Université de Montréal

Beyond the Status Quo in Deep Reinforcement Learning

par

Rishabh Agarwal

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Informatique

2 mai 2024

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Beyond the Status Quo in Deep Reinforcement Learning

présentée par

Rishabh Agarwal

a été évaluée par un jury composé des personnes suivantes :

Yoshua Bengio
(président-rapporteur)

Aaron Courville
(directeur de recherche)

Marc Bellemare
(codirecteur)

Glen Berseth
(membre du jury)

Deepak Pathak
(examinateur externe)

(représentant du doyen de la FESP)

Résumé

L'apprentissage par renforcement profond (RL) a connu d'énormes progrès ces dernières années, mais il est encore difficile d'appliquer le RL aux problèmes de prise de décision du monde réel. Cette thèse identifie trois défis clés avec la façon dont nous faisons la recherche RL elle-même qui entravent les progrès de la recherche RL.

— Évaluation et comparaison peu fiables des algorithmes RL ; les méthodes d'évaluation actuelles conduisent souvent à des résultats peu fiables.

— Manque d'informations préalables dans la recherche RL ; Les algorithmes RL sont souvent formés à partir de zéro, ce qui peut nécessiter de grandes quantités de données ou de ressources informatiques.

— Manque de compréhension de la façon dont les réseaux de neurones profonds interagissent avec RL, ce qui rend difficile le développement de méthodes évolutives de RL.

Pour relever ces défis susmentionnés, cette thèse apporte les contributions suivantes :

— Une méthodologie plus rigoureuse pour évaluer les algorithmes RL.

— Un flux de travail de recherche alternatif qui se concentre sur la réutilisation des progrès existants sur une tâche.

— Identification d'un phénomène de perte de capacité implicite avec un entraînement RL hors ligne prolongé.

Dans l'ensemble, cette thèse remet en question le statu quo dans le RL profond et montre comment cela peut conduire à des algorithmes de RL plus efficaces, fiables et mieux applicables dans le monde réel.

Mots-cles. Apprentissage par renforcement profond, RL profond, évaluation, réutilisation du calcul, RL réincarné, RL hors ligne, régularisation implicite

Abstract

Deep reinforcement learning (RL) has seen tremendous progress in recent years, but it is still difficult to apply RL to real-world decision-making problems. This thesis identifies three key challenges with how we do RL research itself that hinder the progress of RL research.

— Unreliable evaluation and comparison of RL algorithms; current evaluation methods often lead to unreliable results.

— Lack of prior information in RL research; RL algorithms are often trained from scratch, which can require large amounts of data or computational resources.

— Lack of understanding of how deep neural networks interact with RL, making it hard to develop scalable RL methods.

To tackle these aforementioned challenges, this thesis makes the following contributions:

— A more rigorous methodology for evaluating RL algorithms.

— An alternative research workflow that focuses on reusing existing progress on a task.

— Identifying an implicit capacity loss phenomenon with prolonged offline RL training.

Overall, this thesis challenges the status quo in deep reinforcement learning and shows that doing so can make RL more efficient, reliable and improve its real-world applicability.
Keywords. Deep Reinforcement Learning, Deep RL, Evaluation, Reusing Computation, Reincarnating RL, Offline RL, Implicit Regularization

Contents

Acknowledgments

In 2018, I started as an AI resident at Google Brain Toronto in Geoff Hinton's team. Geoff taught me to be optimistic about research even in the face of failure. Mohammad Norouzi and Dale Schuurmans also made a huge impact by teaching me how to do research as well as how to deal with its emotional highs and lows. Their mentorship, which included weekly meetings filled with animated discussions, fueled my excitement for research. Somewhat ironically, Geoff advised me to not work on reinforcement learning (RL) but also that I should do what I thought was best, even if it meant ignoring advice from established researchers, just like he did. As such, I dove headfirst into RL, becoming so fascinated that I decided to pursue a PhD.

Next year, I interviewed for a PhD position at Mila with Aaron Courville and Marc Bellemare. During my interview with Aaron, he learned about my work dealing with under-specified rewards in RL for program synthesis. Serendipitously, he had been working on the exact same problem and offered me a spot in his research group, which I accepted happily. I was thrilled at the opportunity to live in Montreal, a charming city with an active RL community, with the added bonus of keeping my full-time role at Google Brain team in Montreal. The next four years were a wonderful journey.

I am indebted to my advisors, Aaron and Marc, for their guidance throughout my PhD journey. Their knowledge, expertise, and encouragement have been invaluable to me. Their guidance has helped me grow as a researcher, both in terms of my scientific skills and my professional development. I learned to remain open-minded in research from Aaron, while Marc taught me how to present ideas and write well. Moreover, they both emphasized the importance of doing good science rather than focusing on the quantity of publications. For example, Aaron encouraged me to work on reliable evaluation, covered by chapter 2, even though it meant abandoning another project, because he believed it would help others do good science. Similarly, Marc challenged me to come up with a new experimental paradigm in which one continuously improves on an existing trained (RL) agent, which led to the work covered in chapter 3.

In addition to my advisors, I would like to thank my primary collaborators without whom this thesis is not possible: Max Schwarzer, Pablo Samuel Castro, Aviral Kumar. Chapters

2 and 3 are based on collaboration with Max and Pablo, while chapter 4 includes research co-led by Aviral. In particular, I am grateful to Aviral Kumar for the numerous discussions we had ranging from implicit regularization in RL to where the field is headed, and for being a great collaborator.

In the last few years of my research, I was extremely lucky to have several good collaborators. Apart from the ones mentioned above, it was a pleasure to collaborate with the following people: Sergey Levine, Nick Frosst, Rich Caruana, Marlos C Machado, Caglar Gulcehre, Jacob Buckman, George Tucker, Justin Fu, Adrien Ali Taiga, Chen Liang, William Fedus, Dibya Ghosh, Evegenii Nikishin, Siddharth Aggarwal, Joshua Greaves, Jesse Farebrother, Utku Evci, Prajit Ramachandran, Stephen Tu, Sergio Gómez, Mark Rowland, Will Dabney, Yoshua Bengio, and Nando De Freitas. During my PhD, I was also lucky to mentor Siddarth Aggarwal, Ghada Sokar, Joshua P Zitovsky, Johan Obando-Ceron, and Charline Le Lan. I would like to thank all of them for the knowledge that I gained through mentoring them.

I've also engaged with the broader research community by organizing 5 workshops on topics I worked on during my PhD. These workshops would not have been possible without my wonderful co-organizers. Specifically, I'd like to thank Aviral Kumar, Doina Precup, Lihong Li, Nan Jiang, Aravind Rajeshwaran, Justin Fu, and Wenxuan Zhou for co-organizing offline RL workshops at NeurIPS 2020, 2021 and 2022. I am grateful to Stephanie Chan, Xavier Bouthillier, Jesse Dodge, and Caglar Gulcehre for co-organizing the ML evaluation standards workshop at ICLR 2022. It is also a pleasure to co-organize the Reincarnating RL workshop at ICLR 2023 with Ted Xiao, Max Schwarzer, Susan Zhang, and Yanchao Sun.

I was also lucky to have amazing peers at Google Brain and Mila, who provided the best possible environment for research. At Google, I have had interesting discussions with Simon Kornblith, Lihong Li, Hossein Mobahi, Taylor Killian, John D Martin, Erin Grant, Zafarali Ahmed, Hanie Sedghi, Rosanne Liu, Michal Valko, Ben Poole, Sara Hooker, Kevin Swersky, Chen Liang, Ross Goroshin, Jean Harb, Ofir Nachum, Owen He, Igor Mordatch, Sherry Yang, Bo Dai, Ankit Anand, Laura Graessar, Daniel Johnson, Fabio Viola, Daniel Toyoma, Kory Matthewson, Anita Gergely, Nino Veillard, Olivie Bachem, Robert Dadashi, Saurabh Kumar, and Georg Ostrovski. A special thanks to my managers Natacha Mainville and Aleksandra Faust as well as Hugo Larochelle, who leads the Brain Montreal team, for making sure that I succeed in my research and work at Google. At Mila, I am thankful to the amazing set of friends and colleagues who were always eager to talk through ideas and encourage each other, in particular, Max Schwarzer, Ankesh Anand, Hattie Zhou, Michael Noukhovitch, Emmanuel Bengio, Jacob Buckman, Bogdan Mazoure, Scott Fujimoto, and Evgenii Nikishin for stimulating discussions.

Last but not the least, I am grateful to my family and friends for their love and support. They have always believed in me, even when I didn't believe in myself. I would especially like to thank my mom and partner for their unwavering support.

Introduction

Decision making is a complex process that is fundamental to human intelligence. It involves the selection of an action from among a set of alternatives based on various factors, such as personal preferences, situational context, and available information. It is a critical aspect of our daily lives, from mundane choices like what to wear or eat to significant life-altering decisions like career choices and life partners. Moreover, our current decisions often shape our future outcomes, making it essential to understand the mechanisms underlying effective decision making. One of the key challenges in decision making is making a series of decisions that lead to a desired outcome, known as sequential decision making. This is the fundamental problem studied in the field of reinforcement learning.

Reinforcement learning (RL) is a branch of machine learning that focuses on developing computational approaches for creating artificial agents that can learn to make effective decisions in sequential decision-making problems. In RL, an agent interacts with an environment by selecting actions to take and receiving a scalar reward signal in response to those actions. The goal of the agent is to learn a policy that maximizes its cumulative reward over time. One significant development in the last decade has been the integration of RL with deep neural networks, known as deep reinforcement learning. Deep RL has lead to tremendous advances in several challenging decision-making problems such as playing challenging games [155, 196], flying stratospheric balloons [23], control for nuclear fusion [55], robotic manipulation [9, 1], and aligning models with human preferences [167]. Despite such successes, it is still difficult to apply deep RL to tackle real-world decision making problems.

One major issue with deep RL is that it often results in high variability in performance, making it hard to evaluate and compare different algorithms. Additionally, the agents used in deep RL typically learn from their own experience by interacting with their environment, which means they need to collect large amounts of data for each experiment. This can be time-consuming and expensive in practical settings. Moreover, deep RL agents are usually trained to perform well only in the environments they were trained on, and they may not be able to generalize to new, unseen environments. Another challenge is that deep RL approaches usually learn without making use of any prior information, which can limit their scalability to more complex problems due to their sample and compute inefficiency. Furthermore, the

RL community still does not have a good understanding of how the "deep" part interacts with RL in deep RL.

This thesis pushes beyond the status quo in deep RL and takes steps towards mitigating some of the issues that hinder the practical applicability of deep RL. First, the thesis proposes reliable protocols for evaluating RL algorithms for avoiding false impressions of progress in the field and ensuring continual advancement. Second, many challenging decision-making problems are currently infeasible with deep RL, and the thesis suggests building upon existing computational work to overcome this limitation. This proposal can further democratize RL and is especially relevant for large-scale decision-making systems. Third, the thesis explores how the implicit regularization of gradient descent with neural networks can be problematic in RL, resulting in a peculiar underfitting phenomenon in offline RL with prolonged training.

Thesis Outline

— Chapter 1 discusses the pre-requisite concepts and terminology required in order to understand this thesis. Additionally, it thoroughly covers prior work to contextualize the work covered in this thesis.

— Chapter 2 discusses evidence of unreliable results in numerous published deep RL papers and proposes a more rigorous evaluation methodology for RL and ML benchmarks that is easily applicable with even a handful of experiment runs. The content is taken from Agarwal et al. [6]:

  — **Rishabh Agarwal**, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. NeurIPS (2021) [Outstanding Paper Award].

— Chapter 3 discusses our work on reincarnating RL, a research workflow that argues for reusing prior computational work in RL. This work is especially relevant for large-scale decision-making systems, where learning from scratch can be impractical. I also organized a workshop on reincarnating RL at ICLR 2023, which was well-received and timely given the availability of large-scale pretrained models in NLP and vision. The content is taken from Agarwal et al. [7]:

  — **Rishabh Agarwal**, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Reincarnating Reinforcement Learning: Reusing Prior Computation to Accelerate Progress. NeurIPS (2022).

— Chapter 4 investigates the "deep" part in deep RL. Specifically, it studies the capacity loss in deep Q-learning with prolonged training, particularly offline RL, which we show to stem from how implicit regularization of gradient descent with neural networks interacts with Q-learning. The content is taken from Agarwal* et al. [3]:

— **Rishabh Agarwal***, Aviral Kumar*, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. ICLR (2021) [* denotes equal contribution].

The papers included above were a result of a multi-author collaboration but I was the primary contributor for the papers included in each of the chapters. As such, I contributed towards formulating the research, writing the paper, implementing most of the code, and running the experiments. Details about motivation, impact and my contributions are included in the preamble for each chapter.

While not discussed in this thesis, I briefly list other promising directions for improving the real-world applicability of RL. One of the reasons behind the success of deep learning is being able to learn from large and diverse datasets and RL using only existing logged datasets [*e.g.,* 4, 243] and using such datasets as a launchpad for future learning would greatly enable RL for real-world applications. Furthermore, deploying RL agents developed in simulation in the wild requires developing RL approaches that can easily transfer knowledge from training tasks to "similar" tasks, such as zero-shot generalization [*e.g.,* 5, 213, 8] or sim-to-real transfer. Moreover, RL methods whose performance scales with model capacity and compute, would enable progress akin to benefits that arise from scaling in deep learning. Some of our recent work finds that scaling RL can enable RL-trained generalist models and achieve significant sample efficiency improvements. Finally, I strongly believe that further democratizing RL to the broader research community would accelerate progress in deep RL.

I have done some work related to the above directions as well as on several other topics that was published during my PhD, but these have not been included in this thesis:

— **Rishabh Agarwal**, Dale Schuurmans, and Mohammad Norouzi. "An optimistic perspective on offline reinforcement learning." ICML (2020). [Oral]

— **Rishabh Agarwal**, Marlos C. Machado, Pablo Samuel Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. ICLR (2021). [Spotlight]

— **Rishabh Agarwal**, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E. Hinton. Neural additive models: Interpretable machine learning with neural nets. NeurIPS (2021). [Spotlight]

— Sokar, Ghada, **Rishabh Agarwal**, Pablo Samuel Castro, and Utku Evci. The Dormant Neuron Phenomenon in Deep Reinforcement Learning. ICML (2023) [Oral].

— Kumar, Aviral, **Rishabh Agarwal**, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization. ICLR (2022). [Spotlight]

— Aviral Kumar, **Rishabh Agarwal**, Xinyang Geng, George Tucker, Sergey Levine. Offline Q-Learning on Diverse Multi-Task Data Both Scales And Generalizes. ICLR (2023) [Top 5%].

— Max Schwarzer, Johan Obando Ceron, Aaron Courville, Marc Bellemare, **Rishabh Agarwal***, Pablo Samuel Castro*. Bigger, Better, Faster: Human-level Atari with human-level efficiency. ICML (2023).

— Taiga, Adrien Ali, **Rishabh Agarwal**, Jesse Farebrother, Aaron Courville, and Marc G. Bellemare. Investigating Multi-task Pretraining and Generalization in Reinforcement Learning. ICLR (2023).

— Farebrother, Jesse, Joshua Greaves, **Rishabh Agarwal**, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G. Bellemare. Proto-Value Networks: Scaling Representation Learning with Auxiliary Tasks. ICLR (2023).

— Zitovsky, Joshua P., Daniel de Marchi, **Rishabh Agarwal**, and Michael R. Kosorok. Revisiting Bellman Errors for Offline Model Selection. ICML (2023).

— Lan, Charline Le, Stephen Tu, Adam Oberman, **Rishabh Agarwal**, and Marc G. Bellemare. On the Generalization of Representations in Reinforcement Learning. AISTATS (2022).

— Agarwal, Siddhant, Aaron Courville, and **Rishabh Agarwal**. Behavior Predictive Representations for Generalization in Reinforcement Learning. RLDM (2022).

— Nikishin, Evgenii, Romina Abachi, **Rishabh Agarwal**, and Pierre-Luc Bacon. Control-Oriented Model-Based Reinforcement Learning with Implicit Differentiation. AAAI (2022).

— Gulcehre, Caglar, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, **Rishabh Agarwal** et al. RL unplugged: A suite of benchmarks for offline reinforcement learning. NeurIPS (2020).

— Fedus, William, Prajit Ramachandran, **Rishabh Agarwal**, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. ICML (2020).

Chapter 1

Background

The field of machine learning (ML) is concerned with the problem of learning from data to make predictions or decisions. Reinforcement learning (RL) is an area of machine learning concerned with sequential decision-making problems by training artificial agents that learn to act optimally based on maximizing their long-term reward. The entity which makes the decision is referred to as the *agent*, and everything other than the agent is considered as the *environment*. In a typical RL formulation, an agent learns through trial and error in an environment, receiving feedback in terms of reward when taking a particular action from a given state. See Sutton & Barto [208] for a more thorough introduction.

1.1. Markov Decision Process

Formally, an interactive environment in RL is typically described as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \gamma)$ [174], with an observation space $\mathcal{S}$, an action space $\mathcal{A}$, a stochastic real-valued reward function $R(s, a)$, transition dynamics $P(s'|s, a)$ and a discount factor $\gamma \in [0, 1)$, which encourages the agent to accumulate rewards sooner rather than later. A stochastic policy $\pi(\cdot \mid s)$ maps each state $s \in \mathcal{S}$ to a distribution (density) over actions.

At each time-step t, the agent receives observation $s_t \in \mathcal{S}$ from its environment $\mathcal{M}$ and executes an action $a_t \in \mathcal{A}$ according to its behavior policy. The environment then provides a feedback signal in form of reward $r_{t+1} \in \mathbb{R}$ and a new observation $s_{t+1} \in \mathcal{S}$ to the agent. This series of actions, observations and rewards defines the agent's *experience*. The goal of RL is to improve the agent's future reward given its past experience. The MDP formulation assumes that the next observation and reward depends only on the current observation and action,

$$Pr(s_{t+1}, r_{t+1}|s_1, a_1, r_1, ..., s_t, a_t, r_t) = Pr(s_{t+1}, r_{t+1}|s_t, a_t) \tag{1.1.1}$$

For a MDP, the observation s_t summarizes all the information present in the previous time steps and is also referred as the *markovian state*. At any given time t, the discounted

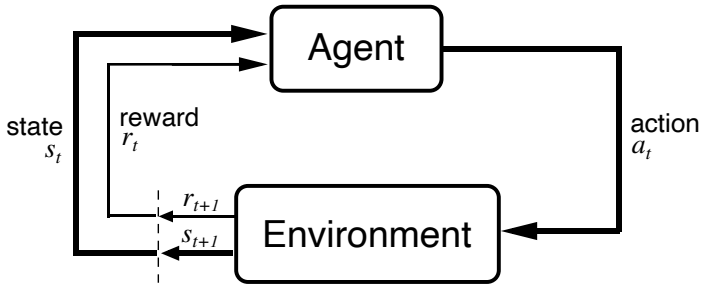

Figure 1. Image from Sutton & Barto [209] showing interaction between an agent and an RL environment.

return G_t can be considered as a summary of future rewards:

$$\begin{aligned} G_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \\ &= r_{t+1} + \gamma G_{t+1} \end{aligned}$$

The goal of RL is to learn a policy π that maximizes the expected long-term discounted return $\mathbb{E}_\pi[G_0]$ starting from the initial observation s_0. To do so, we focus on RL methods, that can learn directly from experience and do not assume any access to knowledge of the environment's dynamics. Specifically, we focus on *value-based* methods that learn the value of states $V(s)$ or state-action pairs $Q(s,a)$. Optimal policies then involve taking actions with the maximum expected value.

1.2. Value-based Reinforcement Learning

Our experiments in chapter 3 build on value-based deep RL methods, which we discuss in detail here. Many successful examples of RL use a *value function* to summarize the expected reward for a decision-making policy [*e.g.,* 215, 157, 196]. The value function $V^\pi(s)$ is the expected return from state s when following policy π. Similarly, for an agent following the policy π, the action-value function (also known as Q-function), denoted $Q^\pi(s,a)$, is defined as the expectation of cumulative discounted return, *i.e.,*

$$Q^\pi(s,a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t)\right], s_0 = s, a_0 = \quad a, s_t \sim P(\cdot \mid s_{t-1}, a_{t-1}), a_t \sim \pi(\cdot \mid s_t).$$

The goal of RL is to find an optimal policy π^* that attains maximum expected return, for which $Q^{\pi^*}(s,a) \geq Q^\pi(s,a)$ for all π, s, a. The Bellman optimality equations [25] characterize the optimal policy in terms of the optimal Q-values, denoted $Q^* = Q^{\pi^*}$, via:

$$Q^*(s,a) = \mathbb{E}\ R(s,a) + \gamma \mathbb{E}_{s' \sim P} \max_{a' \in \mathcal{A}} Q^*(s',a')\ . \tag{1.2.1}$$

To learn a policy from interaction with the environment, **Q**-learning. [230] iteratively improves an approximate estimate of Q^*, denoted Q_θ, by repeatedly regressing the LHS of (1.2.1) to target values defined by samples from the RHS of (1.2.1).

Deep Q-learning. For large and complex state spaces, approximate Q-values are obtained using a neural network as the function approximator. To further stabilize optimization, a target network $Q_{\theta'}$ with frozen parameters may be used for computing the learning target [155]. The target network parameters θ' are updated to the current Q-network parameters θ after a fixed number of time steps.

DQN [155, 157] parameterizes Q_θ with a convolutional neural network [128] and uses Q-learning with a target network while following an ϵ-greedy policy for exploration, *i.e.*,, taking a random action with probability ϵ and otherwise taking the greedy action with respect to Q_θ, for data collection. DQN minimizes the temporal difference (TD) error Δ_θ using the loss $\mathcal{L}_{TD}(\mathcal{D}_S)$ on mini-batches of agent's past experience tuples, (s, a, r, s'), sampled from an experience replay buffer $\mathcal{D}_S$ [137] collected during training:

$$\begin{aligned} \mathcal{L}_{TD}(\mathcal{D}) &= \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\ell_\lambda \left(\Delta_\theta(s, a, r, s') \right) \right], \\ \Delta_\theta(s, a, r, s') &= \; Q_\theta(s, a) - r - \gamma \max_{a'} Q_{\theta'}(s', a') \end{aligned} \tag{1.2.2}$$

where l_λ is the Huber loss [99] given by

$$\ell_\lambda(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \leq \lambda \\ \lambda(|u| - \frac{1}{2}\lambda), & \text{otherwise.} \end{cases} \tag{1.2.3}$$

Q-learning is an *off-policy* algorithm [209], since the learning target can be computed without any consideration of how the experience was generated.

Distributional RL algorithms are a family of recent off-policy deep RL algorithms that estimate a density over returns for each state-action pair, denoted $Z^\pi(s, a)$, instead of directly estimating the mean $Q^\pi(s, a)$. Accordingly, one can express a form of distributional Bellman optimality as

$$\begin{aligned} Z^*(s, a) &\overset{D}{=} r + \gamma Z^*(s', \text{argmax}_{a' \in \mathcal{A}} \; Q^*(s', a')), \\ &\text{where } r \sim R(s, a),\; s' \sim P(\cdot \mid s, a). \end{aligned} \tag{1.2.4}$$

and $\overset{D}{=}$ denotes distributional equivalence and $Q^*(s', a')$ is estimated by taking an expectation with respect to $Z^*(s', a')$. C51 [22] approximates $Z^*(s, a)$ by using a categorical distribution over a set of pre-specified anchor points, and distributional QR-DQN [52] approximates the return density by using a uniform mixture of K Dirac delta functions, *i.e.,*

$$Z_\theta(s, a) := \frac{1}{K} \sum_{i=1}^{K} \delta_{\theta_i(s,a)},\; Q_\theta(s, a) = \frac{1}{K} \sum_{i=1}^{K} \theta_i(s, a).$$

QR-DQN outperforms C51 and DQN on Atari 2600 games [20], a widely-used benchmark for evaluating deep RL algorithms. Please refer to [24] for a more thorough introduction to distributional RL.

Hessel et al. [92] combined six components into a single agent they called *Rainbow*: prioritized experience [188], n-step learning [206], distributional RL [21], double Q-learning [220], dueling architecture [227] and NoisyNets [72]. Ceron & Castro [37] and Hessel et al. [92] both showed that n-step learning is one of the most crucial components of Rainbow, in that removing it caused a large drop in performance.

In n-step learning, instead of computing the temporal difference error using a single-step transition, one can use n-step targets instead [206], where for a trajectory $(s_0, a_0, r_0, s_1, a_1, \cdots)$ and update horizon n: $R_t^{(n)} := \sum_{k=0}^{n-1} \gamma^k r_{t+k+1}$, yielding the multi-step temporal difference: $R_t^{(n)} + \gamma^n \max_{a'} Q_\theta(s_{t+n}, a') - Q_\theta(s_t, a_t)$.

1.3. Offline Reinforcement Learning

Modern off-policy deep RL algorithms (as discussed above) perform remarkably well on common benchmarks such as the Atari 2600 games [20] and continuous control MuJoCo tasks [216]. Such off-policy algorithms are considered "online" because they alternate between optimizing a policy and using that policy to collect more data. Typically, these algorithms keep a sliding window of most recent experiences in a finite replay buffer, throwing away stale data to incorporate most on-policy experiences.

Offline RL, in contrast to online RL, describes the fully off-policy setting of learning using a fixed dataset of experiences, without any further interactions with the environment. Akin to supervised learning, offline RL also utilizes a fixed dataset but unlike supervised learning, we don't have access to the true labels (*i.e.,* optimal actions) and is harder than supervised learning. Offline RL is considered challenging due to the *distribution mismatch* [74, 120, 232, 195] between the current policy and the offline data collection policy, *i.e.,* when the policy being learned takes a different action than the data collection policy, we don't know the reward that should be provided. Such papers propose remedies by regularizing the learned policy to stay close to training trajectories and we investigate one such approach in chapter 3. Please refer to Levine et al. [133] for a survey on offline RL.

1.4. Evaluation in Deep RL

Deep RL algorithms are known to have high variability in performance, which makes it difficult to reliably evaluate such algorithms. Recently, various efforts explicitly investigate the reproducibility of reported results [91, 104, 47]. These efforts are in part due to the inadequate experimental practices and reporting in RL and general machine learning [173, 138]. Similar to such efforts, our work in chapter 2 was motivated by the need for more reliable evaluation protocols to compare RL algorithms. We discuss differences with prior work in detail below.

While prior work [104, 91, 147] highlights various reproducibility issues in policy-gradient methods, chapter 2 focuses specifically on the reliability of evaluation procedures on RL

benchmarks and provides an extensive analysis on common deep RL algorithms on widely-used benchmarks. For more rigorous performance comparisons on a single RL task, Henderson et al. [91], Colas et al. [48] provide guidelines for statistical significance testing while Colas et al. [47] focuses on determining the minimum number of runs needed for such comparisons to be statistically significant. Instead, we focus on reliable comparisons on a suite of tasks and mainly recommend reporting stratified bootstrap CIs due to the dichotomous nature and wide misinterpretation of statistical significance tests (see Remark in Section 2.2). Henderson et al. [91], Colas et al. [47, 48] also discuss bootstrap CIs but for reporting single task mean scores – however, 3-5 runs is a small sample size for bootstrapping on Atari 100k, to achieve true coverage close to 95%, such CIs require at least 20-30 runs per task as opposed to 5-10 runs for stratified bootstrap CIs we present later.

Chan et al. [38] propose metrics to measure the reliability of RL algorithms in terms of their stability during training and their variability and risk in returns across multiple episodes. While our work focuses on reliability of evaluation in deep RL itself, performance profiles (subsection 2.4.2) showing the tail distribution of episodic returns, applicable for even a single task with multiple runs, can be useful for measuring reliability of an algorithm's performance.

Jordan et al. [109] propose a game-theoretic evaluation procedure for "complete" algorithms that do not require any hyperparameter tuning and recommend evaluating between 1,000 to 10,000 runs per task to detect statistically significant results. Instead, our work focuses on reliably evaluating performance obtained after the hyperparameter tuning phase, even with just a handful of runs. That said, run-score distributions based on runs with different hyperparameter configurations might reveal sensitivity to hyperparameter tuning.

In chapter 2, we propose reporting *score distributions*, which corresponds to performance profiles [59] with performance thresholds on the x-axis and fractions of all runs above that threshold on the y-axis, to reveal the variability in performance across tasks. Recht [178] proposed an alternative to *score distributions*, which we call R-profiles, where scores in a performance profile are replaced by the probability that average task scores of a given method outperforms the best method (among a given set of methods). The probability in R-profiles is computed using the Welsh's t-test [231]. However, R-profiles have several limitations compared to score distributions. R-profiles lead to a biased estimate and are less robust to outlier runs. Furthermore, such profiles are insensitive to the size of performance differences, *i.e.,* two methods that are uniformly 1% and 100% worse than the best method are assigned the same probability. Moreover, R-profile is only sensible when task score distributions are Gaussian, as required by Welsh's t-test. Finally, the ranking of methods depends on the specific set of methods being compared in such profiles.

1.5. Prior Computation in RL

While learning tabula rasa - that is without any prior task-specific knowledge such as offline datasets or prior policies - is the prevalent workflow in RL research, it is computationally and sample inefficient. To address these inefficiencies, chapter 3 introduces an alternative workflow, which we call reincarnating RL, that focuses on leveraging prior computational work (*e.g.,* learned policies) to speed up research progress in RL. To discuss challenges in setting up this workflow, we investigate policy-to-value reincarnating RL (PVRL) that focuses on accelerating a value-based RL agent given access to an existing suboptimal policy. Next, we discuss prior works related to reincarnating RL, including several large-scale deep RL efforts.

Large-scale reincarnation efforts. Several high-profile RL achievements employ a limited form of reincarnating RL. OpenAI Five [27], which can play Dota2 at a superhuman level, required 10 months of large-scale RL training and went through continual changes in code and environment (*e.g.,* expanding observation spaces) during development. To avoid restarting from scratch after such changes, OpenAI Five used "**surgery**" akin to Net2Net [41] style transformations to convert a trained model to certain bigger architectures with custom weight initializations. AlphaStar [225] employs population-based training (**PBT**) [106], which periodically copies weights of the best performing value-based agents and mutates hyperparameters during training. Although PBT and surgery methods are efficient, they *can not* be used for reincarnating RL when switching to arbitrary architectures (*e.g.,* feed-forward to recurrent networks) or from one model class to another (*e.g.,* policy to a value function). In this work, we focus on reincarnating RL methods that allow such architecture and algorithm changes. Akkaya et al. [9] trained RL policies for several months to manipulate a robot hand for solving Rubik's cube. To do so, they "rarely trained experiments from scratch" but instead initialized new policies, with architectural changes, from previous trained policies using **behavior cloning** via on-policy distillation [170, 49]. AlphaGo [196] also used behavior cloning on human replays for initializing the policy and fine-tuning it further with RL. However, behavior cloning is only applicable for policy to policy transfer and is inadequate for the PVRL setting of transferring a policy to a value function [*e.g.,* 164, 219]. Several prior works also **fine-tune** existing agents with deep RL for reducing training time, especially on real-world tasks such as chip floor-planning [154], robotic manipulation [110], aligning language models [15], and compiler optimization [218]. In line with these works, we find that fine-tuning a value-based agent can be an effective reincarnation strategy (Figure 7). However, fine-tuning is *constrained* to use the same architecture as the agent being fine-tuned. Instead, we focus on reincarnating RL methods that do not have this limitation.

Leveraging existing agents. Existing policies have been previously used for improving data collection in RL [200, 28, 39, 233, 71]; we evaluate one such recent approach, JSRL [219],

which improves exploration in goal-reaching RL tasks. However, our PVRL experiments indicate that JSRL performs poorly on ALE. Closely related to PVRL, Schmitt et al. [190] propose kickstarting to speed-up actor-critic agents using an interactive teacher policy by combining on-policy distillation [49, 170] with RL. Empirically, we find that kickstarting is a strong baseline for PVRL, however it exhibits unstable behavior without n-step returns and underperforms our proposed QDagger, which we discuss further in chapter 3. PVRL also falls under the framework of agents teaching agents (ATA) [50] with RL-based students and teachers. While ATA approaches, such as action advice [217], emphasize how and when to query the teacher or evaluating the utility of teacher advice, PVRL focuses on sample-efficient transfer and does not impose constraints on querying the teacher. PVRL is also different from prior work on accelerating RL using a heuristic or oracle value function [43, 204, 18], as PVRL only assumes access to a suboptimal policy. Lee et al. [132] tackle robotic manipulation tasks given a teacher policy and find that training on both the teacher and student collected data, akin to our proposed method, enables best performance. However, unlike PVRL methods that wean off the teacher, Lee et al. [132] propose methods that are constrained to stay close to the suboptimal teacher, which can limit the student's performance with continued training (Figure 9).

Leveraging prior data. Learning from demonstrations (LfD) [187, 11, 94, 76, 100] approaches focus on accelerating RL training using demonstrations. Such approaches typically assume access to optimal or near-optimal trajectories, often obtained from human demonstrators, and aim to match the demonstrator's performance. Instead, PVRL focuses on leveraging a suboptimal teacher policy, which can be obtained from any trained RL agent, that we wean off during training. Empirically, we find that DQfD [94], a well-known LfD approach to accelerate deep Q-learning, when applied to PVRL, exhibits severe performance degradation when weaning off the teacher. Rehearsal approaches [169, 163, 199] focus on improving exploration by replaying demonstrations during learning; we find that such approaches are ineffective for leveraging the teacher in PVRL. Offline RL [125, 133, 4] focuses on learning *solely* from fixed datasets while reincarnating RL focuses on leveraging prior information, which can also be presented as offline datasets, for speeding up further learning from environment interactions. Recent work [143, 164, 116, 132] use offline RL to pretrain on prior data and then fine-tune online. We also evaluate this pretraining approach for PVRL. However, PVRL is more flexible than only using a fixed dataset for pretraining, as it assumes access to an interactive teacher policy.

Chapter 2

Reliable Evaluation in Deep RL

***Origin and Impact**: I started this project as I wasn't able to get reliable results on the Atari 100K benchmark: each time I added more seeds, median normalized score improved by a substantial amount (in hindsight, this was simply the statistical bias of the median). Aaron and Marc encouraged me to further investigate this finding as opposed to continue my ongoing project and move to another benchmark. The investigation resulted in this paper, which also received an outstanding paper award at NeurIPS 2021! The RL community has started to pick up some of the recommendations by this work, which has already been employed by 250+ papers. Following this work, I also co-led a workshop on ML evaluation standards at ICLR 2022 to further bring attention to this important topic.*

***Contribution**: I initiated the project and drove it from start-to-finish: (1) identifying problems in prevalent evaluation protocols, (2) proposing more rigorous statistical protocols and running extensive experiments, (3) carefully and diplomatically interacting with external authors whose results were brought into question, and (4) writing the first draft.*

***Abstract**: Deep RL algorithms are predominantly evaluated by comparing their relative performance on a large suite of tasks. Most published results on deep RL benchmarks compare* point estimates *of aggregate performance such as mean and median scores across tasks, ignoring the statistical uncertainty implied by the use of a finite number of training runs. Beginning with the Arcade Learning Environment (ALE), the shift towards computationally-demanding benchmarks has led to the practice of evaluating only a small number of runs per task, exacerbating the statistical uncertainty in point estimates. In this paper, we argue that reliable evaluation in the few-run deep RL regime cannot ignore the uncertainty in results without running the risk of slowing down progress in the field. We illustrate this point using a case study on the Atari 100k benchmark, where we find substantial discrepancies between*

conclusions drawn from point estimates alone versus a more thorough statistical analysis. With the aim of increasing the field's confidence in reported results with a handful of runs*, we advocate for reporting interval estimates of aggregate performance and propose performance profiles to account for the variability in results, as well as present more robust and efficient aggregate metrics, such as interquartile mean scores, to achieve small uncertainty in results. Using such statistical tools, we scrutinize performance evaluations of existing algorithms on other widely used RL benchmarks including the ALE, Procgen, and the DeepMind Control Suite, again revealing discrepancies in prior comparisons. Our findings call for a change in how we evaluate performance in deep RL, for which we present a more rigorous evaluation methodology, accompanied with an open-source library* rliable*, to prevent unreliable results from stagnating the field.*

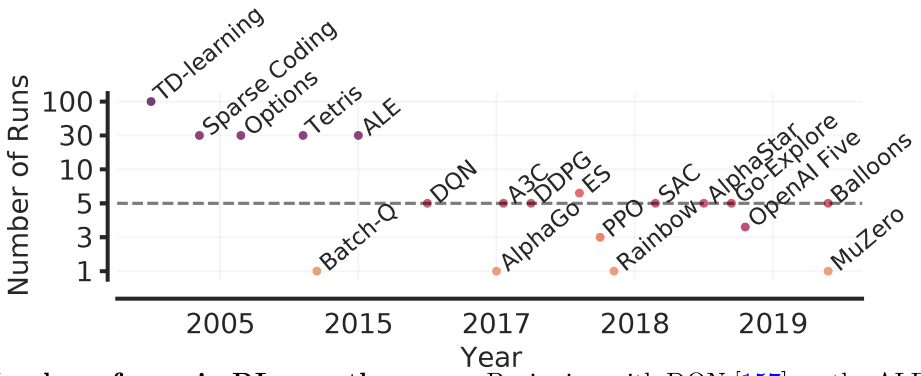

Figure 1. Number of runs in RL over the years. Beginning with DQN [157] on the ALE, 5 or less runs are common in the field. Here, we show representative RL papers with empirical results, in the order of their publication year: TD-learning [205], Sparse coding [207], Options [210], Tetris (CEM) [212], Batch-Q [66], ALE [20], DQN [157], AlphaGo [196], A3C [158], DDPG [135], ES [185], PPO [192], SAC [86], Rainbow [93], AlphaStar [225], Go-Explore [62], OpenAI Five [27], Balloon navigation [23] and MuZero [191].

2.1. Introduction

Research in artificial intelligence, and particularly deep reinforcement learning (RL), relies on evaluating *aggregate* performance on a diverse suite of tasks to assess progress. Quantitative evaluation on a suite of tasks, such as Atari games [20], reveals strengths and limitations of methods while simultaneously guiding researchers towards methods with promising results. Performance of RL algorithms is usually summarized with a *point estimate* of task performance measure, such as mean and median performance across tasks, aggregated over independent training runs [1].

A small number of training runs (Figure 1) coupled with high variability in performance of deep RL algorithms [91, 150, 44, 38, 147], often leads to substantial statistical uncertainty in reported point estimates. While evaluating more runs per task has been prescribed to reduce uncertainty and obtain reliable estimates [91, 47, 109], 3-10 runs are prevalent in deep RL as it is often computationally prohibitive to evaluate more runs. For example, 5 runs each on 50+ Atari 2600 games in ALE using standard protocol requires more than 1000 GPU training days [37]. As we move towards more challenging and complex RL benchmarks (*e.g.,* StarCraft [225]), evaluating more than a handful of runs will become increasingly demanding due to increased amount of compute and data needed to tackle such tasks. Additional confounding factors, such as exploration in the low-data regime, exacerbates the performance variability in deep RL – as seen on the Atari 100k benchmark [111] – often requiring many more runs to achieve negligible statistical uncertainty in reported estimates.

Ignoring the statistical uncertainty in deep RL results gives a false impression of fast scientific progress in the field. It inevitably evades the question: "Would similar findings be obtained with new independent runs under different random conditions?" This could steer researchers towards superficially beneficial methods [30, 58, 31], often at the expense

1. A "run" refers to training an RL agent in an environment. It typically involves the agent interacting with the environment, collecting experiences, and updating its learned parameters based on those experiences.

of better methods being neglected or even rejected early [153, 144] as such methods fail to outperform inferior methods simply due to less favorable random conditions. Furthermore, only reporting point estimates obscures nuances in comparisons [181] and can erroneously lead the field to conclude which methods are *state-of-the-art* [136, 179], ensuing wasted effort when applied in practice [223]. Moreover, not reporting the uncertainty in deep RL results makes them difficult to reproduce except under the *exact* same random conditions, which could lead to a *reproducibility crisis* similar to the one that plagues other fields [102, 171, 16]. Finally, unreliable results could erode trust in deep RL research itself [103].

In this work, we show that recent deep RL papers compare unreliable point estimates, which are dominated by statistical uncertainty, as well as exploit non-standard evaluation protocols, using a case study on Atari 100k (Section 2.3). Then, we illustrate how to reliably evaluate performance with only *a handful of runs* using a more rigorous evaluation methodology that accounts for uncertainty in results (Section 2.4). To exemplify the necessity of such methodology, we scrutinize performance evaluations of existing algorithms on widely used benchmarks, including the ALE [20] (Atari 100k, Atari 200M), Procgen [45] and DeepMind Control Suite [214], again revealing discrepancies in prior comparisons (Section 2.5). Our findings call for a change in how we evaluate performance in deep RL, for which we present a better methodology to prevent unreliable results from stagnating the field.

How do we reliably evaluate performance on deep RL benchmarks with only a handful of runs? As a practical solution that is easily applicable with 3-10 runs per task, we identify three statistical tools for improving the quality of experimental reporting. Since any performance estimate based on a finite number of runs is a *random variable*, we argue that it should be treated as such. Specifically, we argue for reporting aggregate performance measures using *interval estimates* via stratified bootstrap confidence intervals, as opposed to point estimates. Among prevalent aggregate measures, mean can be easily dominated by performance on a few outlier tasks, while median has high variability and zero performance on nearly half of the tasks does not change it. To address these deficiencies, we present more *efficient* and *robust* alternatives, such as *interquartile mean*, which are not unduly affected by outliers and have small uncertainty even with a handful of runs. Furthermore, to reveal the variability in performance across tasks, we propose reporting performance distributions across all runs. Compared to prior work [20, 178], these distributions result in *performance profiles* [59] that are statistically unbiased, more robust to outliers, and require fewer runs for smaller uncertainty.

2.2. Formalism

We consider the setting in which a reinforcement learning algorithm is evaluated on M tasks. For each of these tasks, we perform N independent runs [2] which each provide a scalar, *normalized score* $x_{m,n}$, $m = 1, \ldots, M$ and $n = 1, \ldots, N$. These normalized scores are obtained by linearly rescaling per-task scores [3] based on two reference points; for example, performance on the Atari games is typically normalized with respect to a random agent and an average human, who are assigned a normalized score of 0 and 1 respectively [157]. We denote the set of normalized scores by $x_{1:M,1:N}$.

In most experiments, there is inherent randomness in the scores obtained from different runs. This randomness can arise from stochasticity in the task, exploratory choices made during learning, randomized initial parameters, but also software and hardware considerations such as non-determinism in GPUs and in machine learning frameworks [242]. Thus, we model the algorithm's normalized score on the m^{th} task as a real-valued random variable X_m. Then, the score $x_{m,n}$ is a realization of the random variable $X_{m,n}$, which is identically distributed as X_m. For $\tau \in \mathbb{R}$, we define the tail distribution function of X_m as $F_m(\tau) = \mathrm{P}(X_m > \tau)$. For any collection of scores $y_{1:K}$, the *empirical tail distribution function* is given by $\hat{F}(\tau; y_{1:K}) = \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}[y_k > \tau]$. In particular, we write $\hat{F}_m(\tau) = \hat{F}(\tau; x_{m,1:N})$.

The *aggregate performance* of an algorithm maps the set of normalized scores $x_{1:M,1:N}$ to a scalar value. Two prevalent aggregate performance metrics are the mean and median normalized scores. If we denote by $\bar{x}_m = \frac{1}{N}\sum_{n=1}^{N} x_{m,n}$ the average score on task m across N runs, then these aggregate metrics are $\mathrm{Mean}(\bar{x}_{1:M})$ and $\mathrm{Median}(\bar{x}_{1:M})$. More precisely, we call these *sample mean* and *sample median* over the task means since they are computed from a finite set of N runs. Since $\bar{x}_m$ is a realization of the random variable $\bar{X}_m = \frac{1}{N}\sum_{n=1}^{N} X_{m,n}$, the sample mean and median scores are *point estimates* of the random variables $\mathrm{Mean}(\bar{X}_{1:M})$ and $\mathrm{Median}(\bar{X}_{1:M})$ respectively. We call *true mean* and *true median* the metrics that would be obtained if we had unlimited experimental capacity ($N \to \infty$), given by $\mathrm{Mean}(\mathbb{E}[X_{1:M}])$ and $\mathrm{Median}(\mathbb{E}[X_{1:M}])$ respectively.

Confidence intervals (CIs) for a finite-sample score can be interpreted as an estimate of plausible values for the true score. A $\alpha \times 100\%$ CI computes an interval such that if we rerun the experiment and construct the CI using a different set of runs, the fraction of calculated CIs (which would differ for each set of runs) that contain the true score would tend towards $\alpha \times 100\%$, where $\alpha \in [0, 1]$ is the nominal coverage rate. 95% CIs are typically used in practice. If the true score lies outside the 95% CI, then a sampling event has occurred which had a probability of 5% of happening by chance.

2. A run can be different from using a fixed random seed. Indeed, fixing the seed may not be able to control all sources of randomness such as non-determinism of ML frameworks with GPUs.

3. Often the average undiscounted return obtained during an episode (see Chapter 1 for an explanation of the reinforcement learning setting).

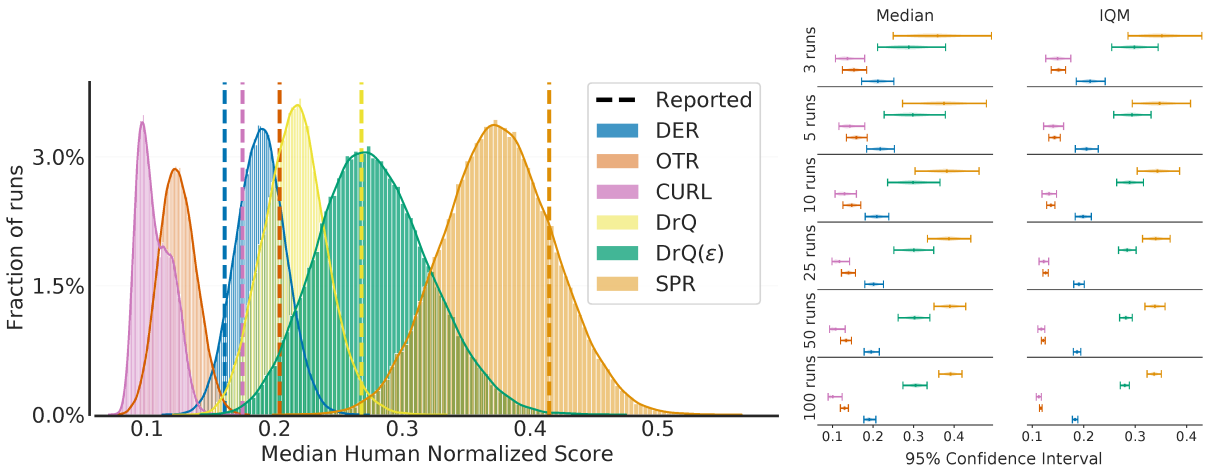

Figure 2. Left. Distribution of median normalized scores computed using 100,000 different sets of N runs subsampled uniformly with replacement from 100 runs. For a given algorithm, the sampling distribution shows the variation in the median scores when re-estimated using a different set of runs. The reported *point estimates* of median in publications, as shown by dashed lines, do not provide any information about the variability in median scores and severely overestimate or underestimate the expected median. We use the same number of runs as reported by publications: $N = 5$ runs for DER, OTR and DrQ, $N = 10$ runs for SPR and $N = 20$ runs for CURL. **Right. 95% CIs** for median and IQM scores (Section 2.4.3) for varying N. There is a substantial uncertainty in median scores even with 50 runs. IQM has much smaller CIs than median. Note that when CIs overlap, properly accounting for uncertainty entails computing CIs for score differences.

Remark. Following Amrhein et al. [10], Wasserstein et al. [229], Romer [183], we recommend using confidence intervals for measuring the uncertainty in results and showing effect sizes (*e.g.,* performance improvements over baseline) that are compatible with the given data. Furthermore, we emphasize using statistical thinking but avoid statistical significance tests (*e.g.,* p-value < 0.05) because of their dichotomous nature (significant *vs.* not significant) and common misinterpretations [82, 78, 152] such as 1) lack of statistically significant results does not demonstrate the absence of effect (Figure 2, right), and 2) given enough data, any trivial effect can be statistically significant but may not be practically significant.

2.3. Case Study: The Atari 100k benchmark

We begin with a case study to illustrate the pitfalls arising from the naïve use of point estimates in the few-run regime. Our case study concerns the Atari 100k benchmark [111], an offshoot of the ALE for evaluating data-efficiency in deep RL. In this benchmark, algorithms are evaluated on only 100k steps (2-3 hours of game-play) for each of its 26 games, versus 200M frames in the ALE benchmark. Prior reported results on this benchmark have been computed mostly from 3 [119, 89, 130, 151, 186, 194] or 5 runs [111, 222, 117, 113, 182, 241, 139, 118, 141], and more rarely, 10 [193, 140] or 20 runs [127].

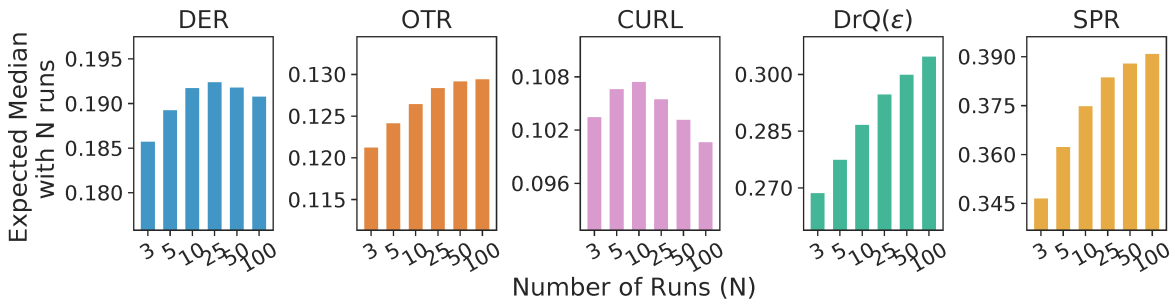

Figure 3. Expected sample median of task means. The expected score for N runs is computed by repeatedly subsampling N runs with replacement out of 100 runs for 100,000 times.

Our case study compares the performance of five recent deep RL algorithms, namely: (1) DER [222] and (2) OTR [113], (3) DrQ [4] [117], (4) CURL [127], and (5) SPR [193]. We chose these methods as representative of influential algorithms within this benchmark. Since good performance on one game can result in unduly high sample means without providing much information about performance on other games, it is common to measure performance on Atari 100k using sample medians.

We investigate statistical variations in the few-run regime by evaluating 100 independent runs for each algorithm, where the score for a run is the average returns obtained in 100 evaluation episodes taking place after training. Each run corresponds to training one algorithm on each of the 26 games in Atari 100k. This provides us with 26×100 scores per algorithm, which we then subsample with replacement to 3–100 runs. The subsampled scores are then used to produce a collection of point estimates whose statistical variability can be measured. We begin by using this experimental protocol to highlight statistical concerns regarding median normalized scores.

High variability in reported results. Our first observation is that the sample medians reported in the literature exhibit substantial variability when viewed as random quantities that depend on a small number of sample runs (Figure 2, left). This shows that there is a fairly large potential for drawing erroneous conclusions based on point estimates alone. As a concrete example, our analysis suggests that DER may in fact be better than OTR, unlike what the reported point estimates suggest. We conclude that in the few-run regime, point estimates are unlikely to provide definitive answers to the question: "Would we draw the same conclusions were we to re-evaluate our algorithm with a different set of runs?"

Substantial bias in sample medians. The sample median is a biased estimator of the true median: $\mathbb{E}[\text{Median}(\bar{X}_{1:M})] \neq \text{Median}(\mathbb{E}[X_{1:M}])$ in general. In the few-run regime, we find that this bias can dominate the comparison between algorithms, as evidenced in Figure 3. For example, the score difference between sample medians with 5 and 100 runs for SPR (+0.03 points) is about 36% of its mean improvement over DrQ(ε) (+0.08 points). Adding to the issue, the magnitude and sign of this bias strongly depends on the algorithm being evaluated.

4. DrQ codebase uses non-standard evaluation hyperparameters. Instead, DrQ(ε) corresponds to DrQ with standard ε-greedy parameters [36, Table 1] in ALE. See Appendix for more details.

Statistical concerns cannot be satisfactorily addressed with few runs. While claiming improvements with 3 or fewer runs may naturally raise eyebrows, folk wisdom in experimental RL suggests that 20 or 30 runs are enough. By calculating 95% confidence interval [5] on sample medians for a varying number of runs (Figure 2, right), we find that this number is closer to 50–100 runs in Atari 100k – far too many to be computationally feasible for most research projects.

Consider a setting in which an algorithm is known to be better – what is the reliability of median and IQM (Section 2.4.3) for accurately assessing performance differences as the number of runs varies? Specifically, we consider two identical N-run experiments involving SPR, except that we artificially inflate one of the experiments' scores by a fixed fraction or *lift* of $+\ell\%$ (Figure 4). In particular, $\ell = 0$ corresponds to running the same experiment twice but with different runs. We find that statistically defensible improvements with median scores is only achieved for 25 runs ($\ell = 25$) and 100 runs ($\ell = 10$). With $\ell = 0$, even 100 runs are insufficient, with deviations of 20% possible.

Changes in evaluation protocols invalidates comparisons to prior work. A typical and relatively safe approach for measuring the performance of an RL algorithm is to average the scores received in their final training episodes [148]. However, the field has seen a number of alternative protocols used, including reporting the maximum evaluation score achieved during training [157, 4, 14] or across multiple runs [107, 67, 177]. A similar protocol is also used by CURL and SUNRISE [130].

Results produced under alternative protocols involving maximum are generally incomparable with end-performance reported results. On Atari 100k, we find that the two protocols produce substantially different results (Figure 5), of a magnitude greater than the actual difference in score. In particular, evaluating DER with CURL's protocol results in scores far above those reported for CURL. In other words, this gap in evaluation procedures resulted in CURL being assessed as achieving a greater true median than DER, where our experiment gives strong support to DER being superior. Similarly, we find that a lot of SUNRISE's improvement over DER can be explained by the change in evaluation protocol (Figure 5).

2.4. Recommendations and Tools for Reliable Evaluation

Our case study shows that the increase in the number of runs required to address the statistical uncertainty issues is typically infeasible for computationally demanding deep RL benchmarks. In this section, we identify three tools for improving the quality of experimental

5. Specifically, we use the m/n bootstrap [29] to calculate the interval between $[2.5^{th}, 97.5^{th}]$ percentiles of the distribution of sample medians (95% CIs).

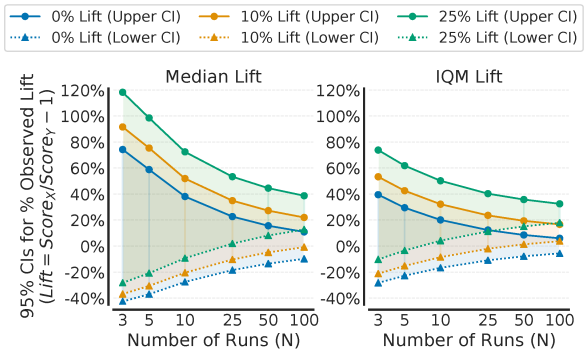

Figure 4. Detecting score lifts. Left. 95% CIs for observed lift with median scores, and **Right.** 95% CIs for observed lift with IQM (Section 2.4.3) when comparing SPR with an algorithm that performs ℓ% better. IQM requires fewer runs than median for small uncertainty.

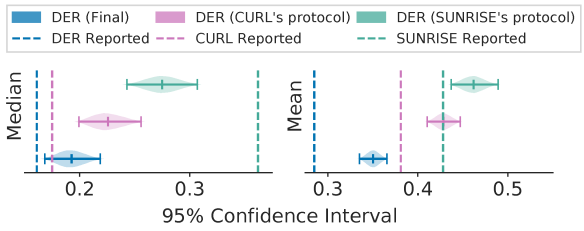

Figure 5. Normalized DER scores with non-standard evaluation protocols. Gains from SUNRISE and CURL over DER can mostly be explained by such protocols.

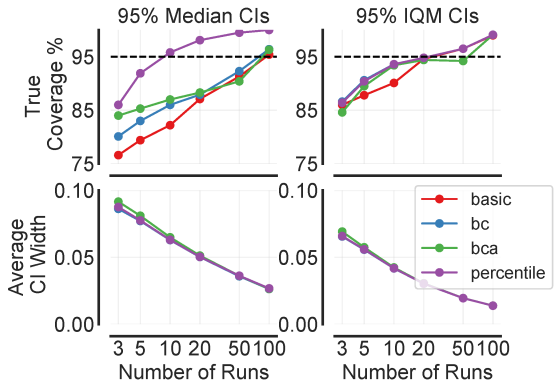

Figure 6. Validating 95% Stratified Bootstrap CIs for a varying number of runs for median and IQM scores for DER. The true coverage % is computed by sampling 10,000 sets of K runs without replacement from 200 runs and checking the fraction of 95% CIs that contains the true estimate approximation based on 200 runs. Note that we evaluate additional 100 runs for DER for an accurate point estimate. Percentile CIs has the best coverage while achieving a small width compared to other methods. Also, CI widths for IQM are much smaller than that of median. We also note that with 3 runs, bootstrap CIs underestimate the true 95% CIs and might require a larger nominal coverage rate to achieve true 95% coverage.

reporting in the few-run regime, all aligned with the principle of accounting for statistical uncertainty in results.

2.4.1. Stratified Bootstrap Confidence Intervals

We first reaffirm the importance of reporting interval estimates to indicate the range within which an algorithm's aggregate performance is believed to lie. Concretely, we propose using bootstrap CIs [63] with stratified sampling for aggregate performance, a method that can be applied to small sample sizes and is better justified than reporting sample standard deviations in this context. While prior work has recommended using bootstrap CIs for reporting uncertainty in single task mean scores with N runs [91, 38, 47], this is less useful when N is small, as *bootstrapping* assumes that re-sampling from the data approximates sampling from the true distribution. We can do better by aggregating samples across tasks, for a total of MN random samples.

To compute the stratified bootstrap CIs, we re-sample runs with replacement independently for each task to construct an empirical bootstrap sample with N runs each for M tasks from which we calculate a statistic and repeat this process many times to approximate the sampling distribution of the statistic. We measure the reliability of this technique in Atari 100k for

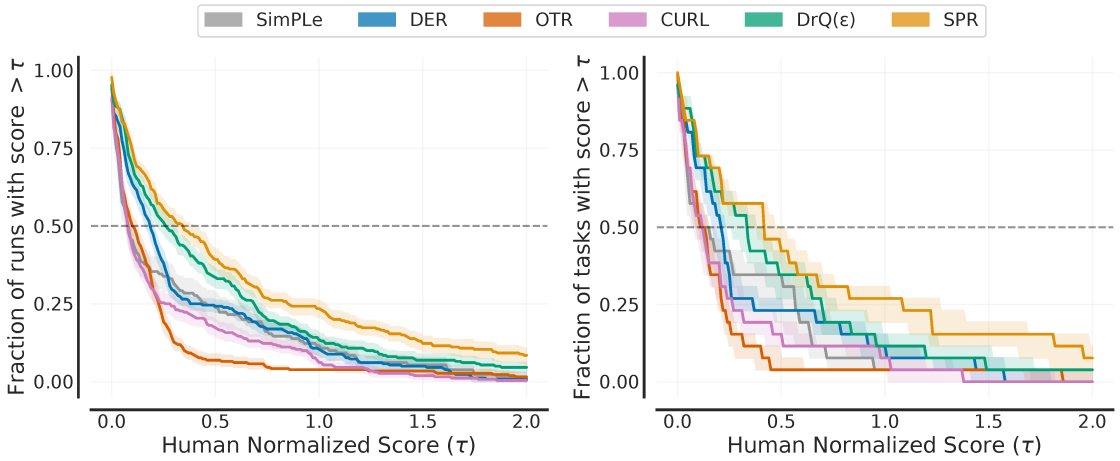

Figure 7. Performance profiles on Atari 100k based on score distributions **(left)**, which we recommend, and average score distributions **(right)**. Shaded regions show pointwise 95% confidence bands based on percentile bootstrap with stratified sampling. The profiles on the left are more robust to outliers and have smaller confidence bands. We use 10 runs to show the robustness of profiles with a few runs. For SimPLe [111], we use the 5 runs from their reported results. The τ value where the profiles intersect $y = 0.5$ shows the median while for a non-negative random variable, area under the performance profile corresponds to the mean.

variable N, by comparing the nominal coverage of 95% to the "true" coverage from the estimated CIs (Figure 6) for different bootstrap methods (see [64]). We find that percentile CIs provide good interval estimates for as few as $N = 10$ runs for both median and IQM scores (Section 2.4.3).

2.4.2. Performance Profiles

Most deep RL benchmarks yield scores that vary widely between tasks and may be heavy-tailed, multimodal, or possess outliers. In this regime, both point estimates, such as mean and median scores, and interval estimates of these quantities paint an incomplete picture of an algorithm's performance [56, Section 3]. Instead, we recommend the use of *performance profiles* [59], commonly used in benchmarking optimization software. While performance profiles from Dolan & Moré [59] correspond to empirical cumulative distribution functions without any uncertainty estimates, profiles proposed herein visualize the empirical tail distribution function (Section 2.2) of a random score (higher curve is better), with pointwise confidence bands based on stratified bootstrap.

By representing the entire set of normalized scores $x_{1:M,1:N}$ visually, performance profiles reveal performance variability across tasks much better than interval estimates of aggregate metrics. Although tables containing per-task mean scores and standard deviations can reveal this variability, such tables tend to be overwhelming for more than a few tasks.[6] In addition, performance profiles are robust to outlier runs and insensitive to small changes in performance across all tasks [59].

6. In addition, standard deviations are sometimes omitted from tables due to space constraints.

In this paper, we propose the use of a performance profile we call run-score distributions or simply *score distributions* (Figure 7, right), particularly well-suited to the few-run regime. A score distribution shows the fraction of runs above a certain normalized score and is given by

$$\hat{F}_X(\tau) = \hat{F}(\tau; x_{1:M,1:N}) = \frac{1}{M}\sum_{m=1}^{M}\hat{F}_m(\tau) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}[x_{m,n} > \tau]. \tag{2.4.1}$$

One advantage of the score distribution is that it is an unbiased estimator of the underlying distribution $F(\tau) = \frac{1}{N}\sum_{m=1}^{M} F_m(\tau)$. Another advantage is that an outlier run with extremely high score can change the output of score distribution for any τ by at most a value of $\frac{1}{MN}$.

It is useful to contrast score distributions to average-score distributions, originally proposed in the context of the ALE [20] as a generalization of the median score. Average-score distributions correspond to the performance profile of a random variable $\bar{X}$, $\hat{F}_{\bar{X}}(\tau) = \hat{F}(\tau; \bar{x}_{1:M})$, which shows the fraction of tasks on which an algorithm performs better than a certain score. However, such distributions are a biased estimate of the thing they seek to represent. Run-score distributions are more robust than average-score distributions, as they are a step function in $1/MN$ versus $1/M$ intervals, and typically has less variance: $\sigma^2_X = \frac{1}{M^2N}\sum_{m=1}^{M} F_m(\tau)(1 - F_m(\tau))$ versus $\sigma^2_{\bar{X}} = \frac{1}{M^2}\sum_{m=1}^{M} F_{\bar{X}_m}(\tau)(1 - F_{\bar{X}_m}(\tau))$. Figure 7 illustrates these differences.

Another alternative [178] is to replace scores in a performance profile by the probability that average task scores of a given method outperforms the best method (among a given set of methods), computed using the Welsh's t-test [231]. However, this profile is (1) also a biased estimate, (2) less robust to outlier runs, (3) is insensitive to the size of performance differences, *i.e.,* two methods that are uniformly 1% and 100% worse than the best method are assigned the same probability, (4) is only sensible when task score distributions are Gaussian, as required by Welsh's t-test, and finally, (5) the ranking of methods depends on the specific set of methods being compared in such profiles.

2.4.3. Robust and Efficient Aggregate Metrics

Performance profiles allow us to compare different methods at a glance. If one curve is strictly above another, the better method is said to *stochastically dominate*[7] the other [134, 60]. In RL benchmarks with a large number of tasks, however, stochastic dominance is rarely observed: performance profiles often intersect at multiple points. Finer quantitative comparisons must therefore entail aggregate metrics.

We can extract a number of aggregate metrics from score distributions, including median (mixing runs and tasks) and mean normalized scores (matching our usual definition). As we already argued that these metrics are deficient, we now consider interesting alternatives also derived from score distributions.

7. A random variable X has stochastic dominance over random variable Y if $P(X > \tau) \geq P(Y > \tau)$ for all τ, and for some τ, $P(X > \tau) > P(Y > \tau)$.

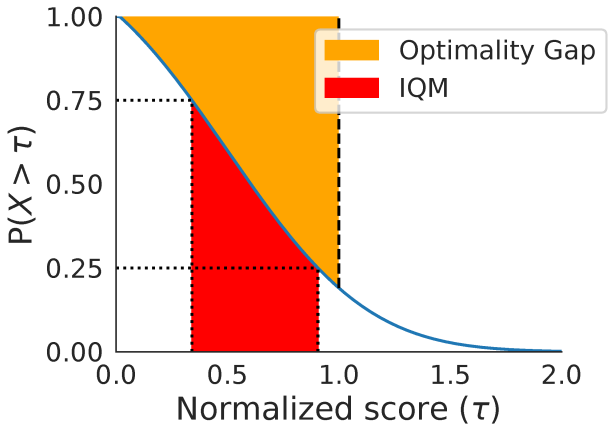

Figure 8. Aggregate metrics. For a non-negative random variable X, IQM corresponds to the red shaded region while optimality gap corresponds to the orange shaded region in the performance profile of X.

As an alternative to median, we recommend using the **interquartile mean** (IQM). Also called 25% trimmed mean, IQM discards the bottom and top 25% of the runs and calculates the mean score of the remaining 50% runs ($=\lfloor NM/2 \rfloor$ for N runs each on M tasks). IQM interpolates between mean and median across runs, which are 0% and almost 50% trimmed means respectively. Compared to sample median, IQM is a better indicator of overall performance as it is calculated using 50% of the combined runs while median only depends on the performance ordering across tasks and not on the magnitude except at most 2 tasks. For example, zero scores on nearly half of the tasks does not affect the median while IQM exhibits a severe degradation. Compared to mean, IQM is robust to outliers, yet has considerably less bias than median. While median is more robust to outliers than IQM, this robustness comes at the expense of statistical efficiency, which is crucial in the few-run regime: IQM results in much smaller CIs (Figure 2 (right) and 6) and is able to detect a given improvement with far fewer runs (Figure 4).

As a robust alternative to mean, we recommend using the **optimality gap**: the amount by which the algorithm fails to meet a minimum score of $\gamma = 1.0$ (orange region in Figure 8). This assumes that a score of 1.0 is a desirable target beyond which improvements are not very important, for example when the aim is to obtain human-level performance [*e.g.,* 52, 14].

If one is interested in knowing how robust an improvement from an algorithm X over an algorithm Y is, another possible metric to consider is the average **probability of improvement** – this metric shows how likely it is for X to outperform Y on a randomly selected task. Specifically, $P(X > Y) = \frac{1}{M}\sum_{m=1}^{M} P(X_m > Y_m)$, where $P(X_m > Y_m)$ is the probability that X is better than Y on task m. Note that, unlike IQM and optimality gap, this metric does not account for the size of improvement. While finding the best aggregate metric is still an open question and is often dependent on underlying normalized score distribution, our proposed alternatives avoid the failure modes of prevalent metrics while being robust and requiring fewer runs to reduce uncertainty.

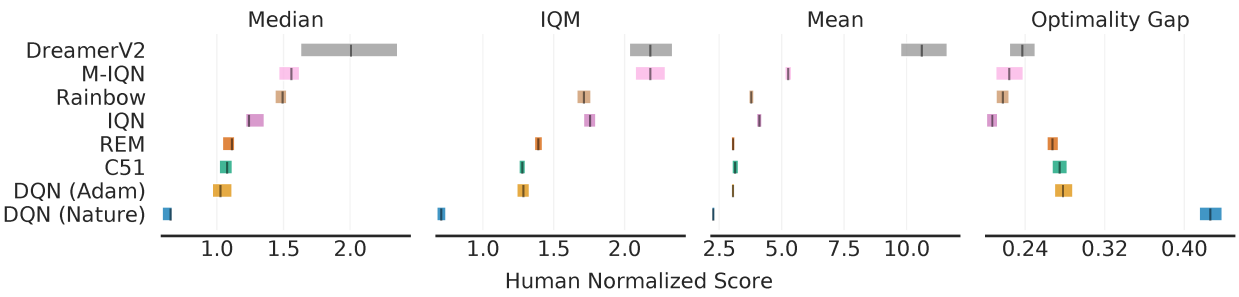

Figure 9. Aggregate metrics on Atari 200M with 95% CIs based on 55 games with sticky actions [148]. Higher mean, median and IQM scores and lower optimality gap are better. The CIs are estimated using the percentile bootstrap with stratified sampling. IQM typically results in smaller CIs than median scores. Large values of mean scores relative to median and IQM indicate being dominated by a few high performing tasks, for example, DreamerV2 and M-IQN obtain normalized scores above 50 on the game JamesBond. Optimality gap is less susceptible to outliers compared to mean scores. We compare DQN (Nature) [157], DQN with Adam optimizer, C51 [22], REM [4], Rainbow [93], IQN [51], Munchausen-IQN (M-IQN) [224], and DreamerV2 [88]. All results are based on 5 runs per game except for M-IQN and DreamerV2 which report results with 3 and 11 runs.

2.5. Re-evaluating Evaluation on Deep RL Benchmarks

Arcade Learning Environment. Training RL agents for 200M frames on the ALE [20, 148] is the most widely recognized benchmark in deep RL. We revisit some popular methods which demonstrated progress on this benchmark and reveal discrepancies in their findings as a consequence of ignoring the uncertainty in their results (Figure 9). For example, DreamerV2 [88] exhibits a large amount of uncertainty in aggregate scores. While M-IQN [224] claimed better performance than Dopamine Rainbow[8] [93] in terms of median normalized scores, their interval estimates strikingly overlap. Similarly, while C51 [20] is considered substantially better than DQN [157], the interval estimates as well as performance profiles for DQN (Adam) and C51 overlap significantly.

Figure 9 reveals an interesting limitation of aggregate metrics: depending on the choice of metric, the ordering between algorithms changes (*e.g.,* Median *vs.* IQM). The inconsistency in ranking across aggregate metrics arises from the fact that such metrics only capture a specific aspect of overall performance across tasks and runs. Additionally, the change of algorithm ranking between optimality gap and IQM/median scores reveal that while recent algorithms typically show performance gains relative to humans on average, their performance seems to be worse on games below human performance. Since performance profiles capture the full picture, they would often illustrate why such inconsistencies exist. For example, optimality gap and IQM can be both read as areas in the profile (Figure 8). The performance profile in Figure 10 (left) illustrates the nuances present when comparing different algorithms. For example, IQN seems to be better than Rainbow for $\tau \geq 2$, but worse for $\tau < 2$. Similarly, the profiles of DreamerV2 and M-IQN for $\tau < 8$ intersect at multiple points. To compare

8. Dopamine Rainbow differs from that of Hessel et al. [93] by not including double DQN, dueling architecture and noisy networks. Also, results in [93] were reported using a single run without sticky actions.

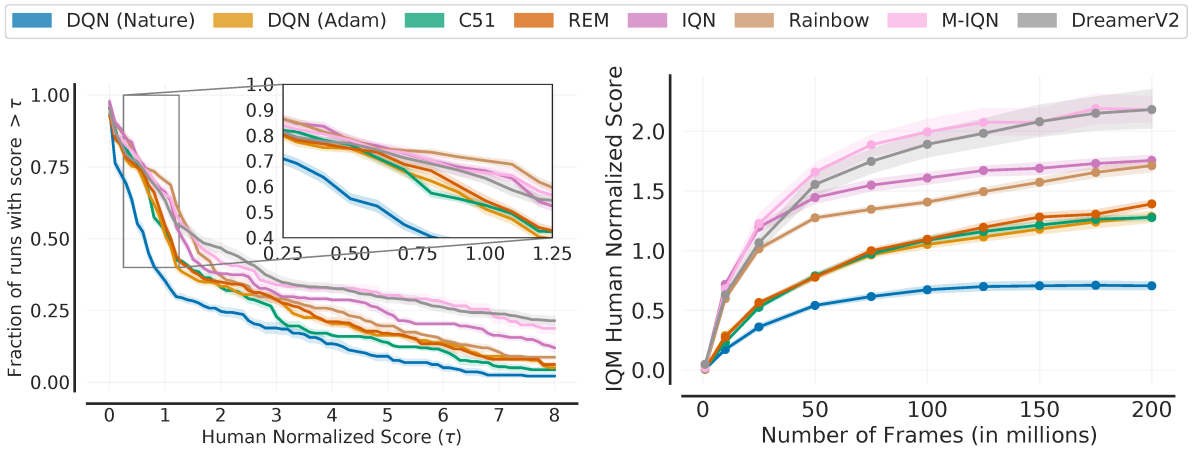

Figure 10. Atari 200M evaluation. **Left**. Score distributions using human-normalized scores obtained after training for 200M frames. **Right**. Sample-efficiency of agents as a function of number of frames measured via IQM human-normalized scores. Shaded regions show pointwise 95% percentile stratified bootstrap CIs.

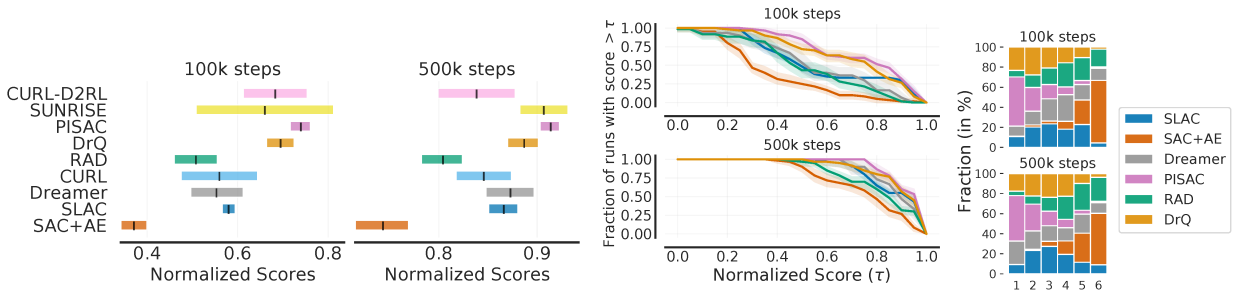

Figure 11. DeepMind Control Suite evaluation results, averaged across 6 tasks, on the 100k and 500k benchmark. We compare SAC+AE [238], SLAC [129], Dreamer [87], CURL [202], RAD [126], DrQ [117], PISAC [131], SUNRISE [130], and CURL-D2RL [198]. The **ordering** of the algorithms in the left figure is based on their claimed relative performance – all algorithms except Dreamer claimed improvement over at least one algorithm placed below them. **(a)** Interval estimates show 95% stratified bootstrap CIs for methods with individual runs provided by their respective authors and 95% studentized CIs for CURL, CURL-D2RL, and SUNRISE. Normalized scores are computed by dividing by the maximum score (=1000). **(b)** Score distributions. **(c)** The i^{th} column in the rank distribution plots show the probability that a given method is assigned rank i, averaged across all tasks. The ranks are estimated using 200,000 stratified bootstrap re-samples.

sample efficiency of the agents, we also present their IQM scores as a function of number of frames in Figure 10 (right).

DeepMind Control Suite. Recent continuous control papers benchmark performance on 6 tasks in DM Control [214] at 100k and 500k steps. Typically, such papers claim improvement based on higher mean scores per task regardless of the variability in those scores. However, we find that when accounting for uncertainty in results, most algorithms do not consistently rank above algorithms they claimed to improve upon (Figure 11). Furthermore, there are huge overlaps in 95% CIs of mean normalized scores for most algorithms (Figure 11(left)). These findings suggest that a lot of the reported improvements are spurious, resulting from randomness in the experimental protocol.

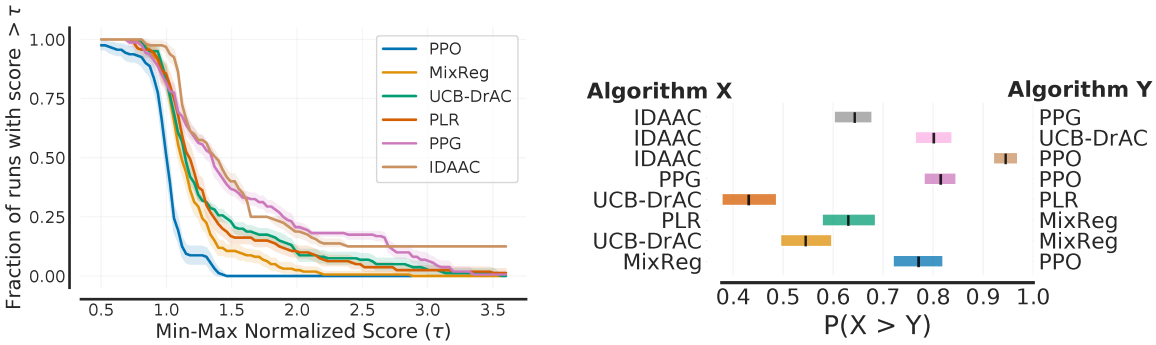

Figure 12. Procgen evaluation results based on easy mode comparisons [175] with 16 tasks. **Left**. Score distributions which compare PPO [192], MixReg [226], UCB-DrAC [176], PLR [108], PPG [46] and IDAAC [175]. Shaded regions indicate 95% percentile stratified bootstrap CIs. **Right**. Each row shows the probability of improvement, with 95% bootstrap CIs, that the algorithm X on the left outperforms algorithm Y on the right, given that X was claimed to be better than Y. For all algorithms, results are based on 10 runs per task.

Procgen benchmark. Procgen [45] is a popular benchmark, consisting of 16 diverse tasks, for evaluating generalization in RL. Recent papers report mean PPO-normalized scores on this benchmark to emphasize the gains relative to PPO [192] as most methods are built on top of it. However, Figure 12 (left) shows that PPO-normalized scores typically have a heavy-tailed distribution making the mean scores highly dependent on performance on a small fraction of tasks. Instead, we recommend using normalization based on the estimated minimum and maximum scores on ProcGen [45] and reporting aggregate metrics based on such scores. While publications sometimes make binary claims about whether they improve over prior methods, such improvements are inherently probabilistic. To reveal this discrepancy, we investigate the following question: "What is the probability that an algorithm which claimed improvement over a prior algorithm performs better than it?" (Figure 12, right). While this probability does not distinguish between two algorithms which uniformly improve on all tasks by 1% and 100%, it does highlight how likely an improvement is. For example, there is only a $40-50\%$ chance that UCB-DrAC [176] improves upon PLR [108]. We note that a number of improvements reported in the existing literature are only $50-70\%$ likely.

2.6. Discussion

We saw, both in our case study on the Atari 100k benchmark and with our analysis of other widely-used RL benchmarks, that statistical issues can have a sizeable influence on reported results, in particular when point estimates are used or evaluation protocols are not kept constant within comparisons. Despite earlier calls for more experimental rigor in deep RL [91, 47, 48, 109, 38, 178], our analysis shows that the field has not yet found sure footing in this regards.

In part, this is because the issue of reproducibility is a complex one; where our work is concerned with our confidence about and interpretation of reported results (what Goodman

et al. [80] calls *results reproducibility*), others [172] have highlighted that there might be missing information about the experiments themselves (*methods reproducibility*). We remark that the problem is not solved by fixing random seeds, as has sometimes been proposed [162, 115], since it does not really address the question of whether an algorithm would perform well under similar conditions but with different seeds. Furthermore, fixed seeds might benefit certain algorithms more than others. Nor can the problem be solved by the use of dichotomous statistical significance tests, as discussed in Section 2.2.

One way to minimize the risks associated with statistical effects is to report results in a more complete fashion, paying close attention to bias and uncertainty within these estimates. To further support RL researchers in this endeavour, we released an easy-to-use Python library, `rliable` along with a Colab notebook for implementing our recommendations, as well as all the individual runs used in our experiments. Again, we emphasize the importance of published papers providing results for all runs to allow for future statistical analyses.

A barrier to adoption of evaluation protocols proposed in this work, and more generally, rigorous evaluation, is whether there are clear incentives for researchers to do so, as more rigor generally entails more nuanced and tempered claims. Arguably, doing good and reproducible science is one such incentive. We hope that our findings about erroneous conclusions in published papers would encourage researchers to avoid fooling themselves, even if that requires tempered claims. That said, a more pragmatic incentive would be if conferences and reviewers required more rigorous evaluation for publication, *e.g.,* NeurIPS 2021 checklist asks whether error bars are reported. Moving towards reliable evaluation is an ongoing process and we believe that this paper would greatly benefit it.

Given the substantial influence of statistical considerations in experiments involving 40-year old Atari 2600 video games and low-DOF robotic simulations, we argue that it is unlikely that an increase in available computation will resolve the problem for the future generation of RL benchmarks. Instead, just as a well-prepared rock-climber can skirt the edge of the steepest precipices, it seems likely that ongoing progress in reinforcement learning will require greater experimental discipline.

Chapter 3

Reusing Prior Computation In RL

***Origin and Impact**: This paper was a response to a open-ended challenge posed by Marc: "Come up with a new experimental paradigm in which one continuously improves on an existing trained (RL) agent." Another motivation for me was to normalize reusing prior computation in RL research. To further facilitate research, I also organized a workshop on reincarnating RL at ICLR 2023, which was well-received. With the recent rise of large-scale pretrained language models, the RL community already recognizes that "tabula rasa" RL is not worthwile in domains involving natural language. With this work, my hope is that reusing prior computation becomes the go-to approach for most domains where we can apply deep RL.*

***Contribution**: I led this project and contributed in the following ways: (1) Formalized reincarnation and scoped the project to focus on policy to value reincarnating RL (PVRL), (2) Identified the existing literature state and came up with a generally applicable PVRL algorithm, and (3) Designed, implemented and owned most of the experiments, and wrote the paper.*

***Abstract**: Learning* tabula rasa*, that is without any prior knowledge, is the prevalent workflow in reinforcement learning (RL) research. However, RL systems, when applied to large-scale settings, rarely operate tabula rasa. Such large-scale systems undergo multiple design or algorithmic changes during their development cycle and use* ad hoc *approaches for incorporating these changes without re-training from scratch, which would have been prohibitively expensive. Additionally, the inefficiency of tabula rasa RL typically excludes researchers without access to industrial-scale resources from tackling computationally-demanding problems. To address these issues, we present* reincarnating *RL as an alternative workflow, where prior computational work (*e.g.*, learned policies) is reused or transferred between design iterations of an RL agent, or from one RL agent to another. As a step towards enabling reincarnating RL*

from any agent to any other agent, we focus on the specific setting of efficiently transferring an existing sub-optimal policy to a standalone value-based RL agent. We find that existing approaches fail in this setting and propose a simple algorithm to address their limitations. Equipped with this algorithm, we demonstrate reincarnating RL's gains over tabula rasa RL on Atari 2600 games, a challenging locomotion task, and the real-world problem of navigating stratospheric balloons. Overall, this work argues for an alternate approach to RL research, which we believe could significantly improve real-world RL adoption and help democratize it further.

3.1. Introduction

Reinforcement learning (RL) is a general-purpose paradigm for making data-driven decisions. Due to this generality, the prevailing trend in RL research is to learn systems that can operate efficiently *tabula rasa*, that is without much previously learned knowledge about the problem. However, tabula rasa RL systems are typically the exception rather than the norm for solving large-scale RL problems [196, 9, 27, 225]. Such large-scale RL systems often need to function for long periods of time and continually experience new data; restarting them from scratch may require weeks if not months of computation, and there may be billions of data points to re-process – this makes the tabula rasa approach impractical. For example, the system that plays Dota 2 at a superhuman level [27] or the system that manipulates a robot hand for solving Rubik's cube [9], underwent several months of RL training with continual changes (*e.g.,* in model architecture, environment, *etc.*) during their development; this necessitated building upon the previously trained system after such changes to circumvent re-training from scratch, which was done using *ad hoc* approaches. Furthermore, tackling challenging problems with deep RL often incurs substantial computational and financial cost: AlphaStar [225], which achieves grandmaster level in StarCraft, was trained using TPUs for more than a month and replicating it would cost several million dollarsAs a result, the majority of the RL community outside certain resource-rich labs is currently excluded from tackling such problems.

To address both the computational and sample inefficiencies of tabula rasa RL, we present *reincarnating* RL as an alternative research workflow. Reincarnating RL focuses on maximally leveraging existing computational work, such as learned network weights and collected data, to accelerate training across design iterations of an RL agent or when moving from one agent to another. In this workflow, agents need not be trained tabula rasa, except for initial forays into new problems. Reincarnating RL can be seen as an attempt to provide a more formal foundation for the prior *ad hoc* strategies to confront the challenges of large-scale RL model development [*e.g.,* 27, 9].

Reincarnating RL also suggests a different benchmarking paradigm that can democratize RL by allowing the broader community to continually improve and update existing trained agents, and even collaboratively tackle problems that are currently infeasible with tabula rasa RL. For example, imagine a researcher who has trained a deep RL agent $\mathcal{A}_1$ for a long time (*e.g.,* weeks), but now this or another researcher wants to experiment with better algorithms or architectures. While the tabula rasa workflow requires re-training another agent from scratch, reincarnating RL provides the more viable option of transferring $\mathcal{A}_1$ to another agent and training this agent further, or simply fine-tuning $\mathcal{A}_1$ (Figure 1). However, beyond some large-scale reincarnation efforts (Section 4.3), the research community has not focused much

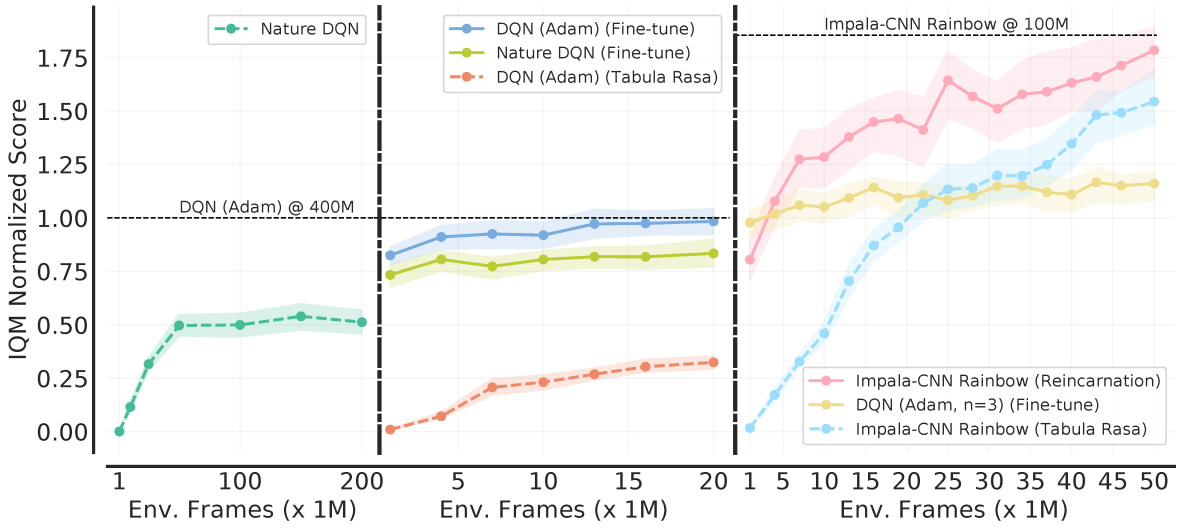

Figure 1. A reincarnating RL workflow on ALE. The plots show Interquartile mean (IQM) [6] normalized scores over training, computed using 50 seeds, aggregated across 10 Atari games. The two vertical separators correspond to loading network weights and replay buffer for fine-tuning while offline pre-training on replay buffer using QDagger (Section 3.2.1) for reincarnation. Shaded regions show 95% confidence intervals. We assign a score of 1 to DQN (Adam) trained for 400M frames and 0 to a random agent. **(Panel 1)** *Tabula rasa* Nature DQN [157] nearly converges in performance after training for 200M environment frames. **(Panel 2)** Reincarnation via fine-tuning Nature DQN with a reduced learning rate leads to 50% higher IQM with only 1M additional frames (leftmost point). Furthermore, fine-tuning Nature DQN while switching from RMSProp [95] to Adam [114] matches the performance of DQN (Adam) trained from scratch for 400M frames, using only 20M frames. **(Panel 3)** A modern ResNet (Impala-CNN [67]) with a better algorithm (Rainbow [93]) outperforms further fine-tuning n-step DQN. Reincarnated Impala-CNN Rainbow outperforms tabula rasa Impala-CNN Rainbow throughout training and requires only 50M frames to nearly match its performance at 100M frames.

on reincarnating RL, in part due to the prevalence of tabula rasa RL. To this end, this work investigates this setting and its potential for accelerating RL research.

As a step towards developing broadly applicable reincarnation approaches, we focus on the specific setting of *policy-to-value* reincarnating RL (PVRL) for efficiently transferring a suboptimal teacher policy to a value-based RL student agent (Section 3.2). Since it is undesirable to maintain dependency on past teachers for successive reincarnations, we require a PVRL algorithm to "wean" off the teacher dependence as training progresses. We find that prior approaches, when evaluated for PVRL on the Arcade Learning Environment (ALE) [20], either result in small improvements over the tabula rasa student or exhibit degradation when weaning off the teacher. To address these limitations, we introduce QDagger, which combines Dagger [184] with n-step Q-learning, and outperforms prior approaches. Equipped with QDagger, we demonstrate the sample and compute-efficiency gains of reincarnating RL over tabula rasa RL, on ALE and a humanoid locomotion task. Reincarnating RL also makes it easier to make progress on a simulated real-world problem of navigating stratospheric balloons [23] in a computationally feasible manner (Section 3.3). We also discuss some considerations in reincarnating RL that require further investigation (Section 3.4). Finally,

to improve the benchmarking ecosystem in reincarnating RL, we will open-source our code and trained agents.

3.2. Case Study: Policy to Value Reincarnating RL

Reincarnating RL can leverage different ways of representing prior knowledge: logged datasets, learned policies, pretrained models (*e.g.,* value functions, dynamics models), representations, and others. While prior large-scale efforts have used a limited form of reincarnating RL (Section 4.3), such as transferring one policy to another, it is unclear how to design reincarnation approaches that can be incorporated in any RL research project. To exemplify the challenges of designing such approaches, we focus on *policy-to-value* reincarnating RL (**PVRL**) for transferring a suboptimal teacher policy to a value-based student agent to accelerate learning. While we can obtain a policy from any RL agent, we chose this setting because value-based RL methods (Q-learning, actor-critic) can leverage off-policy data for better sample efficiency. The difficulty in PVRL arises from the fact that policies do not estimate discounted returns but only distribution over actions, while value functions do. To be broadly useful for reincarnating agents, a PVRL algorithm should satisfy the following desiderata:

— **Teacher-agnostic**. Reincarnating RL has limited utility if the student is constrained by the teacher's architecture or learning algorithm. Thus, we require the student to be teacher-agnostic.

— **Weaning**. It is undesirable to maintain dependency on past teachers when reincarnation may occur several times over the course of a project, or one project to another. Thus, it is necessary that the student's dependence on the teacher policy can be weaned off, as training progresses.

— **Sample-efficient**. Naturally, reincarnating RL is only useful if it is computationally cheaper than training from scratch. Thus, it is desirable that the student can recover and possibly improve upon the teacher's performance using fewer environment samples than training tabula rasa.

PVRL on Atari 2600 games. Given the above desiderata for PVRL, we now empirically investigate whether existing methods that leverage existing data or agents (see Section 4.3) suffice for PVRL. The specific methods that we consider were chosen because they are simple to implement, and also because they have been designed with closely related goals in mind.

Experimental setup. We conduct experiments on ALE with sticky actions [148]. To reduce the computational cost of our experiments, we use a subset of 10 commonly-used Atari 2600 games: Asterix, Breakout, Space Invaders, Seaquest, Q*Bert, Beam Rider, Enduro, Ms Pacman, Bowling and River Raid. We obtain the teacher policy π_T by running DQN [157] with Adam optimizer for 400 million environment frames, requiring 7 days of training per

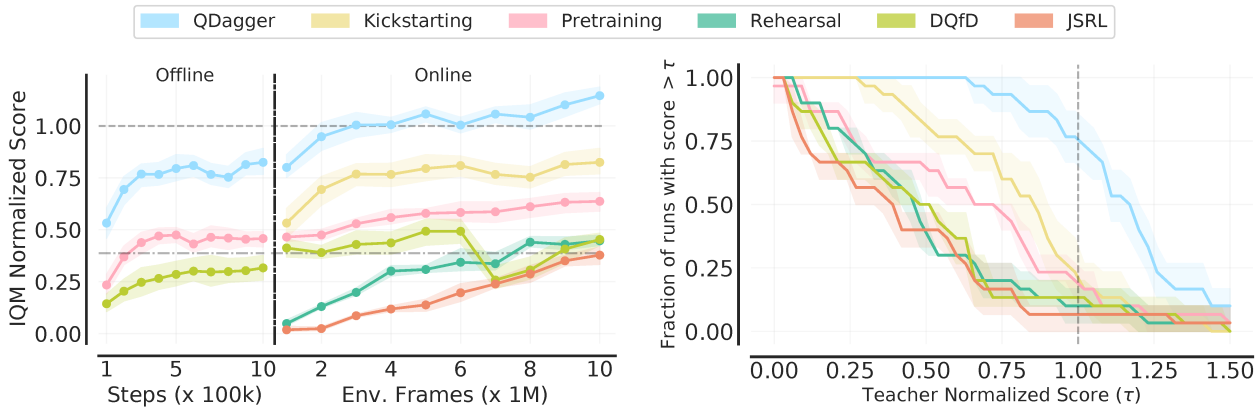

Figure 2. Comparing PVRL algorithms for reincarnating a student DQN agent given a teacher policy (with normalized score of 1), obtained from a DQN agent trained for 400M frames (Section 3.2). Tabula rasa 3-step DQN student ($-\cdot$ line) obtains an IQM teacher normalized score around 0.39. Shaded regions show 95% bootstrap CIs. **Left**. Sample efficiency curves based on IQM normalized scores, aggregated across 10 games and 3 runs, over the course of training. Among all algorithms, only QDagger (Section 3.2.1) surpasses teacher performance within 10 million frames. **Right**. Performance profiles [6] showing the distribution of normalized scores across all 30 runs at the end of training (higher is better). The area under an algorithm's profile corresponds to its mean performance while τ value where the profile intersects $y = 0.5$ shows its median performance. QDagger stochastically dominates other algorithms and outperforms the teacher in 75% of runs.

run with Dopamine [36] on P100 GPUs. Without loss of generality, we assume access to a dataset $\mathcal{D}_T$ that can be generated by the teacher. For this work, $\mathcal{D}_T$ corresponds to the final replay buffer (1 million transitions) of the teacher DQN. For a challenging PVRL setting, we use DQN as the student since tabula rasa DQN requires a substantial amount of training to reach the teacher's performance. To emphasize sample-efficient reincarnation, we train this student for only 10 million frames, a 40 *times* smaller sample budget than the teacher. Furthermore, we wean off the teacher at 6 million frames.

Evaluation. For reliable evaluation, we follow the recommendations of Agarwal et al. [6] and report the Interquartile Mean (IQM) normalized scores with 95% bootstrap confidence intervals (CIs), aggregated across 10 Atari games with 3 seeds each. The normalization is done such that the random policy corresponds to a score of 0 and the teacher policy π_T corresponds to a score of 1. This differs from typically reported human-normalized scores, as we wanted to highlight the performance differences between the student and the teacher. Next, we describe the approaches we investigate.

— **Rehearsal**: Since the student, in principle, can learn using any off-policy data, we can replay teacher data $\mathcal{D}_T$ along with the student's own interactions during training. Following Paine et al. [169], the student minimizes the TD loss on mini-batches that contain ρ% of the samples from $\mathcal{D}_T$ and the rest from the student's replay $\mathcal{D}_S$.

— **JSRL** (Figure 3, left): JSRL [219] uses an interactive teacher policy as a "guide" to improve exploration and rolls in with the guide for a random number of environment

steps. To evaluate JSRL, we vary the maximum number of roll-in steps, α, that can be taken by the teacher and sample a random number of roll-in steps between $[0, \alpha]$ every episode. As the student improves, we decay the steps taken by the teacher every iteration (1M frames) by a factor of β.

— **RL Pretraining**: Given access to teacher data $\mathcal{D}_T$, we can pre-train the student using offline RL. To do so, we use CQL [122], a widely used offline RL algorithm, which jointly minimizes the TD and behavior cloning on logged transitions in $\mathcal{D}_T$ (Equation 3.2.1). Following pretraining, we fine-tune the learned Q-network using TD loss on the student's replay $\mathcal{D}_S$.

$$\mathcal{L}_{Pretrain} = \mathcal{L}_{TD}(\mathcal{D}_T) + \lambda \mathbb{E}_{s,a \sim \mathcal{D}_T} \left[\log \Big(\sum_{a'} Q(s, a') \Big) - Q(s,a) \right] \tag{3.2.1}$$

— **Kickstarting** (Figure 3, right): Akin to kickstarting [190], we jointly optimize the TD loss with an on-policy distillation loss on the student's self-collected data in $\mathcal{D}_S$. The distillation loss uses the cross-entropy between teacher's policy π_T and the student policy $\pi(\cdot|s) = \text{softmax}(Q(s, \cdot)/\tau)$, where τ corresponds to temperature. To wean off the teacher, we decay the distillation coefficient as training progresses. Note that kickstarting does not pretrain on teacher data.

— **DQfD** (Figure 4, left): Following DQfD [93], we initially pretrain the student on teacher data D_T using a combination of TD loss with a large margin classification loss to imitate the teacher actions (Equation 3.2.2). After pretraining, we train the student on its replay data $\mathcal{D}_S$, again using a combination of TD and margin loss. While DQfD minimizes the margin loss throughout training, we decay the margin loss coefficient during the online phase, akin to kickstarting.

$$\mathcal{L}_{DQfD}(\mathcal{D}) = \mathcal{L}_{TD}(\mathcal{D}) + \eta_t \mathbb{E}_{s \sim \mathcal{D}} \left[\max_a \left(Q(s,a) + f(a_T(s), a) \right) - Q(s, a_T(s)) \right] \tag{3.2.2}$$

Results. Rehearsal, with best-performing teacher data ratio ($\rho = 1/16$), is marginally better than tabula rasa DQN but significantly underperforms the teacher (Figure 2, teal), which seems related to the difficulty of standard value-based methods to learn from off-policy teacher data [168]. JSRL does not improve performance compared to tabula rasa DQN and even hurts performance with a large number of teacher roll-in steps (Figure 3, right). The ineffectiveness of JSRL on ALE is likely due to the state-distribution mismatch between the student and the teacher, as the student may never visit the states visited by the teacher and as a result, doesn't learn to correct for its previous mistakes [39].

Pretraining with offline RL on logged teacher data recovers around 50% of the teacher's performance and fine-tuning this pretrained Q-function online marginally improves performance (Figure 2, pink). However, fine-tuning degrades performance with 1-step returns,

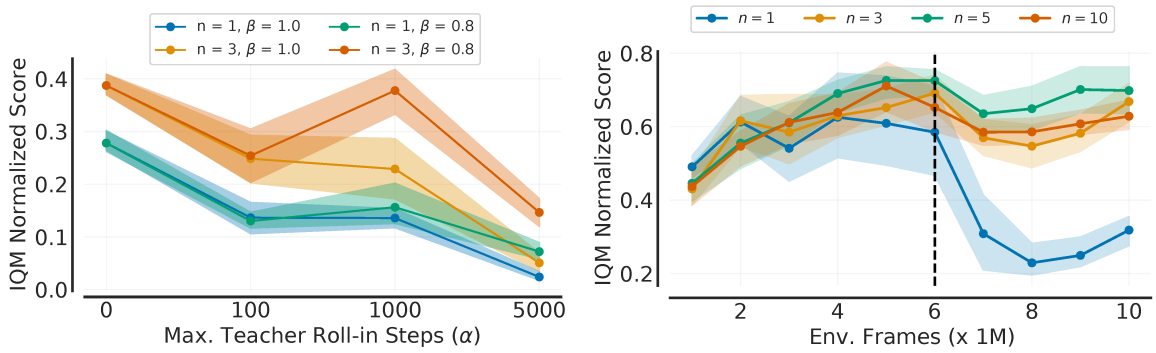

Figure 3. Left. **JSRL**. The plot shows teacher normalized scores with 95% CIs, after training for 10M frames, aggregated using IQM across 10 Atari games with 3 seeds each. Each point corresponds to a different experiment, evaluated using 30 seeds, with specific values of JSRL parameters (α, β) and n-step returns. **Right**. **Kickstarting**, with different n-step returns. The plots show IQM scores over the coures of training. Kickstarting exhibits performance degradation, which is severe with 1-step, and is unable to surpass teacher's performance.

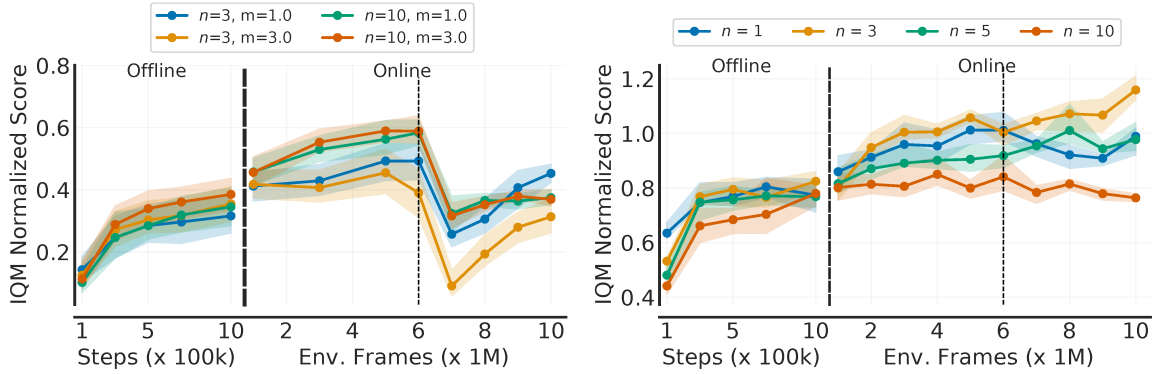

Figure 4. Left. **DQfD**. Here, m is the margin loss parameter, which is the loss penalty when the student's action is different from the teacher. **Right**. **QDagger**, with different n-step returns. In both, the 1^{st} vertical line separates pretraining phase from online phase while the 2^{nd} one indicates completely weaning off the teacher.

which is more pronounced with higher values of CQL loss coefficient. We also find that kickstarting exhibits performance degradation (Figure 3, right), which is severe with 1-step returns, once we wean off the teacher policy. Akin to kickstarting, we again observe a severe performance collapse when weaning off the the teacher dependence in DQfD (Figure 4, left), even when using n-step returns. We hypothesize that this performance degradation is caused by the inconsistency between Q-values trained using a combination of imitation learning and TD losses, as opposed to only minimizing the TD loss. Further investigation of this phenomenon is left for future work. We also find that using intermediate values of n-step returns, such as $n = 3$ (also used by Rainbow [93]), quickly recovers after the performance drop from weaning while larger n-step values impede learning, possibly due to stale target Q-values. See Fedus et al. [70] for a discussion about benefits of n-step returns.

3.2.1. QDagger: A simple PVRL baseline

To address the limitations of prior approaches, we propose QDagger, a simple method for PVRL that combines Dagger [184], an interactive imitation learning algorithm, with n-step Q-learning (Figure 4, right). Specifically, we first pre-train the student on teacher data $\mathcal{D}_T$ by minimizing $\mathcal{L}_{QDagger}(\mathcal{D}_T)$, which combines distillation loss with the TD loss, weighted by a constant λ. This pretraining phase helps the student to mimic the teacher's state distribution, akin to the behavior cloning phase in Dagger. After pretraining, we minimize $\mathcal{L}_{QDagger}(\mathcal{D}_S)$ on the student's replay $\mathcal{D}_S$, akin to kickstarting, where the teacher "corrects" the mistakes on the states visited by the student. As opposed to minimizing the Dagger loss indefinitely, QDagger decays the distillation loss coefficient λ_t ($\lambda_0 = \lambda$) as training progresses, to satisfy the weaning desiderata for PVRL. Weaning allows QDagger to deviate from the suboptimal teacher policy π_T, as opposed to being perpetually constrained to stay close to π_T (Figure 9). We find that both decaying λ_t linearly over training steps or using an affine function of the ratio of student and teacher performance worked well. Assuming the student policy $\pi(\cdot|s) = \text{softmax}(Q(s,\cdot)/\tau)$, the QDagger loss is given by:

$$\mathcal{L}_{QDagger}(\mathcal{D}) = \mathcal{L}_{TD}(\mathcal{D}) + \lambda_t \mathbb{E}_{s\sim\mathcal{D}}\Big[\sum_a \pi_T(a|s)\log\pi(a|s)\Big] \tag{3.2.3}$$

Figure 2 shows that QDagger outperforms prior methods and surpasses the teacher. We remark that DQfD can be viewed as a QDagger ablation that uses a margin loss instead of a distillation loss, while kickstarting as another ablation that does not pretrain on teacher data. Equipped with QDagger, we show how to incorporate PVRL into our workflow and demonstrate its benefits over tabula rasa RL.

3.3. Reincarnating RL as a research workflow

Revisiting ALE. As Mnih et al. [157]'s development of Nature DQN established the tabula rasa workflow on ALE, we demonstrate how iterating on ALE agents' design can be significantly accelerated using a reincarnating RL workflow, starting from Nature DQN, in Figure 1. Although Nature DQN used RMSProp, Adam yields better performance than RMSProp [4, 166]. While we can train another DQN agent from scratch with Adam, fine-tuning Nature DQN with Adam, with a reduced learning rate (see Figure 7), matches the performance of this tabula rasa DQN trained for 400M frames, using a 20 *times* smaller sample budget (Panel 2 in Figure 1). As such, on a P100 GPU, fine-tuning only requires training for a few hours rather than a week needed for tabula rasa training. Given this fine-tuned DQN, fine-tuning it further results in diminishing returns with additional frames due to being constrained to use the 3-layer convolutional neural network (CNN) with the DQN algorithm.

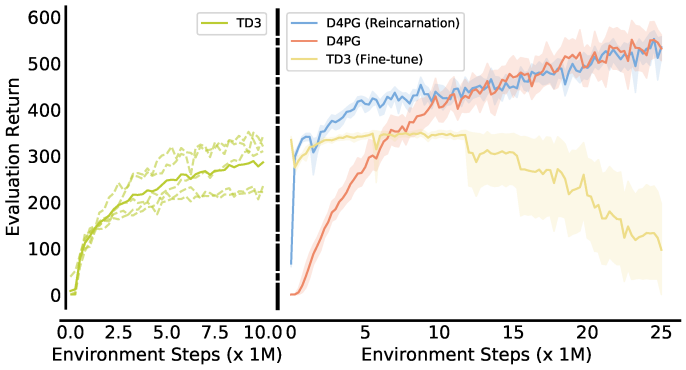

Figure 5. Reincarnating RL on humanoid:run. (Panel 1). We observe that TD3 nearly saturates in performance after training for 10M environment steps. The dashed traces show individual runs while the solid line shows the mean return. **(Panel 2).** Reincarnated D4PG performs better than its tabula rasa counterpart until the first 10M environment steps and then converges to similar performance (with lower variance). Furthermore, training TD3 for a large number of steps eventually results in performance collapse. We use identically parameterized MLP critic and policy networks with 2 hidden layers of size $(256, 256)$ for TD3 but larger networks with 3 hidden layers for D4PG. Shaded regions show 95% CIs based on 10 seeds.

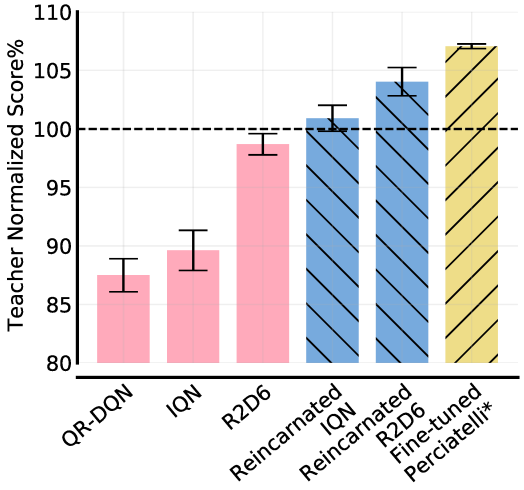

Figure 6. Comparing BLE agents. $*$: See main text. We compare QR-DQN [52] with the same MLP architecture as Perciatelli, IQN [51] with DenseNet [98], and R2D6. Reincarnated R2D6 outperforms Perciatelli as well as the tabula rasa agents, but lags behind fine-tuned Perciatelli. We report the mean score (TWR50) across 10,000 evaluation seeds with varying wind difficulty, averaged over 2 independent runs. Error bars show minimum and maximum scores on those runs.

Let us now consider how one might use a more general reincarnation approach to improve on fine-tuning, by leveraging architectural and algorithmic advances since DQN, without the sample complexity of training from scratch (Panel 3 in Figure 1). Specifically, using QDagger to transfer the fine-tuned DQN, we reincarnate Impala-CNN Rainbow that combines Dopamine Rainbow [93], which incorporates distributional RL [24], prioritized replay [189] and n-step returns, with an Impala-CNN architecture [67], a deep ResNet with 15 convolutional layers. Tabula rasa Impala-CNN Rainbow outperforms fine-tuning DQN further within 25M frames. Reincarnated Impala-CNN Rainbow quickly outperforms its teacher policy within 5M frames and maintains superior performance over its tabula rasa counterpart throughout training for 50M frames. To catch up with the performance of this reincarnated agent's performance, the tabula rasa Impala-CNN Rainbow requires additional training for 50M frames (48 hours on a P100 GPU). Overall, these results indicate how past research on ALE could have been accelerated by incorporating a reincarnating RL approach to designing agents, instead of always re-training agents from scratch.

Tackling a challenging control task. To show how reincarnating RL can enable faster experimentation, we apply PVRL on the *humanoid:run* locomotion task, one of the hardest control problems in DMC [214] due to its large action space (21 degrees of freedom). For this experiment, shown in Figure 5, we use actor-critic agents in Acme [96]. For the teacher policy, we use TD3 [75] trained for 10M environment steps and pick the best run. We find

that fine-tuning this TD3 agent degrades performance after 15M environment steps, which may be related to capacity loss in value-based RL with prolonged training [3, 146]. For reincarnation, we use single-actor D4PG [17], a distributional RL variant of DDPG [135], with a larger policy and critic architecture than TD3. Reincarnated D4PG performs better than its tabula rasa counterpart for the first 10M environment interactions. Both these agents converge to similar performance, which is likely a limitation of QDagger. This result also raises the question of whether better PVRL methods can lead to reincarnated agents that outperform their tabula rasa counterpart throughout learning. Nevertheless, tabula rasa D4PG requires additional training for 10-12 hours on a V100 GPU to match reincarnated D4PG's performance, which might quickly add up to a substantial savings in compute when running a large set of experiments (*e.g.,* architectural or hyperparameter sweeps).

Balloon Learning Environment (BLE) [81]. One of the motivations for our work is to be able to use deep RL in real-world tasks in a data and computationally efficient manner. To this end, the BLE provides a high-fidelity simulator for navigating stratospheric balloons using RL [23]. An agent in the BLE can choose from three actions to control the balloon: move up, down, or stay in place. The balloon can only move laterally by "surfing" the winds at its altitude; the winds change over time and vary as the balloon changes position and altitude. Thus, the agent is interacting with a *partially observable* and non-stationary system, rendering this environment quite challenging. For the teacher, we use the Perciatelli QR-DQN agent provided by BLE, trained using large-scale distributed RL for 40 days on the *production-level* Loon simulator by Bellemare et al. [23] and further fine-tuned in BLE. For our experiments, we train agents using Acme with 64 distributed actors for a budget of 50,000 episodes on a single cloud TPU-v2 (costs $4.5/hour), taking approximately 10-12 hours per run.

In Figure 6, we compare the final performance of agents trained tabula rasa (in pink), with reincarnation (in blue), and fine-tuned (in yellow). We consider three agents, QR-DQN [52] with an MLP architecture (same as Perciatelli), IQN [51] with a Densenet architecture [98], and a recurrent agent R2D6 [1] for addressing the partial observability in BLE. When trained tabula rasa, none of these agents are able to match the teacher performance, with the teacher-lookalike QR-DQN agent performing particularly poorly. As R2D6 and IQN have substantial architectural differences from the teacher, we utilize PVRL for transferring the teacher. Reincarnation allows IQN to match and R2D6 to surpass teacher performance, although both lag behind fine-tuning the teacher.

When fine-tuning, we are reloading the weights from Perciatelli, which was notably trained on a broader geographical region than BLE and whose training distribution can be considered a *superset* of what is used by the other agents; this is likely the reason that fine-tuning

1. R2D6 builds on recurrent replay distributed DQN (R2D2) [112], which uses a LSTM-based policy, and incorporates dueling networks [228], distributional RL [24], DenseNet [98], and double Q-learning [221].

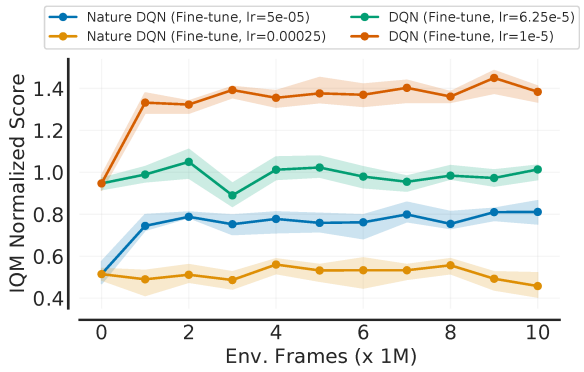

Figure 7. Reincarnation via fine-tuning with same and reduced lr, relative to the original agent.

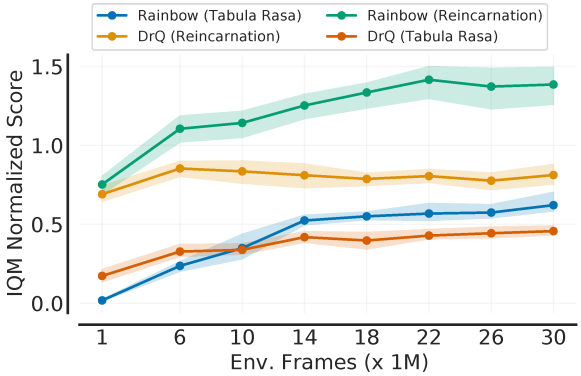

Figure 8. Contrasting benchmarking results under tabula rasa and PVRL settings.

does remarkably well relative to other agents in BLE. Efficiently transferring information in Perciatelli's weights to another agent *without* the replay data from the Loon simulator presents an interesting challenge for future work. Overall, the improved efficiency of reincarnating RL (fine-tuning and PVRL) over tabula rasa RL, as evident on the BLE, could make deep RL more accessible to researchers without access to industrial-scale resources as they can build upon prior computational work, such as model checkpoints, enabling the possible reuse of months of prior computation (*e.g.,* Perciatelli).

3.4. Considerations in Reincarnating RL

Reincarnation via fine-tuning. Given access to model weights and replay of a value-based agent, a simple reincarnation strategy is to fine-tune this agent. While naive fine-tuning with the same learning rate (lr) as the original agent does not exhibit improvement, fine-tuning with a reduced lr, for only 1 million additional frames, results in 25% IQM improvement for DQN (Adam) and 50% IQM improvement for Nature DQN trained with RMSProp (Figure 7). As reincarnating RL leverages existing computational work (*e.g.,* model checkpoints), it allows us to easily experiment with such hyperparameter schedules, which can be expensive in the tabula rasa setting. Note that when fine-tuning, one is *forced* to keep the same network architecture; in contrast, reincarnating RL grants flexibility in architecture and algorithmic choices, which can surpass fine-tuning performance (Figures 1 and 5).

Difference with tabula rasa benchmarking. Are student agents that are more data-efficient when trained from scratch also better for reincarnating RL? In Figure 8, we answer this question in the negative, indicating the possibility of developing better students for utilizing existing knowledge. Specifically, we compare Dopamine Rainbow [93] and DrQ [239], under tabula rasa and PVRL settings. DrQ outperforms Rainbow in low-data regime when trained from scratch but underperforms Rainbow in the PVRL setting as well as when training longer from scratch. Based on this, we speculate that reincarnating RL comparisons might be more consistent with asymptotic tabula rasa comparisons.

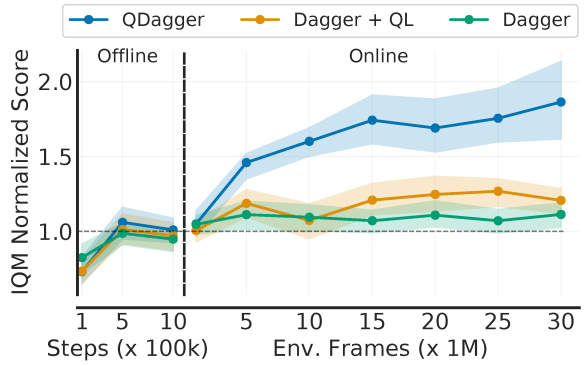

Figure 9. Reincarnation *vs.* Distillation. Reincarnating Impala-CNN Rainbow from a DQN (Adam) policy trained for 400M frames, using QDagger, and comparing it to Dagger (imitation learning) and Dagger + Q-learning (imitation-regularized RL).

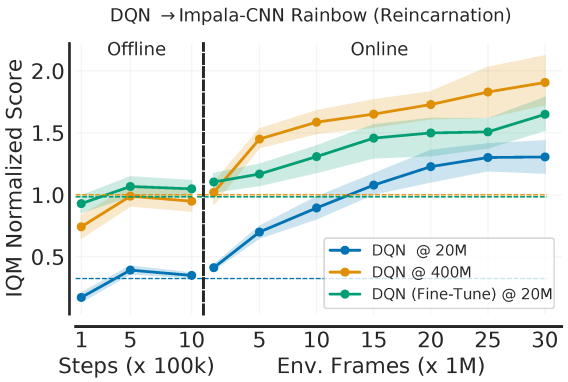

Figure 10. Reincarnation from different teachers, namely, a DQN (Adam) policy trained for 20M and 400M frames and fine-tuned Nature DQN in Figure 1 that achieves similar performance to DQN (Adam) trained for 400M frames.

Reincarnation *vs.* Distillation. PVRL is different from imitation learning or imitation-regularized RL as it focuses on using an existing policy only as a launchpad for further learning, as opposed to imitating or staying close to it. To contrast these settings, we run two ablations of QDagger for reincarnating Impala-CNN Rainbow given a DQN teacher policy: (1) Dagger [184], which only minimizes the on-policy distillation loss in QDagger, and (2) Dagger + QL, which uses a fixed distillation loss coefficient throughout training (as opposed to QDagger, which decays it; see Equation 3.2.3). As shown in Figure 9, Dagger performs similarly to the teacher while Dagger + QL improves over the teacher but quickly saturates in performance. On the contrary, QDagger substantially outperforms these ablations and shows continual improvement with additional environment interactions.

Dependency on prior work. Findings in reincarnating RL are dependent on existing computational work (*e.g.,* teacher policies). This is similar to machine learning areas, such as NLP and computer vision, where building upon pretrained models is the dominant paradigm [*e.g.,* 57, 97, 90, 42]. To investigate teacher dependence in PVRL, we reincarnate a fixed student from three different DQN teachers (Figure 10). As expected, we observe that a higher performing teacher results in a better performing student. However, reincarnation from two distinct policies that perform similarly results in different performance trends. This suggests that a reincarnated student's performance depends not only on the teacher's performance but also on its behavior, which opens up the possibility for creating teacher policies that may be more suited for reincarnation (*e.g.,* pre-trained exploratory policies [35]).

3.5. Conclusion

This work shows that reincarnating RL is a more compute and data efficient workflow than tabula rasa RL. Nevertheless, our results also open several avenues for future work.

Particularly, more research is needed for enabling workflows that can incorporate knowledge provided in a form other than a policy, such as pretrained models or representations, developing better methods for PVRL, and extending PVRL to transfer a policy to model-based agents. We hope that this work motivates RL researchers to release computational work (*e.g.,* model checkpoints) for their publications, which would allow others to directly build on their work. In this regards, we have open-sourced our code and trained agents. Concurrent to this work, Gogianu et al. [79] released 25,000 trained Atari agents, which we believe would further facilitate reincarnating RL. As Newton put it "If I have seen further it is by standing on the shoulders of giants", we argue that reincarnating RL can substantially accelerate progress by building on prior computational work, as opposed to always redoing this work from scratch.

Chapter 4

Implicit Capacity Loss in Data-Efficient RL

Origin & Impact: *Based on our prior experience in offline RL, both and Aviral I knew that offline RL methods typically collapse in performance with prolonged training. As such, we wanted to investigate why this collapse happens and in doing so, we ended up connecting results in deep learning generalization theory (self-distillation, implicit regularization of SGD) to bootstrapping in Q-learning and theoretically derived an phenomenon that also happens empirically. This work led to several follow-ups on studying capacity loss in both online and offline RL [e.g. 165, 146], including some of my own work [*e.g., *123, 201]. Particularly, the feature regularization method proposed by [123] to tackle capacity loss ended up being a crucial ingredient for our work on scaling offline RL to train a generalist RL agent [124].*

Contribution: *I co-led the project, steered the narrative from start to end (e.g, this doc), designed and ran most of the Atari experiments, and jointly proved theoretical results.*

Abstract: *We identify an implicit under-parameterization phenomenon in value-based deep RL methods that use bootstrapping: when value functions, approximated using deep neural networks, are trained with gradient descent using iterated regression onto target values generated by previous instances of the value network, more gradient updates decrease the expressivity of the current value network. We characterize this loss of expressivity via a drop in the rank of the learned value network features, and show that this typically corresponds to a performance drop. We demonstrate this phenomenon on Atari and Gym benchmarks, in both offline and online RL settings. We formally analyze this phenomenon and show that it results from a pathological interaction between bootstrapping and gradient-based optimization.*

4.1. Introduction

Many commonly used deep reinforcement learning (RL) algorithms estimate value functions using bootstrapping, which corresponds to sequentially fitting value functions to target value estimates generated from the value function learned in the previous iteration. Despite high-profile achievements [197], these algorithms are highly unreliable due to poorly understood optimization issues. Although a number of hypotheses have been proposed to explain these issues [2, 26, 73, 101, 142, 121], a complete understanding remains elusive.

We identify an "implicit under-parameterization" phenomenon that emerges when value networks are trained using gradient descent combined with bootstrapping. This phenomenon manifests as an excessive aliasing of features learned by the value network across states, which is exacerbated with more gradient updates. While the supervised deep learning literature suggests that some feature aliasing is desirable for generalization [*e.g.,* 83, 13], implicit under-parameterization results in more pronounced aliasing than in supervised learning. This over-aliasing causes an otherwise expressive value network to *implicitly* behave as an *under-parameterized* network, often resulting in poor performance.

Implicit under-parameterization becomes aggravated when the rate of data re-use is increased, restricting the sample efficiency of deep RL methods. In online RL, increasing the number of gradient steps in between data collection steps for *data-efficient* RL [73, 70] causes the problem to emerge more frequently. In the extreme case when no additional data is collected, referred to as *offline* RL [125, 4, 133], implicit under-parameterization manifests consistently, limiting the viability of offline methods.

We demonstrate the existence of implicit under-parameterization in common value-based deep RL methods, including Q-learning [156, 93] and actor-critic [85], as well as neural fitted-Q iteration [180, 65]. To isolate the issue, we study the effective rank of the features in the penultimate layer of the value network (Section 4.4). We observe that after an initial learning period, the rank of the learned features drops steeply. As the rank decreases, the ability of the features to fit subsequent target values and the optimal value function generally deteriorates and results in a sharp decrease in performance (Section 4.4.1).

To better understand the emergence of implicit under-parameterization, we formally study the dynamics of Q-learning under two distinct models of neural net behavior (Section 4.5): kernel regression [105, 159] and deep linear networks [12]. We corroborate the existence of this phenomenon in both models, and show that implicit under-parameterization stems from a pathological interaction between bootstrapping and the implicit regularization of gradient descent. Since value networks are trained to regress towards targets generated by a previous version of the same model, this leads to a sequence of value networks of potentially decreasing expressivity, which can result in degenerate behavior and a drop in performance.

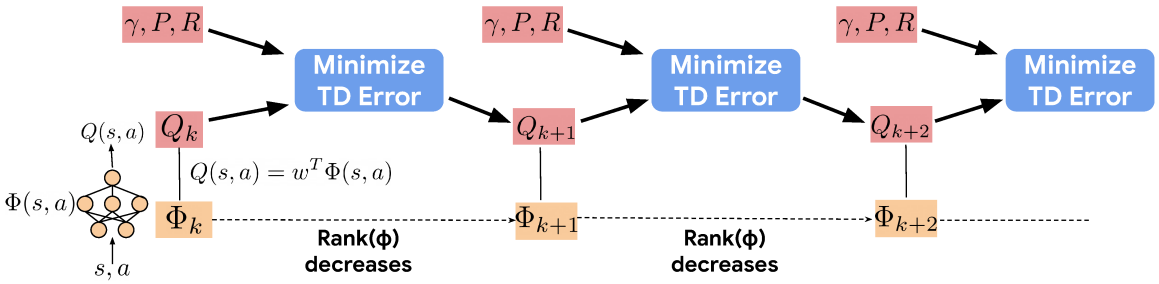

Figure 1. Implicit under-parameterization. Schematic diagram depicting the emergence of an *effective rank* collapse in deep Q-learning. Minimizing TD errors using gradient descent with deep neural network Q-function leads to a collapse in the effective rank of the learned features Φ, which is exacerbated with further training.

The main contribution of this work is the identification of implicit under-parameterization in deep RL methods that use bootstrapping. Empirically, we demonstrate a collapse in the rank of the learned features during training, and show it typically corresponds to a drop in performance in the Atari [19] and continuous control Gym [33] benchmarks in both the offline and data-efficient online RL settings. We verify the emergence of this phenomenon theoretically and characterize settings where implicit under-parameterization can emerge. We then show that mitigating this phenomenon via a simple penalty on the singular values of the learned features improves performance of value-based RL methods in the offline setting on Atari.

4.2. Preliminaries

The goal in RL is to maximize long-term discounted reward in a Markov decision process (MDP), defined as a tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ [174], with state space $\mathcal{S}$, action space $\mathcal{A}$, a reward function $R(\mathbf{s}, \mathbf{a})$, transition dynamics $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ and a discount factor $\gamma \in [0, 1)$. The Q-function $Q^\pi(\mathbf{s}, \mathbf{a})$ for a policy $\pi(\mathbf{a}|\mathbf{s})$, is the expected long-term discounted reward obtained by executing action $\mathbf{a}$ at state $\mathbf{s}$ and following $\pi(\mathbf{a}|\mathbf{s})$ thereafter, $Q^\pi(\mathbf{s}, \mathbf{a}) := E\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}_t, \mathbf{a}_t)\right]$. $Q^\pi(\mathbf{s}, \mathbf{a})$ is the fixed point of the Bellman operator $\mathcal{T}^\pi$, $\forall \mathbf{s}, \mathbf{a}$: $\mathcal{T}^\pi Q(\mathbf{s}, \mathbf{a}) := R(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a}), \mathbf{a}' \sim \pi(\cdot|\mathbf{s}')}\left[Q(\mathbf{s}', \mathbf{a}')\right]$, which can be written in vector form as: $\mathbf{Q}^\pi = \mathbf{R} + \gamma P^\pi \mathbf{Q}^\pi$. The optimal Q-function, $Q^*(\mathbf{s}, \mathbf{a})$, is the fixed point of the Bellman optimality operator $\mathcal{T}$: $\mathcal{T}Q(\mathbf{s}, \mathbf{a}) := R(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim P(\cdot|\mathbf{s},\mathbf{a})}\left[\max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')\right]$.

Practical Q-learning methods [*e.g.*, 156, 93, 85] convert the Bellman equation into a bootstrapping-based objective for training a Q-network, Q_θ, via gradient descent. This objective, known as mean-squared temporal difference (TD) error, is given by: $L(\theta) = \sum_{\mathbf{s},\mathbf{a}} \left(R(\mathbf{s}, \mathbf{a}) + \gamma \bar{Q}_\theta(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a})\right)^2$, where $\bar{Q}_\theta$ is a delayed copy of the Q-function, typically referred to as the *target network*. These methods train Q-networks via gradient descent and slowly update the target network via Polyak averaging on its parameters. We refer the output

Algorithm 1 Fitted Q-Iteration (FQI)

1: Initialize Q-network $\mathbf{Q}_\theta$, buffer μ.
2: **for** fitting iteration k in $\{1, \ldots, \mathrm{N}\}$ **do**
3: Compute $\mathbf{Q}_\theta(\mathbf{s},\mathbf{a})$ and target values $y_k(\mathbf{s},\mathbf{a}) = r + \gamma \max_{\mathbf{a}'} \mathbf{Q}_{k-1}(\mathbf{s}',\mathbf{a}')$ on $\{(\mathbf{s},\mathbf{a})\} \sim \mu$ for training
4: Minimize TD error for $\mathbf{Q}_\theta$ via $t = 1, \cdots, \mathrm{T}$ gradient descent updates, $\min_\theta\ (Q_\theta(\mathbf{s},\mathbf{a}) - \mathbf{y}_k)^2$
5: **end for**

of the penultimate layer of the deep Q-network as the learned *feature matrix* Φ, such that $Q(\mathbf{s},\mathbf{a}) = \mathbf{w}^T\Phi(\mathbf{s},\mathbf{a})$, where $\mathbf{w} \in \mathbb{R}^d$ and $\Phi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times d}$.

For simplicity of analysis, we abstract deep Q-learning methods into a generic fitted Q-iteration (**FQI**) framework [65]. We refer to FQI with neural nets as neural FQI [180]. In the k-th *fitting* iteration, FQI trains the Q-function, $\mathbf{Q}_k$, to match the target values, $\mathbf{y}_k = \mathbf{R} + \gamma P^\pi \mathbf{Q}_{k-1}$ generated using previous Q-function, $\mathbf{Q}_{k-1}$ (Algorithm 1). Practical methods can be instantiated as variants of FQI, with different target update styles, different optimizers, etc.

4.3. Related Work

Prior work has extensively studied the learning dynamics of Q-learning with tabular and linear function approximation, to study error propagation [160, 68] and to prevent divergence [54, 149, 211, 53], as opposed to deep Q-learning analyzed in this work. Q-learning has been shown to have favorable optimization properties with certain classes of features [77], but our work shows that the features learned by a neural net when minimizing TD error do not enjoy such guarantees, and instead suffer from rank collapse. Recent theoretical analyses of deep Q-learning have shown convergence under restrictive assumptions [237, 34, 240, 234], but Theorem 2 shows that implicit under-parameterization appears when the estimates of the value function approach the optimum, potentially preventing convergence. Xu et al. [235**?**] present variants of LSTD [32], which model the Q-function as a kernel-machine but do not take into account the regularization from gradient descent, as done in Equation 4.5.1, which is essential for implicit under-parameterization. Igl et al. [101], Fedus et al. [69] argue that non-stationarity arising from distribution shift hinders generalization and recommend periodic network re-initialization. Under-parameterization is not caused by this distribution shift, and we find that network re-initialization does little to prevent rank collapse (Figure 7). Luo et al. [145] proposes a regularization similar to ours, but in a different setting, finding that more expressive features increases performance of on-policy RL methods. Finally, Yang et al. [236] study the effective rank of the Q^*-values when expressed as a $|\mathcal{S}| \times |\mathcal{A}|$ matrix in online RL and find that low ranks for this Q^*-matrix are preferable. We analyze a fundamentally

different object: the learned features (and illustrate that a rank-collapse of features can hurt), not the Q^*-matrix, whose rank is upper-bounded by the number of actions (*e.g.,* 24 for Atari).

4.4. Implicit Under-Parameterization in Deep Q-Learning

In this section, we empirically demonstrate the existence of implicit under-parameterization in deep RL methods that use bootstrapping. We characterize implicit under-parameterization in terms of the *effective rank* [236] of the features learned by a Q-network. The effective rank of the feature matrix Φ, for a threshold δ (we choose $\delta = 0.01$), denoted as $\text{srank}_\delta(\Phi)$, is given by $\text{srank}_\delta(\Phi) = \min \left\{ k : \frac{\sum_{i=1}^{k} \sigma_i(\Phi)}{\sum_{i=1}^{d} \sigma_i(\Phi)} \geq 1 - \delta \right\}$, where $\{\sigma_i(\Phi)\}$ are the singular values of Φ in decreasing order, i.e., $\sigma_1 \geq \cdots \geq \sigma_d \geq 0$. Intuitively, $\text{srank}_\delta(\Phi)$ represents the number of "effective" unique components of the feature matrix Φ that form the basis for linearly approximating the Q-values. When the network maps different states to orthogonal feature vectors, then $\text{srank}_\delta(\Phi)$ has high values close to d. When the network "aliases" state-action pairs by mapping them to a smaller subspace, Φ has only a few active singular directions, and $\text{srank}_\delta(\Phi)$ takes on a small value.

Definition 1. Implicit under-parameterization *refers to a reduction in the effective rank of the features, $\text{srank}_\delta(\Phi)$, that occurs implicitly as a by-product of learning deep neural network Q-functions.*

While rank decrease also occurs in supervised learning, it is usually beneficial for obtaining generalizable solutions [83, 13]. However, we will show that in deep Q-learning, an interaction between bootstrapping and gradient descent can lead to more aggressive rank reduction (or rank collapse), which can hurt performance.

Experimental setup. To study implicit under-parameterization empirically, we compute $\text{srank}_\delta(\Phi)$ on a minibatch of state-action pairs sampled i.i.d. from the training data (*i.e.,* the dataset in the offline setting, and the replay buffer in the online setting). We investigate offline and online RL settings on benchmarks including Atari games [19] and Gym environments [33]. We also utilize gridworlds described by Fu et al. [73] to compare the learned Q-function against the oracle solution computed using tabular value iteration. We evaluate DQN [156] on gridworld and Atari and SAC [85] on Gym domains.

Data-efficient offline RL. In offline RL, our goal is to learn effective policies by performing Q-learning on a fixed dataset of transitions. We investigate the presence of rank collapse when deep Q-learning is used with broad state coverage offline datasets from Agarwal et al. [4]. In the top row of Figure 2, we show that after an initial learning period, $\text{srank}_\delta(\Phi)$ decreases in all domains (Atari, Gym and the gridworld). The final value of $\text{srank}_\delta(\Phi)$ is often quite small – *e.g.,* in Atari, only 20-100 singular components are active for 512-dimensional features, implying significant underutilization of network capacity. Since under-parameterization is *implicitly* induced by the learning process, even high-capacity value

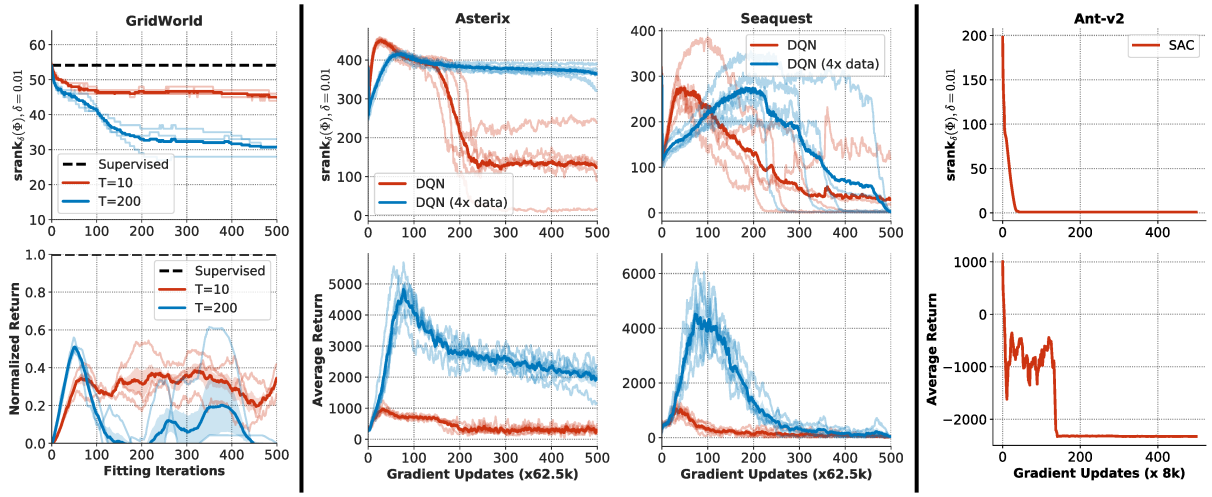

Figure 2. Offline RL. $\text{srank}_\delta(\Phi)$ and performance of neural FQI on gridworld, DQN on Atari and SAC on Gym environments in the offline RL setting. Note that low rank (top row) generally corresponds to worse policy performance (bottom row). Rank collapse is worse with more gradient steps per fitting iteration (T= 10 *vs.* 200 on gridworld). Even when a larger, high coverage dataset is used, marked as DQN (4x data), rank collapse occurs.

networks behave as low-capacity networks as more training is performed with a bootstrapped objective (*e.g.,* mean squared TD error).

On the gridworld environment, regressing to Q^* using supervised regression results in a much higher $\text{srank}_\delta(\Phi)$ (black dashed line in Figure 2(left)) than when using neural FQI. On Atari, even when a 4x larger offline dataset with much broader coverage is used (blue line in Figure 2), rank collapse still persists, indicating that implicit under-parameterization is not due to limited offline dataset size. Figure 2 (2^{nd} row) illustrates that policy performance generally deteriorates as $\text{srank}(\Phi)$ drops, and eventually collapses simultaneously with the rank collapse. While we do not claim that implicit under-parameterization is the only issue in deep Q-learning, the results in Figure 2 show that the emergence of this under-parameterization is *strongly* associated with poor performance.

To prevent confounding effects from the distribution mismatch between the learned policy and the offline dataset, which often affects the performance of Q-learning methods, we also study CQL [122], an offline RL algorithm designed to handle distribution mismatch. We find a similar degradation in effective rank and performance for CQL, implying that under-parameterization does not stem from distribution mismatch and arises even when the resulting policy is within the behavior distribution (though the policy may not be exactly pick actions observed in the dataset).

Data-efficient online RL. Deep Q-learning methods typically use very few gradient updates (n) per environment step (*e.g.,* DQN takes 1 update every 4 steps on Atari, $n = 0.25$). Improving the sample efficiency of these methods requires increasing n to utilize the replay data more effectively. However, we find that using larger values of n results in higher levels

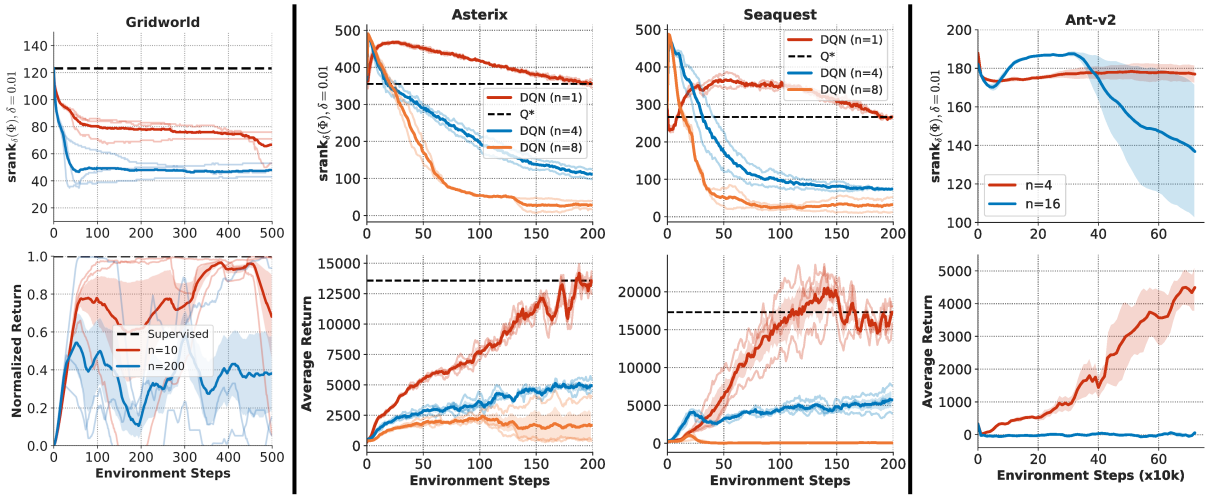

Figure 3. Data Efficient Online RL. $\text{srank}_\delta(\Phi)$ and performance of neural FQI on gridworld, DQN on Atari and SAC on Gym domains in the online RL setting, with varying numbers of gradient steps per environment step (n). Rank collapse happens earlier with more gradient steps, and the corresponding performance is poor.

of rank collapse as well as performance degradation. In the top row of Figure 3, we show that larger values of n lead to a more aggressive drop in $\text{srank}_\delta(\Phi)$ (red *vs.* blue/orange lines), and that rank continues to decrease with more training. Furthermore, the bottom row illustrates that larger values of n result in worse performance, corroborating Fu et al. [73], Fedus et al. [70]. We find similar results with the Rainbow algorithm [93]. As in the offline setting, directly regressing to Q^* via supervised learning does not cause rank collapse (black line in Figure 3).

4.4.1. Understanding Implicit Under-parameterization and its Implications

How does implicit under-parameterization degrade performance? Having established the presence of rank collapse in data-efficient RL, we now discuss how it can adversely affect performance. As the effective rank of the network features Φ decreases, so does the network's ability to fit the subsequent target values, and eventually results in inability to fit Q^*. In the gridworld domain, we measure this loss of expressivity by measuring the error in fitting oracle-computed Q^* values via a linear transformation of Φ. When rank collapse occurs, the error in fitting Q^* steadily increases during training, and the consequent network is not able to predict Q^* at all by the end of training (Figure 5) – this entails a drop in performance. In Atari domains, we do not have access to Q^*, and so we instead measure TD error, that is, the error in fitting the target value estimates, $\mathbf{R} + \gamma P^\pi \mathbf{Q}_k$. In SEAQUEST, as rank decreases, the TD error increases (Figure 8) and the value function is unable to fit the target values, culminating in a performance plateau (Figure 3). This observation is consistent across other environments.

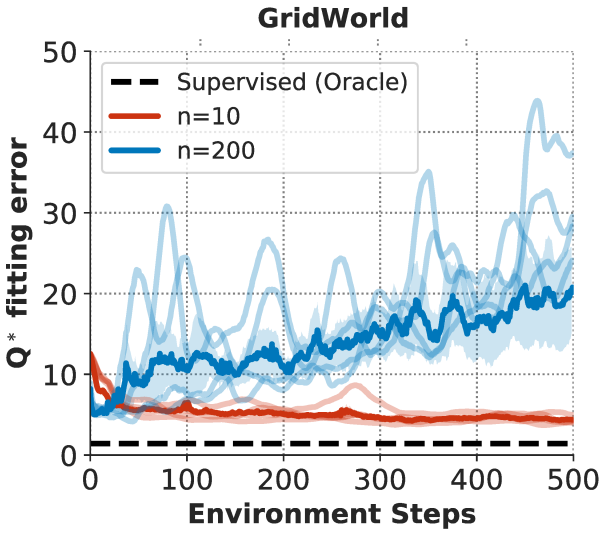

Figure 4. Q* Fitting Error. Fitting error for Q^* prediction for $n = 10$ vs $n = 200$ steps in Figure 3. Observe that rank collapse inhibits fitting Q^* as the fitting error rises over training while rank collapses.

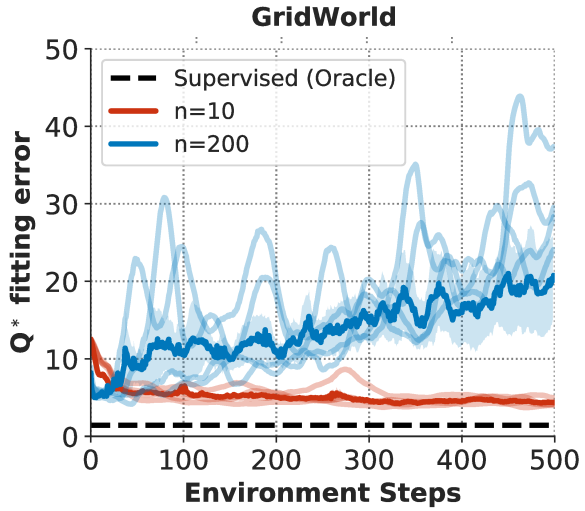

Figure 5. Q* Fitting Error. Fitting error for Q^* prediction for $n = 10$ vs $n = 200$ steps in Figure 3. Observe that rank collapse inhibits fitting Q^* as the fitting error rises over training while rank collapses.

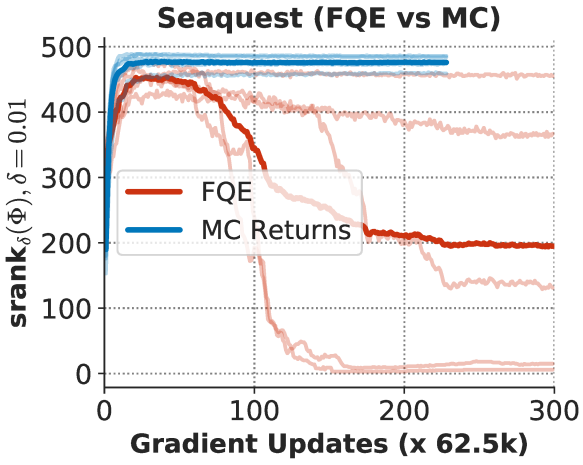

Figure 6. FQE versus MC. Trend of srank for policy evaluation based on bootstrapped updates (FQE) vs Monte-Carlo returns (no bootstrapping). Note that rank-collapse still persists with reinitialization and FQE, but goes away in the absence of bootstrapping.

Does bootstrapping cause implicit under-parameterization? We perform a number of controlled experiments in the gridworld and Atari environments to isolate the connection between rank collapse and bootstrapping. We first remove confounding issues of poor network initialization [69] and non-stationarity [101] by showing that rank collapse occurs even when the Q-network is re-initialized from scratch at the start of each fitting iteration (Figure 7). To show that the problem is not isolated to the control setting, we show evidence of rank collapse in the policy evaluation setting as well. We trained a value network using fitted Q-evaluation for a fixed policy π (*i.e.*, using the Bellman operator $\mathcal{T}^\pi$ instead of $\mathcal{T}$), and

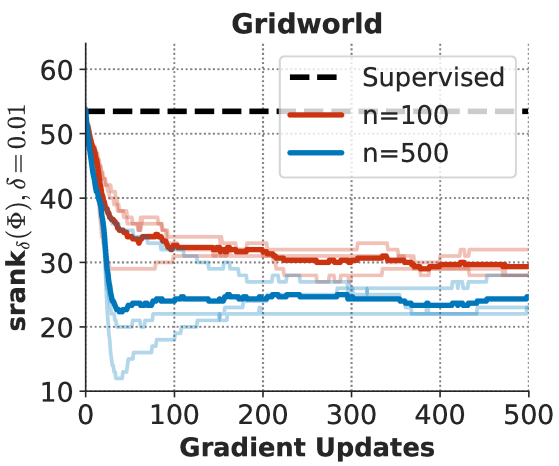

Figure 7. Q-REINITIALIZATION. Q-network re-initialization in each fitting iteration on gridworld.

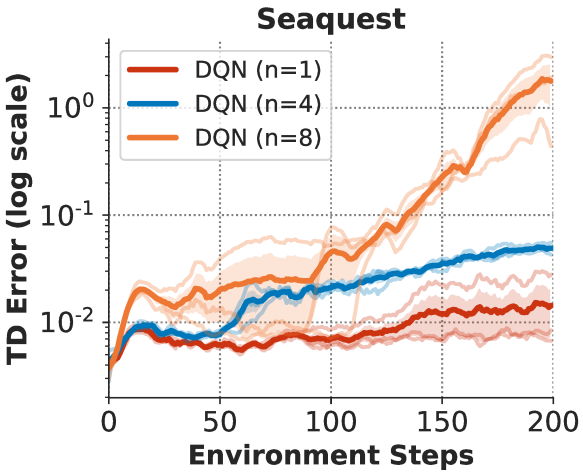

Figure 8. TD ERROR. TD error for varying values of n for SEAQUEST in Figure 3 (middle).

found that rank drop still occurs (FQE in Figure 6). Finally, we show that by removing bootstrapped updates and instead regressing directly to Monte-Carlo (MC) estimates of the value, the effective rank *does not* collapse (MC Returns in Figure 6). These results, along with similar findings on other Atari environments, our analysis indicates that bootstrapping is at the core of implicit under-parameterization.

4.5. Theoretical Analysis of IUP

In this section, we formally analyze implicit under-parameterization and prove that training neural networks with bootstrapping reduces the effective rank of the Q-network, corroborating the empirical observations in the previous section. We focus on policy evaluation (Figure 6), where we aim to learn a Q-function that satisfies $\mathbf{Q} = \mathbf{R} + \gamma P^\pi \mathbf{Q}$ for a fixed π, for ease of analysis. We also presume a fixed dataset of transitions, $\mathcal{D}$, to learn the Q-function.

4.5.1. Analysis via Kernel Regression

We first study bootstrapping with neural networks through a mathematical abstraction that treats the Q-network as a kernel machine, following the neural tangent kernel (NTK) formalism [105]. Building on prior analysis of self-distillation [159], we assume that each iteration of bootstrapping, the Q-function optimizes the squared TD error to target labels $\mathbf{y}_k$ with a kernel regularizer. This regularizer captures the inductive bias from gradient-based optimization of TD error and resembles the regularization imposed by gradient descent under NTK [159]. The error is computed on $(\mathbf{s}_i, \mathbf{a}_i) \in \mathcal{D}$ whereas the regularization imposed by a universal kernel u with a coefficient of $c \geq 0$ is applied to the Q-values at *all* state-action pairs as shown in Equation 4.5.1. We consider a setting $c > 0$ for all rounds of bootstrapping,

which corresponds to the solution obtained by performing gradient descent on TD error for a small number of iterations with early stopping in each round [203] and thus, resembles how the updates in Algorithm 1 are typically implemented in practice.

$$\mathbf{Q}_{k+1} \leftarrow \arg\min_{\mathbf{Q}\in\mathcal{Q}} \sum_{\mathbf{s}_i,\mathbf{a}_i\in\mathcal{D}} \left(Q(\mathbf{s}_i,\mathbf{a}_i) - y_k(\mathbf{s}_i,\mathbf{a}_i)\right)^2 + c \sum_{(\mathbf{s},\mathbf{a})} \sum_{(\mathbf{s}',\mathbf{a}')} u((\mathbf{s},\mathbf{a}),(\mathbf{s}',\mathbf{a}'))Q(\mathbf{s},\mathbf{a})Q(\mathbf{s}',\mathbf{a}'). \tag{4.5.1}$$

The solution to Equation 4.5.1 can be expressed as $Q_{k+1}(\mathbf{s},\mathbf{a}) = \mathbf{g}_{(\mathbf{s},\mathbf{a})}^T(c\mathbf{I}+)^{-1}\mathbf{y}_k$, where is the Gram matrix for a special positive-definite kernel [61] and $\mathbf{g}_{(\mathbf{s},\mathbf{a})}$ denotes the row of corresponding to the input $(\mathbf{s},\mathbf{a})$ [159, Proposition 1]. When combined with the fitted Q-iteration recursion, setting labels $\mathbf{y}_k = \mathbf{R} + \gamma P^\pi \mathbf{Q}_{k-1}$, we recover a recurrence that relates subsequent value function iterates

$$\mathbf{Q}_{k+1} = (c\mathbf{I}+)^{-1}\mathbf{y}_k = \underbrace{(c\mathbf{I}+)^{-1}}_{\mathbf{A}} \left[\mathbf{R} + \gamma P^\pi \mathbf{Q}_k\right] = \mathbf{A}\left(\sum_{i=1}^{k} \gamma^{k-i} \left(P^\pi \mathbf{A}\right)^{k-i}\right)\mathbf{R} := \mathbf{A}\mathbf{M}_k\mathbf{R}. \tag{4.5.2}$$

Equation 4.5.2 follows from unrolling the recurrence and setting the algorithm-agnostic initial Q-value, $\mathbf{Q}_0$, to be $\mathbf{0}$. We now show that the sparsity of singular values of the matrix $\mathbf{M}_k$ generally increases over fitting iterations, implying that the effective rank of $\mathbf{M}_k$ diminishes with more iterations. For this result, we assume that the matrix $\mathbf{S}$ is normal, *i.e.,* the norm of the (complex) eigenvalues of $\mathbf{S}$ is equal to its singular values.

Theorem 1. *Let* $\mathbf{S}$ *be a shorthand for* $\mathbf{S} = \gamma P^\pi \mathbf{A}$ *and assume* $\mathbf{S}$ *is a normal matrix. Then there exists an infinite, strictly increasing sequence of fitting iterations,* $(k_l)_{l=1}^\infty$ *starting from* $k_1 = 0$*, such that, for any two singular-values* $\sigma_i(\mathbf{S})$ *and* $\sigma_j(\mathbf{S})$ *of* $\mathbf{S}$ *with* $\sigma_i(\mathbf{S}) < \sigma_j(\mathbf{S})$*,*

$$\forall\ l \in \mathbb{N} \quad \textit{and} \quad l' \geq l, \quad \frac{\sigma_i(\mathbf{M}_{k_{l'}})}{\sigma_j(\mathbf{M}_{k_{l'}})} < \frac{\sigma_i(\mathbf{M}_{k_l})}{\sigma_j(\mathbf{M}_{k_l})} \leq \frac{\sigma_i(\mathbf{S})}{\sigma_j(\mathbf{S})}. \tag{4.5.3}$$

Hence, $\textit{srank}_\delta(\mathbf{M}_{k_{l'}}) \leq \textit{srank}_\delta(\mathbf{M}_{k_l})$*. Moreover, if* $\mathbf{S}$ *is positive semi-definite, then* $(k_l)_{l=1}^\infty = \mathbb{N}$*, i.e., srank continuously decreases in each fitting iteration.*

A stronger variant that shows a gradual decrease in the effective rank for fitting iterations outside this infinite sequence can also be proved. As k increases along the sequence of iterations given by $k = (k_l)_{l=1}^\infty$, the effective rank of the matrix $\mathbf{M}_k$ drops, leading to low expressivity of this matrix. Since $\mathbf{M}_k$ linearly maps rewards to the Q-function (Equation 4.5.2), drop in expressivity results of $\mathbf{M}_k$ in the inability to model the actual $\mathbf{Q}^\pi$.

Summary of our analysis. Our analysis of bootstrapping and gradient descent from the view of regularized kernel regression suggests that rank drop happens with more training (i.e., with more rounds of bootstrapping). In contrast to self-distillation [159], rank drop may

not happen in every iteration (and rank may increase between two consecutive iterations occasionally), but srank_δ exhibits a generally decreasing trend.

4.5.2. Analysis with Deep Linear Networks under Gradient Descent

While Section 4.5.1 demonstrates rank collapse will occur in a kernel-regression model of Q-learning, it does not illustrate *when* the rank collapse occurs. To better specify a point in training when rank collapse emerges, we present a complementary derivation for the case when the Q-function is represented as a deep linear neural network [13], which is a widely-studied setting for analyzing implicit regularization of gradient descent in supervised learning [83, 84, 12, 13]. Our analysis will show that rank collapse can emerge as the generated target values begin to approach the previous value estimate, in particular, when in the vicinity of the optimal Q-function.

Proof strategy. Our proof consists of two steps: **(1)** We show that the effective rank of the feature matrix decreases within one fitting iteration (for a given target value) due to the low-rank affinity, **(2)** We show that this effective rank drop is "compounded" as we train using a bootstrapped objective. Proposition 1 explains **(1)** and Proposition 2, Theorem 2 discuss **(2)**.

Additional notation and assumptions. We represent the Q-function as a deep linear network with at ≥ 3 layers, such that $Q(\mathbf{s}, \mathbf{a}) = \mathbf{W}_N \mathbf{W}_\phi[\mathbf{s}; \mathbf{a}]$, where $N \geq 3$, $\mathbf{W}_N \in \mathbb{R}^{1 \times d_{N-1}}$ and $\mathbf{W}_\phi = \mathbf{W}_{N-1}\mathbf{W}_{N-2}\cdots\mathbf{W}_1$ with $\mathbf{W}_i \in \mathbb{R}^{d_i \times d_{i-1}}$ for $i = 1, \ldots, N-1$. $\mathbf{W}_\phi$ maps an input $[\mathbf{s}; \mathbf{a}]$ to corresponding penultimate layer' features $\Phi(\mathbf{s}, \mathbf{a})$. Let $\mathbf{W}_j(k, t)$ denotes the weight matrix $\mathbf{W}_j$ at the t-th step of gradient descent during the k-th fitting iteration (Algorithm 1). We define $\mathbf{W}_{k,t} = \mathbf{W}_N(k, t)\mathbf{W}_\phi(k, t)$ and $L_{N,k+1}(\mathbf{W}_{k,t})$ as the TD error objective in the k-th fitting iteration. We study $\text{srank}_\delta(\mathbf{W}_\phi(k, t))$ since the rank of features $\Phi = \mathbf{W}_\phi(k, t)[\mathcal{S}, \mathcal{A}]$ is equal to rank of $\mathbf{W}_\phi(k, t)$ provided the state-action inputs have high rank.

We assume that the evolution of the weights is governed by a continuous-time differential equation [12] *within* each fitting iteration k. To simplify analysis, we also assume that all *except the last-layer* weights follow a "balancedness" property, which suggests that the weight matrices in the consecutive layers in the deep linear network share the same singular values (but with different permutations). However, note that we do not assume balancedness for the last layer which trivially leads to rank-1 features, making our analysis strictly more general than conventionally studied deep linear networks. In this model, we can characterize the evolution of the singular values of the feature matrix $\mathbf{W}_\phi(k, t)$, using techniques analogous to Arora et al. [13]:

Proposition 1. *The singular values of the feature matrix $\mathbf{W}_\phi(k, t)$ evolve according to:*

$$\dot{\sigma}_r(k, t) = -N \cdot \left(\sigma_r^2(k, t)\right)^{1 - \frac{1}{N-1}} \cdot \left\langle \mathbf{W}_N(k, t)^T \frac{dL_{N,k+1}(\mathbf{W}_{K,t})}{d\mathbf{W}}, \mathbf{u}_r(k, t)\mathbf{v}_r(k, t)^T \right\rangle, \quad (4.5.4)$$

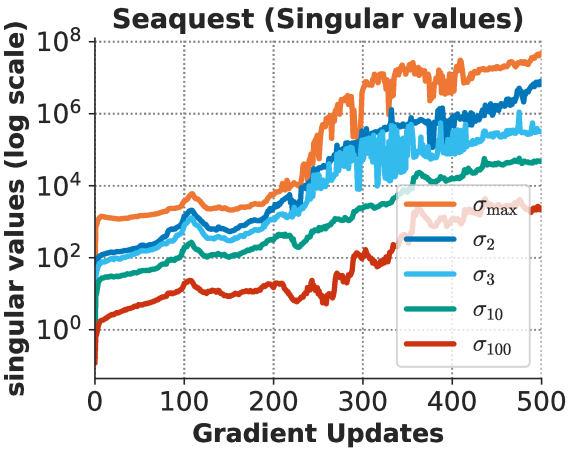

Figure 9. Evolution of singular values of W_ϕ on SEAQUEST.

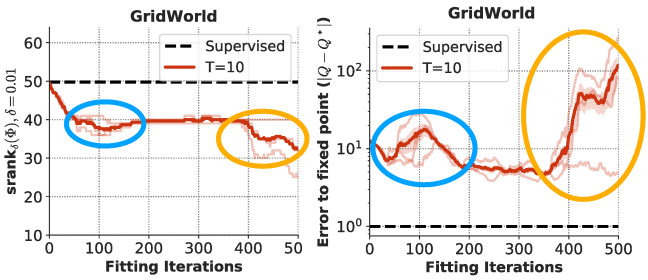

Figure 10. Trend of $\text{srank}_\delta(\Phi)$ *v.s.* error on log scale to the projected TD fixed point. A drop in $\text{srank}_\delta(\Phi)$ (shown as blue and yellow circles) corresponds to a corresponding increase in distance to the fixed point.

for $r = 1, \cdots, \min_{i=1}^{N-1} d_i$, *where* $\mathbf{u}_r(k,t)$ *and* $\mathbf{v}_r(k,t)$ *denote the left and right singular vectors of the feature matrix,* $\mathbf{W}_\phi(k,t)$, *respectively.*

Solving the differential equation (4.5.4) indicates that larger singular values will evolve at a exponentially faster rate than smaller singular values and the difference in their magnitudes disproportionately increase with increasing t. This behavior also occurs empirically, illustrated in the figure on the right, where larger singular values are orders of magnitude larger than smaller singular values. Hence, the effective rank, $\text{srank}_\delta(\mathbf{W}_\phi(k,t))$, will decrease with more gradient steps *within* a fitting iteration k.

Abstract optimization problem for the low-rank solution. Building on Proposition 1, we note that the final solution obtained in a bootstrapping round (*i.e.,* fitting iteration) can be equivalently expressed as the solution that minimizes a weighted sum of the TD error and a data-dependent implicit regularizer $h_\mathcal{D}(\mathbf{W}_\phi, \mathbf{W}_N)$ that encourages disproportionate singular values of $\mathbf{W}_\phi$, and hence, a low effective rank of $\mathbf{W}_\phi$. While the actual form for h is unknown, to facilitate our analysis of bootstrapping, we make a simplification and express this solution as the minimum of Equation 4.5.5.

$$\min_{\mathbf{W}_\phi, \mathbf{W}_N \in \mathcal{M}} ||\mathbf{W}_N \mathbf{W}_\phi [\mathbf{s}; \mathbf{a}] - \mathbf{y}_k(\mathbf{s}, \mathbf{a})||^2 + \lambda_k \text{srank}_\delta(\mathbf{W}_\phi). \tag{4.5.5}$$

Note that the entire optimization path may not correspond to the objective in Equation 4.5.5, but the Equation 4.5.5 represents the final solution of a given fitting iteration. $\mathcal{M}$ denotes the set of constraints that $\mathbf{W}_N$ obtained via gradient optimization of TD error must satisfy, however we do not need to explicitly quantify $\mathcal{M}$ in our analysis. λ_k is a constant that denotes the strength of rank regularization. Since srank_δ is always regularized, our analysis assumes that $\lambda_k > 0$.

Rank drop within a fitting iteration "compounds" due to bootstrapping. In the RL setting, the target values are given by $y_k(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma P^\pi Q_{k-1}(\mathbf{s}, \mathbf{a})$. First

note that when $r(\mathbf{s}, \mathbf{a}) = 0$ and $P^\pi = \mathbf{I}$, i.e., when the bootstrapping update resembles self-regression, we first note that just "copying over weights" from iteration $k-1$ to iteration k is a feasible point for solving Equation 4.5.5, which attains *zero* TD error with no increase in srank_δ. A better solution to Equation 4.5.5 can thus be obtained by incurring non-zero TD error at the benefit of a decreased srank, indicating that in this setting, $\text{srank}_\delta(\mathbf{W}_\phi)$ drops in each fitting iteration, leading to a compounding rank drop effect.

We next extend this analysis to the full bootstrapping setting. Unlike the self-training setting, $y_k(\mathbf{s}, \mathbf{a})$ is not directly expressible as a function of the previous $\mathbf{W}_\phi(k, T)$ due to additional reward and dynamics transformations. Assuming closure of the function class under the Bellman update [161, 40], we reason about the compounding effect of rank drop across iterations in Proposition 2. Specifically, srank_δ can increase in each fitting iteration due to $\mathbf{R}$ and P^π transformations, but will decrease due to low rank preference of gradient descent. This change in rank then compounds as shown below.

Proposition 2. *Assume that the Q-function is initialized to $\mathbf{W}_\phi(0)$ and $\mathbf{W}_N(0)$. Let the Q-function class be closed under the backup, i.e., $\exists\ \mathbf{W}_N^P, \mathbf{W}_\phi^P$, s.t. $(\mathbf{R} + \gamma P^\pi \mathbf{Q}_{k-1})^T = \mathbf{W}_N^P(k)\mathbf{W}_\phi^P(k)[\mathcal{S}; \mathcal{A}]^T$, and assume that the change in srank due to dynamics and reward transformations is bounded: $\text{srank}_\delta(\mathbf{W}_\phi^P(k)) \leq \text{srank}_\delta(\mathbf{W}_\phi(k-1)) + c_k$. Then,*

$$\text{srank}_\delta(\mathbf{W}_\phi(k)) \leq \text{srank}_\delta(\mathbf{W}_\phi(0)) + \sum_{j=1}^{k} c_j - \sum_{j=1}^{k} \frac{||\mathbf{Q}_j - \mathbf{y}_j||}{\lambda_j}.$$

Proposition 2 provides a bound on the value of srank after k rounds of bootstrapping. srank decreases in each iteration due to non-zero TD errors, but potentially increases due to reward and bootstrapping transformations. To instantiate a concrete case where rank clearly collapses, we investigate c_k as the value function gets closer to the Bellman fixed point, which is a favourable initialization for the Q-function in Theorem 2. In this case, the learning dynamics begins to resemble the self-training regime, as the target values approach the previous value iterate $\mathbf{y}_k \approx \mathbf{Q}_{k-1}$, and thus, as we show next, the potential increase in srank (c_k in Proposition 2) converges to 0.

Theorem 2. *Suppose target values $\mathbf{y}_k = \mathbf{R} + \gamma P^\pi \mathbf{Q}_{k-1}$ are close to the previous value estimate $\mathbf{Q}_{k-1}$, i.e. $\forall\ \mathbf{s}, \mathbf{a}$, $y_k(\mathbf{s}, \mathbf{a}) = Q_{k-1}(\mathbf{s}, \mathbf{a}) + \varepsilon(\mathbf{s}, \mathbf{a})$, with $|\varepsilon(\mathbf{s}, \mathbf{a})| \ll |Q_{k-1}(\mathbf{s}, \mathbf{a})|$. Then, there is a constant ϵ_0 depending upon $\mathbf{W}_N$ and $\mathbf{W}_\phi$, such that for all $\|\varepsilon\| < \varepsilon_0$, $c_k = 0$. Thus, srank decreases in iteration k: $\text{srank}_\delta(\mathbf{W}_\phi(k)) \leq \text{srank}_\delta(\mathbf{W}_\phi(k-1)) - ||\mathbf{Q}_k - y_k||/\lambda_k$.*

To empirically show the **consequence of Theorem 2** that a decrease in $\text{srank}_\delta(\mathbf{W}_\phi)$ values can lead to an increase in the distance to the fixed point in a neighborhood around the fixed point, we performed a controlled experiment on a deep linear net shown in Figure 10 that measures the relationship between of $\text{srank}_\delta(\Phi)$ and the error to the projected TD fixed

point $|\mathbf{Q} - \mathbf{Q}^*|$. Note that a drop in $\text{srank}_\delta(\Phi)$ corresponds to a increased value of $|\mathbf{Q} - \mathbf{Q}^*|$ indicating that rank drop when $\mathbf{Q}$ get close to a fixed point can affect convergence to it.

4.6. Conclusion

We identified an implicit under-parameterization phenomenon in deep RL algorithms that use bootstrapping, where gradient-based optimization of a bootstrapped objective can lead to a reduction in the expressive power of the value network. This effect manifests as a collapse of the rank of the features learned by the value network, causing aliasing across states and often leading to poor performance. Our analysis reveals that this phenomenon is caused by the implicit regularization due to gradient descent on bootstrapped objectives.

More broadly, understanding the effects of neural nets and associated factors such as initialization, choice of optimizer, *etc.* on the learning dynamics of deep RL algorithms, using tools from deep learning theory, is likely to be key towards developing robust and data-efficient deep RL algorithms.

Conclusion

"The greatest threat to progress is the belief that the status quo is sufficient." – ChatGPT

This thesis addressed several challenges that hinder the real-world application of reinforcement learning (RL). Our proposed solutions could enable a number of advances, including:

— **Reliable and reproducible RL research**: Our proposed methodology for evaluating RL algorithms could help make RL research more reliable and reproducible. This can allow researchers to more easily compare different algorithms and identify the best ones for a particular task.

— **Efficient and practical RL**: Our proposed research workflow in Chapter 3, which focuses on reusing existing progress, could help make RL research more efficient and practical. This would allow researchers to build on the work of others and avoid having to start from scratch every time they work on a new task. Future research with our proposed workflow could further improve the practical applicability of RL.

— **Generalizable RL algorithms**: Our work on understanding how deep neural networks interact with RL could help develop more generalizable RL algorithms. This means that they are based on sound theoretical foundations and can be applied to a wider range of tasks and environments.

Overall, I believe that these contributions can significantly enhance the capabilities of reinforcement learning (RL) as a powerful and versatile tool for solving real-world problems. I hope that this thesis will inspire others to continue to push the boundaries of RL research and development. I am excited to see how this work will be used by researchers in the future and what breakthroughs RL will enable.

References

[1] Abeyruwan, S. W., Graesser, L., D'Ambrosio, D. B., Singh, A., Shankar, A., Bewley, A., Jain, D., Choromanski, K. M., and Sanketi, P. R. i-sim2real: Reinforcement learning of robotic policies in tight human-robot interaction loops. In Liu, K., Kulic, D., and Ichnowski, J. (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 212–224. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/abeyruwan23a.html.

[2] Achiam, J., Knight, E., and Abbeel, P. Towards characterizing divergence in deep q-learning. *ArXiv*, abs/1903.08894, 2019.

[3] Agarwal*, R., Kumar*, A., Ghosh, D., and Levine, S. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020.

[4] Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

[5] Agarwal, R., Machado, M. C., Castro, P. S., and Bellemare, M. G. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021.

[6] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34: 29304–29320, 2021.

[7] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. *NeurIPS*, 2022.

[8] Aggarwal, S., Courville, A., and Agarwal, R. Behavior predictive representations for generalization in reinforcement learning. In *RLDM*, 2022.

[9] Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[10] Amrhein, V., Greenland, S., and McShane, B. Scientists rise up against statistical significance. *Nature*, 2019.

[11] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[12] Arora, S., Cohen, N., and Hazan, E. On the optimization of deep networks: Implicit acceleration by overparameterization. *arXiv preprint arXiv:1802.06509*, 2018.

[13] Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 7413–7424, 2019.

[14] Badia, A., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., and Blundell, C. Agent57: Outperforming the human atari benchmark. In *Proceedings of the 37th International Conference on Machine Learning, Online, PMLR*, volume 119, pp. 2020, 2020.

[15] Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[16] Baker, M. 1,500 scientists lift the lid on reproducibility. *Nature News*, 2016.

[17] Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Tb, D., Muldal, A., Heess, N., and Lillicrap, T. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[18] Bejjani, W., Papallas, R., Leonetti, M., and Dogar, M. R. Planning with a receding horizon for manipulation in clutter using a learned value function. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pp. 1–9. IEEE, 2018.

[19] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, May 2013. ISSN 1076-9757.

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

[21] Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *ICML*, 2017.

[22] Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458. PMLR, 2017.

[23] Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 2020.

[24] Bellemare, M. G., Dabney, W., and Rowland, M. *Distributional Reinforcement Learning*. MIT Press, 2022. http://www.distributional-rl.org.

[25] Bellman, R. *Dynamic Programming*. Princeton University Press, 1957.

[26] Bengio, E., Pineau, J., and Precup, D. Interference and generalization in temporal difference learning. *arXiv preprint arXiv:2003.06350*, 2020.

[27] Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[28] Bianchi, R. A., Ribeiro, C. H., and Costa, A. H. Heuristically accelerated q–learning: a new approach to speed up reinforcement learning. In *Brazilian Symposium on Artificial Intelligence*, pp. 245–254. Springer, 2004.

[29] Bickel, P. J., Götze, F., and van Zwet, W. R. Resampling fewer than n observations: gains, losses, and remedies for losses. In *Selected works of Willem van Zwet*, pp. 267–297. Springer, 2012.

[30] Bouthillier, X., Laurent, C., and Vincent, P. Unreproducible research is reproducible. In *International Conference on Machine Learning*, pp. 725–734, 2019.

[31] Bouthillier, X., Delaunay, P., Bronzi, M., Trofimov, A., Nichyporuk, B., Szeto, J., Mohammadi Sepahvand, N., Raff, E., Madan, K., Voleti, V., et al. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3, 2021.

[32] Boyan, J. A. Least-squares temporal difference learning. In *ICML*, pp. 49–56. Citeseer, 1999.

[33] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

[34] Cai, Q., Yang, Z., Lee, J. D., and Wang, Z. Neural temporal-difference and q-learning provably converge to global optima. *arXiv preprint arXiv:1905.10027*, 2019.

[35] Campos, V., Sprechmann, P., Hansen, S., Barreto, A., Kapturowski, S., Vitvitskyi, A., Badia, A. P., and Blundell, C. Beyond fine-tuning: Transferring behavior in reinforcement learning. *arXiv preprint arXiv:2102.13515*, 2021.

[36] Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

[37] Ceron, J. S. O. and Castro, P. S. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, pp. 1373–1383. PMLR, 2021.

[38] Chan, S. C., Fishman, S., Korattikara, A., Canny, J., and Guadarrama, S. Measuring the reliability of reinforcement learning algorithms. In *International Conference on Learning Representations*, 2020.

[39] Chang, K.-W., Krishnamurthy, A., Agarwal, A., Daumé III, H., and Langford, J. Learning to search better than your teacher. In *International Conference on Machine Learning*, pp. 2058–2066. PMLR, 2015.

[40] Chen, J. and Jiang, N. Information-theoretic considerations in batch reinforcement learning. *ICML*, 2019.

[41] Chen, T., Goodfellow, I., and Shlens, J. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[42] Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[43] Cheng, C.-A., Kolobov, A., and Swaminathan, A. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[44] Clary, K., Tosch, E., Foley, J., and Jensen, D. Let's play again: Variability of deep reinforcement learning agents in atari environments. *arXiv preprint arXiv:1904.06312*, 2019.

[45] Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

[46] Cobbe, K., Hilton, J., Klimov, O., and Schulman, J. Phasic policy gradient. *arXiv preprint arXiv:2009.04416*, 2020.

[47] Colas, C., Sigaud, O., and Oudeyer, P.-Y. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018.

[48] Colas, C., Sigaud, O., and Oudeyer, P.-Y. A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms. *arXiv preprint arXiv:1904.06979*, 2019.

[49] Czarnecki, W. M., Pascanu, R., Osindero, S., Jayakumar, S., Swirszcz, G., and Jaderberg, M. Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1331–1340. PMLR, 2019.

[50] Da Silva, F. L., Warnell, G., Costa, A. H. R., and Stone, P. Agents teaching agents: a survey on inter-agent transfer learning. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–17, 2020.

[51] Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pp. 1096–1105. PMLR, 2018.

[52] Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[53] Dai, B., Shaw, A., Li, L., Xiao, L., He, N., Liu, Z., Chen, J., and Song, L. Sbeed: Convergent reinforcement learning with nonlinear function approximation. In *International Conference on Machine Learning*, pp. 1133–1142, 2018.

[54] De Farias, D. P. *The linear programming approach to approximate dynamic programming: Theory and application*. PhD thesis, 2002.

[55] Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[56] Dehghani, M., Tay, Y., Gritsenko, A. A., Zhao, Z., Houlsby, N., Diaz, F., Metzler, D., and Vinyals, O. The benchmark lottery. *arXiv preprint arXiv:2107.07002*, 2021.

[57] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[58] Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., and Smith, N. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

[59] Dolan, E. D. and Moré, J. J. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[60] Dror, R., Shlomov, S., and Reichart, R. Deep dominance-how to properly compare deep neural models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[61] Duffy, D. G. *Green's functions with applications*. CRC Press, 2015.

[62] Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[63] Efron, B. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7:1–26, 1979.

[64] Efron, B. Better bootstrap confidence intervals. *Journal of the American statistical Association*, 1987.

[65] Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[66] Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[67] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, 2018.

[68] Farahmand, A.-m., Szepesvári, C., and Munos, R. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[69] Fedus, W., Ghosh, D., Martin, J. D., Bellemare, M. G., Bengio, Y., and Larochelle, H. On catastrophic interference in atari 2600 games. *arXiv preprint arXiv:2002.12499*, 2020.

[70] Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pp. 3061–3071. PMLR, 2020.

[71] Fernández, F. and Veloso, M. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 720–727, 2006.

[72] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. 2018.

[73] Fu, J., Kumar, A., Soh, M., and Levine, S. Diagnosing bottlenecks in deep q-learning algorithms. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019.

[74] Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[75] Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pp. 1587–1596, 2018.

[76] Gao, Y., Xu, H., Lin, J., Yu, F., Levine, S., and Darrell, T. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[77] Ghosh, D. and Bellemare, M. G. Representations for stable off-policy reinforcement learning. *arXiv preprint arXiv:2007.05520*, 2020.

[78] Gigerenzer, G. Statistical rituals: The replication delusion and how we got there. *Advances in Methods and Practices in Psychological Science*, 1(2):198–218, 2018.

[79] Gogianu, F., Berariu, T., Buşoniu, L., and Burceanu, E. Atari agents, 2022. URL https://github.com/floringogianu/atari-agents.

[80] Goodman, S. N., Fanelli, D., and Ioannidis, J. P. What does research reproducibility mean? *Science translational medicine*, 8(341):341ps12–341ps12, 2016.

[81] Greaves, J., Candido, S., Dumoulin, V., Goroshin, R., Ponda, S. S., Bellemare, M. G., and Castro, P. S. Balloon Learning Environment, 12 2021. URL https://github.com/google/balloon-learning-environment.

[82] Greenland, S., Senn, S. J., Rothman, K. J., Carlin, J. B., Poole, C., Goodman, S. N., and Altman, D. G. Statistical tests, p values, confidence intervals, and power: a guide to misinterpretations. *European journal of epidemiology*, 2016.

[83] Gunasekar, S., Woodworth, B. E., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 6151–6159, 2017.

[84] Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 9461–9471, 2018.

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

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

[87] Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

[88] Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[89] Hansen, S., Dabney, W., Barreto, A., Warde-Farley, D., de Wiele, T. V., and Mnih, V. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations*, 2020.

[90] He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[91] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. 2018.

[92] Hessel, M., Modayil, J., Hasselt, H. V., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

[93] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. 2018.

[94] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[95] Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. 2012.

[96] Hoffman, M., Shahriari, B., Aslanides, J., Barth-Maron, G., Behbahani, F., Norman, T., Abdolmaleki, A., Cassirer, A., Yang, F., Baumli, K., et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

[97] Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[98] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[99] Huber, P. Robust estimation of a location parameter. *Ann. Math. Stat.*, 1964.

[100] Humphreys, P. C., Raposo, D., Pohlen, T., Thornton, G., Chhaparia, R., Muldal, A., Abramson, J., Georgiev, P., Goldin, A., Santoro, A., et al. A data-driven approach for learning to control computers. *arXiv preprint arXiv:2202.08137*, 2022.

[101] Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

[102] Ioannidis, J. P. Why most published research findings are false. *PLoS medicine*, 2(8):e124, 2005.

[103] Irpan, A. Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[104] Islam, R., Henderson, P., Gomrokchi, M., and Precup, D. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

[105] Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31*. 2018.

[106] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[107] Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. 2017.

[108] Jiang, M., Grefenstette, E., and Rocktäschel, T. Prioritized level replay. *International Conference on Machine Learning*, 2021.

[109] Jordan, S., Chandak, Y., Cohen, D., Zhang, M., and Thomas, P. Evaluating the performance of reinforcement learning algorithms. In *International Conference on Machine Learning*, pp. 4962–4973. PMLR, 2020.

[110] Julian, R., Swanson, B., Sukhatme, G. S., Levine, S., Finn, C., and Hausman, K. Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning. *arXiv preprint arXiv:2004.10190*, 2020.

[111] Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[112] Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

[113] Kielak, K. Do recent advancements in model-based deep reinforcement learning really improve data efficiency? *arXiv preprint arXiv:2003.10181*, 2020.

[114] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[115] Kolesnikov, S. and Hrinchuk, O. Catalyst. rl: a distributed framework for reproducible rl research. *arXiv preprint arXiv:1903.00027*, 2019.

[116] Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

[117] Kostrikov*, I., Yarats*, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[118] Kozakowski, P., Kaiser, L., Michalewski, H., Mohiuddin, A., and Kańska, K. Q-value weighted regression: Reinforcement learning with limited data. *arXiv preprint arXiv:2102.06782*, 2021.

[119] Kulkarni, T. D., Gupta, A., Ionescu, C., Borgeaud, S., Reynolds, M., Zisserman, A., and Mnih, V. Unsupervised learning of object keypoints for perception and control. *NeurIPS*, 32:10724–10734, 2019.

[120] Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pp. 11761–11771, 2019.

[121] Kumar, A., Gupta, A., and Levine, S. Discor: Corrective feedback in reinforcement learning via distribution correction. *arXiv preprint arXiv:2003.07305*, 2020.

[122] Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[123] Kumar, A., Agarwal, R., Ma, T., Courville, A., Tucker, G., and Levine, S. Dr3: Value-based deep reinforcement learning requires explicit regularization. *arXiv preprint arXiv:2112.04716*, 2021.

[124] Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. Offline q-learning on diverse multi-task data both scales and generalizes. *ICLR*, 2023.

[125] Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

[126] Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[127] Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

[128] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[129] Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Conference on Neural Information Processing Systems*, 2020.

[130] Lee, K., Laskin, M., Srinivas, A., and Abbeel, P. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. *International Conference on Machine Learning*, 2021.

[131] Lee, K.-H., Fischer, I., Liu, A., Guo, Y., Lee, H., Canny, J., and Guadarrama, S. Predictive information accelerates learning in rl. *arXiv preprint arXiv:2007.12401*, 2020.

[132] Lee, S., Seo, Y., Lee, K., Abbeel, P., and Shin, J. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

[133] Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[134] Levy, H. Stochastic dominance and expected utility: Survey and analysis. *Management science*, 38(4): 555–593, 1992.

[135] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[136] Lin, J., Campos, D., Craswell, N., Mitra, B., and Yilmaz, E. Significant improvements over the state of the art? a case study of the ms marco document ranking leaderboard. *arXiv preprint arXiv:2102.12887*, 2021.

[137] Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[138] Lipton, Z. C. and Steinhardt, J. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

[139] Liu, G., Zhang, C., Zhao, L., Qin, T., Zhu, J., Jian, L., Yu, N., and Liu, T.-Y. Return-based contrastive representation learning for reinforcement learning. In *International Conference on Learning Representations*, 2021.

[140] Liu, H. and Abbeel, P. Behavior from the void: Unsupervised active pre-training. *arXiv preprint arXiv:2103.04551*, 2021.

[141] Liu, H. and Abbeel, P. Aps: Active pretraining with successor features. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[142] Liu, V., Kumaraswamy, R., Le, L., and White, M. The utility of sparse representations for control in reinforcement learning. *CoRR*, abs/1811.06626, 2018. URL http://arxiv.org/abs/1811.06626.

[143] Lu, Y., Hausman, K., Chebotar, Y., Yan, M., Jang, E., Herzog, A., Xiao, T., Irpan, A., Khansari, M., Kalashnikov, D., et al. Aw-opt: Learning robotic skills with imitation andreinforcement at scale. In *Conference on Robot Learning*, pp. 1078–1088. PMLR, 2022.

[144] Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. Are gans created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

[145] Luo, X., Meng, Q., He, D., Chen, W., and Wang, Y. I4r: Promoting deep reinforcement learning by the indicator for expressive representations. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 2669–2675. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/370. URL https://doi.org/10.24963/ijcai.2020/370. Main track.

[146] Lyle, C., Rowland, M., and Dabney, W. Understanding and preventing capacity loss in reinforcement learning. *arXiv preprint arXiv:2204.09560*, 2022.

[147] Lynnerup, N. A., Nolling, L., Hasle, R., and Hallam, J. A survey on reproducibility by evaluating deep reinforcement learning algorithms on real-world robots. In *Conference on Robot Learning*, 2020.

[148] Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 2018.

[149] Maei, H. R., Szepesvári, C., Bhatnagar, S., Precup, D., Silver, D., and Sutton, R. S. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, 2009.

[150] Mania, H., Guy, A., and Recht, B. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

[151] Maulana, M. R. and Lee, W. S. Ensemble and auxiliary tasks for data-efficient deep reinforcement learning. *arXiv preprint arXiv:2107.01904*, 2021.

[152] McShane, B. B., Gal, D., Gelman, A., Robert, C., and Tackett, J. L. Abandon statistical significance. *The American Statistician*, 2019.

[153] Melis, G., Dyer, C., and Blunsom, P. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018.

[154] Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A., et al. A graph placement methodology for fast chip design. *Nature*, 594(7862): 207–212, 2021.

[155] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning, 2013.

[156] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015. ISSN 0028-0836.

[157] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[158] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

[159] Mobahi, H., Farajtabar, M., and Bartlett, P. L. Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*, 2020.

[160] Munos, R. Error bounds for approximate policy iteration. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 560–567. AAAI Press, 2003. ISBN 1577351894.

[161] Munos, R. and Szepesvari, C. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857, 2008.

[162] Nagarajan, P., Warnell, G., and Stone, P. Deterministic implementations for reproducibility in deep reinforcement learning. *arXiv preprint arXiv:1809.05676*, 2018.

[163] Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

[164] Nair, A., Gupta, A., Dalal, M., and Levine, S. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[165] Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 16828–16847. PMLR, 2022.

[166] Obando-Ceron, J. S. and Castro, P. S. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning (ICML)*, 2021.

[167] OpenAI. Chatgpt. https://openai.com/blog/chatgpt, November 2022. Accessed on April 8th, 2023.

[168] Ostrovski, G., Castro, P. S., and Dabney, W. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[169] Paine, T. L., Gulcehre, C., Shahriari, B., Denil, M., Hoffman, M., Soyer, H., Tanburn, R., Kapturowski, S., Rabinowitz, N., Williams, D., et al. Making efficient use of demonstrations to solve hard exploration problems. *arXiv preprint arXiv:1909.01387*, 2019.

[170] Parisotto, E., Ba, J. L., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[171] Pashler, H. and Wagenmakers, E.-J. Editors' introduction to the special section on replicability in psychological science: A crisis of confidence? *Perspectives on psychological science*, 7(6):528–530, 2012.

[172] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Larochelle, H. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *arXiv preprint arXiv:2003.12206*, 2020.

[173] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Larochelle, H. Improving reproducibility in machine learning research: a report from the neurips 2019 reproducibility program. *Journal of Machine Learning Research*, 22, 2021.

[174] Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[175] Raileanu, R. and Fergus, R. Decoupling value and policy for generalization in reinforcement learning. *International Conference on Machine Learning*, 2021.

[176] Raileanu, R., Goldstein, M., Yarats, D., Kostrikov, I., and Fergus, R. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

[177] Raposo, D., Ritter, S., Santoro, A., Wayne, G., Weber, T., Botvinick, M., van Hasselt, H., and Song, F. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425*, 2021.

[178] Recht, B. Benchmarking Machine Learning with Performance Profiles, 2018. URL http://www.argmin.net/2018/03/26/performance-profiles/.

[179] Reimers, N. and Gurevych, I. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 338–348, 2017.

[180] Riedmiller, M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.

[181] Ritter, S., Barrett, D. G., Santoro, A., and Botvinick, M. M. Cognitive psychology for deep neural networks: A shape bias case study. In *International conference on machine learning*, 2017.

[182] Robine, J., Uelwer, T., and Harmeling, S. Smaller world models for reinforcement learning. *arXiv preprint arXiv:2010.05767*, 2020.

[183] Romer, D. In praise of confidence intervals. In *AEA Papers and Proceedings*, volume 110, pp. 55–60, 2020.

[184] Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635, 2011.

[185] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[186] Saphal, R., Ravindran, B., Mudigere, D., Avancha, S., and Kaul, B. Seerl: Sample efficient ensemble reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1100–1108, 2021.

[187] Schaal, S. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[188] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *CoRR*, abs/1511.05952, 2016.

[189] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. 2016.

[190] Schmitt, S., Hudson, J. J., Zidek, A., Osindero, S., Doersch, C., Czarnecki, W. M., Leibo, J. Z., Kuttler, H., Zisserman, A., Simonyan, K., et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

[191] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[192] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[193] Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uCQfPZwRaUu.

[194] Seo, Y., Chen, L., Shin, J., Lee, H., Abbeel, P., and Lee, K. State entropy maximization with random encoders for efficient exploration. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[195] Siegel, N. Y., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., and Riedmiller, M. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[196] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[197] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676): 354–359, 2017.

[198] Sinha, S., Bharadhwaj, H., Srinivas, A., and Garg, A. D2rl: Deep dense architectures in reinforcement learning. *arXiv preprint arXiv:2010.09163*, 2020.

[199] Skrynnik, A., Staroverov, A., Aitygulov, E., Aksenov, K., Davydov, V., and Panov, A. I. Forgetful experience replay in hierarchical reinforcement learning from expert demonstrations. *Knowledge-Based Systems*, 218:106844, 2021.

[200] Smart, W. D. and Kaelbling, L. P. Effective reinforcement learning for mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pp. 3404–3410. IEEE, 2002.

[201] Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. The dormant neuron phenomenon in deep reinforcement learning. *ICML*, 2023.

[202] Srinivas, A., Laskin, M., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136v2*, 2020.

[203] Suggala, A., Prasad, A., and Ravikumar, P. K. Connecting optimization and regularization paths. In *Advances in Neural Information Processing Systems*, pp. 10608–10619, 2018.

[204] Sun, W., Bagnell, J. A., and Boots, B. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *arXiv preprint arXiv:1805.11240*, 2018.

[205] Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 1988.

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

[207] Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 1996.

[208] Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, 2nd edition, 2018.

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

[210] Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

[211] Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)*, 2009.

[212] Szita, I. and Lörincz, A. Learning tetris using the noisy cross-entropy method. *Neural computation*, 2006.

[213] Taiga, A. A., Agarwal, R., Farebrother, J., Courville, A., and Bellemare, M. G. Investigating multi-task pretraining and generalization in reinforcement learning. In *ICLR*, 2023.

[214] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[215] Tesauro, G. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219, 1994.

[216] Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. 2012.

[217] Torrey, L. and Taylor, M. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1053–1060, 2013.

[218] Trofin, M., Qian, Y., Brevdo, E., Lin, Z., Choromanski, K., and Li, D. Mlgo: a machine learning guided compiler optimizations framework. *arXiv preprint arXiv:2101.04808*, 2021.

[219] Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J., Bennice, M., Fu, C., Ma, C., Jiao, J., et al. Jump-start reinforcement learning. *arXiv preprint arXiv:2204.02372*, 2022.

[220] van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirthieth AAAI Conference On Artificial Intelligence (AAAI), 2016*, 2016. cite arxiv:1509.06461Comment: AAAI 2016.

[221] Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double Q-learning. 2016.

[222] Van Hasselt, H. P., Hessel, M., and Aslanides, J. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

[223] Varoquaux, G. and Cheplygina, V. How i failed machine learning in medical imaging–shortcomings and recommendations. *arXiv preprint arXiv:2103.10292*, 2021.

[224] Vieillard, N., Pietquin, O., and Geist, M. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[225] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[226] Wang, K., Kang, B., Shao, J., and Feng, J. Improving generalization in reinforcement learning with mixture regularization. *arXiv preprint arXiv:2010.10814*, 2020.

[227] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 1995–2003, 2016.

[228] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

[229] Wasserstein, R. L., Schirm, A. L., and Lazar, N. A. Moving to a world beyond "p < 0.05". *The American Statistician*, 2019.

[230] Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[231] Welch, B. L. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[232] Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[233] Xie, L., Wang, S., Rosa, S., Markham, A., and Trigoni, N. Learning with training wheels: speeding up training with a simple controller for deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6276–6283. IEEE, 2018.

[234] Xu, P. and Gu, Q. A finite-time analysis of q-learning with neural network function approximation. *arXiv preprint arXiv:1912.04511*, 2019.

[235] Xu, X., Xie, T., Hu, D., and Lu, X. Kernel least-squares temporal difference learning. *International Journal of Information Technology*, 11(9):54–63, 2005.

[236] Yang, Y., Zhang, G., Xu, Z., and Katabi, D. Harnessing structures for value-based planning and reinforcement learning. *arXiv preprint arXiv:1909.12255*, 2019.

[237] Yang, Z., Xie, Y., and Wang, Z. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pp. 486–489. PMLR, 2020.

[238] Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

[239] Yarats, D., Kostrikov, I., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.

[240] Zhang, Y., Cai, Q., Yang, Z., Chen, Y., and Wang, Z. Can temporal-difference and q-learning learn representation? a mean-field theory. *arXiv preprint arXiv:2006.04761*, 2020.

[241] Zhu, J., Xia, Y., Wu, L., Deng, J., Zhou, W., Qin, T., and Li, H. Masked contrastive representation learning for reinforcement learning. *arXiv preprint arXiv:2010.07470*, 2020.

[242] Zhuang, D., Zhang, X., Song, S. L., and Hooker, S. Randomness in neural network training: Characterizing the impact of tooling. *arXiv preprint arXiv:2106.11872*, 2021.

[243] Zitovsky, J. P., de Marchi, D., Agarwal, R., and Kosorok, M. R. Revisiting bellman errors for offline model selection. *arXiv preprint arXiv:2302.00141*, 2023.