University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Michał Filip Nauman

Sample-efficient actor-critic algorithms in reinforcement learning

PhD dissertation
in COMPUTER SCIENCE

Supervisor:
dr hab. Marek Cygan
University of Warsaw

Warsaw, September 2025

Supervisor's Statement

I confirm that the presented thesis was prepared under my supervision and that it fulfills the requirements for the doctoral degree in the field of Natural Sciences, in the discipline of Computer Science.

Date Supervisor's Signature

Author's Statement

I declare that the presented thesis was prepared by me and that none of its content was obtained through unlawful means.

The thesis has never before been subject to any procedure for obtaining an academic degree. Furthermore, I declare that the presented version of the thesis is identical to the attached electronic version.

Date Author's Signature

Oświadczenie kierującego pracą

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie stopnia doktora w dziedzinie nauk ścisłych i przyrodniczych w dyscyplinie informatyka.

Data Podpis kierującego pracą

Oświadczenie autora pracy

Oświadczam, że niniejsza rozprawa doktorska została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem stopnia doktora w innej jednostce. Niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Abstract

This dissertation investigates and improves the sample efficiency of deep reinforcement learning algorithms. Our key contributions are:

1. We developed the Many-Actions optimality condition, which can determine the optimal sampling strategy for on-policy actor-critic methods. Following these findings, we proposed the model-based approach for sample-efficient estimation of low-variance policy gradient.

2. We conducted an empirical analysis of issues related to plasticity loss, overestimation, and overfitting in off-policy actor-critic methods. Our findings show that combining specific algorithmic improvements results in significant synergistic effects. Building on these findings, we proposed a simple model-free algorithm that achieved significant improvements on the complex simulated dog tasks.

3. We presented an analysis linking optimistic and pessimistic actor-critic updates with the expected utility hypothesis. We proposed a novel approach that integrates optimistic exploration with pessimistic value learning for improved sample efficiency in actor-critic.

4. We demonstrated that the approximation error of value models trained via temporal difference can be calculated using a dynamic programming approach similar to the Bellman equation. Building on these insights, we developed an online pessimism adjustment method to minimize the approximation error in critic networks.

5. We investigated the impact of scaling the parameter count in actor-critic models. We showed that applying specific regularization techniques can enhance the performance of agents as Q-value models are scaled. Based on these insights, we proposed a state-of-the-art actor-critic algorithm.

6. We demonstrated that Q-learning can be predictably scaled, with larger compute investments leading to better performance. We showed that the relationship between Q-learning's computational budget and its sample efficiency can be modeled via a power-law relationship. We presented how our framework can be used to find well-performing settings of batch size and learning rate efficiently.

Keywords

reinforcement learning, model-based reinforcement learning, model-free reinforcement learning, foundation models, score-based models

ACM subject classification

Computing methodologies → Machine learning → Learning paradigms → Reinforcement learning → Sequential decision making

Title

Sample-efficient actor-critic algorithms in reinforcement learning

Streszczenie

W rozprawie badamy możliwości zwiększenia wydajności próbkowej algorytmów głębokiego uczenia ze wzmocnieniem i przedstawiamy następujące wyniki:

1. Zaproponowaliśmy warunek optymalności próbkowania wielu akcji w algorytmach aktora-krytyka typu *online*. Budując na tej obserwacji, zaproponowaliśmy algorytm uczenia ze wzmocnieniem w oparciu o model który pozwala na modelowanie próbkowania wielu akcji w kontekście uczenia ze wzmocnieniem.

2. Przeprowadziliśmy empiryczną analizę problemów związanych z utratą plastyczności, przeszacowaniem nagród oraz przeuczeniem w metodach aktora-krytyka. Uzyskane wyniki wskazują, że integracja określonych usprawnień algorytmicznych prowadzi do istotnych efektów synergii. W oparciu o te obserwacje zaproponowaliśmy prosty algorytm, który wykazał znaczące ulepszenia w wielowymiarowych symulacjach biegu psa.

3. Przedstawiliśmy analizę łączącą optymistyczne i pesymistyczne metody aktora-krytyka z hipotezą użyteczności oczekiwanej. Zaproponowaliśmy nowy algorytm, który integruje optymistyczną eksplorację z pesymistycznym uczeniem krytyka w celu poprawy wydajności próbkowej algorytmu.

4. Wykazaliśmy, że błąd aproksymacji modelu krytyka trenowanego za pomocą równania Bellmana może być obliczany z wykorzystaniem podejścia programowania dynamicznego, analogicznego do równania Bellmana. W oparciu o te spostrzeżenia opracowaliśmy metodę dynamicznej korekty pesymizmu w trybie online, mającą na celu minimalizację błędu aproksymacji.

5. Zbadaliśmy wpływ skalowania liczby trenowalnych parametrów modelu krytyka na wydajność próbkową. Wykazaliśmy, że zastosowanie określonych technik regularyzacyjnych może poprawić wydajność w miarę zwiększania skali modeli. Zaproponowaliśmy nowy algorytm, który w czasie publikacji osiągał najlepszą wydajność w swojej klasie algorytmicznej.

6. Wykazaliśmy, że pewne algorytmy aktora-krytyka mogą być przewidywalnie skalowane, gdzie większe nakłady obliczeniowe prowadzą do lepszej wydajności. Pokazaliśmy, że zależność pomiędzy budżetem obliczeniowym a wydajnością próbkową może być modelowana za pomocą prawa potęgowego. Przedstawiliśmy sposób do efektywnego wyznaczania dobrze działających ustawień hiperparametrów.

Słowa kluczowe

uczenie ze wzmocnieniem, uczenie ze wzmocnieniem w oparciu o model, uczenie ze wzmocnieniem bez modelu, modele podstawowe, modele oparte na gradiencie gęstości

Klasyfikacja ACM

Computing methodologies → Machine learning → Learning paradigms → Reinforcement learning → Sequential decision making

Tytuł

Wydajność próbkowa algorytmów typu aktor-krytyk w uczeniu ze wzmocnieniem

Contents

List of Works

The body of this doctoral thesis comprises seven publications listed below. The main authors are indicated with an asterisk (*). We also detail the contribution of the candidate to each of these works in Section 4.1.

[P1] **M. Nauman** and M. Cygan. *On many-actions policy gradient.*
Proceedings of the 40th International Conference on Machine Learning (2023).

[P2] **M. Nauman***, M. Bortkiewicz*, P. Miłoś, T. Trzciński, M. Ostaszewski, and M. Cygan. *Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning.*
Proceedings of the 41st International Conference on Machine Learning (2024).

[P3] **M. Nauman** and M. Cygan. *Decoupled policy actor-critic: bridging pessimism and risk awareness in reinforcement learning.*
Proceedings of the 39th AAAI Conference on Artificial Intelligence (2025).

[P4] **M. Nauman**, M. Ostaszewski and M. Cygan. *A case for validation buffer in pessimistic actor-critic.*
Proceedings of the 33rd International Joint Conference on Artificial Intelligence (2025).

[P5] **M. Nauman**, M. Ostaszewski, K. Jankowski, P. Miłoś, and M. Cygan. *Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control.*
Advances in Neural Information Processing Systems (2024). **Spotlight** (top 3%).

[P6] O. Rybkin, **M. Nauman**, P. Fu, C. Snell, P. Abbeel, S. Levine, A. Kumar. *Value-based deep RL scales predictably.*
Proceedings of the 42nd International Conference on Machine Learning (2025).

Chapter 1

Introduction

During the last decade, progress in deep learning has unlocked significant capabilities in several fields, including natural language processing [56, 42], and computer vision [65, 24]. These advances have led to breakthroughs such as the development of highly accurate image recognition systems [45], language translation [80], and sophisticated generative models that can create realistic images [64, 65], text [14, 42], and even music [79, 47]. The success of deep learning in these domains is largely attributed to its ability to learn complex representations from vast amounts of data [66, 45, 8, 24], allowing deep learning-driven models to perform tasks that were previously considered too challenging for machine learning.

However, while deep learning has proven to be remarkably effective in supervised learning tasks [14, 24], the field of reinforcement learning (RL) has seen more gradual progress. RL involves learning to make decisions through trial and error, guided by feedback from the environment in the form of rewards [77]. Despite its potential, early RL methods struggled with issues such as instability during training, sample inefficiency, and difficulty scaling to high-dimensional state and action spaces [82, 78]. Integrating deep learning with RL has begun to address some of these challenges. By leveraging the representational power of deep neural networks, RL has made it possible to tackle more complex problems, such as playing video games at superhuman levels [81, 67, 71], controlling robotic systems [75, 84, 18], or optimizing large-scale operations in fields like finance and logistics [86, 31]. Nevertheless, many challenges remain, particularly in improving sample efficiency, enhancing stability, and ensuring the robustness of these models in real-world applications.

1.1. Reinforcement Learning

Reinforcement Learning (RL) is a learning approach focused on determining the best actions to take in various situations such that a numerical reward is maximized [38, 77]. Unlike supervised learning, where correct actions are provided in advance, RL requires the learner to explore and discover which actions lead to the highest rewards through direct interaction with the environment [83]. This involves trial and error, where the learner must navigate not only immediate outcomes but also how current actions will affect future situations and rewards. Unlike other learning paradigms that typically assume the data is stationary [11], RL explicitly assumes non-stationarity where the conditions and returns can evolve. As such, RL framework is not limited to solving isolated problems, such as classifying data [87] or making predictions based on static inputs [76]. Instead, it integrates real-time decision-making, planning, and continuous learning from ongoing interaction with the environment, providing a robust approach for managing the complexities of non-stationary data and adaptive

systems [50, 28].

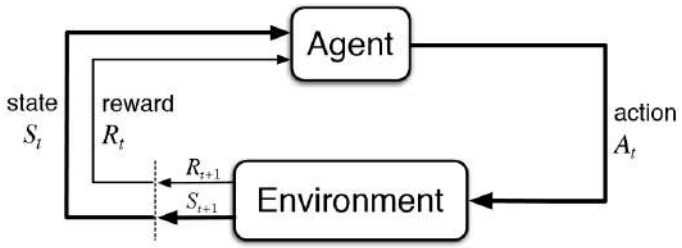


Figure 1.1: The RL framework. Image taken from the classic reinforcement learning book [77].

In the RL framework illustrated in Figure 1.1, an agent interacts with an environment in order to progressively improve its policy [10]. The environment is modeled as a Markov Decision Process (MDP) [10, 51, 63], formally defined by the tuple (S, A, p, r, γ). Here, $s \in S$ denotes a state from the set of possible environment states, and $a \in A$ denotes an action from the agent's action space. Upon taking an action, the agent transitions to a new state according to the transition function p, where $p(s'|s, a)$ specifies the probability of moving to state s' given state s and action a. The initial state s_0 is drawn from the starting state distribution p_0. The agent receives a reward $r(s, a, s')$ for transitioning from s to s' after taking action a. The discount factor $\gamma \in (0, 1]$ balances immediate and future rewards. The sequence of interactions produces trajectories $T = (s_0, a_0, r_0, \ldots, s_B, a_B, r_B)$, where B is the trajectory length. The agent's objective is to learn a policy $\pi : S \to P(A|S)$ that maximizes the expected discounted return from the initial states of the MDP. The sum of expected discounted rewards achieved by following the policy π in state s is referred to as value function $V^\pi(s)$:

$$V^\pi(s) = \mathop{\mathbb{E}}_{s' \sim p} \mathop{\mathbb{E}}_{a \sim \pi} \big(r(s, a, s') + \gamma V^\pi(s')\big). \tag{1.1}$$

Similarly, Q-value is defined as the expected discounted sum of rewards stemming from performing action a in s and following the policy afterward:

$$Q^\pi(s, a) = \mathop{\mathbb{E}}_{s' \sim p} \big(r(s, a, s') + \gamma V^\pi(s')\big). \tag{1.2}$$

Finally, the optimal policy denoted as π^* is defined by:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathop{\mathbb{E}}_{s_0 \sim T_0} V^\pi(s_0) \tag{1.3}$$

Where Π is the considered set of policies.

1.2. Actor-Critic Algorithms

Actor-Critic (AC) algorithms are a popular class of methods in RL that are characterized by using separate structures to represent the policy (called *actor*) and the value or Q-value function (called *critic*) [43, 78]. The actor-critic mechanism is incorporated in many modern algorithms with both actor and critic being modeled by neural networks [74, 27, 29, 36, 17, 35, 32, 30, 33], including breakthrough methods like champion-level drone racing [40] or tokamak plasma control [23]. We denote the parameters of actor and critic by ϕ and θ respectively. AC methods can be either on-policy (with the algorithm updated using data generated by

the current policy) [68, 70] or off-policy (with the algorithm updated using data generated by an arbitrary policy) [73, 27]. We characterize these approaches in the subsections below.

1.2.1. On-Policy Actor-Critic

On-policy Actor–Critic (AC) methods build on the Policy Gradient theorem [78]. This theorem enables the computation of the gradient of the policy objective (Equation 1.3) even when the transition and reward functions are non-differentiable. The policy parameters are updated according to:

$$\phi^{t+1} = \phi^t + \kappa \mathop{\mathbb{E}}_{s\sim p^\pi_\gamma} \mathop{\mathbb{E}}_{a\sim\pi} A^\pi(s,a) \nabla_\phi \log \pi_\phi(a|s). \tag{1.4}$$

The policy gradient is usually approximated using a finite trajectory of interactions. The outer expectation is then approximated using B interactions, and the inner expectation is approximated using a single action sample per state [83, 78, 70, 22]. Above, p^π_γ denotes the discounted stationary distribution under the policy π [62, 78], κ denotes the learning rate, and $A^\pi(s,a)$ denotes the advantage function used for baseline variance reduction and is defined by:

$$A^\pi(s,a) = \hat{Q}^\pi(s,a) - V_\theta(s), \tag{1.5}$$

where $\hat{Q}^\pi(s,a)$ is the estimated Q-value for the state–action pair, and $V_\theta(s)$ is the critic's value estimate. The Q-value may be approximated using Monte Carlo returns, temporal-difference updates, or λ-returns [69]. The critic parameters are learned by minimizing the mean-squared error between the critic's predictions and the target Q-value:

$$\theta^{t+1} = \theta^t - \kappa \mathop{\mathbb{E}}_{s\sim p^\pi_\gamma} \nabla_\theta \big(V_\theta(s) - \hat{Q}^\pi(s,a)\big)^2, \quad \text{with} \quad a \sim \pi_\phi(s). \tag{1.6}$$

Similarly to the actor, the critic's gradient is approximated from on-policy trajectories. Although policy methods are generally less sample efficient than their non-policy counterparts, they are amenable to large-scale parallelization [58, 52]. This scalability is particularly attractive in simulation platforms that leverage massively parallel execution on graphics processing units (GPUs).

1.2.2. Off-Policy Actor-Critic

In off-policy AC the critic function represents the Q-value. As such, the critic $Q_\theta : S, A \to \mathbb{R}$ is conditioned on both the state and the action. This setup allows the actor to be optimized directly with respect to the critic, such that the actor learns to output actions that maximize the critic's output. This strategy yields the following approach for calculating the policy gradient:

$$\phi^{t+1} = \phi^t + \kappa \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \nabla_\phi Q_\theta(s, a(\phi, s)). \tag{1.7}$$

Above, $\mathcal{D}$ denotes the replay buffer used to store previous experiences [59], and Q_θ denotes the critic network. If the policy models a Dirac delta distribution then $a(\phi, s) = \pi_\phi(s)$ and the policy gradient is called *deterministic* [73]. Alternatively, the policy can model a parameterized distribution with the actor outputting the parameters of the considered distribution class. Usually, a hyperbolic tangent Gaussian is chosen due to the finite support and expressiveness of this distribution [29]. Then, the action is given by the reparametrization trick [41]:

$$a(\phi, s) = Tanh\big(\mu^{\phi}(s) + \sigma^{\phi}(s)\epsilon\big). \tag{1.8}$$

Above, $\mu^{\phi}(s)$ and $\sigma^{\phi}(s)$ denote the parameters of the Gaussian which are given by the actor network, $Tanh(x)$ is the hyperbolic tangent function, and $\epsilon \sim N(0, 1)$ is the reparametrization noise. The critic network learns via minimizing a SARSA temporal difference variant (i.e., temporal difference where the target value is sampled according to the current policy). The critic loss is defined as follows:

$$\theta^{t+1} = \theta^{t} - \kappa \mathop{\mathbb{E}}_{s,a,r,s' \sim \mathcal{D}} \nabla_{\theta} \big(Q_{\theta}(s, a) - r - \gamma \min(Q^{1}_{\bar{\theta}}(s', a'), Q^{2}_{\bar{\theta}}(s', a'))\big)^2, \text{ with } a' \sim \pi_{\phi}(s'). \tag{1.9}$$

Above, $\bar{\theta}$ denotes the target network parameters which are updated via Polyak averaging [27, 29], and $\min(Q^{1}_{\bar{\theta}}(s', a'), Q^{2}_{\bar{\theta}}(s', a'))$ denotes the pessimistic value target. Using the minimum to calculate a pessimistic value target is supposed to limit the Q-value overestimation and is referred to as Clipped Double Q-learning [27].

Chapter 2

Motivation

In this section, we discuss the motivation behind studying the sample efficiency of RL algorithms. Later, we discuss research themes that might lead to RL algorithms with better sample efficiency.

2.1. Sample Efficiency in Reinforcement Learning

Although there are various important research questions in RL, we focused on the issue of sample efficiency. We discuss why we find sample efficiency important below.

Data Collection - In many real-world applications, interacting with the environment can be expensive, time-consuming, or even dangerous. For example, training an autonomous vehicle or a medical robot requires data that could be risky or costly to gather.

Computational Resources - given constraints on computational power or time, sample efficiency ensures that the agent can learn effectively without requiring excessive resources.

Non-Stationary Environments - In non-stationary environments, sample efficiency is vital because the agent needs to continually adapt to new conditions. Sample efficiency allows the agent to adjust its behavior based on a few recent interactions, rather than relying on extensive data to retrain.

Ethical Considerations - certain training scenarios, such as healthcare or behavioral training, might involve humans or animals. Then, it is crucial to minimize the number of interactions to reduce discomfort, risk, and other negative impacts.

2.2. How to Improve Sample Efficiency?

We discuss research questions designed to better understand solutions that might improve the sample efficiency of AC algorithms in the following paragraphs.

Variance Reduction - Reducing the variance of policy gradient directly translates to sample efficiency (we show this phenomenon in Figure 4.1 in Section 4.1.1). This is because in on-policy implementations, the gradient is calculated after gathering a trajectory of states, and this trajectory often has to be of substantial length to achieve a satisfactory quality of the gradient [83, 58, 44]. Variance reduction might allow for calculating the policy gradient updates with less data and, therefore, performing more updates within a given budget of

environment interactions. We perform a theoretical investigation of strategies for variance reduction, as well as propose a novel low-variance on-policy AC algorithm in [**P1**].

Training Stability - Learning from non-stationary data, as is the case in policy and temporal difference learning, is known to suffer from various training instabilities. In particular, recent literature investigates training instabilities associated with value overestimation [27, 9], critic overfitting [49], and plasticity loss [54, 53]. In particular, value overestimation and plasticity loss are directly associated with sample efficiency: value overestimation results in a policy that explores the actions for which the value is overestimated, and this self-correction mechanism requires environment interactions; plasticity loss, by definition, implies that the neural network requires more gradient steps to achieve a satisfactory solution. We perform an extensive empirical analysis of various algorithmic improvements tackling value overestimation, critic overfitting, and plasticity loss, as well as uncover novel synergy effects between certain techniques in [**P2**].

Exploration - Various theoretical results showcase that certain exploration strategies offer lower regret during the training (i.e., they allow for faster discovery of the optimal policy) [6, 13, 3]. One prominent approach, referred to as optimism in the face of uncertainty, focuses on assigning higher probability to actions that lead to previously unseen states and was shown to improve sample efficiency when implemented with various algorithms [16, 20]. Unfortunately, due to the risk of value overestimation, many modern algorithms leverage the pessimistic Clipped Double Q-learning, which, while controlling overestimation, was shown to result in the so-called pessimistic underexploration [20]. We propose a novel algorithm for simultaneous pessimistic value learning and optimistic exploration in [**P3**].

Value Overestimation - As discussed in the paragraph above, controlling the value overestimation is important for stable learning with AC algorithms and is directly related to sample efficiency. However, overly pessimistic value learning can lead to value underestimation [27, 15, 37], which can impact the performance of the AC algorithm more profoundly, since value underestimation may never be corrected while pursuing a greedy policy. We analyze the approximation error associated with learning via temporal difference presented in Equation 1.9, as well as propose a novel algorithm for online adjustment of the level of pessimism to limit the value over- and underestimation in [**P4**].

Model Scaling - Scaling the number of network parameters was shown to improve the learning efficiency as well as the final performance in domains such as natural language processing [39, 42] and computer vision [24]. Whereas it is not entirely clear why increasing the model size might improve the sample efficiency, there are hypotheses related to the proneness of models to being stuck in local optima [85], the complexity of loss function manifold stemming from the model architecture [26], as well as generalization capabilities of larger models [5, 21]. Unfortunately, previous works have shown that naive scaling of models in deep RL often leads to training instabilities and poor performance [12, 71]. We show neural network architectures that allow for model scaling in off-policy AC methods, as well as propose a state-of-the-art algorithm in [**P5**].

Replay Ratio Scaling - Prior works showed that certain techniques allow for an increase in the number of gradient updates performed per environment interaction (*replay ratio*), leading to improved sample efficiency [36, 35, 25]. However, it remains unclear how scaling the replay ratio interacts with other hyperparameters, such as batch size or learning rate. This raises the question of whether observed efficiency gains could be further amplified by joint tuning

of related hyperparameters, and if the relationship between replay ratio and optimal batch size/learning rate is predictable. We show that, given optimal batch size and learning rate, the sample efficiency gains stemming from increasing the replay ratio follow a power-law relationship, as well as propose heuristics for selection of batch size and learning rate as replay ratio is varied in [**P6**].

Chapter 3

Contributions

In this section, we discuss the scientific contributions made in our work, as well as individual contributions of the candidate.

3.1. Summary of Scientific Contributions

In our work, we explored strategies to enhance the sample efficiency of AC algorithms within deep RL. Our contributions span both theoretical insights and the development of novel, efficient algorithms.

In **[P1]**, we provided theoretical arguments for estimating policy gradients using multiple action samples per state. We introduced a novel model-based algorithm that implements our proposed many-actions sampling approach, demonstrating improved sample efficiency compared to baseline algorithms when tested on a variety of locomotion problems. The work was published in the proceedings of the International Conference on Machine Learning (ICML).

In **[P2]**, we conducted an extensive empirical analysis of recent algorithmic innovations addressing value overestimation, overfitting, and plasticity loss in off-policy actor-critic settings. Our analysis identified combinations of techniques with synergistic effects, leading to substantial improvements over prior work in complex simulated dog tasks. The work was published in the proceedings of the International Conference on Machine Learning (ICML).

In **[P3]**, we showed that commonly implemented pessimistic learning techniques, such as Clipped Double Q-learning, can be understood through the lens of the expected utility hypothesis, with the policies stemming from maximizing Clipped Double Q-learning being approximately equivalent to policies that maximize exponential utility of returns. Additionally, we proposed a new algorithm to enhance exploration in off-policy actor-critic. The work was published in the proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI).

In **[P4]**, we demonstrated that the approximation error of a pessimistic critic can be learned using a recursive objective similar to the Bellman equation. We also introduced an algorithm that dynamically adjusts the level of pessimism in off-policy actor-critic to minimize the value approximation error. The work was published in the proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI).

In **[P5]**, we proposed a neural network architecture that allows for scaling the critic parameter count, leading to significant performance improvements. We proposed an approach

that combines parameter scaling with a variety of other design choices, leading to a state-of-the-art algorithm for continuous control. The work was published in the proceedings of the Neural Information Processing Systems conference (NeurIPS).

In **[P6]**, we showed that data and compute requirements to attain a given performance level lie on a Pareto frontier, controlled by replay ratio. We showed that this frontier can be estimated and used to predict data requirements when given more compute, and compute requirements when given more data. We proposed a framework for efficient tuning of batch size and learning rate when the replay ratio is varied. The work was published in the proceedings of the International Conference on Machine Learning (ICML).

3.2. Summary of Author Contributions

We summarize the candidate's contributions in the paragraphs below.

[P1] has 2 authors (the candidate and the supervisor), the candidate was primarily responsible for the contributions of the article, including the theoretical results, the design and implementation of the proposed and baseline algorithms, the execution and analysis of the experiments, and the writing of the report. The author estimates his contributions to be 85% of the work.

[P2] has 6 authors, including two main authors. In this work, the candidate contributed by proposing and implementing 6 out of the 9 studied techniques, running and analyzing a significant portion of the experiments, and contributing to writing the manuscript. The author estimates his contributions to be 40% of the work.

[P3] has 2 authors (the candidate and the supervisor), the candidate was primarily responsible for the contributions of the article, including the theoretical results, the design and implementation of the proposed and baseline algorithms, the execution and analysis of the experiments, and the writing of the report. The author estimates his contributions to be 85% of the work.

[P4] had 3 authors. The candidate was responsible for the initial derivation of the theoretical results, designing and implementing the proposed algorithm, running and analyzing the majority of the experiments, and contributing to the writing of the paper. The author estimates his contributions to be 60% of the work.

[P5] had 5 authors. The candidate proposed and implemented the method, conducted and analyzed the majority of the experiments, and wrote substantial portions of the paper. The author estimates his contributions to be 75% of the work.

[P6] had 7 authors. The candidate implemented and analyzed the experiments in the DeepMind Control Suite benchmark and formalized the optimization problem tackled in the paper. The author estimates his contributions to be 15% of the work.

Chapter 4

Description of Works

4.1. [P1] On Many-Actions Policy Gradient

In this work, we address the variance in Policy Gradient (PG) estimation and explore efficient methods for reducing this variance. Reducing variance is crucial for enhancing the sample efficiency of on-policy algorithms, as the gradient must be evaluated over long trajectories of states to achieve a high-quality PG estimate [58]. Instead of extending trajectory length, some previous studies have focused on variance reduction through the Many-Action PG (MA) approach [68, 19]. In MA, the gradient is calculated using multiple action samples per state without considering the follow-up states of these additional actions. This method, based on conditional Monte-Carlo [55], results in a variance that is equal to or less than that of single-action PG for a constant trajectory length [44, 19].

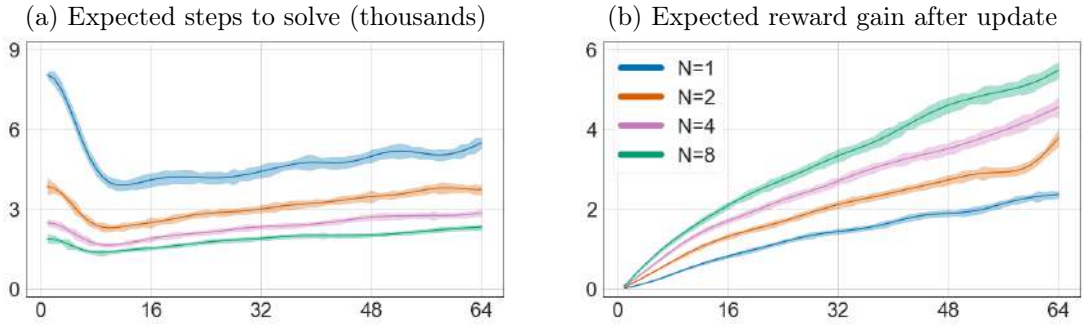


Figure 4.1: Variance reduction leads to better sample efficiency. We train a CartPole Actor-Critic agent with different batch sizes and many action samples per state (denoted as N). In Figures 4.1a and 4.1b X-axis denotes batch size (ie. trajectory length) and Y-axis denotes thousands of steps and average performance gain resulting from a single policy update. Increasing batch size leads to better gradient quality at the cost of fewer updates during training. Sampling more actions yields better gradient quality with fewer environment steps.

In the theoretical section, we present a new analysis addressing the following question: *Given a fixed sequence length and sampling cost, is PG variance more favorably reduced by sampling additional actions or by extending the sequence length?*. We quantify the variance reduction from sampling multiple actions per state versus extending the sequence length of a single-action agent. Finally, we derive conditions under which MA estimation offers

greater variance reduction than extending the trajectory length proportionally. We show that these conditions are often met in Markov Decision Processes, but not in contextual bandit problems. We also propose the Model-Based Many-Actions (MBMA) module, which leverages a learned dynamics model to simulate state-action gradients and can be integrated with any on-policy PG algorithm. Our comparisons with various baseline algorithms demonstrate that MBMA has a favorable bias and variance structure. Finally, we show that this structure leads to better sample efficiency and higher reward sums across various continuous action environments compared to many-actions, model-based, and model-free baselines.

4.2. [P2] Overestimation, Overfitting, and Plasticity in Actor-Critic: the Bitter Lesson of Reinforcement Learning

In this work, we conduct a comprehensive investigation into the effectiveness of algorithmic improvements within the Soft Actor-Critic (SAC) framework for off-policy reinforcement learning. Specifically, we explore improvements aiming at controlling value overestimation [34, 27, 60], reducing overfitting of the critic network [7, 49], and increasing neural network plasticity [25, 48]. We explore how combining these techniques influences the performance of agents across a diverse set of tasks and environments, including both locomotion and manipulation tasks. In total, our study evaluates over 60 design choices, tested within two simulation benchmarks and different hyperparameter configurations.

Table 4.1: We consider 9 techniques tackling value overestimation, critic overfitting, and plasticity loss. We implement over 60 Soft Actor-Critic agents augmented with various combinations of these techniques. We report on the synergy effects between different improvements.

Value Overestimation	Critic Overfitting	Plasticity Loss
Clipped Double Q [27]	Layer Normalization [7]	CReLU [1]
Optimism & Pessimism [60]	Spectral Normalization [57]	Resets [61]
Pessimism Learning [15]	Weight Decay [46]	SAM Optimizer [26]

We aim to address the following research questions: *Are there combinations of techniques that consistently lead to robust performance improvements across varied tasks?*, and *Can general, RL agnostic methods outperform domain-specific reinforcement learning techniques?*. Our findings highlight a "bitter lesson": general neural network regularizers, particularly those motivated by stabilizing gradient-based learning, significantly outperform most reinforcement learning-specific algorithmic improvements. Notably, these regularizers enable agents to achieve effective policies in complex tasks, such as the simulated dog domain, where model-free agents typically fail. We also show that layer normalization [7] is more effective at reducing Q-value overestimation than specialized methods like Clipped Double Q-learning. Furthermore, our study also investigates how overestimation, overfitting, and plasticity loss affect the underlying algorithm performance. We find that the studied dependencies are highly environment dependent, underscoring the necessity for diverse benchmarking in reinforcement learning research. Ultimately, this work provides a deeper understanding of how various techniques can be leveraged to enhance the robustness and effectiveness of off-policy reinforcement learning algorithms across a wide range of tasks.

4.3. [P3] Decoupled Policy Actor-Critic: Bridging Pessimism and Risk Awareness in Reinforcement Learning

In this work, we explore the theoretical underpinnings of optimistic and pessimistic update objectives commonly implemented within off-policy RL [27, 20, 17, 60, 32, 15]. These objectives are used to improve two aspects of reinforcement learning agents: the pessimistic objective is motivated by controlling value overestimation [34, 27]; the optimistic objective is motivated by the popular heuristic of optimism in the face of uncertainty [20, 60, 72]. However, optimizing these objectives is not guaranteed to lead to optimal policies [27, 15].

Algorithm 1 DAC training loop with changes Soft Actor-Critic coloured red.

Models: π_ϕ^p - pessimistic actor; π_η^o - optimistic actor; Q_θ - critic; $Q_{\bar{\theta}}$ - target critic; α - entropy temperature; β^o - optimism; τ - KL weight
Hyperparameters: κ - learning rate, β^p - pessimism; $\mathcal{KL}^*$ - target KL

Sample action from the optimistic actor
$s', r = \text{ENV.STEP}(a) \quad a \sim \pi_\eta^o$
Add transition to the replay buffer
$\text{BUFFER.ADD}(s, a, r, s')$
for $i = 1$ **to** ReplayRatio **do**
  Sample batch of transitions
  $s, a, r, s' \sim \text{BUFFER.SAMPLE}$
  Update the soft critic using the pessimistic actor
  $\theta \leftarrow \theta - \kappa\nabla_\theta\big(Q_\theta(s', a') - \big(r + \gamma(Q_{\bar{\theta}}^{\beta^p}(s, a) - \alpha \log \pi_\phi^p(a'|s')\big)^2, \quad \text{with} \quad a' \sim \pi_\phi^p(s')$
  Update pessimistic actor via maximum entropy objective
  $\phi \leftarrow \phi + \kappa\nabla_\phi\big(Q_\theta^{\beta^p}(s, a) - \alpha \log \pi_\phi^p(a|s)\big), \quad \text{with} \quad a \sim \pi_\phi^p(s)$
  Update optimistic actor via optimistic actor objective
  $\eta \leftarrow \eta + \kappa\nabla_\eta\big(Q_\theta^{\beta^p}(s, a) - \tau KL\big(\pi_\phi^p(s)|\bar{\pi}_\eta^o(s)\big)\big), \quad \text{with} \quad a \sim \pi_\eta^o(s)$
  Update entropy temperature
  $\alpha \leftarrow \alpha - \kappa\nabla_\alpha\alpha\big(\mathcal{H}^* - \mathcal{H}(s)\big)$
  Update optimism
  $\beta^o \leftarrow \beta^o - \kappa\nabla_{\beta^o}(\beta^o - \beta^p)(\frac{1}{|A|}KL(\pi_\phi^p(s)|\pi_\eta^o(s)) - \mathcal{KL}^*)$
  Update KL weight
  $\tau \leftarrow \tau + \kappa\nabla_\tau\tau(\frac{1}{|A|}KL(\pi_\phi^p(s)|\pi_\eta^o(s)) - \mathcal{KL}^*)$
  Update target network
  $\bar{\theta} \leftarrow \text{POLYAK}(\theta, \bar{\theta})$
end for

Our analysis reveals that the optimistic and pessimistic objectives can be derived from the application of decision theory principles to RL. In particular, we show that these objectives correspond to the maximization of an expected utility objective given an exponential utility function. We also introduce a new off-policy actor-critic algorithm called Decoupled Policy Actor-Critic (DAC), which features separate optimistic and pessimistic actors, each trained on distinct objectives. The optimistic actor, used exclusively for exploration, aims to maximize an upper bound of the Q-value, while the pessimistic actor, used for sampling the temporal difference targets, focuses on minimizing a lower bound to address value overestimation. We design gradient-based adjustments of the hyperparameters of the optimistic actor, allowing DAC to adapt dynamically to varying levels of uncertainty and reward scales without manual adjustments of hyperparameters. Our experiments show that DAC achieves sample efficiency improvements as compared to baseline algorithms across various tasks.

4.4. [P4] A Case for Validation Buffer in Pessimistic Actor-Critic

In this paper, we further analyze the effects of pessimistic updates in off-policy RL, particularly focusing on continuous action spaces. We motivate our research by recent findings that show that pessimistic updates can sometimes lead to value underestimation [37]. Whereas overestimation issues are likely to be corrected due to greedy exploration, underestimation can lead to suboptimal solutions that are not necessarily self-correcting because a greedy policy is less likely to pick actions which lead to low critic output. We therefore analyze strategies for online adjustment of pessimism levels.

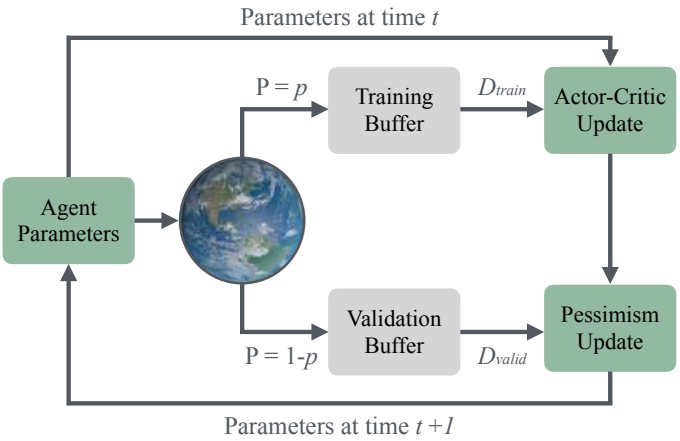


Figure 4.2: High level overview of the proposed approach. After environment step, the transition is stored in either the training buffer (used for updating actor-critic modules) or the validation buffer (used for updating pessimism module). The pessimism is updated via a "reverse" TD loss, optimization of which on the training buffer would be prone to overfitting.

We begin our analysis by characterizing the nature of error propagation in critic networks, attributed mainly to inaccuracies in temporal difference learning and the tendency of maximization algorithms to overestimate values [34, 27]. We find that the critic approximation error itself can be evaluated using a dynamic programming objective similar to the Bellman rule [77] and provide a theoretical framework defining the critic approximation error through a recursive fixed-point model. This analysis also allows us to provide novel insights into the convergence conditions of pessimistic objectives. We introduce the Validation Pessimism Learning (VPL) algorithm as a novel approach to dynamically adjust pessimism during training. VPL utilizes a small validation replay buffer to refine the pessimism levels, aiming to minimize error in the critic's value estimates while preventing overfitting to the accumulated training data. Through empirical analysis, we challenge the convention that all transitions need to be included in the replay buffer for sample-efficient learning, suggesting that strategic use of a validation buffer can enhance performance without compromising the learning efficacy of the RL agent. Our experiments across various tasks illustrate that VPL enhances sample efficiency as compared to other pessimism adjustment strategies.

4.5. [P5] Bigger, Regularized, Optimistic: Scaling for Compute and Sample-Efficient Continuous Control

Whereas in [**P2**, **P3**, and **P4**] we analyzed the improvements in regularization, exploration, and overestimation respectively, this paper considers the potential of combining these in-

sights with scaling the parameter count of the actor-critic models. Traditionally, RL has focused on smaller networks and algorithmic refinements to address challenges like exploration-exploitation dilemma. Recent trends in deep learning suggest that larger models might yield significant benefits [39, 4, 2], previously demonstrated primarily in natural language processing [42] and computer vision [24]. Unfortunately, previous work shows that naive application of scaled models in RL leads to performance collapse [12, 71].

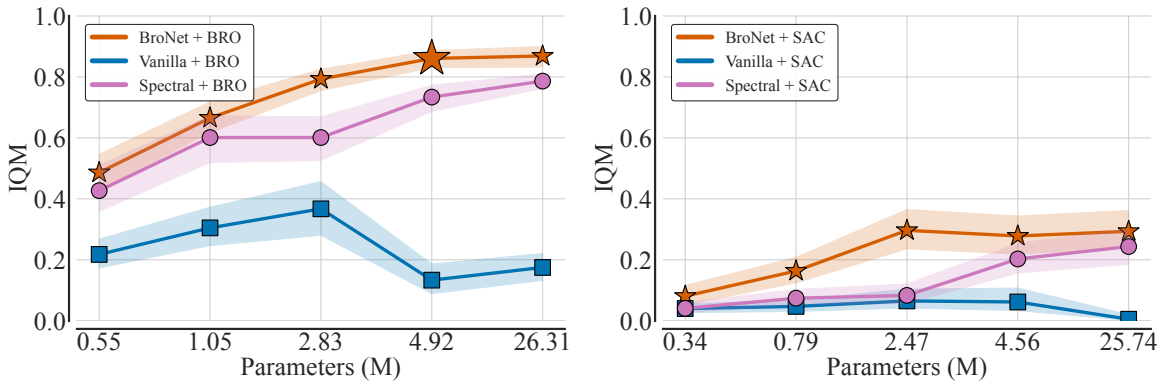


Figure 4.3: Scaling the critic parameter count for vanilla dense [27], spectral normalization ResNet [12], and our BroNet for BRO (left), and SAC [29] (right). We conclude that to achieve the best performance, we need both the right architecture (BroNet) and the correct algorithmic enhancements encapsulated in BRO. We report interquartile mean performance after $1M$ environment steps in 10 complex locomotion and manipulation tasks, with error bars indicating 95% CI from 10 seeds. On the X-axis, we report the approximate parameter count of each model.

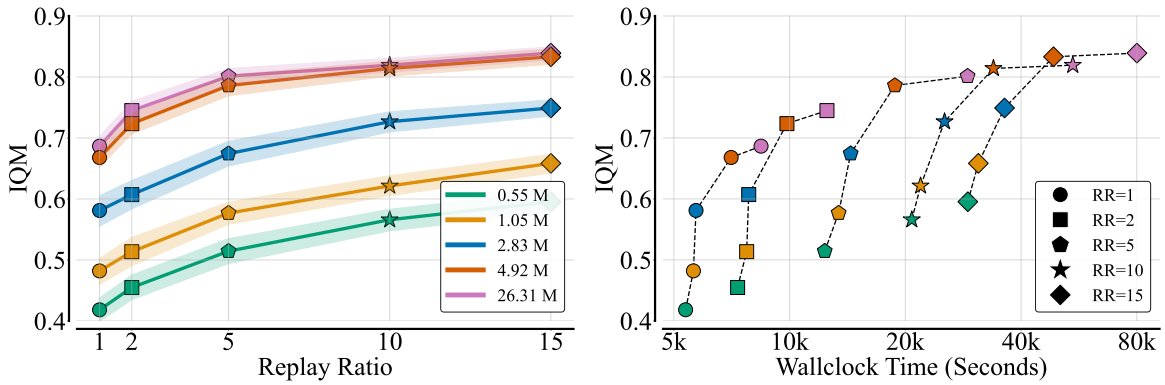


Figure 4.4: We report the averaged interquartile mean across different 5 replay ratios and 5 critic model sizes. All agents were evaluated in 30 complex tasks, and 10 seeds per variant. The left figure shows performance scaling with increasing replay ratios (shapes) and model sizes (colors). The right figure examines the tradeoff between performance and computational cost when scaling replay ratios versus critic model sizes. Increasing model size leads to substantial performance improvements at lower compute costs compared to increasing the replay ratio.

In this paper, we investigate various approaches for scaling models in RL, and combining scaling with design choices such as regularization, normalization, and distributional reinforcement learning. We propose BroNet, a neural network architecture that achieves improved performance when scaling. Importantly, we show that when using BroNet scaling the parameter count leads to similar or better performance improvements as increasing the replay

ratio, while being more computationally efficient. Based on these insights, we introduce the BRO (Bigger, Regularized, Optimistic) algorithm. BRO employs a combination of scaling the parameter count, model regularization, and optimistic exploration to achieve significant performance improvement on 40 complex continuous control tasks. Interestingly, BRO achieves performance improvements over state-of-the-art model-based reinforcement learning algorithm (TD-MPC2 [33]) while requiring 5 times less time to converge.

4.6. [P6] Value-Based Reinforcement Learning Scales Predictably

Whereas in [**P6**] we investigated the feasibility of model scaling in off-policy actor-critic, this paper focuses on scaling the *replay ratio*, that is the number of gradient updates per environment step in online off-policy RL. In particular, we study whether the improvements stemming from increasing the replay ratio compute lead to predictable improvements in sample efficiency.

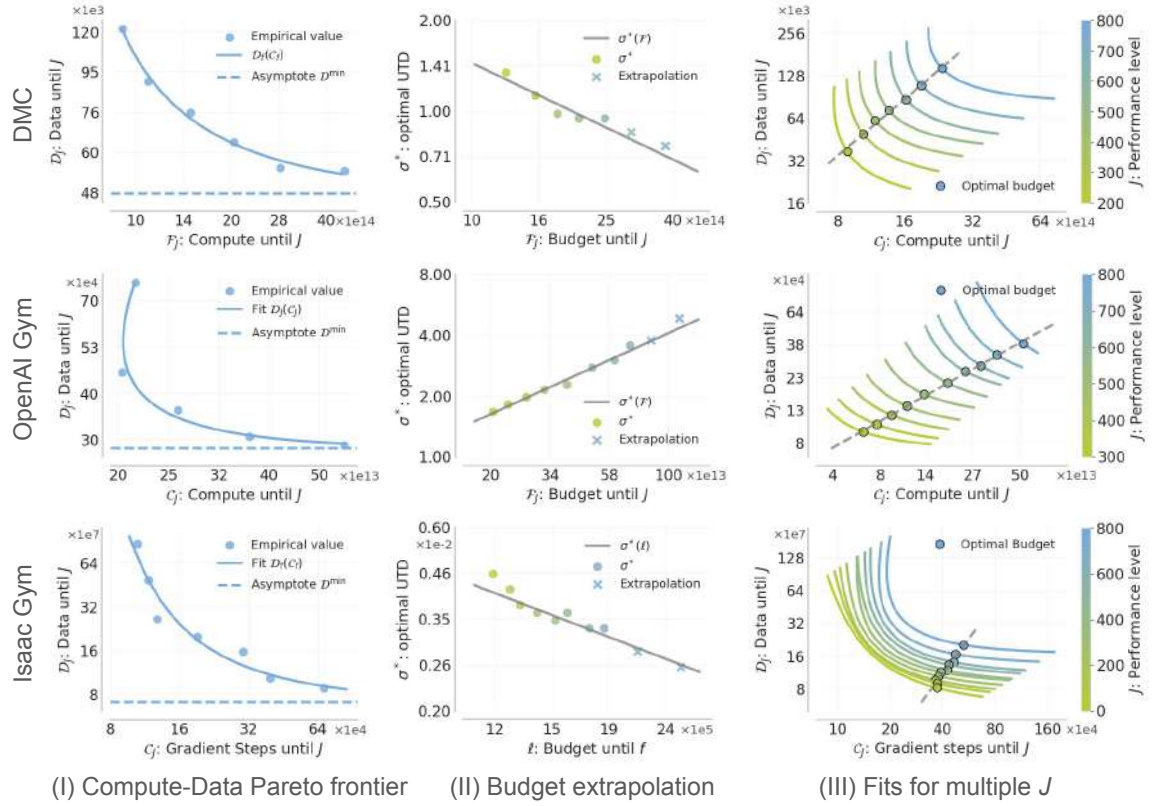


Figure 4.5: Scaling properties when increasing compute $\mathcal{C}$, data $\mathcal{D}$, budget $\mathcal{F}$, or performance J. **Left:** Compute versus data requirements Pareto frontier controlled by the replay ratio σ. We observe that we can trade off data for compute and vice versa, and this relationship is predictable. **Middle:** Extrapolation from low to high performance. We observe that the optimal resource allocation controlled by σ evolves predictably with increasing budget, and can be used to extrapolate from low to high performance. **Right:** Pareto frontiers for several performance levels J.

To demonstrate that the behavior of value-based RL can be predicted reliably at scale, we first formalize this problem as multiple resource optimization questions. Viewing data and compute as two resources, we answer questions of the form: what is the minimum value

of *resource* needed to attain a given target performance? Do optimal settings of hyperparameters, such as batch size and learning rate, change predictably as compute is varied? We answer these questions by fitting empirical laws from low data and compute runs to determine relationships between hyperparameters. Doing so, in turn, enables us to determine how to set hyperparameters and allocate resources to maximize performance, given a larger data and compute budget. While questions of this form have been studied in supervised learning, answering them is different in the context of online RL, because online RL requires the algorithm to collect its own data during training, which ties data and compute in a complex manner and breaks the i.i.d. nature of datapoints.

Chapter 5

Conclusions and Future Work

This dissertation focused on the development of sample-efficient methods for reinforcement learning. It explores a variety of techniques, including model-based and model-free, on-policy and off-policy, as well as online and offline RL. Throughout our research, we have made a range of theoretical and applied contributions. These include improvements to theoretical understanding of sampling strategies for approximation of policy gradients and pessimism in temporal difference learning, as well as development of best-in-class algorithm for continuous control via reinforcement learning.

Despite addressing several challenges specific to RL algorithms, such as exploration or overestimation bias, our most substantial improvements came from techniques that are largely RL-agnostic. Methods such as scaling model capacity, applying regularization to reduce overfitting, and incorporating normalization to stabilize gradient optimization provided much greater performance improvements than domain-specific algorithmic refinements. This observation echoes Richard Sutton's seminal essay "The Bitter Lesson", which argues that progress in AI is more often achieved through leveraging general-purpose computation and scalable function approximation rather than carefully hand-engineered, domain-specific solutions.

Consequently, the evidence suggests that one of the most impactful avenues for future progress in reinforcement learning is not the invention of entirely new algorithmic paradigms, but rather the systematic improvement of the scaling capabilities of existing methods. By scaling, we mean enhancing an algorithm's ability to reliably increase performance as resources grow - whether through larger model capacity, a greater number of gradient updates per unit of data, or expanded offline datasets. Strengthening these scaling properties would not only align RL more closely with the broader trajectory of machine learning but also provide a principled framework for turning additional compute, data, and experience into predictable performance gains.

Chapter 6

Acknowledgments

I would like to thank Marek Cygan, my supervisor, for believing in me, and for his time, support and guidance throughout the studies.

I would also like to thank Pieter Abbeel, who hosted me during my 9 month stay as a visiting scholar at University of California, Berkeley. I have learned a lot during this stay and I am grateful for this amazing experience.

I would also like to thank Piotr Sankowski, who gave me a great research environment in Ideas NCBR, where I stayed for 2 years of my PhD. I have met a lot of great people there and I am thankful for this opportunity.

I would also like to thank my amazing collaborators: Mateusz Ostaszewski, Piotr Miłoś, Aviral Kumar, Oleg Rybkin, Youngyo Seo, Carlo Sferrazza, Alicja Ziarko, Gracjan Góral, Preston Fu, Krzysztof Jankowski, Maciej Wołczyk, Sergey Levine, Michał Kosiński, Haoran Geng, Zhao-Heng Yin, Michał Bortkiewicz, Tomasz Trzciński, and Charlie Snell.

Finally, I would like to thank my family: Lan Anh Do, Katarzyna Smólska and Jan Smólski, Aleksander Nauman and Małgorzata Kazubowska-Nauman, Krystyna Zakrzewska-Walczak, Janusz Nauman and Alicja Nauman, Janusz Nauman and Agata Nauman, Maciej Nauman and Natasza Rakowska.

Bibliography

[1] Zaheer Abbas, Rosie Zhao, Joseph Modayil, Adam White, and Marlos C. Machado. Loss of plasticity in continual deep reinforcement learning. *arXiv preprint arXiv: 2303.07507*, 2023.

[2] Armen Aghajanyan, Lili Yu, Alexis Conneau, Wei-Ning Hsu, Karen Hambardzumyan, Susan Zhang, Stephen Roller, Naman Goyal, Omer Levy, and Luke Zettlemoyer. Scaling laws for generative mixed-modal language models. In *International Conference on Machine Learning*, pages 265–279. PMLR, 2023.

[3] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1. JMLR Workshop and Conference Proceedings, 2012.

[4] Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312, 2022.

[5] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[6] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[8] Gregor Bachmann, Sotiris Anagnostidis, and Thomas Hofmann. Scaling mlps: A tale of inductive bias. *arXiv preprint arXiv: 2306.13575*, 2023.

[9] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. 2023.

[10] Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

[11] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[12] Nils Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning with spectral normalization. *Advances in neural information processing systems*, 34:8242–8255, 2021.

[13] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[14] Tom B Brown. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*, 2020.

[15] Edoardo Cetin and Oya Celiktutan. Learning pessimism for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6971–6979, 2023.

[16] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

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

[18] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.

[19] K Ciosek and S Whiteson. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21(2020), 2020.

[20] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32, 2019.

[21] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pages 1282–1289. PMLR, 2019.

[22] Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.

[23] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[24] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023.

[25] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2022.

[26] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2020.

[27] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[28] Sabino Gadaleta and Gerhard Dangelmayr. Optimal chaos control through reinforcement learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9(3):775–788, 1999.

[29] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[30] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[31] Ben Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *Mathematical Finance*, 33(3):437–503, 2023.

[32] N Hansen, X Wang, and H Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning, PMLR*, 2022.

[33] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv: 2310.16828*, 2023.

[34] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[35] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2021.

[36] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] Tianying Ji, Yu Luo, Fuchun Sun, Xianyuan Zhan, Jianwei Zhang, and Huazhe Xu. Seizing serendipity: Exploiting the value of past success in off-policy actor-critic. *arXiv preprint arXiv:2306.02865*, 2023.

[38] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[39] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[40] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[41] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[42] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[43] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[44] Wouter Kool, Herke van Hoof, and Max Welling. Estimating gradients for discrete random variables by sampling without replacement. In *International Conference on Learning Representations*, 2019.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[46] Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

[47] Max WY Lam, Qiao Tian, Tang Li, Zongyu Yin, Siyuan Feng, Ming Tu, Yuliang Ji, Rui Xia, Mingbo Ma, Xuchen Song, et al. Efficient neural music generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[48] Hojoon Lee, Hanseul Cho, Hyunseung Kim, Daehoon Gwak, Joonkee Kim, Jaegul Choo, Se-Young Yun, and Chulhee Yun. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[49] Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. In *The Eleventh International Conference on Learning Representations*, 2022.

[50] Chin-Teng Lin and Chong-Ping Jou. Controlling chaos by ga-based reinforcement learning neural network. *IEEE Transactions on Neural Networks*, 10(4):846–859, 1999.

[51] Michael Lederman Littman. *Algorithms for sequential decision-making*. Brown University, 1996.

[52] Chris Lu, Jakub Kuba, Alistair Letcher, Luke Metz, Christian Schroeder de Witt, and Jakob Foerster. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

[53] Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv: 2402.18762*, 2024.

[54] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. *arXiv preprint arXiv:2303.01486*, 2023.

[55] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[56] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[57] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations*, 2018.

[58] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

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

[60] Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12849–12863, 2021.

[61] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pages 16828–16847. PMLR, 2022.

[62] James R Norris and James Robert Norris. *Markov chains*. Cambridge university press, 1998.

[63] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[64] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[65] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[66] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[67] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

[69] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[71] Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pages 30365–30380. PMLR, 2023.

[72] Tim Seyde, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Learning to plan optimistically: Uncertainty-guided deep exploration via latent model ensembles. In *Conference on Robot Learning*, pages 1156–1167. PMLR, 2022.

[73] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[74] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[75] Laura Smith, Ilya Kostrikov, and Sergey Levine. Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *Robotics: Science and Systems (RSS) Demo*, 2(3):4, 2023.

[76] Donald F Specht et al. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.

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

[78] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[79] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[80] Ashish Vaswani. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

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

[82] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[83] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[84] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

[85] Ji Xu, Daniel J Hsu, and Arian Maleki. Benefits of over-parameterization with em. *Advances in Neural Information Processing Systems*, 31, 2018.

[86] Yimo Yan, Andy HF Chow, Chin Pang Ho, Yong-Hong Kuo, Qihao Wu, and Chengshuo Ying. Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities. *Transportation Research Part E: Logistics and Transportation Review*, 162:102712, 2022.

[87] Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.

Appendix A

Complete Publications

On Many-Actions Policy Gradient

Michal Nauman [1 2] **Marek Cygan** [1 3]

Abstract

We study the variance of stochastic policy gradients (SPGs) with many action samples per state. We derive a many-actions optimality condition, which determines when many-actions SPG yields lower variance as compared to a single-action agent with proportionally extended trajectory. We propose Model-Based Many-Actions (MBMA), an approach leveraging dynamics models for many-actions sampling in the context of SPG. MBMA addresses issues associated with existing implementations of many-actions SPG and yields lower bias and comparable variance to SPG estimated from states in model-simulated rollouts. We find that MBMA bias and variance structure matches that predicted by theory. As a result, MBMA achieves improved sample efficiency and higher returns on a range of continuous action environments as compared to model-free, many-actions, and model-based on-policy SPG baselines.

1. Introduction

Stochastic policy gradient (SPG) is a method of optimizing stochastic policy through gradient ascent in the context of reinforcement learning (RL) (Williams, 1992; Sutton et al., 1999; Peters & Schaal, 2006). When paired with powerful function approximators, SPG-based algorithms have proven to be one of the most effective methods for achieving optimal performance in Markov Decision Processes (MDPs) with unknown transition dynamics (Schulman et al., 2017). Unfortunately, the exact calculation of the gradient is unfeasible and thus the objective has to be estimated (Sutton et al., 1999). Resulting variance is known to impact learning speed, as well as performance of the trained agent (Konda & Tsitsiklis, 1999; Tucker et al., 2018).

[1]Informatics Institute, University of Warsaw [2]Ideas National Centre for Research and Development [3]Nomagic. Correspondence to: Michal Nauman <nauman.mic@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

On-policy sample efficiency (ie. the number of environment interactions needed to achieve a certain performance level) is particularly affected by variance, as the gradient must be evaluated over long sequences in order to produce a sufficient quality of the SPG estimate (Mnih et al., 2016). As such, a variety of methods for SPG variance reduction have been proposed. The most widely used is baseline variance reduction, which has been shown to improve algorithms stability and became indispensable to contemporary SPG implementations (Peters & Schaal, 2006; Schulman et al., 2015b). Alternative approaches include Q-value bootstrapping (Gu et al., 2017), reducing the effect of long-horizon stochasticity via small discount (Baxter & Bartlett, 2001), increasing number of samples via parallel agents (Mnih et al., 2016) or using many-actions estimator (Asadi et al., 2017; Kool et al., 2019b; Petit et al., 2019; Ciosek & Whiteson, 2020).

In many-actions SPG (MA), the gradient is calculated using more than one action sample per state, without including the follow-up states of additional actions. The method builds upon conditional Monte-Carlo and yields variance that is smaller or equal to that of single-action SPG given fixed trajectory length (Bratley et al., 2011). These additional action samples can be drawn with (Ciosek & Whiteson, 2020) or without replacement (Kool et al., 2019b) and can be generated through rewinding the environment (Schulman et al., 2015a) or using a parametrized Q-value approximator (Asadi et al., 2017). However, drawing additional action samples from the environment is unacceptable in certain settings, while using a Q-network may introduce bias to the gradient estimate. Furthermore, a diminishing variance reduction effect can be achieved by extending the trajectory. This leads to the following questions:

1. Given fixed trajectory length and cost of sampling actions, is SPG variance more favorable when sampling additional actions or extending the trajectory?
2. Given that more samples translate to smaller variance, what is the bias associated with simulating such additional samples via neural networks?

The contributions of this paper are twofold. Firstly, we analyze SPG variance theoretically. We quantify the variance reduction stemming from sampling multiple actions per state

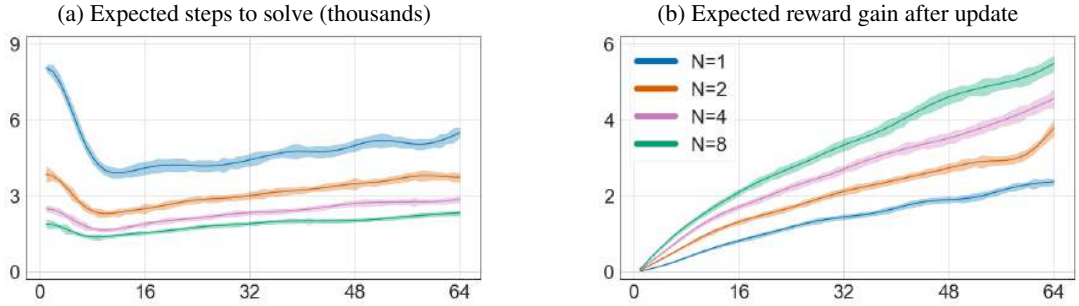


Figure 1. Variance reduction leads to better sample efficiency. We train a CartPole Actor-Critic agent with different batch sizes and many action samples per state (denoted as N). In Figures 1a and 1b X-axis denotes batch size (ie. trajectory length) and Y-axis denotes thousands of steps and average performance gain resulting from a single policy update. Increasing batch size leads to better gradient quality at the cost of fewer updates during training. Sampling more actions yields better gradient quality with fewer environment steps.

as compared to extending the trajectory of a single-action agent. We calculate conditions under which adopting MA estimation leads to greater variance reduction than extending trajectory length. We show that the conditions are often met in RL, but are impossible for contextual bandits. Secondly, we propose an implementation of MA, which we refer to as the Model-Based Many-Actions module (MBMA). The module leverages a learned dynamics model to sample state-action gradients and can be used in conjunction with any on-policy SPG algorithm. MBMA yields a favorable bias/variance structure as compared to learning from states simulated in the dynamics model rollout (Janner et al., 2019; Kaiser et al., 2019; Hafner et al., 2019) in the context of on-policy SPG. We validate our approach and show empirically that using MBMA alongside PPO (Schulman et al., 2017) yields better sample efficiency and higher reward sums on a variety of continuous action environments as compared to many-actions, model-based and model-free PPO baselines.

2. Background

A Markov Decision Process (MDP) (Puterman, 2014) is a tuple (S, A, R, p, γ), where S is a countable set of states, A is a countable set of actions, $R(s, a)$ is the state-action reward, $p(s'|s, a)$ is a transition kernel (with the initial state distribution denoted as p_0) and $\gamma \in (0, 1]$ is a discount factor. A policy $\pi(a|s)$ is a state-conditioned action distribution. Given a policy π, MDP becomes a Markov reward process with a transition kernel $p^\pi(s'|s) = \int_a \pi(a|s)\, p(s'|s, a)\, da$, which we refer to as the underlying Markov chain. The underlying Markov chain is assumed to have finite variance, a unique stationary distribution denoted as p_0^π (Ross et al., 1996; Konda & Tsitsiklis, 1999), t-step stationary transition kernel p_t^π and a unique discounted stationary distribution denoted as p_*^π. Interactions with the MDP according to some policy π are called trajectories and are denoted as $\tau_T^\pi(s_t) = ((s_t, a_t, r_t), ..., (s_{t+T}, a_{t+T}, r_{t+T}))$, where $a_t \sim \pi(a_t|s_t)$, $r_t \sim R(s, a)$ and $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. The value function $V^\pi(s) = \mathrm{E}_{\tau_\infty^\pi(s)}[\sum_{t=0}^\infty \gamma^t R(s_t, a_t)]$ and Q-value function $Q^\pi(s, a) = \mathrm{E}_{\tau_\infty^\pi(s|a)}[\sum_{t=0}^\infty \gamma^t R(s_t, a_t)] = R(s, a) + \gamma \mathrm{E}_{s' \sim p(s'|s,a)}[V^\pi(s')]$ sample a_t according to some fixed policy π. State-action advantage is defined as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. An optimal policy is a policy that maximizes discounted total return $J = \int_{s_0} V^\pi(s_0)\, ds_0$.

2.1. On-policy SPG

Given a policy parametrized by θ, the values of θ can be updated via SPG $\theta \leftarrow \theta + \nabla_\theta J$. Since we are interested only in gradient wrt. θ, we drop it from the gradient notation in further uses. The SPG is given by (Sutton & Barto, 2018):

$$\nabla J = \mathop{\mathbb{E}}_{s \sim p_*^\pi} \mathop{\mathbb{E}}_{a \sim \pi} Q^\pi(s, a)\, \nabla \log \pi(a|s) \tag{1}$$

As such, SPG is proportional to a double expectation of $Q^\pi(s, a) \nabla_\theta \log \pi(a|s)$, with the outer expectation taken wrt. the discounted stationary distribution p_*^π and the inner expectation taken wrt. policy π. The gradient can be estimated via a trajectory sampled according to the policy (Nota & Thomas, 2020; Wu et al., 2022). We denote $\nabla \hat{J}$ as the estimator, $\nabla J(s_t, a_t) = Q^\pi(s_t, a_t) \nabla \log \pi(a_t|s_t)$ with $s_t, a_t \sim p_t^\pi, \pi$. Then, SPG can be calculated:

$$\nabla \hat{J} = \frac{1}{T} \sum_{t=0}^{T-1} \gamma^t\, \nabla J(s_t, a_t) \tag{2}$$

In the setup above, the outer expectation of Equation 1 is estimated via Monte-Carlo (Metropolis & Ulam, 1949) with T state samples drawn from the non-discounted stationary distribution p_0^π; and the inner expectation is estimated with a single action per state drawn from the policy $\pi(a|s)$. The resulting variance can be reduced to a certain degree by a control variate, with state value being a popular choice for such baseline (Schulman et al., 2015b). Then, the Q-value from Equation 1 is replaced by $A^\pi(s_t, a_t)$. If the state value

is learned by a parametrized approximator, it is referred to as the *critic*. Critic bootstrapping (Gu et al., 2017) is defined as $Q^\pi(s,a) = R(s,a) + \gamma V^\pi(s')$ with $s' \sim p(s'|s,a)$ and can be used to balance the bias-variance tradeoff of Q-value approximations.

2.2. On-policy Many-Actions SPG

Given a control variate, the variance of policy gradient can be further reduced by approximating the inner integral of Equation 2 with a quadrature of $N > 1$ action samples. Then, $\nabla \hat{J}$ is equal to:

$$\nabla \hat{J} = \underbrace{\frac{1}{T}\sum_{t=0}^{T-1} \gamma^t \underbrace{\frac{1}{N}\sum_{n=0}^{N-1} \nabla J(s_t, a_t^n)}_{N \text{ actions per state}}}_{T \text{ state samples in a trajectory}} \tag{3}$$

Where a_t^n denotes the n^{th} action sampled at state s_t. Furthermore, MDP transitions are conditioned only on the first action performed (ie. $p^\pi(s_{t+1}|s_t, a_t^n) = p^\pi(s_{t+1}|s_t) \iff n \neq 0$). Implementations of such an approach were called *all-action policy gradient* or *expected policy gradient* (Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020). As follows from the law of iterated expectations, the many-actions (MA) estimator is unbiased and yields lower or equal variance as compared to single-action SPG with equal trajectory length (Petit et al., 2019). Since the policy log probabilities are known, using MA requires approximating the Q-values of additional action samples. As such, MA is often implemented by performing rollouts in a rewinded environment (Schulman et al., 2015a; Kool et al., 2019a;b) or by leveraging a Q-network at the cost of bias (Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020). The variance reduction stemming from using MA has been shown to increase both performance and sample efficiency of SPG algorithms (Schulman et al., 2015a; Kool et al., 2019b).

3. Variance of Stochastic Policy Gradients

Throughout the section, we assume no stochasticity induced by learning Q-values and we treat Q-values as known. Furthermore, when referring to SPG variance, we refer to the diagonal of the policy parameter variance-covariance matrix. Finally, to unburden the notation, we define $\Upsilon^t = \gamma^t \nabla J(s_t, a_t)$ and $\bar{\Upsilon}^t = \gamma^t \mathbb{E}_{a\sim\pi} \nabla J(s_t, a_t)$, where we skip the superscript when $t = 0$. Similarly, we use $\mathbb{O}_a(\cdot) = \mathbb{O}_{a\sim\pi}(\cdot)$, $\mathbb{O}_s(\cdot) = \mathbb{O}_{s\sim p_0^\pi}(\cdot)$ and $\mathbb{O}_{s,a}(\cdot) = \mathbb{O}_{s_t, a_t \sim p_t^\pi, \pi}(\cdot)$, where $\mathbb{O}$ denotes expected value, variance and covariance operators. As shown, given fixed trajectory length T, MA-SPG variance is smaller or equal to the variance of single-action agent Petit et al. (2019); Ciosek & Whiteson (2020). However, approximating the inner expectation of SPG always uses resources (ie. compute or environment interactions), which could be used to reduce the SPG variance through other means (eg. extending the trajectory length). To this end, we extend existing results (Petit et al., 2019; Ciosek & Whiteson, 2020) by comparing the variance reduction stemming from employing MA as opposed to using regular single-action SPG with an extended trajectory length. If the underlying Markov chain is ergodic the variance of SPG, denoted as $\mathbf{V}$, can be calculated via Markov chain Central Limit Theorem (Jones, 2004; Brooks et al., 2011):

$$\mathbf{V} = \frac{1}{T} \operatorname*{Var}_{s,a}\left[\Upsilon\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T^2} \operatorname*{Cov}_{s,a}\left[\Upsilon, \Upsilon^t\right] \tag{4}$$

The states underlying Υ and Υ^t are sampled from the undiscounted stationary distribution p_0^π and the t-step stationary transition kernel p_t^π respectively. As follows from the ergodic theorem (Norris & Norris, 1998), conditional probability of visiting state s_t given starting in state s_0 with action a_0^0 approaches the undiscounted stationary distribution p_0^π exponentially fast as t grows $\lim_{t\to\infty} p(s_t|s_0, a_0^1) = p_0^\pi(s_t)$. Therefore, $\mathrm{Cov}_t \geq \mathrm{Cov}_{t+1}$, as well as $\lim_{t\to\infty} \mathrm{Cov}_t = 0$. Equation 4 shows the well-known result that increasing the trajectory length T decreases $\mathbf{V}$. This result contextualizes the success of parallel SPG (Mnih et al., 2016). Unfortunately, the form above assumes single action per state.

3.1. Variance Decomposition

To quantify variance reduction stemming from many action samples, we decompose $\mathbf{V}$ into sub-components. We include derivations in Appendix A.1.

Lemma 3.1. *Given ergodic MDP, SPG with N action samples per state and T states, $\mathbf{V}$ can be decomposed into:*

$$\begin{aligned}
\operatorname*{Var}_{s,a}\left[\Upsilon\right] &= \operatorname*{Var}_{s}\left[\bar{\Upsilon}\right] + \frac{1}{N}\mathop{\mathbb{E}}_{s}\operatorname*{Var}_{a}\left[\Upsilon\right] \\
\operatorname*{Cov}_{s,a}\left[\Upsilon, \Upsilon^t\right] &= \operatorname*{Cov}_{s,a}\left[\hat{\Upsilon}, \hat{\Upsilon}^t\right] + \frac{1}{N}\mathop{\mathbb{E}}_{s}\operatorname*{Cov}_{s,a}\left[\Upsilon, \Upsilon^t\right]
\end{aligned} \tag{5}$$

Combining Lemma 3.1 with Equation 4 yields an expression for decomposed SPG variance, where we group components according to dependence on N:

$$\begin{aligned}
T\,\mathbf{V} = &\underbrace{\operatorname*{Var}_{s}\left[\hat{\Upsilon}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\operatorname*{Cov}_{s,a}\left[\hat{\Upsilon}, \hat{\Upsilon}^t\right]}_{\text{Marginalized policy variance}} \\
&+ \underbrace{\frac{1}{N}\mathop{\mathbb{E}}_{s}\left(\operatorname*{Var}_{a}\left[\Upsilon\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\operatorname*{Cov}_{s,a}\left[\Upsilon, \Upsilon^t\right]\right)}_{\text{Policy-dependent variance}}
\end{aligned} \tag{6}$$

Table 1. Decomposed trace of variance-covariance matrix divided by the number of parameters. The components were estimated by marginalizing Q-values, with Equation 3 and Lemma 3.1 using 125000 non-baselined interactions. The last two columns record the best performance during $500k$ environment steps (average performance shown in the brackets). The performance of SPG variants was measured during $500k$ training steps with additional action samples drawn from the environment. With most variance depending on the policy, MA often yields better performance than single-action agents with extended trajectories. We detail the setting in Appendix B.

	VARIANCE COMPONENT		PERFORMANCE	
TASK	MARGINALIZED POLICY	POLICY-DEPENDENT	$(T,\ N) = (1024,\ 2)$	$(T,\ N) = (2048,\ 1)$
BALL CATCH	0.026 (3%)	0.819 (97%)	905 (708)	920 (715)
CART SWINGUP	0.006 (1%)	5.736 (99%)	837 (670)	801 (669)
CHEETAH RUN	0.006 (1%)	1.615 (99%)	208 (131)	204 (126)
FINGER SPIN	0.026 (18%)	0.122 (82%)	304 (187)	281 (179)
REACHER EASY	2.269 (39%)	3.565 (61%)	428 (262)	776 (488)
WALKER WALK	0.081 (1%)	11.786 (99%)	509 (315)	465 (287)

Given $N = 1$, the variance simplifies to a single-action case. The statement shows that SPG variance can be decomposed into: marginalized policy variance, which stems from the underlying Markov chain and is decreased only by trajectory length (T); and policy-dependent variance, which indeed is reduced by both sampling more actions per state (N) and increasing trajectory length (T). Table 1 shows estimated variance components and performance of two SPG estimators ($T = 1024; N = 2$ and $T = 2048; N = 1$) for 6 Deepmind Control Suite (DMC) environments. In particular, the table shows that with Q-values marginalized, the policy is responsible for around 90% of SPG variance in tested environments.

3.2. Measuring Variance Reduction

We proceed with the analytical analysis of the variance reduction stemming from increasing N and T.

Lemma 3.2. *Given ergodic MDP, SPG with N action samples per state and T states, variance reduction stemming from increasing N by 1 (denoted as Δ_N) and variance reduction stemming from increasing the trajectory length to $T + \delta T$ with $\delta \in (0, \infty)$ (denoted as Δ_T) are equal to:*

$$\frac{\Delta_N}{\alpha_N} = \mathop{\mathbb{E}}_{s}\Big(\mathop{\mathrm{Var}}_{a}\big[\Upsilon\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\mathop{\mathrm{Cov}}_{s,a}\big[\Upsilon, \Upsilon^t\big]\Big)$$
$$\frac{\Delta_T}{\alpha_T} = \mathop{\mathrm{Var}}_{s,a}\big[\Upsilon\big] + 2\sum_{t=1}^{T-1}\Big(\frac{T-t}{T} - \frac{t}{T+\delta T}\Big)\mathop{\mathrm{Cov}}_{s,a}\big[\Upsilon, \Upsilon^t\big]$$
$$\alpha_N = \frac{-1}{T(N^2+N)} \quad \textit{and} \quad \alpha_T = \frac{-\delta}{T+\delta T} \tag{7}$$

Derivation of Lemma 3.2 is detailed in Appendix A.2. Lemma 3.2 shows the diminishing variance reduction stemming from increasing N by 1 or T by δT. Incorporating δ captures the notion of relative costs of increasing N and T. If $\delta = 1$, then the cost of increasing N by 1 (sampling one more action per state in trajectory) is equal to doubling the trajectory length. Now, it follows that many-actions yield better variance reduction than increasing trajectory length only if $\Delta_N \leq \Delta_T$ for given values of N, T, and δ.

Theorem 3.3. *Given ergodic MDP, SPG with N action samples per T states, variance reduction stemming from increasing N by 1 is bigger than variance reduction stemming from increasing T by δT for $\delta = 1$ and $N = 1$ when:*

$$\sum_{t=1}^{T-1}\frac{t}{T}\mathop{\mathrm{Cov}}_{s,a}\big[\Upsilon, \Upsilon^t\big] \geq \mathop{\mathrm{Var}}_{s}\big[\hat{\Upsilon}\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\mathop{\mathrm{Cov}}_{s,a}\big[\hat{\Upsilon}, \hat{\Upsilon}^t\big] \tag{8}$$

For derivation with $N \geq 1$ and $\delta \in (0, \infty)$ see Equation 14 in Appendix A.3. The theorem represents a condition under which optimal to switch from regular SPG (MA-SPG with $N = 1$) to MA-SPG with $N = 2$. Surprisingly, the optimality condition for $\delta = 1$ and $N = 1$ is dependent solely on the covariance structure of the data. As follows from Theorem 3.3, MA is optimal when the weighted sum of MDP covariances exceeds the variance of the Markov Chain underlying the MDP. As follows, MA is most effective in problems where action-dependent covariance constitutes a sizeable portion of the total SPG variance (ie. problems where future outcomes largely depend on actions taken in the past and consequently, $\nabla_\theta J(s_{t+k}, a_{t+k})$ largely depends upon a_t).

Corollary 3.4. *Given ergodic MDP, SPG with N action samples per state and T states, the SPG variance reduction from increasing $\Delta N = 1$ is bigger than SPG variance reduction from $\Delta T = \delta T$ when:*

$$\frac{\mathop{\mathrm{Var}}_{s}\big[\hat{\Upsilon}\big]}{\mathop{\mathbb{E}}_{s}\mathop{\mathrm{Var}}_{a}\big[\Upsilon\big]} \leq \frac{1-\delta N}{\delta(N^2+N)} \tag{9}$$

The corollary above is a specific case of Theorem 3.3. By assuming a contextual bandit problem, the covariances are equal to zero and the optimality condition is vastly simplified. As follows from the definition of variance, the LHS of Equation 9 is greater or equal to 0. However, the RHS becomes negative when $\delta N > 1$. Since $N \geq 1$, it follows that MA is never optimal for bandits if $\delta \geq 1$ (ie. the cost of acquiring an additional action sample is equal to or greater than the cost of acquiring an additional state sample). Whereas the efficiency of MA for contextual bandits is restricted, Theorem 3.3 shows that MA can be a preferable strategy for gradient estimation in MDPs. We leave researching the optimality condition for setting with sampled Q-values or deterministic policy gradients for future work.

4. Model-Based Many-Actions SPG

Given a fixed amount of interactions with the environment, our theoretical analysis is related to two notions in on-policy SPG algorithms: achieving better quality gradients through MA via Q-network (QMA) (Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020); and achieving better quality gradients through simulating additional transitions via dynamics model in model-based SPG (MB-SPG) (Janner et al., 2019). Building on theoretical insights, we propose Model-Based Many-Actions (MBMA), an approach that bridges the two themes described above. MBMA leverages a learned dynamics model in the context of MA-SPG. As such, MBMA allows for MA estimation by calculating Q-values of additional action samples by simulating a critic-bootstrapped trajectory within a dynamics model, consisting of transition and reward networks (Ha & Schmidhuber, 2018; Hafner et al., 2019; Kaiser et al., 2019; Gelada et al., 2019; Schrittwieser et al., 2020) which we explain in Appendix E. MBMA can be used in conjunction with any on-policy SPG algorithm.

4.1. MBMA and MA-SPG

In contrast to existing implementations of MA-SPG, MBMA does not require Q-network for MA estimation. Using a Q-network to approximate additional action samples yields bias. Whereas the bias can theoretically be reduced to zero, the conditions required for such bias annihilation are unrealistic (Petit et al., 2019). Q-network learns a non-stationary target (Van Hasselt et al., 2016) that is dependent on the current policy. Furthermore, generating informative samples for multiple actions is challenging given single-action supervision. This results in unstable training when Q-network is used to bootstrap the policy gradient (Mnih et al., 2015; Van Hasselt et al., 2016; Gu et al., 2017; Haarnoja et al., 2018). The advantage of MBMA when compared to QMA is that both reward and transition networks learn stationary targets throughout training, thus offering better convergence properties and lower bias. Such bias reduction comes at the cost of additional computation. Whereas QMA approximates Q-values within a single forward calculation, MBMA sequentially unrolls the dynamics model for a fixed amount of steps.

4.2. MBMA and MB-SPG

From the perspective of model-based on-policy SPG, MBMA builds upon on-policy Model-Based Policy Optimization (MPBO) (Janner et al., 2019) but introduces the distinction between two roles for simulated transitions: whereas MBPO calculates gradient at simulated states, we propose to use information from the dynamics model by backpropagating from real states with simulated actions (i.e. simulating Q-values of those actions). As such, we define MBMA as an idea that we do not calculate gradients at simulated states, but instead use the dynamics model to refine the SPG estimator through MA variance reduction. Not calculating gradients at simulated states greatly affects the resulting SPG bias. When backpropagating SPG through simulated states, SPG is biased by two approximates: the Q-value of simulated action; and log-probability calculated at the output of the transition network. The accumulated error of state prediction anchors the gradient on log probabilities which should be associated with different states. MBPO tries to reduce the detrimental effect of compounded dynamics bias by simulating short-horizon trajectories starting from real states. In contrast to that, by calculating gradients at real states, MBMA biases the SPG only through its Q-value approximates, allowing it to omit the effects of biased log probabilities. Such perspective is supported by Lipschitz continuity analysis of approximate MDP models (Asadi et al., 2018; Gelada et al., 2019). We investigate bias stemming from strategies employed by QMA, MBMA, and MBPO in the table below. In light of the above arguments and our theoretical analysis, we hypothesize that using the dynamics model for MA estimation might yield a more favorable bias-variance tradeoff as compared to using the dynamics model to sample additional states given a fixed simulation budget.

Table 2. SPG per-parameter bias associated with action (MA) and state (MS) sample simulation. Q and $\hat{Q}$ denote the true Q-value and approximate Q-value of a given state-action pair respectively; s^* denotes the output of the transition model; and $\mathcal{K}$ denotes the Lipschitz norm of $f_s = \nabla \log \pi(a|s)$. For MS the bias is an upper bound. We include extended calculations in Appendix A.4.

	$\nabla J(s,a) - \nabla \hat{J}(s,a)$
MA $=$	$f_s(Q - \hat{Q})$
MS $\leq$	$f_s(Q - \hat{Q}) + \sqrt{(\mathcal{K}(s - \hat{s}))^2 + f_s^2(Q^2 - Q)}$

5. Experiments

5.1. Experimental Setting

We investigate the effect of bias-variance on the performance of on-policy SPG agents. We compare 4 algorithms implemented with a PPO policy: standard PPO; QMA; MBPO and MBMA. To isolate the effect of bias-variance on agents performance, we implement identical agents that differ only on two dimensions: *which* samples are simulated (ie. no simulation (PPO), state sample simulation (MBPO), action sample simulation (QMA and MBMA)); and *how* samples are simulated (ie. Q-network (QMA) as opposed to dynamics model (MBPO and MBMA)) *ceteris paribus*. We deliberately use the simulated samples only in SPG estimation. As such, the differences in performance stem solely from the bias-variance of specific SPG estimators and the resulting gradient quality. Such an experimental setup reflects the two questions posed in the Introduction:

1. By comparing MBPO-PPO and MBMA-PPO we compare variance reduction of many-actions (MBMA) as opposed to extending the trajectory length (MBPO) in the MB-SPG context and validate our theoretical contribution
2. By comparing QMA-PPO and MBMA-PPO we observe the bias accumulation resulting from simulating action with Q-network (QMA) as opposed to dynamics models (MBMA)
3. By comparing the bias-free high-variance method (PPO) to biased low-variance methods (QMA, MBPO, and MBMA) we investigate how various levels of bias-variance translate to on-policy SPG performance

Note, that we consider on-policy SPG setting. As such, we pair MBPO with an on-policy PPO agent, as opposed to an off-policy SAC agent considered in the original implementation. Algorithm 1 shows the implementation of MBMA and MBPO used in the experiments. Note that the algorithms differ only in the execution of line 8: for MBPO the simulated transitions are single X-step trajectories starting from real states (i.e. representing sampling new states); and for MBMA the simulated transitions are X single-steps starting from each on-policy state (i.e. representing sampling new actions at visited states). Below we describe the algorithms used in our experiments and discuss their bias-variance structure.

PPO Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a model-free on-policy SPG algorithm that leverages multiple actor-critic updates on a single batch of on-policy data. PPO uses a trust-region type of objective that prevents the policy to diverge too much between updates. We use PPO as the single-action agent that performs unbiased policy updates. The algorithm does not reduce the SPG variance beyond using the baseline. As such, we expect PPO variance to be the highest across the tested algorithms.

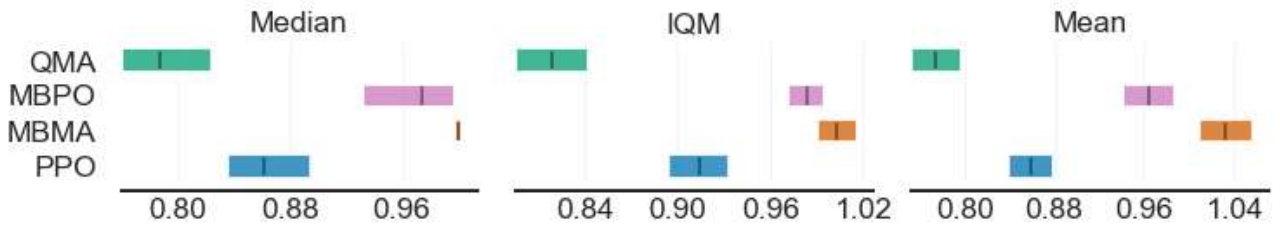
Algorithm 1 MBPO / MBMA with PPO policy

1: **Input:** batch size T, number of simulated samples X
2: Collect T on-policy transitions
3: Compute λ returns
4: Add T transitions to experience buffer
5: **for** $i = 1$ **to** (Epochs * Minibatches) **do**
6: Update dynamics model on buffer data
7: **end for**
8: Simulate $T * X$ new transitions
9: Compute λ returns for simulated transitions
10: **for** $i = 1$ **to** (Epochs * Minibatches) **do**
11: Update policy on T*(X+1) transitions
12: Update value on T transitions
13: **end for**

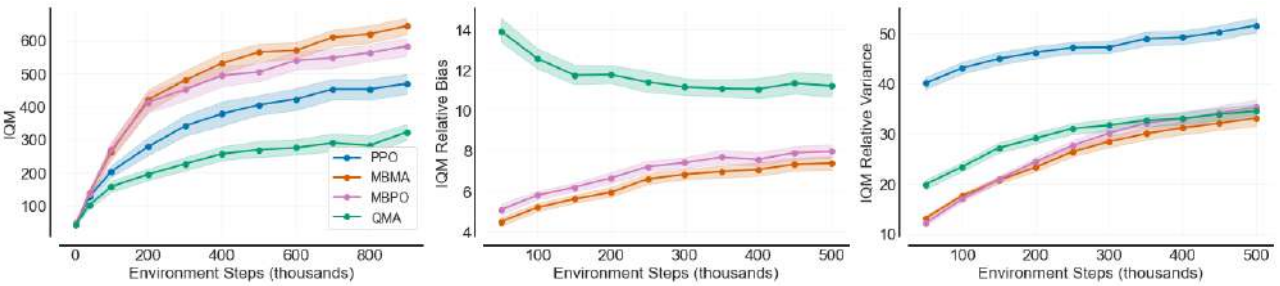
MBMA PPO that leverages the dynamics model to sample additional actions. The algorithm uses simple MLP transition and reward networks that are trained using MSE loss before performing actor updates. Similarly to QMA, the algorithm performs biased policy updates, with the bias stemming only from the dynamics model Q-value approximation error. Since the dynamics model rollouts depend on the sampled actions, the Q-value approximation has a non-zero variance.

QMA PPO that uses an auxiliary Q-network to sample additional actions for every visited state(Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020). To stabilize the training, we implement QMA-PPO using two Q-networks and choose the smaller prediction for a given state-action pair (Van Hasselt et al., 2016; Haarnoja et al., 2018). Q-networks are trained using MSE loss using TD(λ) as targets, which we found to be performing better on average than expected SARSA proposed in the literature (Petit et al., 2019; Ciosek & Whiteson, 2020). The updates performed by QMA are biased, as they depend on the output of a biased Q-network. Q-network determinism reduces the absolute variance beyond the reduction stemming from many-actions.

MBPO PPO that leverages dynamics model to perform finite horizon rollouts branching from the on-policy data (Janner et al., 2019). MBPO allows estimating SPG using a mix of real and simulated states (ie. extend the trajectory length). As such, the algorithm leverages the most common paradigm in model-based SPG - using the dynamics model to generate trajectories (Hafner et al., 2019; Kaiser et al., 2019). Similarly to MBMA, transition and reward networks are trained using MSE loss. Using dynamics model-generated trajectories for SPG updates biases the gradient



(a) Aggregate performance metrics



(b) Performance, bias and variance

Figure 2. Agent performance, bias and variance on DMC-14 (15 seeds, 95% bootstrapped C.I.). We observe that MBMA generates less bias than other methods for comparable variance reduction effects. Because agents differ only in bias-variance of their policy gradient (*ceteris paribus*), the performance differences stem solely from the beneficial bias-variance structure of the MBMA approach. Furthermore, we observe that the average bias gain of QMA overwhelms its variance reduction translating to worse performance than other algorithms.

in two ways. Firstly, similarly to QMA and MBMA, there is bias stemming from Q-value approximation. Secondly, contrary to MA methods, SPG is calculated at states simulated by the model. Due to the extended trajectory, the gradient updates have reduced variance.

We base our implementations on the PPO codebase provided by CleanRL (Huang et al., 2022b) and hyperparameters optimized for PPO Huang et al. (2022a). To accommodate more complex tasks, we increase the number of parameters in actor and critic networks across all tasks. Furthermore, we do not use advantage normalization: it has no grounding in SPG theory and can impact the variance structure of the problem at hand; but it can also adversely impact learning in certain environments (Andrychowicz et al., 2021). All algorithms use the same number of parameters in the actor and critic networks, which are updated the same number of times. QMA, MBPO, and MBMA use an equal number of additional samples (which are tuned for best performance of baselines, see Appendix D): for QMA and MBMA we use additional 8 actions per state; for MBPO we sample rollout of 8 states per state (which results in extending the trajectory 9-fold) and $TD(\lambda)$ for value estimation. We anneal the number of additional samples until 15% step of the training for all methods. Whereas learning dynamics models from images is known to work (Hafner et al., 2019; Schrittwieser et al., 2020), it is known to offer performance benefits over model-free counterparts stemming from backpropagation of additional non-sparse loss functions (Jaderberg et al., 2016; Schwarzer et al., 2020; Yarats et al., 2021b). To mitigate such benefits for algorithms using dynamics models, we use proprioceptive representations given by the environment, with transition and reward networks working on such representations. Similarly, neither MBPO nor MBMA uses an ensemble of dynamics models (Buckman et al., 2018; Kurutach et al., 2018; Janner et al., 2019). Note, that using the same number of simulated samples for all methods yields different computational costs for each method. Calculating Q-value with a dynamics model requires unrolling the model for multiple steps (forward pushes) before bootstrapping it with the critic. In contrast, QMA calculates them in a single step. Relative compute time measurements, hyperparameters, and used netwokrk architectures are detailed in Appendix B.

5.2. Agent Performance, Bias, and Variance

We compare the performance of agents on 14 DMC tasks (Tassa et al., 2018) of varying difficulty for $1M$ environments steps and 15 seeds. During this training, we measure agent performance, as well as bias and variance of policy gradients. Furthermore, to measure algorithms performance in longer training regimes, we record agent performance on 4 difficult DMC tasks (quadruped walk; quadruped run; humanoid stand; and humanoid walk) for $3M$ and $6M$ environment steps respectively. We record robust statistics (Agarwal et al., 2021) for all runs. We provide detailed

Table 3. IQM PPO normalized performance, bias gain (ie. the amount of bias gained as compared to PPO), and variance reduction (ie. the amount of variance reduced as compared to PPO) of the tested approaches. We bold the best-in-class result. 15 seeds.

	PPO NORMALIZED SCORE			BIAS GAIN			VARIANCE REDUCTION		
TASK	MBMA	MBPO	QMA	MBMA	MBPO	QMA	MBMA	MBPO	QMA
ACROBOT SWINGUP	**1.16**	1.11	0.61	**8.09**	8.71	10.6	13.0	**13.9**	13.2
BALL CATCH	**1.03**	1.02	0.98	**5.10**	5.94	12.4	28.7	**30.5**	28.9
CART SWINGUP	**1.08**	1.06	1.02	**3.23**	3.48	9.04	14.2	**14.3**	11.1
CART 2-POLES	**2.05**	1.33	1.09	**6.38**	6.80	10.1	19.0	**20.3**	10.3
CART 3-POLES	0.98	**1.26**	1.15	7.88	**7.67**	11.0	**15.4**	13.1	10.9
CHEETAH RUN	**1.82**	1.74	0.77	**11.4**	12.0	21.3	38.6	**39.5**	26.0
FINGER SPIN	0.87	0.79	**0.88**	**4.49**	4.88	9.58	**14.0**	3.81	10.5
FINGER TURN	**1.02**	0.99	0.88	**3.45**	4.05	10.1	**23.0**	18.1	20.2
POINT EASY	**1.01**	1.00	0.76	**1.36**	1.50	3.91	11.2	**11.9**	11.3
REACHER EASY	1.03	**1.04**	0.68	**3.94**	4.64	10.2	22.3	**22.8**	21.7
REACHER HARD	1.39	**1.40**	0.75	**5.29**	6.20	11.7	20.7	**21.7**	18.6
WALKER STAND	**1.03**	1.02	0.96	**9.70**	11.5	19.2	**29.8**	26.6	18.9
WALKER WALK	**1.76**	1.46	1.01	**11.5**	12.7	16.2	**37.2**	35.5	17.5
WALKER RUN	**1.67**	1.19	1.05	**11.3**	12.2	15.8	36.7	**37.5**	17.5

results, methodology for calculating bias and variance, and further experimental details in Appendix B.

We find that MBMA performs better in 14 out of 18 DMC tasks, while MBPO and PPO have better performance in 3 and 1 environments respectively. However, the performance differences are within the margin of statistical error for some cases. Note that we use hyperparameters tuned wrt. PPO and MBPO. We observe greater performance gaps benefiting MBMA for different hyperparameter settings (see Appendix D, where we compare the performance for different numbers of simulated samples and various simulation horizons).

Table 4. IQM on four complex DMC tasks (8 seeds, 1 std of the mean). $3M$ and $6M$ steps for quadruped and humanoid tasks respectively.

	PPO	MBMA	MBPO
QUAD WALK	667 ± 31	$\mathbf{677 \pm 30}$	590 ± 43
QUAD RUN	455 ± 18	$\mathbf{468 \pm 6}$	460 ± 20
HUM STAND	189 ± 8	$\mathbf{214 \pm 2}$	203 ± 31
HUM WALK	178 ± 14	$\mathbf{210 \pm 8}$	198 ± 16

In line with theory, we find that MBMA produces consistently less bias than other methods while offering greater or comparable variance reduction. On average, MBMA measures the lowest bias and lowest variance. Furthermore, we find that QMA produces smaller gradients than other methods given the same data. This points towards the low variation of the Q-network output and subsequent gradient cancellation. We find that QMA has the highest relative bias despite the MA approach. We find this unsurprising, since as noted in earlier sections, Q-networks pursue a more difficult target than dynamics models. Furthermore, even though QMA has the lowest absolute variance (due to no stochasticity in Q-value estimation), its smallest expected gradient size leads to a greater impact on its variance and thus has the highest relative variance amongst methods.

6. Related Work

6.1. Many-Actions SPG

The idea of sampling many actions per state was proposed in an unfinished preprint[1] by Sutton et al. (2001). Later, the topic was expanded upon by several authors. TRPO (Schulman et al., 2015a) 'vine procedure' uses multiple without-replacement action samples per state generated via environment rewinding. The without-replacement PG estimator was further refined by using the without-replacement samples as a free baseline (Kool et al., 2019b;a). MAC (Asadi et al., 2017) calculates the inner integral of SPG exactly (ie. sample the entire action space for given states) using Q-network, with the scheme applicable only to discrete action spaces and tested on simple environments. Similarly, Petit et al. (2019) propose to estimate the inner integral with a quadrature of N samples given by a Q-network. The authors also derive the basic theoretical properties of MA SPG. Besides expanding on the theoretical framework, Ciosek & Whiteson (2020) propose an off-policy algorithm that, given a Gaussian actor and quadratic critic, can compute the inner integral analytically.

[1] http://incompleteideas.net/papers/SSM-unpublished.pdf

6.2. Model-Based RL

ME-TRPO (Kurutach et al., 2018) leverages an ensemble of environment models to increase the sample efficiency of TRPO. WM (Ha & Schmidhuber, 2018) uses environment interactions to learn the dynamics model, with the policy learning done via evolutionary strategies inside the dynamics model. Similarly, SimPLe (Kaiser et al., 2019) learns the policy by simulating states via the dynamics model. Dreamer (Hafner et al., 2019; 2020) refines the latent dynamics model learning by proposing a sophisticated joint learning scheme for recurrent transition and discrete state representation models, but the policy learning is still done by simulating states inside the dynamics model starting from sampled off-policy transitions. Notably, Dreamer was shown to solve notoriously hard Humanoid task (Yarats et al., 2021a). Differentiable dynamics models allow for direct gradient optimization of the policy as an alternative to traditional SPG. Methods like MAAC (Clavera et al., 2019) and DDPPO (Li et al., 2022) explore policy optimization via backpropagating through the dynamics model. MuZero (Schrittwieser et al., 2020) leverages the dynamics model to perform a Monte-Carlo tree search inside the latent model. Perhaps the closest to the proposed approach is MBVE (Feinberg et al., 2018). There, an off-policy DPG agent uses the dynamics model to estimate n-step Q-values and thus refine the approximation. However, our analysis is restricted to model-based on-policy SPG and we leave the analysis of MBMA in the context of off-policy agents and backpropagating through dynamics model for future work.

7. Conclusions

In this paper, we analyzed the variance of the SPG estimator mathematically. We showed that it can be disaggregated into sub-components dependent on policy stochasticity, as well as the components which are dependent solely on the structure of the Markov process underlying the policy-embedded MDP. By optimizing such components with respect to the number of state and action samples, we derived an optimality condition that shows when MA is a preferable strategy as compared to traditional, single-action SPG. We used the result to show the difficult conditions MA has to meet to be an optimal choice for the case of contextual bandit problems. We hope that those theoretical results will reinvigorate research into MA estimation in the context of RL.

Furthermore, we discussed the bias-variance trade-off induced by using Q-network and dynamics models to simulate action or state samples. We showed that the bias associated with simulating additional states is of more complex form than the bias associated with simulating actions while offering similar variance reduction benefits. We measured the relative bias and variance of policy gradients calculated via each method and found the measurements in line with theoretical predictions, showing the analytical importance of bias and variance of SPG. We hope that those results will impact the domain of model-based on-policy SPG, where leveraging the dynamics model for trajectory simulation is the dominating approach for stochastic policy gradient.

Finally, we proposed an MBMA module - an approach that leverages dynamics models for MA estimation at the cost of additional computations. We evaluated its performance against QMA, MBPO, and PPO on-policy baselines. Our experiments showed that it compares favorably in terms of both sample efficiency and final performance in most of the tested environments. We release the code used for experiments under the following address https://github.com/naumix/On-Many-Actions-Policy-Gradient.

8. Limitations

The main limitation of our theoretical analysis is its dependence on the Markov chain Central Limit Theorem, as such its results hold only if the underlying Markov chain is ergodic. Furthermore, it is conducted in the context of on-policy SPG and its conclusions are applicable only to such settings. Following the theoretical analysis, our experiments tested only on-policy SPG algorithms. We consider expanding MA analysis to off-policy setting an interesting avenue for future research.

9. Acknowledgements

We would like to thank Witold Bednorz, Piotr Miłoś, and Łukasz Kuciński for valuable discussions and notes. Marek Cygan is cofinanced by National Centre for Research and Development as a part of EU supported Smart Growth Operational Programme 2014-2020 (POIR.01.01.01-00-0392/17-00). The experiments were performed using the Entropy cluster funded by NVIDIA, Intel, the Polish National Science Center grant UMO-2017/26/E/ST6/00622, and ERC Starting Grant TOTAL.

References

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.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. What matters in on-policy reinforcement learning? a large-scale empirical study. In *ICLR 2021-Ninth International Conference on Learning Representations*, 2021.

Asadi, K., Allen, C., Roderick, M., Mohamed, A.-r., Konidaris, G., Littman, M., and Amazon, B. U. Mean actor critic. *stat*, 1050:1, 2017.

Asadi, K., Misra, D., and Littman, M. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 264–273. PMLR, 2018.

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Bratley, P., Fox, B. L., and Schrage, L. E. *A guide to simulation*. Springer Science & Business Media, 2011.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. *Handbook of markov chain monte carlo*. CRC press, 2011.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in neural information processing systems*, 31, 2018.

Ciosek, K. and Whiteson, S. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21(2020), 2020.

Clavera, I., Fu, Y., and Abbeel, P. Model-augmented actor-critic: Backpropagating through paths. In *International Conference on Learning Representations*, 2019.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179. PMLR, 2019.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations (ICLR 2017)*, 2017.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

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

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2020.

Huang, S., Dossa, R. F. J., Raffin, A., Kanervisto, A., and Wang, W. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022a. URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and Araújo, J. G. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022b.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Jones, G. L. On the markov chain central limit theorem. *Probability surveys*, 1:299–320, 2004.

Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2019.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Kool, W., van Hoof, H., and Welling, M. Buy 4 reinforce samples, get a baseline for free! *Arxiv*, 2019a.

Kool, W., van Hoof, H., and Welling, M. Estimating gradients for discrete random variables by sampling without replacement. In *International Conference on Learning Representations*, 2019b.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

Li, C., Wang, Y., Chen, W., Liu, Y., Ma, Z.-M., and Liu, T.-Y. Gradient information matters in policy optimization by back-propagating through model. In *International Conference on Learning Representations*, 2022.

Metropolis, N. and Ulam, S. The monte carlo method. *Journal of the American statistical association*, 44(247): 335–341, 1949.

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.

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. PMLR, 2016.

Norris, J. R. and Norris, J. R. *Markov chains*. Cambridge university press, 1998.

Nota, C. and Thomas, P. S. Is the policy gradient a gradient? In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 939–947, 2020.

Peters, J. and Schaal, S. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2219–2225. IEEE, 2006.

Petit, B., Amdahl-Culleton, L., Liu, Y., Smith, J., and Bacon, P.-L. All-action policy gradient methods: A numerical integration approach. *NeurIPS 2019 Optimization Foundations of Reinforcement Learning Workshop*, 2019.

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

Ross, S. M., Kelly, J. J., Sullivan, R. J., Perry, W. J., Mercer, D., Davis, R. M., Washburn, T. D., Sager, E. V., Boyce, J. B., and Bristow, V. L. *Stochastic processes*, volume 2. Wiley New York, 1996.

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.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

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

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*, 2020.

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

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

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.

Tucker, G., Bhupatiraju, S., Gu, S., Turner, R., Ghahramani, Z., and Levine, S. The mirage of action-dependent baselines in reinforcement learning. In *International conference on machine learning*, pp. 5015–5024. PMLR, 2018.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Wu, S., Shi, L., Wang, J., and Tian, G. Understanding policy gradient algorithms: A sensitivity-based approach. In *International Conference on Machine Learning*, pp. 24131–24149. PMLR, 2022.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2021a.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10674–10681, 2021b.

A. Derivations - Variance

First, we augment the notation to encompass many action samples:

$$\Upsilon^t_{s,a^n} = \nabla J(s_t, a^n_t), \quad \Upsilon^t_{s,a} = \nabla J(s_t, a_t) \quad \text{and} \quad \Upsilon^t_s = \mathop{\mathbb{E}}_{a\sim\pi} \Upsilon^t_{s,a}$$

For convenience, throughout the Appendix we will assume finite state and action spaces. However, the same reasoning works for continuous spaces.

A.1. Derivation of Lemma 3.1

Following the MA-SPG definition outlined in Equation 3, $\text{Var}_{s,a\sim p_0^\pi,\pi}\left[\Upsilon_{s,a}\right]$ is equal to:

$$\begin{aligned}
\mathop{\text{Var}}_{s,a\sim p_0^\pi,\pi}\left[\Upsilon_{s,a}\right] &= \sum_s p_0^\pi(s) \prod_{n=1}^N \sum_{a^n} \pi(a^n|s) \left(\frac{\Upsilon_{s,a^1}}{N} + ... + \frac{\Upsilon_{s,a^N}}{N}\right)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{N}{N^2} \sum_s p_0^\pi(s) \sum_a \pi(a|s) \left(\Upsilon_{s,a}\right)^2 + \frac{2}{N^2}\binom{N}{2} \sum_s p_0^\pi(s) \left(\sum_a \pi(a|s)\,\Upsilon_{s,a}\right)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} \left(\Upsilon_{s,a}\right)^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} \left(\Upsilon_{s,a}\right)^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} \left(\Upsilon_{s,a}\right)^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{1}{N}\left(\mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} \left(\Upsilon_{s,a}\right)^2 - \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2\right) + \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \mathop{\text{Var}}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\text{Var}}_{a\sim\pi}\left[\Upsilon_{s,a}\right] \\
&= \mathop{\text{Var}}_{s_0\sim p_0^\pi}\left[\mathop{\mathbb{E}}_{a_0\sim\pi} \nabla_\theta J(s_0, a_0)\right] + \frac{1}{N} \mathop{\mathbb{E}}_{s_0\sim p_0^\pi} \mathop{\text{Var}}_{a_0\sim\pi}\left[\nabla J(s_0, a_0)\right]
\end{aligned} \tag{10}$$

The above result for $N = 1$ is reported in (Petit et al., 2019), noting it as stemming from the law of total variance. However, we could not find the proof in the existing literature. Below, $p_t^\pi(s_t|s_0, a_0^1)$ denotes the t step transition kernel conditioned on s_0 and a_0^1 (ie. the first sampled action in s_0).

$$\begin{aligned}
&\mathbb{E}\left[\Upsilon_{s,a}\Upsilon^t_{s,a}\right] = \\
&= \sum_{s_0} p_0^\pi(s_0) \prod_{n=1}^N \sum_{a_0^n} \pi(a_0^n|s_0) \sum_{s_t} p_t^\pi(s_t|s_0, a_0^1) \prod_{m=1}^N \sum_{a_t^m} \pi(a_t^m|s_t) \left(\frac{\Upsilon_{s,a^1}}{N} + ... + \frac{\Upsilon_{s,a^N}}{N}\right)\left(\frac{\Upsilon^t_{s,a^1}}{N} + ... + \frac{\Upsilon^t_{s,a^N}}{N}\right) \\
&= \sum_{s_0} p_0^\pi(s_0) \prod_{n=1}^N \sum_{a_0^n} \pi(a_0^n|s_0) \sum_{s_t} p_t^\pi(s_t|s_0, a_0^1) \prod_{m=1}^N \sum_{a_t^m} \pi(a_t^m|s_t) \left(\sum_{i=1}^N \sum_{j=1}^N \frac{\Upsilon_{s,a^i}}{N} \frac{\Upsilon^t_{s,a^j}}{N}\right)
\end{aligned}$$

Therefore:

$$
\begin{aligned}
\mathbb{E}\big[\Upsilon_{s,a}\Upsilon^t_{s,a}\big] = &= \frac{1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a^1_0}\pi(a^1_0)\,\Upsilon_{s,a^1_0}\prod_{n=2}^{N}\sum_{a^n_0}\pi(a^n_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a^1_0)\sum_{a_t}\pi(a_t|s_t)\,\Upsilon^t_{s,a} \\
&+ \frac{N-1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a^2_0}\pi(a^2_0)\,\Upsilon_{s,a^2_0}\sum_{a^1_0}\pi(a^1_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a^1_0)\sum_{a_t}\pi(a_t|s_t)\,\Upsilon^t_{s,a} \\
&= \frac{1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a^1_0}\pi(a^1_0)\,\Upsilon_{s,a^1_0}\sum_{s_t}p^\pi_t(s_t|s_0,a^1_0)\,\Upsilon^t_s \\
&+ \frac{N-1}{N}\sum_{s_0} p^\pi_0(s_0)\,\Upsilon_s\sum_{a^1_0}\pi(a^1_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a^1_0)\,\Upsilon^t_s
\end{aligned}
$$

Thus, the t^{th} covariance of MA is equal to:

$$
\begin{aligned}
&\operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\Upsilon_{s,a},\Upsilon^t_{s,a}\big] = \\
&= \frac{1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a_0}\pi(a_0)\,\Upsilon_{s,a}\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s \\
&+ \frac{N-1}{N}\sum_{s_0} p^\pi_0(s_0)\,\Upsilon_s\sum_{a_0}\pi(a_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s \\
&- \Big(\sum_{s_0} p^\pi_0(s_0)\sum_{a_0}\pi(a_0)\,\Upsilon_{s,a}\Big)\Big(\sum_{s_t} p^\pi_t(s_t)\sum_{a_t}\pi(a_t)\,\Upsilon^t_{s,a}\Big) \\
&= \frac{1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a_0}\pi(a_0)\,\Upsilon_{s,a}\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s \\
&+ \frac{N-1}{N}\sum_{s_0} p^\pi_0(s_0)\,\Upsilon_s\sum_{a_0}\pi(a_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s - \big(\mathop{\mathbb{E}}_{a\sim\pi}\Upsilon_{s,a}\big)\big(\mathop{\mathbb{E}}_{a\sim\pi}\Upsilon^t_{s,a}\big) \\
&= \frac{1}{N}\mathop{\mathbb{E}}_{s_0\sim p^\pi_0}\Big(\sum_{a_0}\pi(a_0)\,\Upsilon^0_{s,a}\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s - \Upsilon^0_s\sum_{a_0}\pi(a_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s\Big) \\
&+ \Big(\sum_{s_0} p^\pi_0(s_0)\,\Upsilon^0_s\sum_{a_0}\pi(a_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s - \big(\mathop{\mathbb{E}}_{a\sim\pi}\Upsilon_{s,a}\big)\big(\mathop{\mathbb{E}}_{a\sim\pi}\Upsilon^t_{s,a}\big)\Big) \\
&= \operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\Upsilon_s,\Upsilon^t_s\big] + \frac{1}{N}\mathop{\mathbb{E}}_{s_0\sim p^\pi_0}\operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\Upsilon_{s,a},\Upsilon^t_{s,a}\big] \\
&= \operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\mathop{\mathbb{E}}_{a_0\sim\pi}\nabla_\theta J(s_0,a_0),\mathop{\mathbb{E}}_{a_0\sim\pi}\nabla_\theta J(s_t,a_t)\big] + \frac{1}{N}\mathop{\mathbb{E}}_{s_0\sim p^\pi_0}\operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\nabla J(s_0,a_0),\nabla J(s_t,a_t)\big]
\end{aligned}
\tag{11}
$$

Combining Equations 10 and 11 concludes derivation of Lemma 3.1.

A.2. Derivation of Lemma 3.2

Since N is defined to be a natural number, we calculate the variance reduction effect stemming from increasing N via the forward difference operator:

$$\Delta_N = \mathbf{V}(N+1) - \mathbf{V}(N)$$

We also use the shorthand notation:

$$\alpha_e^t = \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\big[\Upsilon_s,\Upsilon_s^t\big],\quad \alpha^t = \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi|s_0}\big[\Upsilon_{s,a},\Upsilon_{s,a}^t\big]\quad\text{and}\quad \mathrm{C}^t = \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\big[\Upsilon_{s,a},\Upsilon_{s,a}^t\big] = \alpha_e^t + \frac{1}{N}\operatorname*{\mathbb{E}}_{s\sim p_0^\pi}\alpha^t$$

Thus:

$$\mathbf{V} = \frac{1}{T}\bigg(\operatorname*{Var}_{s_0\sim p_0^\pi}\big[\Upsilon_s\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_e^t + \frac{1}{N}\operatorname*{\mathbb{E}}_{s_0\sim p_0^\pi}\big(\operatorname*{Var}_{a\sim\pi}\big[\Upsilon_{s,a}^0\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\big)\bigg)$$

We proceed with the calculation of the forward difference:

$$\begin{aligned}
\Delta_N &= \frac{1}{T}\bigg(\operatorname*{Var}_{s\sim p_0^\pi}\big[\Upsilon_s\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_e^t + \frac{1}{N+1}\operatorname*{\mathbb{E}}_{s\sim p_0^\pi}\big(\operatorname*{Var}_{a\sim\pi}\big[\Upsilon_{s,a}\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\big)\bigg)\\
&- \frac{1}{T}\bigg(\operatorname*{Var}_{s\sim p_0^\pi}\big[\Upsilon_s\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_e^t + \frac{1}{N}\operatorname*{\mathbb{E}}_{s\sim p_0^\pi}\big(\operatorname*{Var}_{a\sim\pi}\big[\Upsilon_{s,a}\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\big)\bigg)\\
&= \frac{1}{T(N+1)}\operatorname*{\mathbb{E}}_{s\sim p_0^\pi}\bigg(\operatorname*{Var}_{a\sim\pi}\big[\Upsilon_{s,a}\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\bigg)\\
&- \frac{1}{T\,N}\operatorname*{\mathbb{E}}_{s\sim p_0^\pi}\bigg(\operatorname*{Var}_{a\sim\pi}\big[\Upsilon_{s,a}\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\bigg)\\
&= \frac{-1}{T(N^2+N)}\operatorname*{\mathbb{E}}_{s\sim p_0^\pi}\bigg(\operatorname*{Var}_{a\sim\pi}\big[\Upsilon_{s,a}\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\bigg)\\
&= \frac{-1}{T(N^2+N)}\operatorname*{\mathbb{E}}_{s\sim p_0^\pi}\bigg(\operatorname*{Var}_{a\sim\pi}\big[\Upsilon_{s,a}\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi|s_0}\big[\Upsilon_{s,a}^0,\Upsilon_{s,a}^t\big]\bigg)\\
&= \frac{-1}{T(N^2+N)}\operatorname*{\mathbb{E}}_{s_0\sim p_0^\pi}\bigg(\operatorname*{Var}_{a\sim\pi}\big[\nabla J(s_0,a_0)\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi|s_0}\big[\nabla J(s_0,a_0),\nabla J(s_t,a_t)\big]\bigg)
\end{aligned}\tag{12}$$

Similarly, we calculate Δ_T:

$$\begin{aligned}
\Delta_T &= \frac{1}{T+\delta T}\operatorname*{Var}_{s,a\sim p_0^\pi,\pi}[\Upsilon_{s,a}] + 2\sum_{t=1}^{T+\delta T-1}\frac{T+\delta T-t}{(T+\delta T)^2}\,\mathrm{C}^t - \frac{1}{T}\operatorname*{Var}_{s,a\sim p_0^\pi,\pi}\big[\Upsilon_{s,a}\big] - 2\sum_{t=1}^{T-1}\frac{T-t}{T^2}\,\mathrm{C}^t\\
&= \frac{-\delta T}{T+\delta T}\operatorname*{Var}_{s,a\sim p_0^\pi,\pi}\big[\Upsilon_{s,a}\big] + 2\sum_{t=1}^{T-1}\bigg(\frac{T+\delta T-t}{(T+\delta T)^2} - \frac{T-t}{T^2}\bigg)\,\mathrm{C}^t + 2\sum_{k=T}^{T+\delta T-1}\frac{T-t}{(T+\delta T)^2}\,\mathrm{C}^t\\
&= \frac{-\delta}{T+\delta T}\bigg(\operatorname*{Var}_{s,a\sim p_0^\pi,\pi}[\Upsilon_{s,a}] + 2\sum_{t=1}^{T-1}\big(\frac{T-t}{T} - \frac{t}{T+\delta T}\big)\,\mathrm{C}^t - \frac{2}{\delta}\sum_{k=T}^{T+\delta T-1}\frac{T+\delta T-k}{T+\delta T}\,\mathrm{C}^k\bigg)
\end{aligned}$$

Now, we assume that the trajectory length guarantees reaching a regenerative state, and thus $\sum_{k=T}^{T+\delta T-1}\frac{T+\delta T-k}{T+\delta T}\,\mathrm{C}^k = 0$:

$$\begin{aligned}
\Delta_T &= \frac{-\delta}{T+\delta T}\bigg(\operatorname*{Var}_{s,a\sim p_0^\pi,\pi}[\Upsilon_{s,a}] + 2\sum_{t=1}^{T-1}\big(\frac{T-t}{T} - \frac{t}{T+\delta T}\big)\,\mathrm{C}^t\bigg)\\
&= \frac{-\delta}{T+\delta T}\bigg(\operatorname*{Var}_{s,a\sim p_0^\pi,\pi}[\Upsilon_{s,a}] + 2\sum_{t=1}^{T-1}\big(\frac{T-t}{T} - \frac{t}{T+\delta T}\big)\operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\big[\Upsilon_{s,a}^0,\Upsilon_{s,a}^t\big]\bigg)
\end{aligned}\tag{13}$$

Combining Equations 12 and 13 concludes derivation of Lemma 3.2.

A.3. Derivation of Theorem 3.3

We start the derivation by stating that MA-SPG is advantageous in terms of variance reduction as compared to increased trajectory length SPG when $-\Delta_N \geq -\Delta_T$. As such:

$$\frac{1+\delta}{\delta(N^2+N)} \mathop{\mathbb{E}}_{s\sim p_0^\pi}\left(\mathop{\text{Var}}_{a\sim\pi}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\right) \geq \mathop{\text{Var}}_{s,a\sim p_0^\pi,\pi}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\left(\frac{T-t}{T} - \frac{t}{T+\delta T}\right)\mathrm{C}^t$$

We use Equations 10 and 11 to expand the RHS:

$$\begin{aligned}
&\mathop{\text{Var}}_{s,a\sim p_0^\pi,\pi}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\left(\frac{T-t}{T} - \frac{t}{T+\delta T}\right)\mathrm{C}^t = \\
&= \mathop{\text{Var}}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + \frac{1}{N}\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\left(\frac{T-t}{T} - \frac{t}{T+\delta T}\right)\left(\alpha_e^t + \frac{1}{N}\,\alpha^t\right)
\end{aligned}$$

We move all terms dependent on the policy to the LHS:

$$\begin{aligned}
&\frac{1-\delta N}{\delta(N^2+N)}\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\left(\frac{(1+\delta-\delta N-\delta^2 N)T-(1-2\delta N-\delta^2 N)t}{(\delta T+\delta^2 T)(N^2+N)}\right)\alpha^t \geq \\
&\mathop{\text{Var}}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1}\left(\frac{T-t}{T} - \frac{t}{T+\delta T}\right)\alpha_e^t
\end{aligned} \tag{14}$$

Now, in order to recover the Corollary 9, we assume a contextual bandit setup (ie. $p^\pi(s'|s) = p^\pi(s')$). Then:

$$\frac{1-\delta N}{\delta(N^2+N)}\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}\left[\Upsilon_{s,a}\right] \geq \mathop{\text{Var}}_{s\sim p_0^\pi}\left[\Upsilon_s\right]$$

Which is equivalent to:

$$\frac{\mathop{\text{Var}}_{s\sim p_0^\pi}\left[\Upsilon_s\right]}{\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}\left[\Upsilon_{s,a}\right]} \leq \frac{1-\delta N}{\delta(N^2+N)}$$

We proceed with the derivation for the MDP setup, where $p^\pi(s'|s) \neq p^\pi(s')$. We write $N = 1$, which implies that we start in the regular single-action SPG setup. Furthermore, we assume $\delta = 1$, which according to the setup implies equal cost of sampling additional action and state samples. Thus, Equation 14 simplifies to:

$$
\begin{aligned}
&\sum_{t=1}^{T-1} \frac{t}{T}\,\alpha^t \geq \operatorname*{Var}_{s\sim p_0^\pi}\big[\Upsilon_s\big] + \sum_{t=1}^{T-1} \frac{2T-3t}{T}\,\alpha_e^t \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T}\,\alpha^t \geq \operatorname*{Var}_{s\sim p_0^\pi}\big[\Upsilon_s\big] + 2\sum_{t=1}^{T-1} \frac{T-t}{T}\,\alpha_e^t - \sum_{t=1}^{T-1} \frac{t}{T}\,\alpha_e^t \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T}\,\big(\alpha^t + \alpha_e^t\big) \geq \operatorname*{Var}_{s\sim p_0^\pi}\big[\Upsilon_s\big] + 2\sum_{t=1}^{T-1} \frac{T-t}{T}\,\alpha_e^t \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T}\,\mathrm{C}_t \geq \operatorname*{Var}_{s\sim p_0^\pi}\big[\Upsilon_s\big] + 2\sum_{t=1}^{T-1} \frac{T-t}{T}\,\alpha_e^t \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T} \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\big[\Upsilon_{s,a}, \Upsilon_{s,a}^t\big] \geq \operatorname*{Var}_{s\sim p_0^\pi}\big[\Upsilon_s\big] + 2\sum_{t=1}^{T-1} \frac{T-t}{T} \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\big[\Upsilon_s, \Upsilon_s^t\big] \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T} \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\big[\Upsilon_{s,a}, \Upsilon_{s,a}^t\big] \geq \operatorname*{Var}_{s\sim p_0^\pi}\big[\Upsilon_s\big] + 2\sum_{t=1}^{T-1} \frac{T-t}{T} \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\big[\Upsilon_s, \Upsilon_s^t\big]
\end{aligned}
\tag{15}
$$

Which concludes the derivation of Theorem 3.3.

A.4. Derivations - Bias

First, we calculate the bias associated with MA (ie. MBMA and QMA), which stems from approximated state-action Q-value. We denote the approximated Q-value as $\hat{Q}^\pi(s,a)$ and write:

$$
\begin{aligned}
\text{bias}^{MA} &= \nabla J(s,a) - \nabla \hat{J}(s,a) \\
&= \nabla \log \pi(a|s)\, Q^\pi(s,a) - \nabla \log \pi(a|s)\, \hat{Q}^\pi(s,a) \\
&= \nabla \log \pi(a|s) \big(Q^\pi(s,a) - \hat{Q}^\pi(s,a)\big)
\end{aligned}
\tag{16}
$$

Furthermore, we calculate the bias associated with using dynamics models to simulate state samples. Firstly, we denote the result of a n-step transition via the dynamics model as s^*, such that the absolute difference between true transition and dynamics model transition is equal to $|s - s^*|$. Furthermore, we denote the Lipschitz norm of $\nabla \log \pi(a|s)$ as $\mathcal{K}$. As such, it follows that:

$$
|\nabla \log \pi(a|s) - \nabla \log \pi(a|s^*)| \leq \mathcal{K}|s - s^*|
$$

We write the bias:

$$
\begin{aligned}
\text{bias}^{MS} &= \nabla J(s,a) - \nabla \hat{J}(s,a) \\
&= \nabla \log \pi(a|s)\, Q^\pi(s,a) - \nabla \log \pi(a|s^*)\, \hat{Q}^\pi(s^*,a) \\
&= \big(\nabla \log \pi(a|s) - \nabla \log \pi(a|s^*)\big)\hat{Q}^\pi(s^*,a) + \nabla \log \pi(a|s)\big(Q^\pi(s,a) - \hat{Q}^\pi(s^*,a)\big)
\end{aligned}
$$

We use the Lipschitz continuity:

$$
\left| \frac{\text{bias}^{MS} - \nabla \log \pi(a|s)\big(Q^\pi(s,a) - \hat{Q}^\pi(s^*,a)\big)}{\hat{Q}^\pi(s^*,a)} \right| \leq \mathcal{K}|s - s^*|
$$

Where we assume that $\hat{Q}^\pi(s^*,a) \neq 0$. Squaring both sides leads to the solution:

$$\text{bias}^{MS} \geq \nabla \log \pi(a|s)\big(Q^{\pi}(s,a) - \hat{Q}^{\pi}(s,a)\big) - \sqrt{\nabla \log \pi(a|s)^2\big(Q^{\pi}(s,a)^2 - Q^{\pi}(s,a)\big) + \big(\mathcal{K}(s-s^*)\big)^2}$$

And: (17)

$$\text{bias}^{MS} \leq \nabla \log \pi(a|s)\big(Q^{\pi}(s,a) - \hat{Q}^{\pi}(s,a)\big) + \sqrt{\nabla \log \pi(a|s)^2\big(Q^{\pi}(s,a)^2 - Q^{\pi}(s,a)\big) + \big(\mathcal{K}(s-s^*)\big)^2}$$

Which concludes the derivation.

B. Experimental Details

B.1. Setting

Figure 1 We use the OpenAI gym CartPole environment. We define solving the environment as reaching an average of 190 rewards during 25 evaluations. We perform policy evaluations every 50 environment steps. If the trajectory length is shorter than environment termination we bootstrap the Q-value with critic. To sample more actions per state we perform environment rewinding. Similarly to regular actions, the Q-values of additional action samples are bootstrapped via critic when reaching the trajectory length. Note that CartPole environment has only two actions, as such there is minimal variance associated with the policy. We smoothen the results with Savitsky-Golay filter and use 45 random seeds.

Table 1 We use a subset of environments from DM Control Suite. We marginalize Q-values by performing 100 rollouts for every state-action pair. We get 125000 on-policy states, with one additional action per state. We use Equation 3 and Lemma 3.1 to isolate the variance components. Note that Q-value marginalization is required by Lemma 3.1. Note that if Q-values are stochastic, we observe more variance reduction stemming from sampling additional actions than expected. The performance of agents was measured during 500000 environment steps, with an average performance recorded in 122 different episodes. Additional action sample is drawn from the environments (via environment rewinding). To reduce the compute load used in the experiment, the performance is measured without Q-value marginalization. We use 10 random seeds.

Table 3 To measure performance we first average across random seeds and take the maximum. We normalize by dividing each seed by maximum best performing PPO seed. To measure bias and variance, we record 125 gradient estimates for every method during 10 points in training for 15 random seeds. Each of 125 gradient estimates is calculated using a batch size of 2500 states. The gradients are always calculated wrt. the same policy. To this end, there is one agent gathering the data and serving as the policy for all methods. The recorded gradients stemming from all methods are never applied to the actor network (ie. using one agent per random seed). We denote $*$ as the tested method and P as the total number of parameters in the model. We calculate relative bias with the following equation:

$$\text{Bias}^* = \frac{1}{P}\sum_{p}^{P} \frac{|\nabla J_p^* - \nabla J_p^{AC}|}{|\nabla J_p^*|}$$

Where ∇J_p^* and ∇J_p^{AC} denote the gradient wrt. p^{th} parameter calculated via the tested method and actor-critic respectively (averaged over 125 gradient examples). As such, at each testing point, we calculate the absolute difference between the 'oracle' AC gradient (which is unbiased) and the respective method average. Furthermore, we calculate relative variance via:

$$\text{Var}^* = \frac{1}{P}\sum_{p}^{P} \frac{\underset{\tau}{\text{Var}}\big[\nabla J_p^*\big]}{(\nabla J_p^*)^2}$$

Where $\text{Var}_{\tau}\big[\nabla J_p^*\big]$ denotes the p^{th} unit of the diagonal of variance-covariance matrix calculated over 125 gradient examples. Dividing bias and variance by the size of the gradient allows us to inspect the relative size (ie. if the gradient is small then bias and variance might also be small, but big in comparison to the gradient that we are looking for).

B.2. Hyperparameters

Below, we provide a detailed list of hyperparameter settings used to generate results presented in Table 3.

HYPERPARAMETER	PPO	QMA	MBMA	MBPO
ACTION REPEAT	4	4	4	4
ACTOR OPTIMIZER	ADAM	ADAM	ADAM	ADAM
CRITIC OPTIMIZER	ADAM	ADAM	ADAM	ADAM
DYNAMICS OPTIMIZER	—	—	ADAM	ADAM
Q-NET OPTIMIZER	—	ADAM	—	—
ACTOR LEARNING RATE	$3e-4$	$3e-4$	$3e-4$	$3e-4$
CRITIC LEARNING RATE	$3e-4$	$3e-4$	$3e-4$	$3e-4$
DYNAMICS LEARNING RATE	—	—	$3e-4$	$3e-4$
Q-NET LEARNING RATE	—	$3e-4$	—	—
ACTOR OPTIMIZER EPSILON	$1e-5$	$1e-5$	$1e-5$	$1e-5$
CRITIC OPTIMIZER EPSILON	$1e-5$	$1e-5$	$1e-5$	$1e-5$
DYNAMICS OPTIMIZER EPSILON	—	—	$1e-5$	$1e-5$
Q-NET OPTIMIZER EPSILON	—	$1e-5$	—	—
ACTOR HIDDEN LAYER SIZE	512	512	512	512
CRITIC HIDDEN LAYER SIZE	1024	1024	1024	1024
DYNAMICS HIDDEN LAYER SIZE	—	—	1024	1024
Q-NETWORK HIDDEN LAYER SIZE	—	1024	—	—
λ	0.95	0.95	0.95	0.95
DISCOUNT RATE	0.99	0.99	0.99	0.99
BATCH SIZE (T)	2048	2048	2048	2048
MINIBATCH SIZE	64	64	64	64
PPO EPOCHS	10	10	10	10
DYNAMICS BUFFER SIZE	—	—	25000	25000
DYNAMICS BATCH SIZE	—	—	128	128
NUMBER OF SIMULATED ACTIONS PER STATE (T*)	—	8	8	—
NUMBER OF SIMULATED STATES PER STATE (N)	—	—	—	8
SIMULATION HORIZON	—	—	12	12
CLIP COEFFICIENT	0.2	0.2	0.2	0.2
MAXIMUM GRADIENT NORM	0.5	0.5	0.5	0.5
VALUE COEFFICIENT	0.5	0.5	0.5	0.5
QUADRUPED AND HUMANOID				
BATCH SIZE (T)	4096	NA	4096	4096
MINIBATCH SIZE	128	NA	128	128
DYNAMICS BUFFER SIZE	—	NA	2000000	2000000
DYNAMICS BATCH SIZE	—	NA	256	256

B.3. Computational Costs

Below, we report the relative computational costs associated with each SPG update type. Note, that code optimization and parallelization would increase the relative performance of QMA and MBMA.

NUMBER OF ACTIONS	PPO	QMA	MBMA
4	1.00	1.22	1.59
8	1.00	1.62	2.31
16	1.00	2.07	3.60

B.4. Unnormalized Results

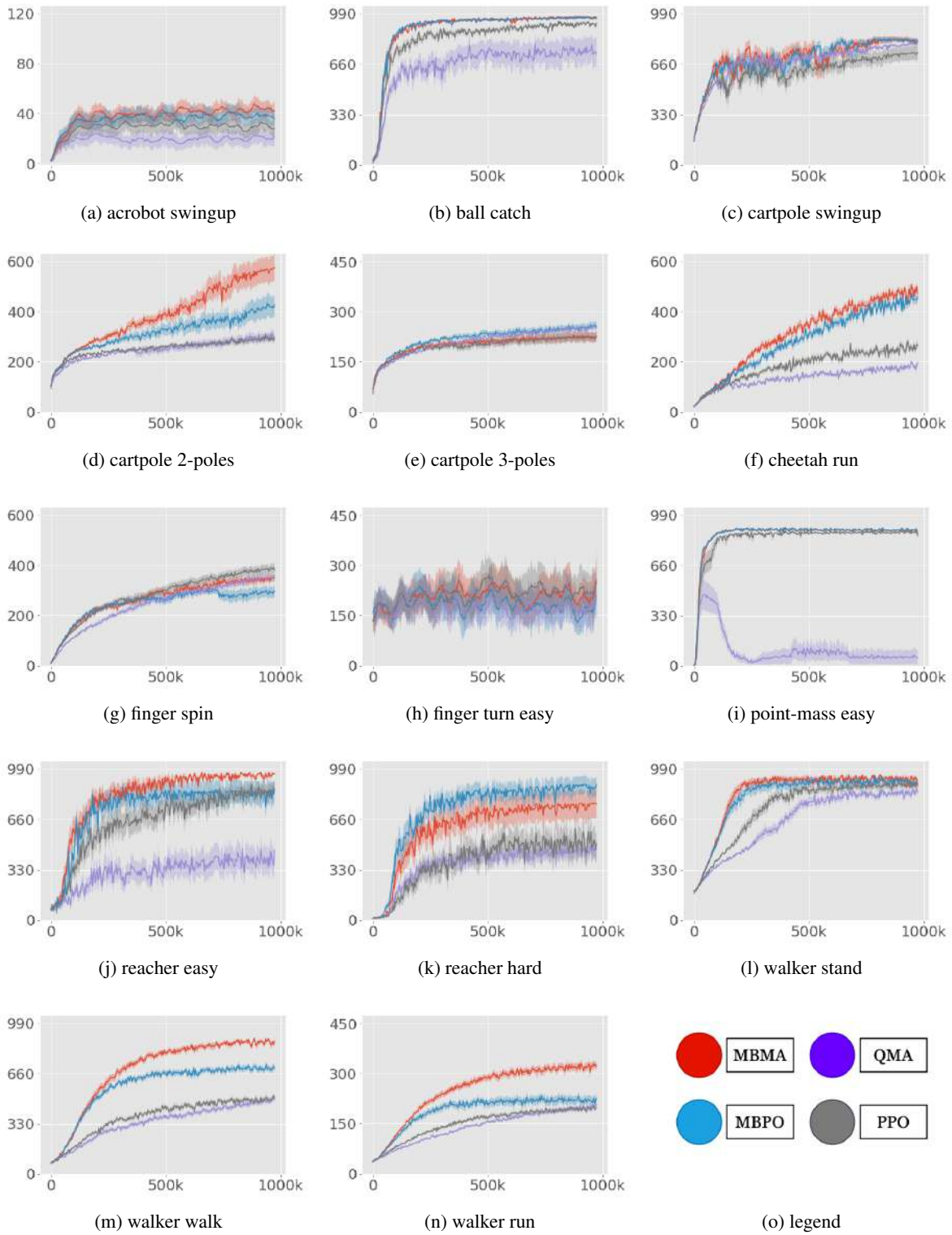
We provide a table of unnormalized results for the performance experiment:

TASK	PPO	MBMA	MBPO	QMA
ACRO SWINGUP	$47 \pm 10\,(32 \pm 7)$	$\mathbf{59 \pm 6}\,(\mathbf{40 \pm 6})$	$56 \pm 8\,(31 \pm 7)$	$34 \pm 10\,(17 \pm 7)$
BALL CATCH	$948 \pm 8\,(831 \pm 20)$	$\mathbf{974 \pm 3}\,(\mathbf{898 \pm 8})$	$969 \pm 2\,(888 \pm 8)$	$934 \pm 5\,(770 \pm 30)$
CART SWINGUP	$736 \pm 46\,(617 \pm 46)$	$\mathbf{828 \pm 16}\,(\mathbf{707 \pm 44})$	$825 \pm 13\,(702 \pm 38)$	$802 \pm 2\,(677 \pm 18)$
CART 2-POLE	$308 \pm 16\,(253 \pm 8)$	$\mathbf{575 \pm 52}\,(\mathbf{388 \pm 27})$	$435 \pm 48\,(317 \pm 22)$	$315 \pm 22\,(248 \pm 12)$
CART 3-POLE	$229 \pm 15\,(199 \pm 10)$	$229 \pm 14\,(202 \pm 10)$	$261 \pm 6\,(\mathbf{221 \pm 9})$	$\mathbf{262 \pm 5}\,(211 \pm 4)$
CHEETAH RUN	$283 \pm 12\,(185 \pm 10)$	$\mathbf{507 \pm 14}\,(\mathbf{316 \pm 14})$	$473 \pm 17\,(284 \pm 14)$	$201 \pm 8\,(135 \pm 7)$
FINGER SPIN	$\mathbf{391 \pm 21}\,(\mathbf{280 \pm 14})$	$350 \pm 14\,(266 \pm 12)$	$305 \pm 16\,(248 \pm 14)$	$359 \pm 15\,(245 \pm 12)$
FINGER TURN	$\mathbf{396 \pm 67}\,(\mathbf{213 \pm 54})$	$368 \pm 59\,(206 \pm 52)$	$318 \pm 73\,(184 \pm 51)$	$296 \pm 75\,(176 \pm 50)$
POINT EASY	$895 \pm 6\,(839 \pm 13)$	$\mathbf{910 \pm 5}\,(866 \pm 7)$	$909 \pm 6\,(\mathbf{867 \pm 7})$	$467 \pm 97\,(106 \pm 50)$
REACHER EASY	$885 \pm 44\,(649 \pm 66)$	$\mathbf{968 \pm 2}\,(\mathbf{815 \pm 39})$	$854 \pm 71\,(729 \pm 74)$	$472 \pm 27\,(316 \pm 72)$
REACHER HARD	$601 \pm 103\,(385 \pm 78)$	$767 \pm 96\,(606 \pm 84)$	$\mathbf{892 \pm 61}\,(\mathbf{722 \pm 61})$	$488 \pm 53\,(361 \pm 47)$
WALKER STAND	$914 \pm 22\,(737 \pm 24)$	$\mathbf{955 \pm 5}\,(\mathbf{839 \pm 22})$	$944 \pm 8\,(815 \pm 29)$	$854 \pm 20\,(654 \pm 21)$
WALKER WALK	$514 \pm 14\,(377 \pm 14)$	$\mathbf{892 \pm 9}\,(\mathbf{686 \pm 18})$	$720 \pm 19\,(576 \pm 19)$	$500 \pm 17\,(340 \pm 14)$
WALKER RUN	$203 \pm 7\,(152 \pm 5)$	$\mathbf{331 \pm 13}\,(\mathbf{251 \pm 9})$	$233 \pm 12\,(190 \pm 11)$	$208 \pm 7\,(141 \pm 4)$

And for the bias-variance experiment:

	RELATIVE BIAS			RELATIVE VARIANCE			
TASK	MBMA	MBPO	QMA	AC	MBMA	MBPO	QMA
ACRO SWINGUP	$\mathbf{8.25 \pm 0.3}$	8.66 ± 0.3	10.5 ± 0.5	44.3 ± 1.5	31.6 ± 1.1	30.8 ± 1.0	$\mathbf{27.3 \pm 1.1}$
BALL CATCH	$\mathbf{5.28 \pm 0.3}$	6.04 ± 0.3	12.1 ± 0.8	48.7 ± 1.4	20.0 ± 0.9	$\mathbf{18.0 \pm 0.7}$	18.8 ± 1.1
CART SWINGUP	$\mathbf{3.16 \pm 0.3}$	3.46 ± 0.3	8.89 ± 1.1	21.2 ± 1.5	8.08 ± 0.6	$\mathbf{7.84 \pm 0.6}$	11.1 ± 1.1
CART 2-POLE	$\mathbf{6.32 \pm 0.4}$	6.79 ± 0.5	10.6 ± 0.7	39.8 ± 1.8	21.6 ± 1.6	$\mathbf{20.5 \pm 1.6}$	29.8 ± 1.4
CART 3-POLE	7.48 ± 0.7	$\mathbf{7.29 \pm 0.6}$	11.0 ± 0.7	43.0 ± 2.7	$\mathbf{27.7 \pm 2.7}$	29.6 ± 2.8	32.4 ± 2.3
CHEETAH RUN	$\mathbf{11.5 \pm 0.4}$	12.0 ± 0.4	21.4 ± 0.8	77.1 ± 2.5	39.0 ± 1.1	$\mathbf{37.7 \pm 1.1}$	52.9 ± 1.7
FINGER SPIN	$\mathbf{4.50 \pm 0.3}$	4.91 ± 0.4	9.60 ± 0.4	38.8 ± 1.1	$\mathbf{24.2 \pm 1.3}$	33.6 ± 2.0	28.2 ± 0.9
FINGER TURN	$\mathbf{3.55 \pm 0.4}$	3.94 ± 0.4	11.3 ± 0.8	51.0 ± 2.8	$\mathbf{25.4 \pm 1.8}$	30.9 ± 2.7	30.4 ± 2.1
POINT EASY	$\mathbf{1.33 \pm 0.1}$	1.49 ± 0.1	3.57 ± 0.6	15.8 ± 1.4	4.40 ± 0.4	$\mathbf{3.84 \pm 0.3}$	4.08 ± 0.5
REACHER EASY	$\mathbf{4.07 \pm 0.2}$	4.85 ± 0.3	10.3 ± 1.0	36.2 ± 1.8	14.3 ± 0.7	$\mathbf{13.8 \pm 0.7}$	15.6 ± 1.4
REACHER HARD	$\mathbf{5.70 \pm 0.3}$	6.58 ± 0.4	13.0 ± 0.7	41.6 ± 1.4	21.1 ± 1.2	$\mathbf{20.0 \pm 1.1}$	23.1 ± 1.4
WALKER STAND	$\mathbf{9.77 \pm 0.4}$	11.4 ± 0.5	18.8 ± 0.8	71.2 ± 2.0	$\mathbf{39.9 \pm 1.2}$	43.0 ± 1.4	51.5 ± 2.1
WALKER WALK	$\mathbf{11.1 \pm 0.4}$	12.3 ± 0.4	16.6 ± 0.8	76.7 ± 1.8	$\mathbf{40.7 \pm 1.2}$	42.0 ± 1.2	59.4 ± 1.1
WALKER RUN	$\mathbf{11.1 \pm 0.4}$	12.1 ± 0.4	15.7 ± 0.8	77.9 ± 1.8	41.3 ± 1.3	$\mathbf{40.7 \pm 1.0}$	59.5 ± 1.6

C. Learning Curves



(a) acrobot swingup (b) ball catch (c) cartpole swingup

(d) cartpole 2-poles (e) cartpole 3-poles (f) cheetah run

(g) finger spin (h) finger turn easy (i) point-mass easy

(j) reacher easy (k) reacher hard (l) walker stand

(m) walker walk (n) walker run (o) legend

Figure 3. Learning curves corresponding to results from Table 3. The shaded region denotes one standard deviation of the mean. 15 seeds.

D. Ablations

We investigate the performance of MBMA and MBPO for different N and T (amount of simulated samples and simulation horizon). We record the average final performance during training of $500k$ steps. First, we investigate the effects of N. The table below shows the mean performance for 5 DMC tasks and various amount of simulated data (10 seeds):

METHOD	N = 8	N = 16	N = 32	N = 64
QMA	**551**	390	303	227
MBPO	**636**	624	570	594
MBMA	683	641	679	**707**

As the table below shows, MBMA is much more robust to the amount of simulated data. Furthermore, we investigate the effects of different simulation horizons. The table below shows the mean performance for 5 DMC tasks and various simulation lengths (10 seeds):

METHOD	H = 12	H = 24	H = 48
MBPO	**636**	592	542
MBMA	**683**	604	551

We find that MBMA is more robust to hyperparameter settings than MBPO. As discussed in the main body of the paper, this most likely stems from the more favorable bias variance structure of MBMA.

E. MBMA Implementation Details

We implement MBMA on top of PPO implementation (Schulman et al., 2017) taken from CleanRL repository (Huang et al., 2022b). Besides actor and critic networks which are standard for PPO, MBMA uses a dynamics model. Following Janner et al. (2019), we implement a simplistic dynamics model consisting of reward and transition models working directly on proprioceptive state representations.

Reward model inputs concatenated state-action and outputs scalar value per state-action. It is trained using MSE loss function with real reward used as a target.

Transition model inputs concatenated state-action and outputs a state vector per state-action pair. It is trained using MSE loss function with future state used as a target.

Critic inputs state and outputs scalar value per state. It is trained using MSE loss function with discounted sum of rollout rewards (ie. Monte Carlo) used as a target.

Actor inputs state and outputs means of a Gaussian distribution. It is trained using PPO clipped objective using a mix of real on-policy data (generated exactly as a regular implementation of PPO would) and simulated on-policy data (which consists of Q-values of additional actions sampled at real on-policy states). The simulated Q-values are calculated by unrolling the dynamics model for some number of steps ("simulation horizon" hyperparameter) and bootstrapping it with future state value given by the critic.

Overestimation, Overfitting, and Plasticity in Actor-Critic: the Bitter Lesson of Reinforcement Learning

Michal Nauman [*1 2] **Michał Bortkiewicz** [*3] **Piotr Miłoś** [1 2 4] **Tomasz Trzciński** [1 3 5] **Mateusz Ostaszewski** [† 3] **Marek Cygan** [† 2 6]

Abstract

Recent advancements in off-policy Reinforcement Learning (RL) have significantly improved sample efficiency, primarily due to the incorporation of various forms of regularization that enable more gradient update steps than traditional agents. However, many of these techniques have been tested in limited settings, often on tasks from single simulation benchmarks and against well-known algorithms rather than a range of regularization approaches. This limits our understanding of the specific mechanisms driving RL improvements. To address this, we implemented over 60 different off-policy agents, each integrating established regularization techniques from recent state-of-the-art algorithms. We tested these agents across 14 diverse tasks from 2 simulation benchmarks, measuring training metrics related to overestimation, overfitting, and plasticity loss — issues that motivate the examined regularization techniques. Our findings reveal that while the effectiveness of a specific regularization setup varies with the task, certain combinations consistently demonstrate robust and superior performance. Notably, a simple Soft Actor-Critic agent, appropriately regularized, reliably finds a better-performing policy within the training regime, which previously was achieved mainly through model-based approaches.

1. Introduction

In recent years, substantial improvements have been made in the domain of deep reinforcement learning, as evidenced by breakthroughs such as mastering complex games like Dota 2 (OpenAI et al., 2019), Go (Silver et al., 2017) and achieving control over nuclear fusion plasma (Degrave et al., 2022). In particular, off-policy RL has witnessed a surge of approaches reporting state-of-the-art results (Li et al., 2022; Hafner et al., 2023; Lee et al., 2023), including application to real robots (Smith et al., 2022). In general, those approaches build upon Soft Actor-Critic (SAC) algorithm with increased number of gradient steps per environment steps (Replay Ratio (RR)) used in conjunction with some form of regularization that stabilizes the learning in high RR setting (Janner et al., 2019; Chen et al., 2020; Hiraoka et al., 2021; Nikishin et al., 2022; Li et al., 2022; D'Oro et al., 2022; Cetin & Celiktutan, 2023). These approaches for regularization encompass considerations of reducing overfitting (Li et al., 2022) (*network regularization*), reducing critic overestimation (Cetin & Celiktutan, 2023) (*critic regularization*) or reducing the rate of plasticity loss (Lee et al., 2023) (*plasticity regularization*).

Despite significant advancements, the understanding of how different regularization techniques synergistically improve off-policy agent performance is still limited (Hiraoka et al., 2021; Lee et al., 2023). Moreover, most methods are tested in narrow contexts, mainly in locomotion or manipulation tasks, often restricted to a single simulation benchmark (Fujimoto et al., 2018; Haarnoja et al., 2018; Chen et al., 2020; Moskovitz et al., 2021; D'Oro et al., 2022), leading to questions about their broad applicability and robustness. In this study, our goal is to consolidate these lessons and address the following research questions: *Which regularization techniques lead to robust performance improvements across diverse tasks and agent designs? Can generic regularization techniques outperform domain-specific RL techniques that directly use the MDP structure?*. We extend the scope of prior research by examining over 60 design choices implemented within the Soft Actor-Critic framework. We test a diverse array of tasks, including both locomotion and manipulation, within two simulation benchmarks and two replay

* Main authors contributed equally to this work; † Senior authors contributed equally to this work. [1]Ideas NCBR [2]University of Warsaw [3]Warsaw University of Technology [4]Polish Academy of Sciences [5]Tooploox [6]Nomagic. Correspondence to: Michal Nauman <nauman.mic@gmail.com>, Michal Bortkiewicz <michalbortkiewicz8@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

ratio regimes. This comprehensive approach offers a deeper understanding of the effectiveness of these regularization techniques in various settings.

Our main result is a bitter lesson: across varied tasks, general neural network regularizers significantly outperform most RL-specific algorithmic improvements in terms of agent performance. Specifically, we find general methods that are motivated by stabilization of gradient-based learning significantly outperform RL-specific algorithmic improvements across a variety of environments. Such emphasis on generality is in line with the celebrated "Bitter Lesson" essay (Sutton, 2019). Notably, network regularization enables agents to find effective policies on tasks previously impossible for model-free agents, such as those in the dog domain. Our findings also show that layer normalisation is more effective in reducing overestimation than techniques specifically designed for mitigating Q-value overestimation in critic networks. Consequently, we show that replacing the ubiquitous Clipped Double Q-learning with network regularization techniques leads to significant performance gains. Our research further explores the impact of overestimation, overfitting, and plasticity loss on agent performance in a unified experimental setup. We examine the correlation between these factors and agent performance, showing a strong negative correlation for value overestimation and agent plasticity metrics. These influences vary significantly across environments, underscoring the complex nature of their effects on learning. A key observation is the environment-dependent performance of various methods. Strategies excelling in locomotion tasks may falter in manipulation scenarios and vice versa. Comparisons between experiments on the DeepMind Control Suite (Tassa et al., 2018) and MetaWorld (Yu et al., 2019) demonstrate the necessity for diverse benchmarking in research, highlighting the value of expansive experimental setups.

1. Our study presents an extensive empirical analysis of various regularization techniques in off-policy RL. We evaluate the effectiveness, robustness, and generality of 12 SAC design choices derived from recent literature, examining their diverse interactions. This encompasses testing 64 model designs across 14 tasks from two benchmarks under two replay ratio regimes.

2. Our findings show that combining well-established network regularization techniques with methods that prevent plasticity loss effectively addresses the value estimation problem, eliminating the need for critic regularization. Specifically, we observe that in network/plasticity regularized agents using critic regularization often leads to significant performance degradation. Leveraging these insights, we demonstrate that integrating specific regularization methods into the basic Soft Actor-Critic framework leads to state-of-the-art performance in dog domain tasks for model-free approaches.

3. Our study investigates the correlation between overestimation, overfitting, and plasticity proxies, and their impact on agent performance. We discover that interventions aimed at one type of issue, such as full-parameter resets, significantly affect proxies for issues other than plasticity such as overestimation and overfitting, often more than interventions specifically designed for those other issues. This suggests that RL agents encounter a range of complex problems that collectively affect the learning process.

2. Background

We consider a Markov Decision Process (MDP) (Puterman, 2014; Sutton & Barto, 2018) which is described via a tuple (S, A, r, p, γ), where states S and actions A are continuous, $r(s, a)$ is the transition reward, $p(s'|s, a)$ is a transition mapping, p_0 is the starting state distribution and $\gamma \in (0, 1]$ is the discount factor. Policy, denoted as $\pi(a|s)$ is a state-conditioned action distribution. Maximum Entropy Reinforcement Learning (MaxEnt RL) objective (Ziebart et al., 2008; Haarnoja et al., 2017) is to find a policy that maximizes the expected sum of discounted returns and policy entropies, or equivalently expected initial state values according to $\pi^* = \arg\max \mathbb{E}_{p_0} V^\pi(s_0)$. The Q-value is defined as $Q^\pi(s, a) = r(s, a) + \gamma V^\pi(s')$. State value is defined by $V^\pi(s) = \mathbb{E}_\pi(Q^\pi(s, a) - \alpha \log \pi(a|s))$, where $\alpha \log \pi(a|s)$ is the maximum entropy term. In actor-critic, policy and Q-value functions are represented by parameterized function approximators (Silver et al., 2014). Policy parameters θ are updated to maximize the value approximation at sampled states s from an off-policy replay buffer $\mathcal{D}$ (Fujimoto et al., 2018; Haarnoja et al., 2018):

$$\theta^* = \arg\max_\theta \mathop{\mathbb{E}}_{\mathcal{D}} Q_\phi(s, a) - \alpha \log \pi_\theta(a|s),\ a \sim \pi_\theta. \quad (1)$$

The critic parameters ϕ are updated by minimizing the temporal-difference (Silver et al., 2014):

$$\phi^* = \arg\min_\phi \mathop{\mathbb{E}}_{\mathcal{D}} \big(Q_\phi(s, a) - r_(s, a) - \gamma \bar{V}_\phi(s')\big)^2, \quad (2)$$

where $\bar{V}_\phi(s')$ is the target network (Mnih et al., 2015).

2.1. Overestimation

Q-learning methods employing function approximation have been observed to exhibit a bias toward overestimation, a phenomenon critical to the training process (Thrun & Schwartz, 2014; Fujimoto et al., 2018). Positive bias stems from the policy being trained to locally maximize action-value estimates, leading its actions to exploit potential model errors for higher scores. Modern actor-critic

algorithms leverage a variety of countermeasures to overestimation of Q-value targets, with Clipped Double Q-learning (**CDQ**) (Fujimoto et al., 2018) being most used by many other algorithms (Haarnoja et al., 2018; Chen et al., 2020; Hiraoka et al., 2021). In CDQ, the algorithm maintains two critics and uses their minimum as an approximate Q-value lower bound. The CDQ was generalized to the following pessimistic objective (Ciosek et al., 2019; Moskovitz et al., 2021; Cetin & Celiktutan, 2023):

$$Q^{\beta}_{\phi}(s,a) = Q^{\mu}_{\phi}(s,a) - \beta Q^{\sigma}_{\phi}(s,a). \tag{3}$$

We denote the level of pessimism as β, and the critic ensemble mean and standard deviation as Q^{μ}_{ϕ} and Q^{σ}_{ϕ} respectively. In particular, for $\beta = 1$, the above rule is exactly equal to the CDQ minimum (Ciosek et al., 2019; Cetin & Celiktutan, 2023). The success of pessimistic updates led to various methods for adjusting β online. A recent approach, Generalized Pessimism Learning (**GPL**) (Cetin & Celiktutan, 2023), estimates the critic approximation error and modifies β accordingly. A different strategy, Tactical Optimism and Pessimism (**TOP**) (Moskovitz et al., 2021), adjusts pessimism independent of the estimated approximation error. Specifically, TOP uses an external bandit controller to maximize online episodic rewards. Whereas this controller is aligned with the RL objective, it only allows for discrete values of pessimism.

2.2. Overfitting

Overfitting, while not commonly scrutinized in reinforcement learning, has gained attention in recent discussions (Li et al., 2022) as a phenomenon correlated with performance decline in models characterized by a high ratio of updates to data. To evaluate overfitting in agents, Li et al. (2022) utilizes a validation dataset that consists of samples gathered using the same policy as the canonical replay buffer. The validation buffer is established to provide an unbiased assessment of the critic error in experiences that were not used in the learning. Although there are many strategies to deal with overfitting in supervised learning, only a few of them were applied in the context of RL. To this end, application of Weight Decay (**WD**) (Schwarzer et al., 2023), Layer Normalization (**LN**) (Ball et al., 2023) or Spectral Normalization (**SN**) (Cetin & Celiktutan, 2023) was shown to greatly effect the performance of the underlying agent.

2.3. Plasticity

Plasticity, in the context of models, refers to their ability to learn new information. The concept of plasticity loss has recently gained prominence in the deep learning community, particularly in supervised learning (Achille et al., 2017; Ash & Adams, 2020; Dohare et al., 2021) and RL (Nikishin et al., 2022; Dohare et al., 2021; Lyle et al., 2023; Lee et al., 2023; Kumar et al., 2023; Nikishin et al., 2023). Numerous hypotheses have been proposed regarding the sources of plasticity loss, including dead or dormant units, rank collapse, and divergence due to large weight magnitudes (Lyle et al., 2022; Sokar et al., 2023; Kumar et al., 2020; Dohare et al., 2021). However, none of these mechanisms alone is sufficient to explain the phenomenon of plasticity loss. Whereas the cause of plasticity loss remains to be discovered, various approaches for regularizing the model plasticity have been proposed. For example, full-parameter **resets** of actor-critic modules were shown to greatly improve the agent's ability to learn (Nikishin et al., 2022; D'Oro et al., 2022). The problem of plasticity was also tackled at the level of the activation function with Concatenated ReLU (**CRLU**) (Abbas et al., 2023) or the optimizer with the Sharpness-Aware Minimization (**SAM**) (Foret et al., 2020).

3. Study Design

In this paper, we analyze the impact of various interventions on SAC performance across seven DeepMind Control Suite (Tassa et al., 2018) (DMC) tasks: acrobot-swingup, hopper-hop, humanoid-walk, humanoid-run, dog-trot, dog-run, quadruped-run and seven MetaWorld (Yu et al., 2019) (MW) tasks: Hammer, Push, Sweep, Coffee-Push, Stick-Pull, Reach, Hand-Insert. We chose a wide spectrum of tasks, ranging from easy (acrobot-swingup, Reach) to barely solvable (dog-run) for generic insights that are not overfitted to only a specific group. We choose tasks that are not easily solved by the baseline high replay SAC, as presented in D'Oro et al. (2022) and Hansen et al. (2022), with added dog tasks (which are generally unsolved in state-based representation). Finally, following (Li et al., 2022), we conduct experiments in low 2 and high replay regimes. Such experimental design allows us to pinpoint if specific regularization targets issues associated with high replay, or if it is universally applicable across varying replay regimes. Categorizing them based on current state-of-the-art methods, we identify three intervention groups:

- Critic Regularizations (CR),
  - Clipped Double Q-learning (CDQ) (Fujimoto et al., 2018),
  - Tactical Optimism Pessimism (TOP) (Moskovitz et al., 2021),
  - Generalized Pessimism Learning (GPL) (Cetin & Celiktutan, 2023),
- Network Regularizations (NR),
  - Layer Norm (LN) (Ba et al., 2016),
  - Spectral Norm (SN) (Miyato et al., 2018; Zhang et al., 2018; Brock et al., 2018),
  - Weight Decay (WD) (Loshchilov & Hutter, 2017),

- Plasticity Regularizations (PR),
  - Resets (Res) (Nikishin et al., 2022),
  - Concatenated ReLU activations (CRLU) (Abbas et al., 2023),
  - Sharpness-Aware Minimization Optimizer (SAM) (Foret et al., 2020).

To explore the interactions between interventions, we systematically run all possible combinations of methods across groups, ensuring that methods from the same group are not combined. Each configuration is evaluated on 10 seeds. In the results analysis, we categorize marginalization into three levels:

First-order marginalization combines all results for a specific intervention. For instance, the marginalized performance of layer norm will be computed as the average performance across all combinations with interventions from other groups (in this example, Critic Regularizations and Plasticity Regularizations).

Second-order marginalization involves evaluating the performance of fixed pairs of methods from two groups and marginalizing results from the third group.

Third-order results involve no intervention marginalization and represent the performance of a specific combination, including one method from each group. The only marginalization is over all tested environments (we present these results in Appendix C.1 due to space constraints). This cumulative result provides insight into the overall impact of a given intervention.

Furthermore, we conduct an analysis of various proxy metrics associated with the problems of overestimation, overfitting, and plasticity loss. For overestimation, we evaluate the state-action critic approximation error, denoted as $b_\phi(s, a)$, is quantified as the disparity between the critic output and the true on-policy Q-value according to $b_\phi(s, a) = Q_\phi(s, a) - Q^\pi(s, a)$, where Q_ϕ denotes the critic Q-value approximation and Q^π represents the on-policy Q-value which we estimate via a Monte-Carlo rollout with 5 samples. To calculate overfitting, we compare average TD errors on evaluation trajectories (which are not used for learning) to average TD errors observed in training according to $o_\phi = \frac{\mathbb{E}_{\mathcal{D}_v} TD_\phi}{\mathbb{E}_{\mathcal{D}} TD_\phi}$, where o_ϕ denotes critic overfitting, $\mathcal{D}_v$ denotes validation data, and TD_ϕ denotes the temporal difference loss. As such, the extent of overestimation is then quantified by the ratio of validation TD error to training TD error. We monitor plasticity loss by the rank of penultimate layer representations (Kumar et al., 2020), dormant neurons or dead units (Sokar et al., 2023), the L2 norm of weights (Nikishin et al., 2022; Lyle et al., 2023), and gradient norm (Nikishin et al., 2022; Lyle et al., 2023) as a proxy for plasticity loss.

4. Experiments

4.1. Combination of interventions – First-order marginalization

Study description: First-order marginalization provides insights into the robust impact of a given intervention on model performance, irrespective of what other type of regularization it is paired with. To measure such robustness, we compare the performance of the baseline SAC model augmented with one specific regularization (e.g., SAC + WD) to the performance of SAC augmented with this regularization paired with some other technique (e.g. SAC + WD + Resets).

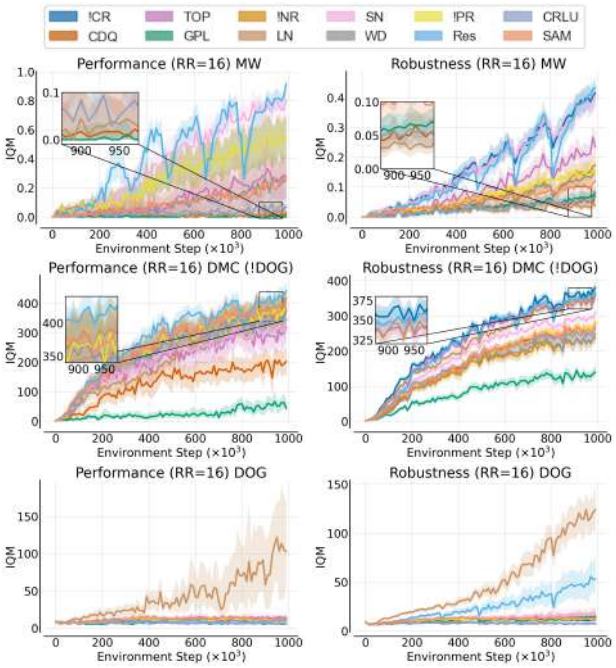


Figure 1. IQM performance of First-Order Marginalization. The left column presents results for baseline SAC augmented with a single regularization technique (and thus uses 10 seeds per task), and the right column presents the aggregate performance of a specific regularization technique when paired with other regularizations (and thus uses 640 seeds per task). Results are presented for MW (top row), DMC without Dog environments (middle row) and only Dog-run and Dog-trot (bottom row) benchmarks. 14 tasks.

Results: Examining the plots in Figure 1 shows that network and plasticity regularization techniques are generally more effective than critic regularization – dark blue line (!CR) on Robustness plots on MW and DMC (!DOC). We observe these results for both simple models using one regularization technique and more complex agents leveraging many regularizations at once. Most notably, avoiding the use of critic regularization interventions (!CR) proves advantageous for both DMC and MW, especially if some other type of regularization is used (such as layer norm or full-

parameter resets). This result is somewhat surprising, as critic regularization methods were designed specifically for off-policy actor-critic agents, whereas the network and plasticity regularization techniques are general. Notably, TOP (Moskovitz et al., 2021) emerges as an exception, particularly showcasing its effectiveness on the DMC benchmark. Conversely, the GPL intervention exhibits the least robust performance in both DMC and MW. Upon further analysis of the impact of CDQL presence (see Section C.4), it becomes apparent that in certain environments, such as Hopper Hop, the adverse effects of this intervention cannot be mitigated even with additional regularizations. Full-parameter resets (Nikishin et al., 2022), tailored for high replay ratio regimes, prove to be one of the most robust approaches in this RR regime. Further analysis reveals discrepancies in conclusions between benchmarks. Clearly, LN is the most effective approach in the DMC Dog environments (bottom row in Figure 1), but it also ranks among the top four interventions in the remaining DMC environments. However, it exhibits very poor performance on the MW benchmark. Therefore, we find SN to be more robust, significantly aiding the MW benchmark and providing moderate assistance in the DMC scenarios.

Takeaways:

- Critic regularization methods exhibit limited effectiveness in enhancing performance. When using network or plasticity regularization, critic regularization leads to reduced performance.
- Periodical network resetting is the most robust intervention across two benchmarks in a high replay ratio regime, and highly surpasses other plasticity regularization techniques in both robustness and performance.
- Layer norm is essential for Dog environments.
- When considering network regularization approaches, layer norm is generally recommended for DMC, while spectral norm is more effective for MW benchmarks. When considering a diverse range of tasks, we find spectral norm to be more robust than layer norm. Weight decay has generally low performance when used alone with SAC.

4.2. Combination of interventions – Second-order marginalization

Study description: This study delves into second-order marginalization to pinpoint the most effective combinations. Results are presented across various replay ratios (2 and 16) and benchmarks (DMC or MW). Given the limited impact of critic regularizations like CDQ or GPL in the first order experiments, our focus is on discerning the most advantageous combinations involving of regularization.

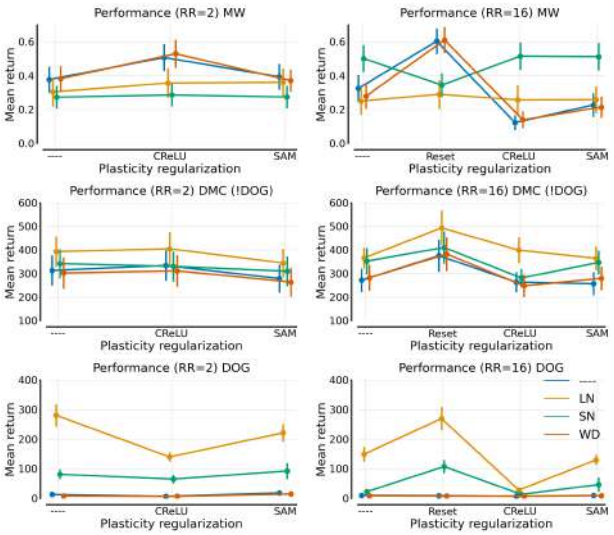


Figure 2. Second-order results marginalizing critic regularization methods. On the x-axis, we have different types of plasticity regularization, and each colour denotes network regularization. For better readability, points within one plasticity regularization are spaced slightly horizontally. Vertical lines indicate standard error.

Results: On the DMC with RR=2 and RR=16 (top row in Figure 2) a clear hierarchy of interventions is observed: layer norm and right below it, spectral norm consistently outperforms others in mean return, irrespective of the plasticity regularization (x-axis). Notably, the combination of layer norm and resets in RR=16 (middle and bottom-right plot in Figure 2) demonstrates exceptional performance across all critic regularization variations on DMC Dog and without Dog environments. In contrast, the hierarchy of interventions on the MW benchmark (top row in Figure 2) diverges significantly from the DMC setup. Furthermore, a higher replay ratio introduces shifts in training dynamics, as evidenced by SN transitioning from a lower position on RR=2 (bottom-left plot in Figure 2) to nearly the most versatile intervention on RR=16 (bottom-right plot in Figure 2). The results generally align with first-order marginalization findings, emphasizing the positive impact of SN, as well as using many different types of regularization at once in general. Deeper analysis (see Appendix C.2) reveals that on MW, indeed, the gradient norm in a higher RR regime is orders of magnitude bigger. The finding that most contrasts with the first-order experiments, is that we observe that weight decay can actually yield significant performance benefits, under the condition that it is paired with other specific methods, namely full-parameter resets. In particular, we observe that

this combination yields synergies surpassing using any of these methods alone.

Takeaways:

- The DMC benchmark can be largely trivialized by using high RR agents combined with layer norm and full-parameter resets.
- Spectral normalization intervention ranks best for the Meta World benchmark, but it's not universally applicable. Whereas weight decay does not perform when used alone, it seems to have high synergy with full-parameter resets.
- Resetting the network significantly outperforms other plasticity-inducing interventions such as CRLU and SAM.

4.3. A closer look on Dog environment performance

Study description: In this study, we delve into the intricacies of two challenging Dog tasks, Dog-Trot and Dog-Run, included in our DMC setup. These environments present considerable difficulties for model-free approaches relying on proprioceptive states, making them of particular interest within the research community. Due to the inherent difficulty of these tasks, we conducted additional experiments using the top three methods identified in Figure 23 (Appendix) for 4 million steps, akin to approaches used in model-based (Hansen et al., 2022) or pixel-based studies (Ji et al., 2024). For the rest of the detailed experimental information, please refer to Appendix A.

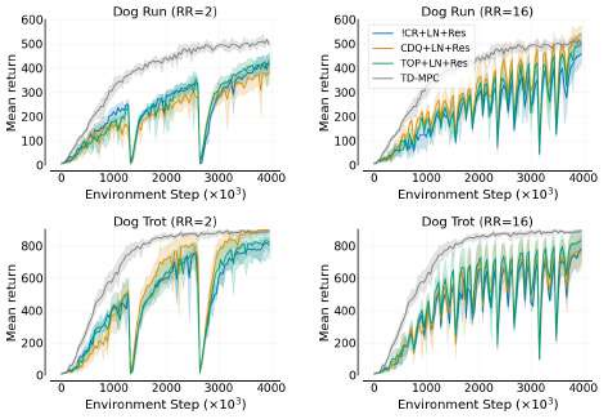


Figure 3. Mean return evolution across 4 million timesteps for Dog-Run (top row) and Dog-Trot (bottom row) environments. Gray plot depicts model-based agent performance. Each plot showcases the top three combinations.

Results: Specific intervention combinations effectively tackle the challenges posed by the Dog environment, as depicted in Figure 3. Analyzing the top three approaches for Dog-Run and Dog-Trot tasks reveals the prevalence of layer norm in nearly all combinations. Additionally, each critic regularization approach contributes to the leading group. Notably, in scenarios with high replay ratios, resets emerge as a crucial intervention. These observations are further substantiated by the analysis of second-order marginalization IQM plots (see 4). Indeed, layer norm without critic regularization excels in RR=2, and layer norm with resets outperforms all others convincingly. This achievement is particularly notable as, to our best knowledge, no model-free agent has previously find a better-performing policy within the training regime the Dog environments using proprioceptive states. Notably, there is a recent study(Ji et al., 2024) where a model-free agent achieved comparable results on the Dog environments but using pixel-based inputs instead. Additionally, our results demonstrate that while a model-based approach on proprioceptive states (Hansen et al., 2022) outperforms slightly, the above model-free approach with simple regularization techniques achieves performance very close to that of the model-based approach. This suggests the efficacy and competitiveness of our approach in challenging environments.

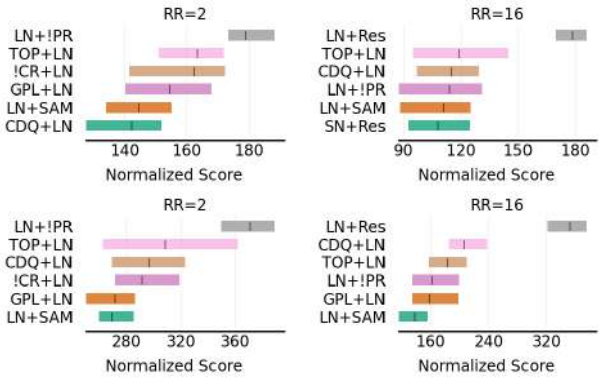


Figure 4. IQM performance of the top six intervention pair combinations based on 1 million steps experiments. The IQM is calculated based on the average of the last ten evaluation points in each run, not the last evaluation point. Results come from 1 million steps experiments. Top row: Dog-Run. Bottom row: Dog-Trot.

Takeaways:

- Well-established network regularization techniques such as layer norm and spectral norm enable finding high-performing policies for Dog-Trot and Dog-Run effectively.
- The choice of domain-specific RL critic regularization has little significance in dog environments when layer norm and resetting interventions are employed.

4.4. Correlation of Overestimation, Overfitting and Plasticity metrics with Performance

Study description: This study analyses the relationships between Overestimation, Overfitting, Plasticity, and model performance. Overestimation is quantified as an approximation error, overfitting as the ratio of TD error on the validation set to TD error on the training set. Expressing Plasticity loss is challenging, so we utilize proxy metrics, including the percentage of Dormant neurons (Sokar et al., 2023), representations rank (Kumar et al., 2020), gradient norm (Nikishin et al., 2022; Lyle et al., 2023), and parameters norm (Nikishin et al., 2022; Lyle et al., 2023).

We employ a Spearman correlation matrix to scrutinize these dependencies. This statistic is chosen because we observe non-linear yet monotonic dependencies between the mentioned metrics. We employ it on the data from all performed experiments, i.e., form runs with different combinations of interventions. Moreover, we do not have a division into RR=2 and RR=16, only the results from both setups are combined, and analyses are made on them. Overestimation, and gradient norm, and parameters norm are analyzed in a logarithmic scale for precision. We exclude metric pairs where the p-value of correlation is above 5% by whitening tiles in the correlation matrix.

Spearman correlation: In Figure 5, we investigate the relations between plasticity loss, overestimation and overfitting metrics and agents return, separately for every benchmark with special separation for Dog environments. Notably, overestimation exhibits the strongest correlation with agent returns on both benchmarks, offering insights into the findings of previous sections regarding the limited robustness of critic regularization methods designed to minimize overestimation. Interestingly, as shown in Figure 6, layer norm and spectral norm effectively mitigate approximation errors, outperforming CR methods. However, further investigation on the DMC benchmark (Figure 5) uncovers a strong negative correlation between the percentage of dormant neurons and performance, closely tied to the rank of representations on the critic's penultimate layer. What is more, the overestimation is not the best predictor for Dog environments where we observe the highest values of overestimation (Figure 5). We hypothesize that overestimation becomes a good predictor of performance only when more fundamental issues, such as plasticity, are mitigated, indicating a multifaceted learning problem in harder environments.

As an observation that supports this hypothesis, we refer to the critic gradient norm, which exhibits the most monotonic relation with the return in dog environments, as indicated by Spearman correlation (Figure 5). Analysing Figure 10 one can see, that Dog environments especially with high replay ratio experience exploding gradients. The high gradient

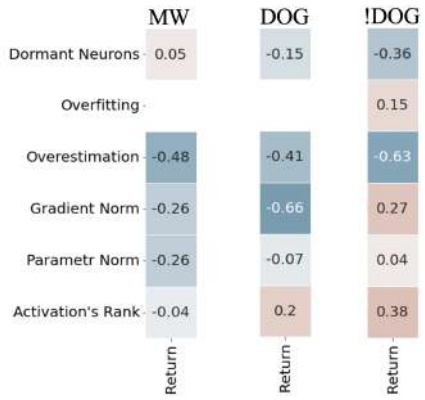


Figure 5. Explanatory metrics correlations for three different groups of environment, namely: MetaWorld, DMC Dog environments, and DMC environments without Dog environments. It's important to observe that not only does the main explanatory metric, gradient norm, vary for dog environments, but the remaining DMC environments also exhibit a different correlation sign for this metric.

norm directly points to high curvature of the loss landscape, which, as indicated by (Lee et al., 2023), describes low input plasticity. For this reason, layer norm primarily smoothens the activation distribution and plays a critical role in making SAC work in dog environments. Environments from the MW benchmark also encounter challenges with high curvature loss landscapes (as indicated by the Spearman correlation between gradient norm and return). This suggests a resemblance between MW and DMC dog environments.

Moreover, on the MW benchmark (Figure 5), the second-best correlated metrics with performance are the critic gradient norm and the critic parameters norm. This aligns with the results from section 4.2, highlighting the significant performance boost provided by spectral norm and weight decay, particularly in the RR=16 setup. An in-depth analysis of how RR increases the negative correlation of gradient norm and return can be found in sections C.2 and C.5 of Appendix.

Takeaways:

- There are distinctive correlations between plasticity loss, overestimation, overfitting metrics, and agent returns in various benchmark suites. These underscore the importance of considering environment-specific factors when assessing model performance and designing effective regularization strategies.

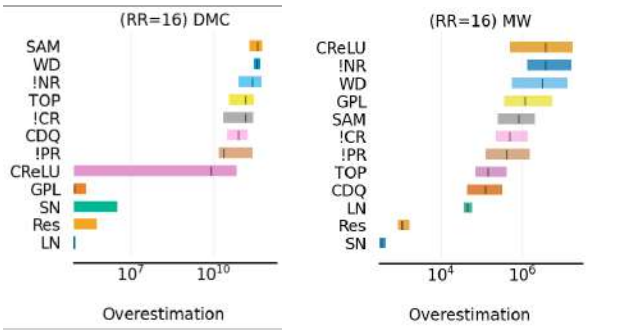


Figure 6. IQM overestimation in logarithmic scale. Each plot presents sorted results. The IQM is calculated based on the average of the last ten evaluation points in each run, not the last evaluation point. Left Figure: DMC benchmark. Right Figure: MW benchmark. Colours indicate hierarchy on the plot, not specific names.

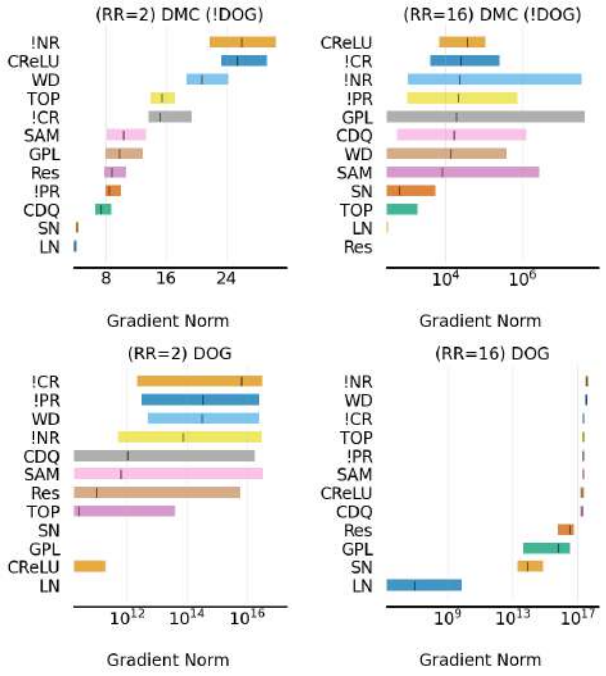


Figure 7. IQM gradient norm of first-order results. The IQM is calculated based on the average of the last ten evaluation points in each run, not the last evaluation point.

- Techniques like layer or spectral norm and resets are particularly effective in mitigating overestimation also compared to methods specifically designed for that purpose.
- The negative correlation of the critic's gradient norm and return becomes more apparent in challenging environments and mainly in a high replay ratio regime.

5. Related Works

The literature on deep reinforcement learning has long explored various factors contributing to performance challenges, approaching the issue from different perspectives. A notable study, akin to our pragmatic approach, investigates the impact of diverse design choices in the training process of on-policy methods (Andrychowicz et al., 2021).

In the realm of off-policy methods, numerous hypotheses have been proposed regarding crucial factors. One key focus is on addressing the overestimation problem, with attempts to harness its potential benefits (Ciosek et al., 2019) and, more prominently, to mitigate the phenomenon by introducing novel loss functions (Fujimoto et al., 2018; Moskovitz et al., 2021; Cetin & Celiktutan, 2023).

Regularization schemes have proven effective in enhancing deep reinforcement learning methods. Neural network regularizations, such as Spectral Norm (Gogianu et al., 2021; Bjorck et al., 2021), Layer Norm (Bjorck et al., 2021; Ball et al., 2023), or weight decay (Liu et al., 2020), have yielded significant improvements in results. Notably, periodic resets of critic weights proposed by (Nikishin et al., 2022) constitute a strong baseline in robotics control. Several regularization schemes have been proposed to address the issue of discarding knowledge caused by fully resetting the critic network. Of particular interest is Shrink and Perturb (Ash & Adams, 2020) and L2 Init (Kumar et al., 2023). In both of these methods the benefit comes from the regularization towards the distribution of "freshly" initialized weights. A concurrent work finds that using unit-ball normalization allows for learning with a high-replay ratio without full-parameter resets (Hussing et al., 2024). An ensemble approach was also suggested to address the challenge of determining the optimal number of gradients per environment step, where decisions are based on the validation TD error (Li et al., 2022).

6. Limitations

In this work, we found crucial choices that drive SAC effectiveness in a wide range of control tasks from two popular benchmarks. Through extensive experiments, we uncovered perplexities concerning explanatory metrics correlations and complex dynamics of overestimation, which is successfully mitigated by widely used regularizations. Nevertheless, our study has certain constraints. Our empirical evaluations were limited to proprioceptive tasks on DMC and Meta-World benchmarks and only for SAC method.

7. Conclusions

This study explored different common RL design choices, as well as interactions thereof, evaluating their impact on

agent learning. Specifically, we consider three types of regularization families: critic regularization (motivated by value overestimation); network regularization (motivated by model overfitting); and plasticity regularization (motivated by plasticity loss). Our analysis revealed that generic network regularization methods such as layer normalization, especially when paired with full-parameter resets, can have a vastly greater impact on the final performance than domain-specific RL approaches. To this end, the same network regularization methods can lead to strong performing policies on domains previously solved by model-based agents, such as the dog domain. Furthermore, we studied a variety of metrics that were shown to co-occur with the deterioration of learning in low and high replay regimes. Our analysis revealed the complex interactions of the considered metrics with agent performance. Surprisingly, we found that interventions motivated by a specific problem, for example, overfitting, can have a pronounced impact on the metrics associated with overestimation or plasticity. Finally, we found that the effectiveness of considered critic, network, and plasticity regularization techniques is not only highly dependent on the simulation benchmark but also type of simulated task. The most prominent example is Clipped Double Q-learning, a technique used in a majority of modern actor-critic algorithms, which is effective in DMC locomotion tasks, but leads to significant performance deterioration on the MetaWorld manipulation tasks. To this end, we highlighted the need to test new algorithms on a diverse set of tasks, preferably stemming from more than one suite.

Acknowledgements

Marek Cygan was partially supported by an NCBiR grant POIR.01.01.01-00-0433/20. Mateusz Ostaszewski was funded by the National Science Center Poland under the grant agreement 2020/39/B/ST6/01511. Michał Bortkiewicz was funded by the Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) programme. This research was also supported by National Science Centre, Poland grant no 2020/39/B/ST6/01511 and grant no 2022/45/B/ST6/02817. This work was partially funded by the European Union under the Horizon Europe grant OMINO (grant number 101086321) and Horizon Europe Program (HORIZON-CL4-2022-HUMAN-02) under the project "ELIAS: European Lighthouse of AI for Sustainability", GA no. 101120237. We also gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2023/016783.

Impact Statement

This paper focuses on the issue of interplay between different design choices within Reinforcement Learning (RL) algorithms. While the successful application of RL has the potential to influence society in many ways, our work, focused primarily on a technical advancement in RL algorithms, does not introduce novel ethical considerations beyond those already inherent in the broader field of RL.

References

Abbas, Z., Zhao, R., Modayil, J., White, A., and Machado, M. C. Loss of plasticity in continual deep reinforcement learning. *arXiv preprint arXiv: 2303.07507*, 2023.

Achille, A., Rovere, M., and Soatto, S. Critical learning periods in deep neural networks. *arXiv preprint arXiv: 1711.08856*, 2017.

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.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. What matters in on-policy reinforcement learning? a large-scale empirical study. In *ICLR 2021-Ninth International Conference on Learning Representations*, 2021.

Ash, J. and Adams, R. P. On warm-starting neural network training. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3884–3894. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/288cd2567953f06e460a33951f55daaf-Paper.pdf.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Ball, P. J., Smith, L., Kostrikov, I., and Levine, S. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv: 2302.02948*, 2023.

Bjorck, J., Gomes, C. P., and Weinberger, K. Q. Towards deeper deep reinforcement learning with spectral normalization. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 8242–8255, 2021. URL https://proceedings.

neurips.cc/paper/2021/hash/4588e674d3f0faf985047d4c3f13ed0d-Abstract.html.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *International Conference on Learning Representations*, 2018.

Cetin, E. and Celiktutan, O. Learning pessimism for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6971–6979, 2023.

Chen, X., Wang, C., Zhou, Z., and Ross, K. W. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2020.

Ciosek, K., Vuong, Q., Loftin, R., and Hofmann, K. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32, 2019.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., Donner, C., Fritz, L., Galperti, C., Huber, A., Keeling, J., Tsimpoukelli, M., Kay, J., Merle, A., Moret, J.-M., Noury, S., Pesamosca, F., Pfau, D., Sauter, O., Sommariva, C., Coda, S., Duval, B., Fasoli, A., Kohli, P., Kavukcuoglu, K., Hassabis, D., and Riedmiller, M. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602 (7897):414—419, February 2022. ISSN 0028-0836. doi: 10.1038/s41586-021-04301-9.

Dohare, S., Sutton, R. S., and Mahmood, A. R. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv: 2108.06325*, 2021.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2022.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv: 2010.01412*, 2020.

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

Gogianu, F., Berariu, T., Rosca, M. C., Clopath, C., Busoniu, L., and Pascanu, R. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, pp. 3734–3744. PMLR, 2021.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. In *International Conference on Machine Learning, PMLR*, 2022.

Hiraoka, T., Imagawa, T., Hashimoto, T., Onishi, T., and Tsuruoka, Y. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2021.

Hussing, M., Voelcker, C., Gilitschenski, I., Farahmand, A.-m., and Eaton, E. Dissecting deep rl with high update ratios: Combatting value overestimation and divergence. *arXiv preprint arXiv:2403.05996*, 2024.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Ji, T., Luo, Y., Sun, F., Zhan, X., Zhang, J., and Xu, H. Seizing serendipity: Exploiting the value of past success in off-policy actor-critic, 2024.

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

Kumar, S., Marklund, H., and Roy, B. V. Maintaining plasticity in continual learning via regenerative regularization. *arXiv preprint arXiv: 2308.11958*, 2023.

Lee, H., Cho, H., Kim, H., Gwak, D., Kim, J., Choo, J., Yun, S.-Y., and Yun, C. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. *NEURIPS*, 2023.

Li, Q., Kumar, A., Kostrikov, I., and Levine, S. Efficient deep reinforcement learning requires regulating overfitting. In *The Eleventh International Conference on Learning Representations*, 2022.

Liu, Z., Li, X., Kang, B., and Darrell, T. Regularization matters in policy optimization-an empirical study on continuous control. In *International Conference on Learning Representations*, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2017.

Lyle, C., Rowland, M., and Dabney, W. Understanding and preventing capacity loss in reinforcement learning. *International Conference on Learning Representations*, 2022. doi: 10.48550/arXiv.2204.09560.

Lyle, C., Zheng, Z., Nikishin, E., Avila Pires, B., Pascanu, R., and Dabney, W. Understanding plasticity in neural networks. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23190–23211. PMLR, 23-29 Jul 2023.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations*, 2018.

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.

Moskovitz, T., Parker-Holder, J., Pacchiano, A., Arbel, M., and Jordan, M. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12849–12863, 2021.

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.

Nikishin, E., Oh, J., Ostrovski, G., Lyle, C., Pascanu, R., Dabney, W., and Barreto, A. Deep reinforcement learning with plasticity injection. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023. URL https://openreview.net/forum?id=O9cJADBZT1.

OpenAI, :, Berner, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv: 1912.06680*, 2019.

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

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

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

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.

Smith, L., Kostrikov, I., and Levine, S. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. The dormant neuron phenomenon in deep reinforcement learning. *International Conference on Machine Learning*, 2023. doi: 10.48550/arXiv.2302.12902.

Sutton, R. The bitter lesson. *Incomplete Ideas (blog)*, 13(1), 2019.

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

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.

Thrun, S. and Schwartz, A. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*, pp. 255–263. Psychology Press, 2014.

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.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In Kaelbling, L. P., Kragic, D., and Sugiura, K. (eds.), *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2019. URL http://proceedings.mlr.press/v100/yu20a.html.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. Self-attention generative adversarial networks. *arXiv preprint arXiv: 1805.08318*, 2018.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A. Details of experiments

Results reporting Interquartile mean (IQM) are based on 500 bootstrapping points as calculated by *rliable* package (Agarwal et al., 2021). The final performance is defined as the average of the last 10 policy evaluations.

B. Architecture details

In all experiments, the Actor and Critic are represented by three-layer MLP networks, each containing 256 neurons in the hidden layers, utilizing the ReLU activation function (except for the CReLU variant, which effectively doubles the number of activations).

In the scenario involving Layer Norm, it is applied to each hidden layer (Li et al., 2022; Ball et al., 2023), while the spectral norm is applied exclusively to the last hidden layer (Gogianu et al., 2021; Li et al., 2022). Weight decay is uniformly applied across all layers (Li et al., 2022). It's important to note that all network regularizations are exclusively applied to the Critic network.

B.1. Hyperparameters

All hyperparameters are taken from original papers introducing the given intervention.

Table 1. Hyperparameter values used in the experiments.

HYPERPARAMETER	NOTATION	VALUE
JOINT		
NETWORK SIZE	NA	$(256, 256)$
ACTION REPEAT	NA	1
OPTIMIZER	NA	ADAM
LEARNING RATE	NA	$3e-4$
BATCH SIZE	B	256
DISCOUNT	γ	0.99
INITIAL TEMPERATURE	α_0	1.0
INITIAL STEPS	NA	10000
TARGET ENTROPY	$\mathcal{H}^*$	$\lvert\mathcal{A}\rvert/2$
POLYAK WEIGHT	τ	0.005
TOP		
PESSIMISM VALUES	β	$\{0, 1\}$
BANDIT LEARNING RATE	NA	0.1
GPL		
PESSIMISM LEARNING RATE	NA	$1e-5$

C. Further Experiments

C.1. Third-order marginalization

Examining the plots without marginalization (Figure 8) provides further insights into the conclusions drawn from the previous experiment. Specifically, most combinations without critic regularization (red points) consistently perform well across all setups. Additionally, the results for GPL (pink points) affirm the overall subpar performance of this method. On the DMC with RR=2 plot (top-left in Figure 8), layer norm (points labeled "L") consistently outperforms others in mean return, irrespective of the Plasticity regularization (x-axis) or Critic Regularization (color). Notably, the combination of layer norm and resets in RR=16 (top-right plot in Figure 8) demonstrates exceptional performance across all critic regularization variations.

For the MW benchmark with RR=16 (bottom-right plot in Figure 8), the results align with first-order marginalization findings, highlighting the positive impact of Spectral Normalization. Particularly interesting is the role of Weight Decay, forming

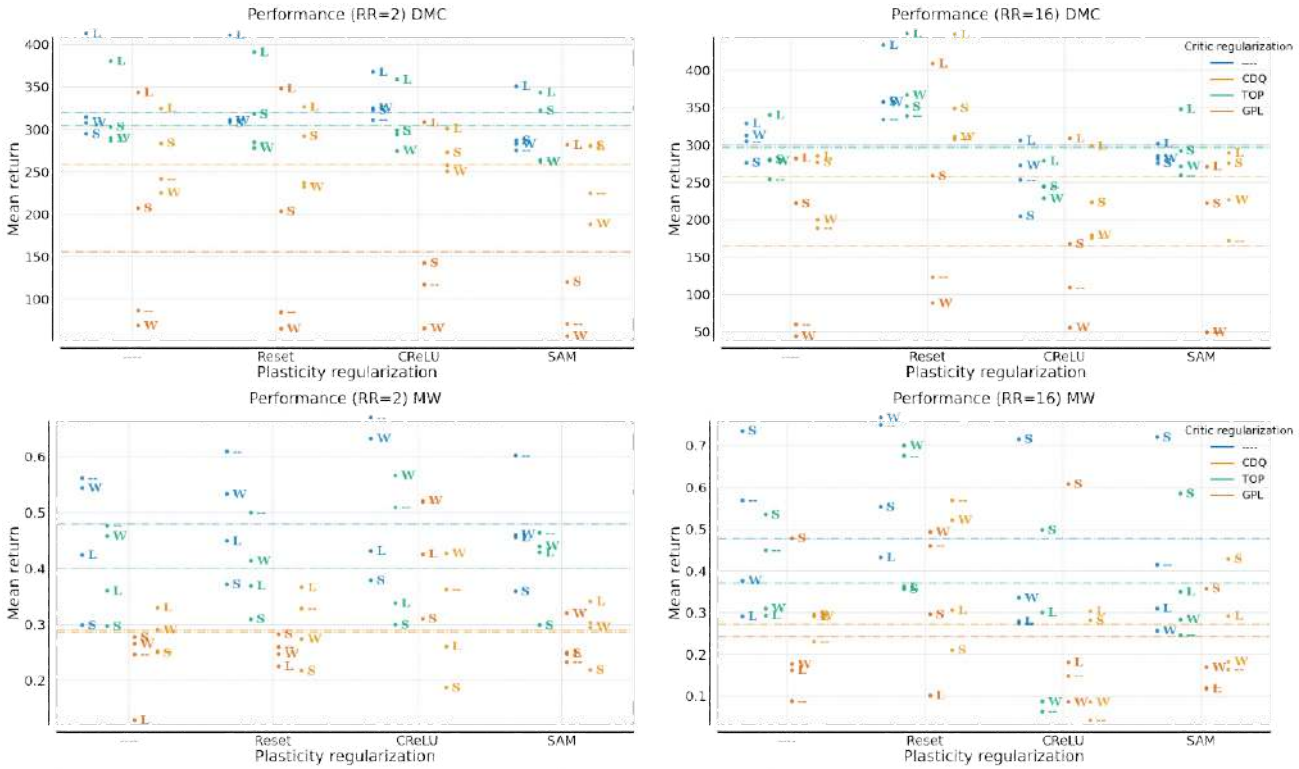


Figure 8. DMC (top) MW (bottom).

optimal combinations with the Reset method on RR=16 and consistently performing well across various combinations on RR=2 (bottom-left plot in Figure 8).

In a scenario with a high replay ratio, the resets are the most important intervention. They work best in combination with Layer norm, but other neural network regularization methods combined with resets are at the forefront regarding performance in the rr=16 scenario (see the top of Figure 8). We present more results in Figures 22 and 23.

C.2. Gradient Norm Analysis

Drawing insights from the findings in section 4.2, where we highlighted the significance of spectral norm in enhancing agent performance on the MW benchmark, we now delve deeper into the behavior of the critic's gradient norm. In Figure 9, one can compare orders of magnitude of IQM of gradient norm with respect to different replay ratio (RR) regimes and on different benchmarks. Referring to results from section 4.2, Figure 9 underscores that the gradient norm on MetaWorld, particularly in the RR=16 setup, exhibits orders of magnitude higher values compared to the DMC benchmark.

A similar phenomenon can be observed on the DMC benchmark, but the layer norm proves more robust in mitigating exploding gradients than the Spectral Norm. Interestingly, training on very complex environments such as Dog causes enormous gradient explosion, even in a small replay ratio regime 10.

C.3. Comparison of ReDO to other plasticity-inducing methods

The ReDO method, as proposed by Sokar et al. (Sokar et al., 2023), is a technique for inducing network plasticity. It shares similarities with the full reset approach but involves more targeted interventions within the network. In particular, ReDo does not reset the full network; it only resets weights connected to dormant neurons. Specifically, incoming weights to dormant neurons are initialized as in full reset, but outgoing weights from dormant neurons are zeroed out, resulting in less severe network output changes.

Figure 11 presents results from Figure2 updated with the ReDo method for both MetaWorld and DMC environments for

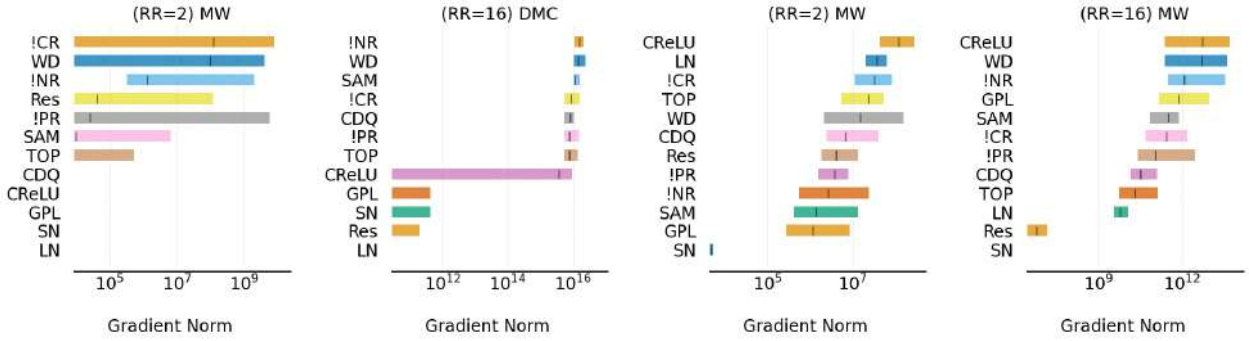


Figure 9. IQM gradient norm of first-order results. The IQM is calculated based on the average of the last ten evaluation points in each run, not the last evaluation point.

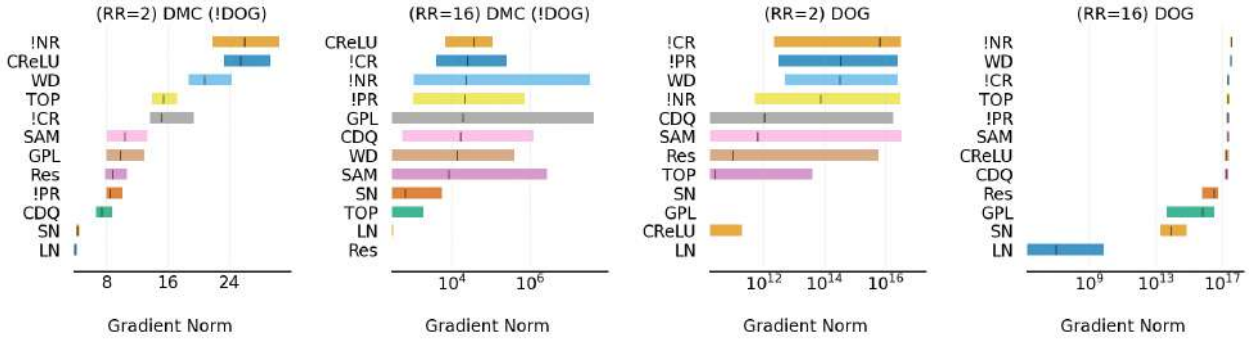


Figure 10. IQM gradient norm of first-order results. The IQM is calculated based on the average of the last ten evaluation points in each run, not the last evaluation point.

a high replay ratio regime. ReDo does not perform as well as resets in the Metaworld and DMC suites. However, both methods effectively reduce the critic gradient norm, overestimation and the number of dormant neurons for both Metaworld and DMC without dog benchmarks, as shown in Figure 12. In dog environments, we observed that ReDo was unstable for runs without SN or LN, and some runs crashed. We report results for the last ten timesteps before the crash for these runs. Interestingly, all methods except SN and LN cannot reduce the critic gradient norm, as shown in the bottom right plot in Figure 12.

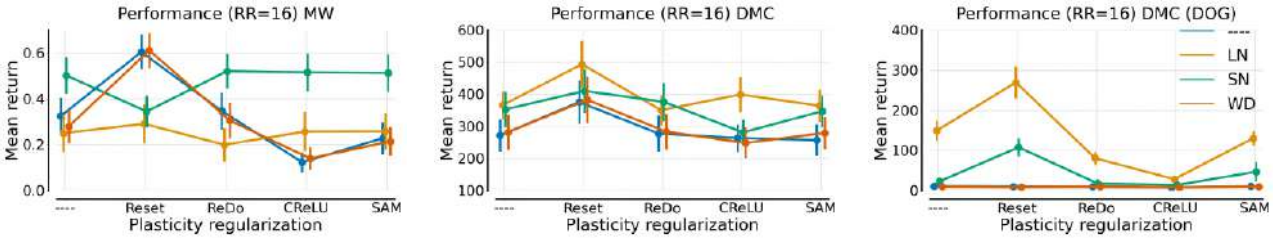


Figure 11. Second-order results marginalizing critic regularization methods with ReDO.

C.4. Closer look on CDQL performance on Hopper and Quadruped environments

Figure 13 illustrates the outcomes of applying Clipped Double Q-learning (CDQL) (Haarnoja et al., 2018; Fujimoto et al., 2018; Hansen et al., 2022), a widely used and straightforward critic regularization method, across various environments. In the case of the hopper hop environment or some MW environments (bottom row), CDQL exhibits a detrimental effect on performance, with additional regularization techniques such as resets or layer normalization failing to alleviate this effect. Similarly, in the quadruped run task, CDQL demonstrates a comparable negative impact, although the application of

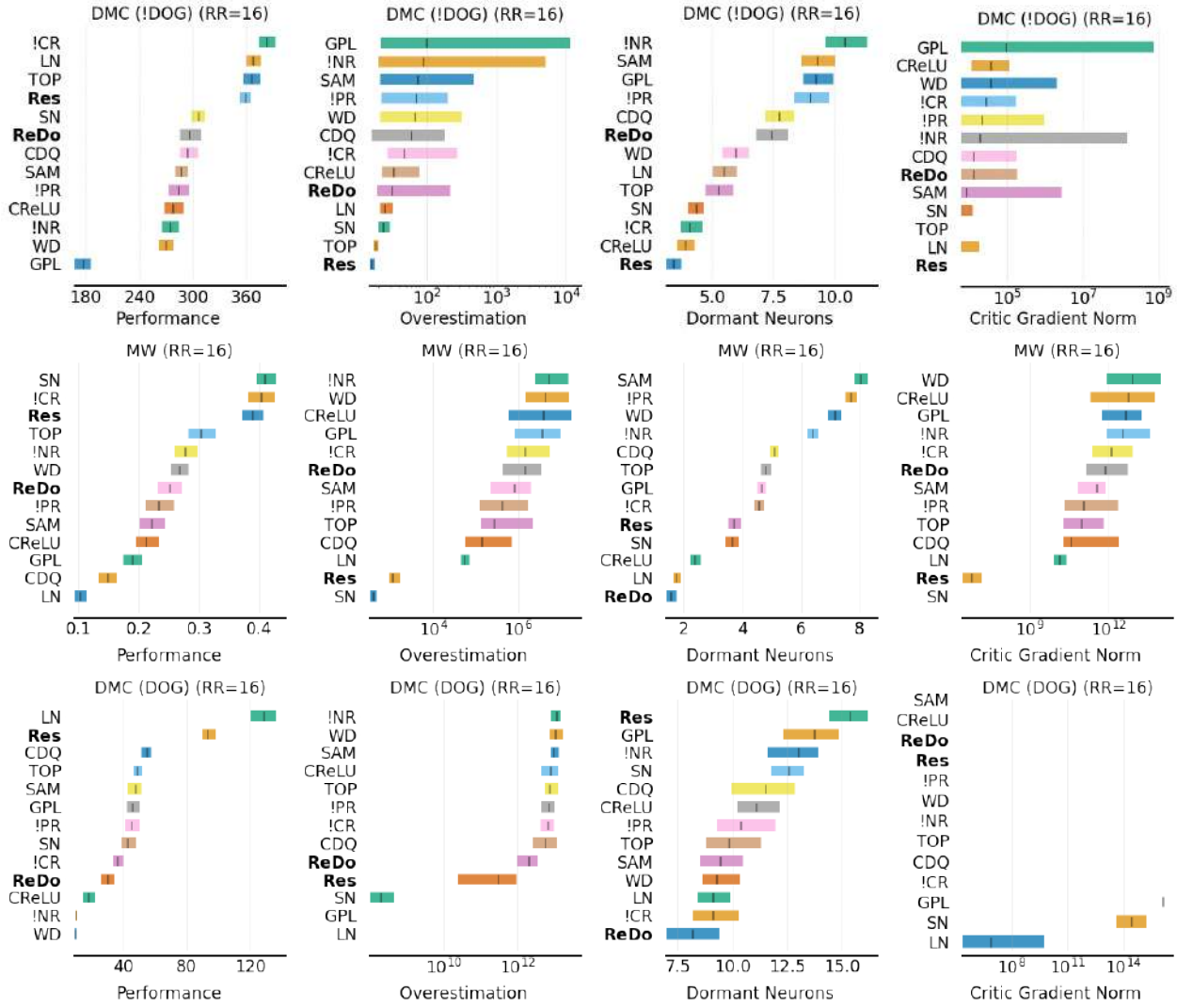


Figure 12. IQM of performance, overestimation, dormant neurons, and gradient norm, of first-order results. The IQM is calculated based on the average of the last ten evaluation points in each run, not the last evaluation point.

additional regularization methods successfully mitigates it.

C.5. Regression and Spearman correlation analysis

Deepening analysis from section 4.4, we investigate the coefficients of Ordinary Least Square regression for overestimation, overfitting, dormant neurons and critic's gradient norm with return. Except for dormant neurons, we take the symmetrical logarithm of every metric. We perform regression fitting for every variable separately, which results in four different regression analyses. As previously stated, because these relations are highly non-linear, however, often monotonic, goodness of fit coefficients such as R^2 for linear regression are low. Besides regression coefficients (β), we report a standard error of regression (se), R^2, p-value p in the plot legend. In the axis titles, the Spearman correlation and its p-value are shown.

In figures 14 and 15, we observe consistent negative coefficients, except for the Coffee environment for overestimation and return. Interestingly, for the MetaWorld benchmark, most environments exhibit a symmetrical degree of overestimation and underestimation as two distinct clusters of well-performing runs are mirrored by the Y axis. For the DMC environments, we observe a much higher amount of runs in which overestimation is present.

For the gradient norm, we observe a higher value of this metric for runs with RR=16 both for DMC and MW benchmarks

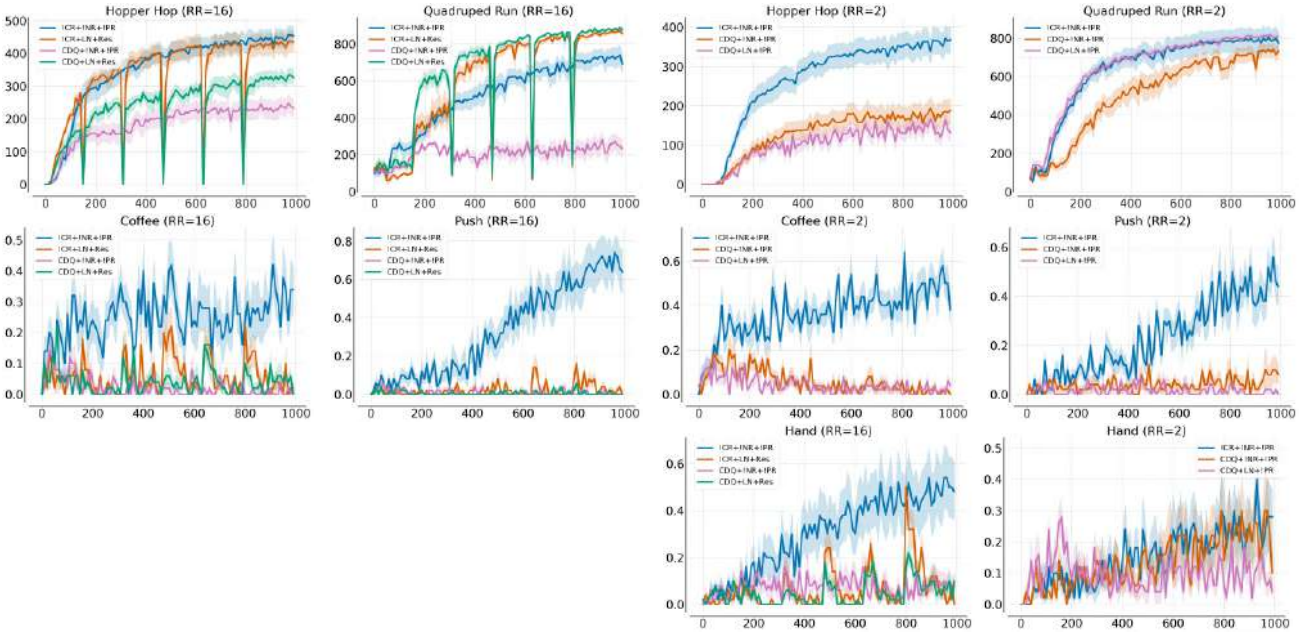


Figure 13. Examining the influence of CDQL on performance in the Hopper Hop and Quadruped Run environments within the DMC benchmark (top row), as well as the Push, Coffee, and Hand environments within the MW benchmark (bottom row).

(Figures 16 and 17. There is consistency in coefficients for MW; however, in DMC, we observe that only four out of 7 environments exhibit negative signs of the coefficient. However, for most of the extremely high values of gradient norm, we systematically observe low returns regardless of the environment.

Dormant neurons, presented in Figures 18 and 19, clearly correlate negatively with the return, especially for DMC environments. However, there are cases, such as the Reach environment from MetaWorld, where a high percentage of dormant neurons benefit the agent, probably because this particular environment is especially easy.

Overfitting is the metric that exhibits the highest variance in coefficient signs across environments and benchmarks. As shown in Figures 20 and 21, the best-performing runs are located for low absolute values of overfitting. In addition, RR=16 clearly results in higher overfitting values.

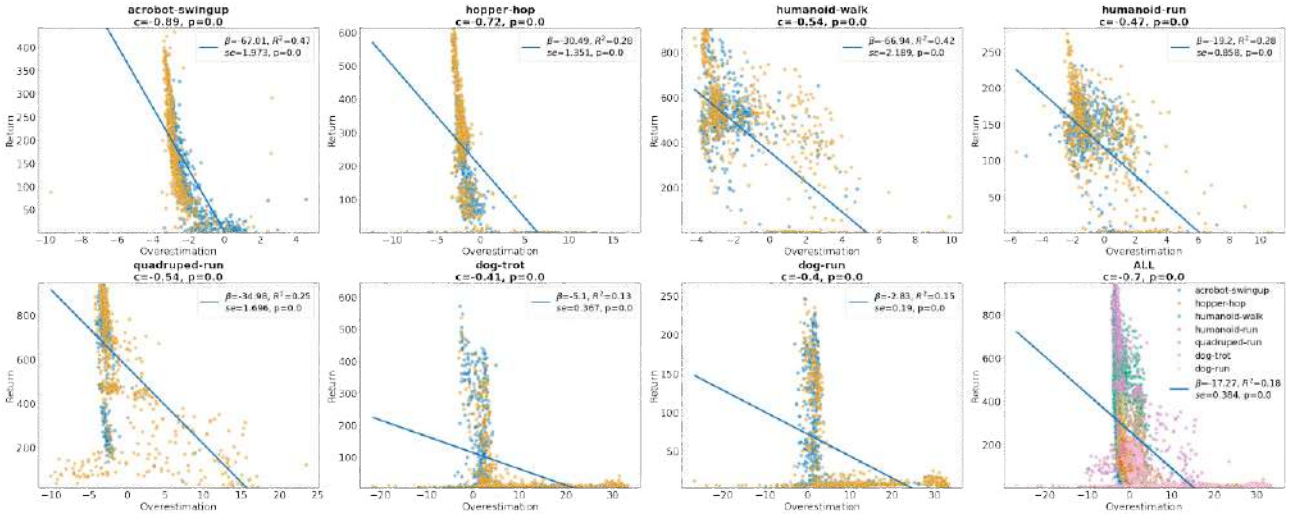


Figure 14. Overestimation logarithm scatter plots with regression line for DMC environments.

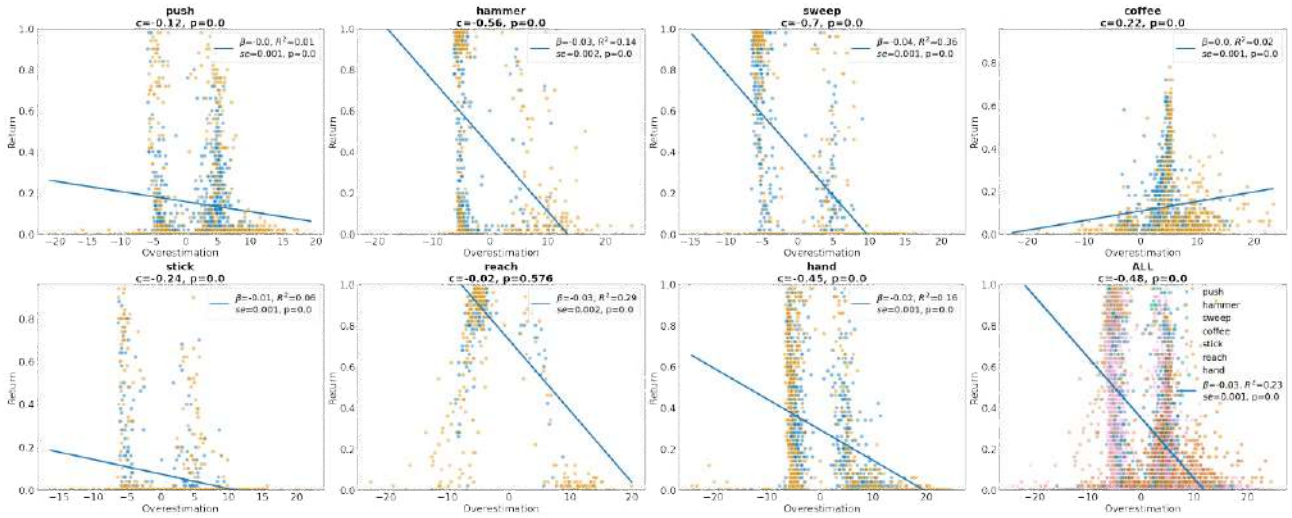


Figure 15. Overestimation logarithm scatter plots with regression line for MW environments.

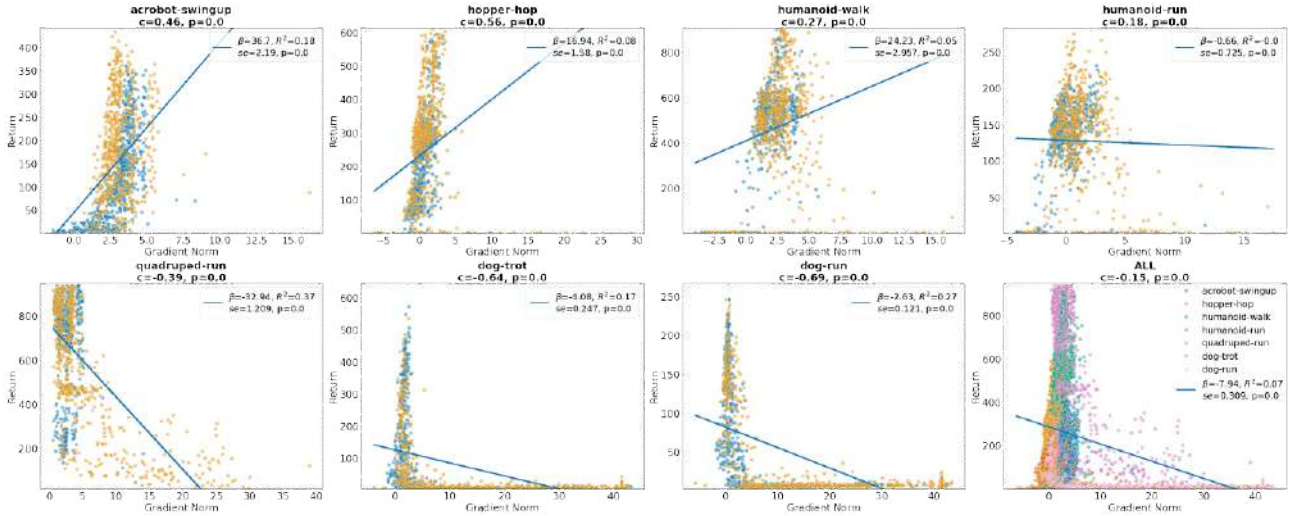


Figure 16. Gradient norm logarithm scatter plots with regression line for DMC environments.

C.6. Image-based DeepMind Control

We test whether the results achieved on proprioceptive state representation transfer to image-based control. To this end, we run 4 versions of the DrQ agent (Yarats et al., 2020):

1. Vanilla DrQ (DrQ)
2. DrQ with layer normalization on the critic network (DrQ + LN)
3. DrQ with full-parameter resets every 200k environment steps (DrQ + Res)
4. DrQ with both normalization on the critic network and full-parameter resets every 200k environment steps (DrQ + LN + Res)

We run these variations on 6 tasks from the DeepMind Control benchmark: Acrobot Swingup, Cheetah Run, Hopper Hop, Humanoid Run, Humanoid Stand, and Humanoid Walk. We run the humanoid tasks for 3mln frames and the other tasks for

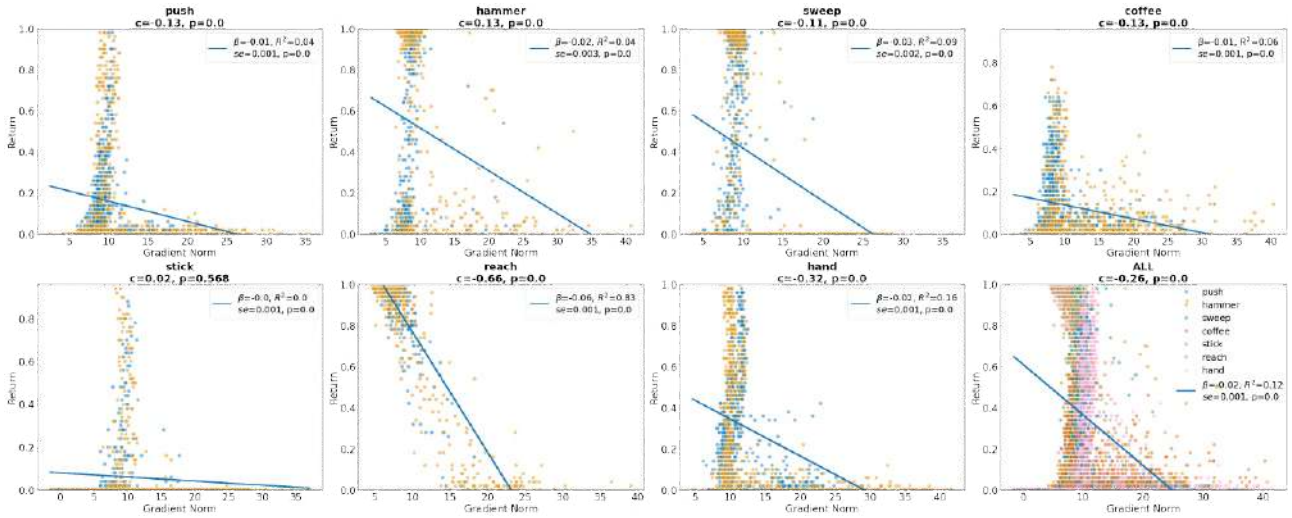


Figure 17. Gradient norm logarithm scatter plots with regression line for MW environments.

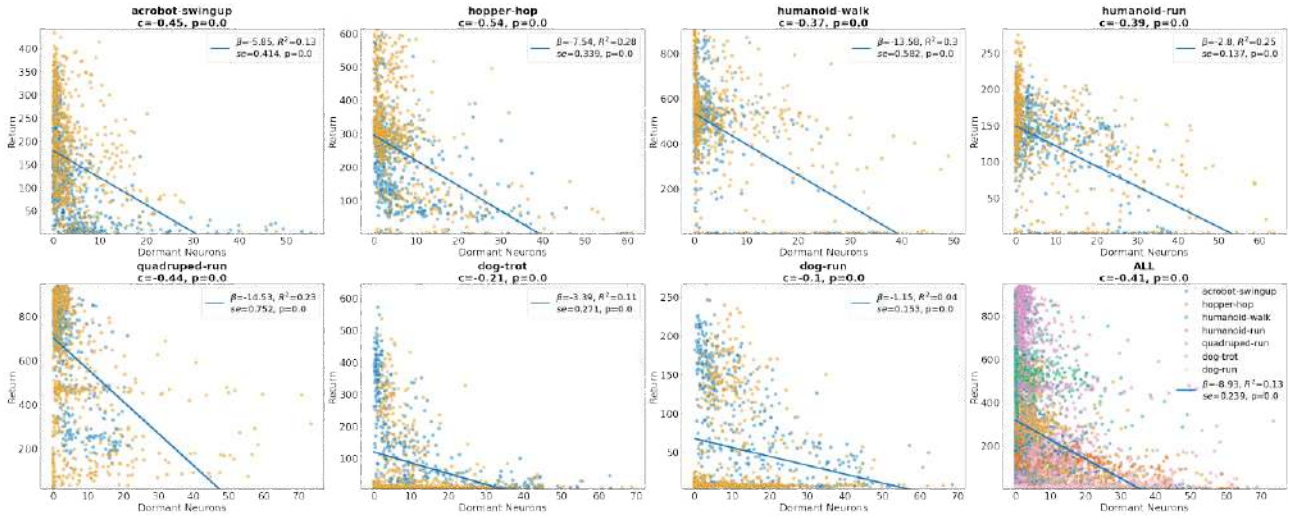


Figure 18. Dormant neurons scatter plots with regression line for DMC environments.

1mln frames with a replay ratio of 1. We calculate the relationship between frames and environment steps according to the methodology presented in Yarats et al. (2020). We present the results in the table below.

Unfortunately, the humanoid agents were mostly unable to achieve non-random policies in the budget of 3mln frames. Interestingly, the proprioceptive results do not seem to directly transfer to image-based agents with a low-replay ratio. As such, we believe the image-based benchmark requires further studies.

C.7. Best combinations of intervention performance plots

C.8. Other

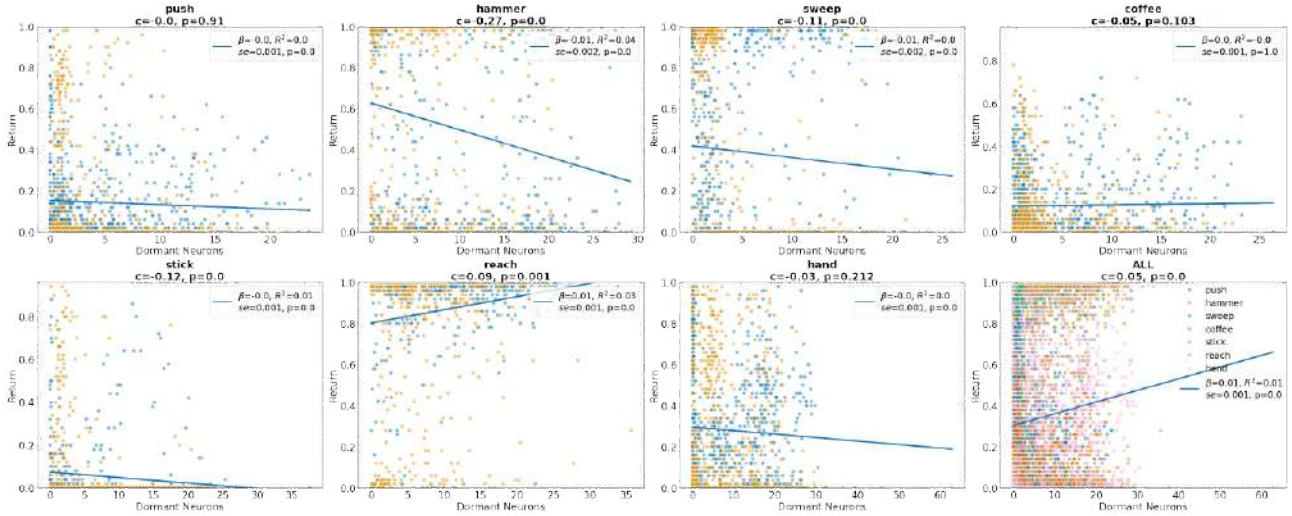


Figure 19. Dormant neurons scatter plots with regression line for MW environments.

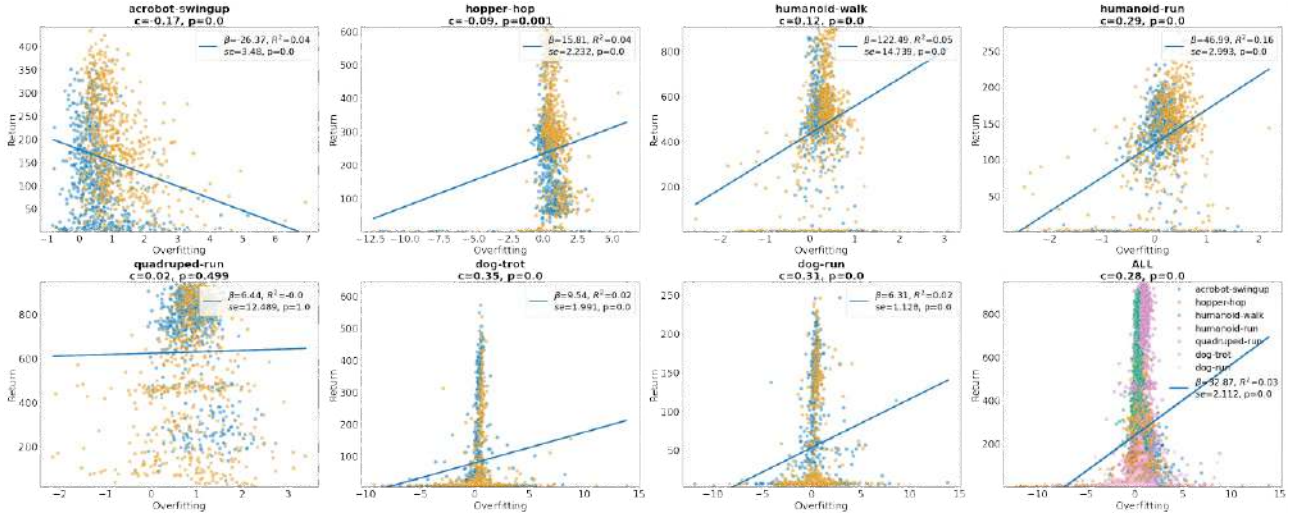


Figure 20. Overfitting logarithm scatter plots with regression line for DMC environments.

Table 2. Final performance in image-based environments. 3 seeds per task.

	DrQ	**DrQ + LN**	**DrQ + Res**	**DrQ + LN + Res**
Acrobot Swingup	172.9 ± 22.1	87.6 ± 19.7	52.3 ± 11.2	88.8 ± 8.2
Cheetah Run	727.4 ± 7.3	680.0 ± 2.3	715.6 ± 7.1	653.9 ± 2.1
Hopper Hop	74.6 ± 34.9	116.3 ± 27.2	135.7 ± 24.1	167.3 ± 14.3
Humanoid Stand	7.8 ± 0.2	8.1 ± 0.2	7.5 ± 0.4	7.8 ± 0.4

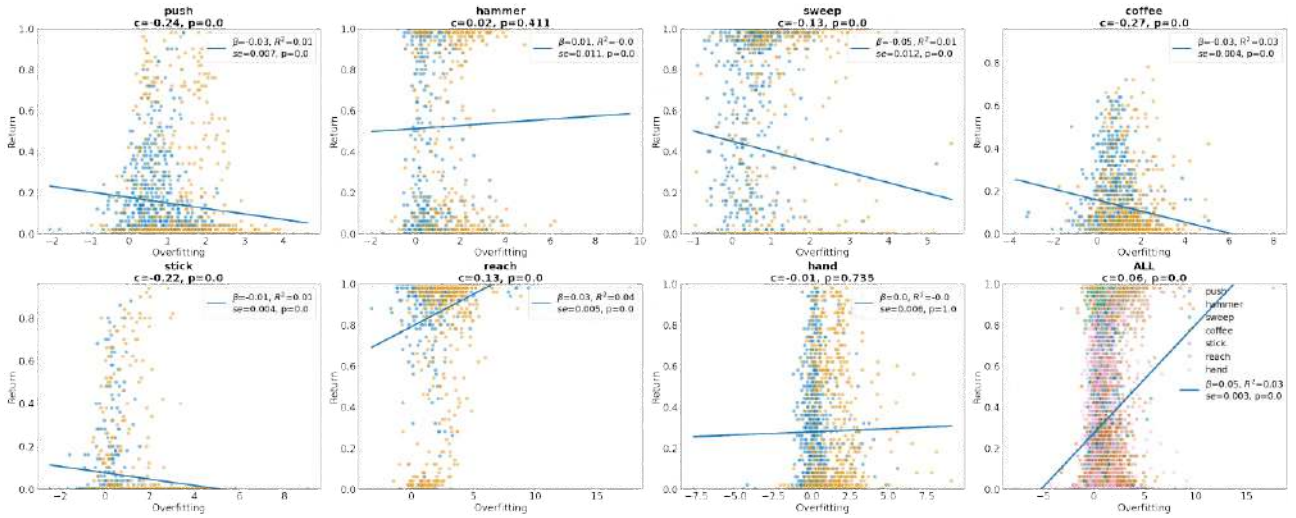


Figure 21. Overfitting logarithm scatter plots with regression line for MW environments.

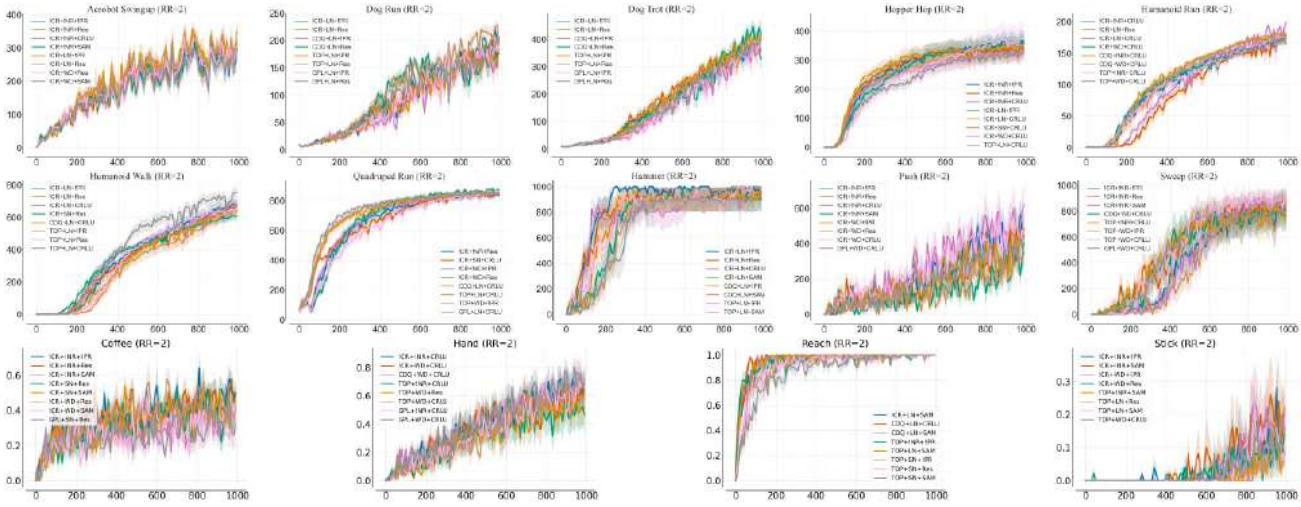


Figure 22. Top performing configuration in the low replay regime. 10 seeds per task per algorithm.

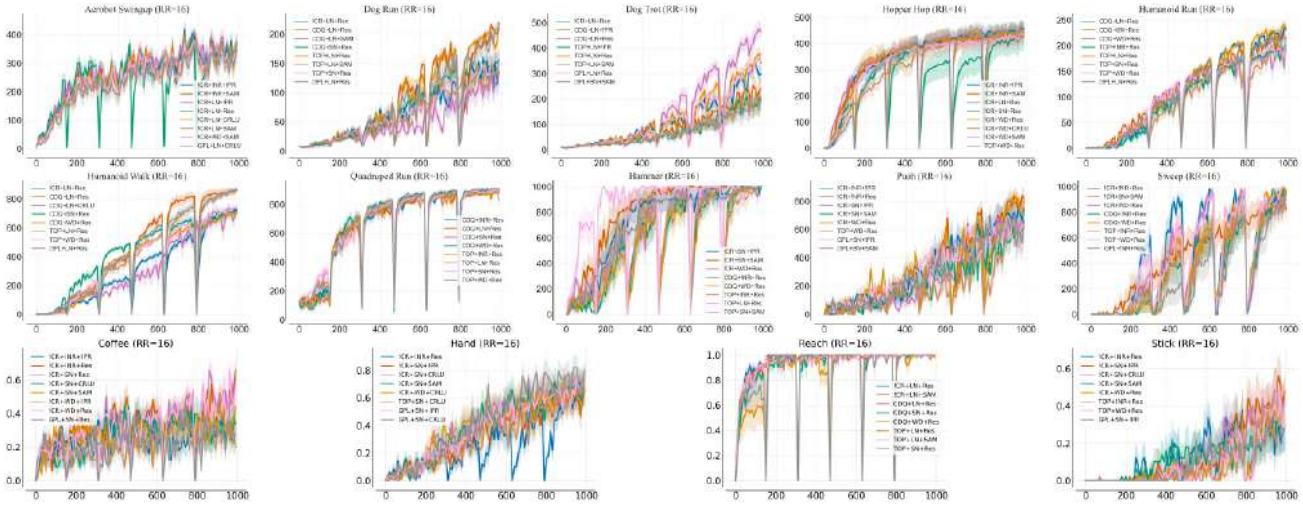


Figure 23. Top performing configuration in the high replay regime. 10 seeds per task per algorithm.

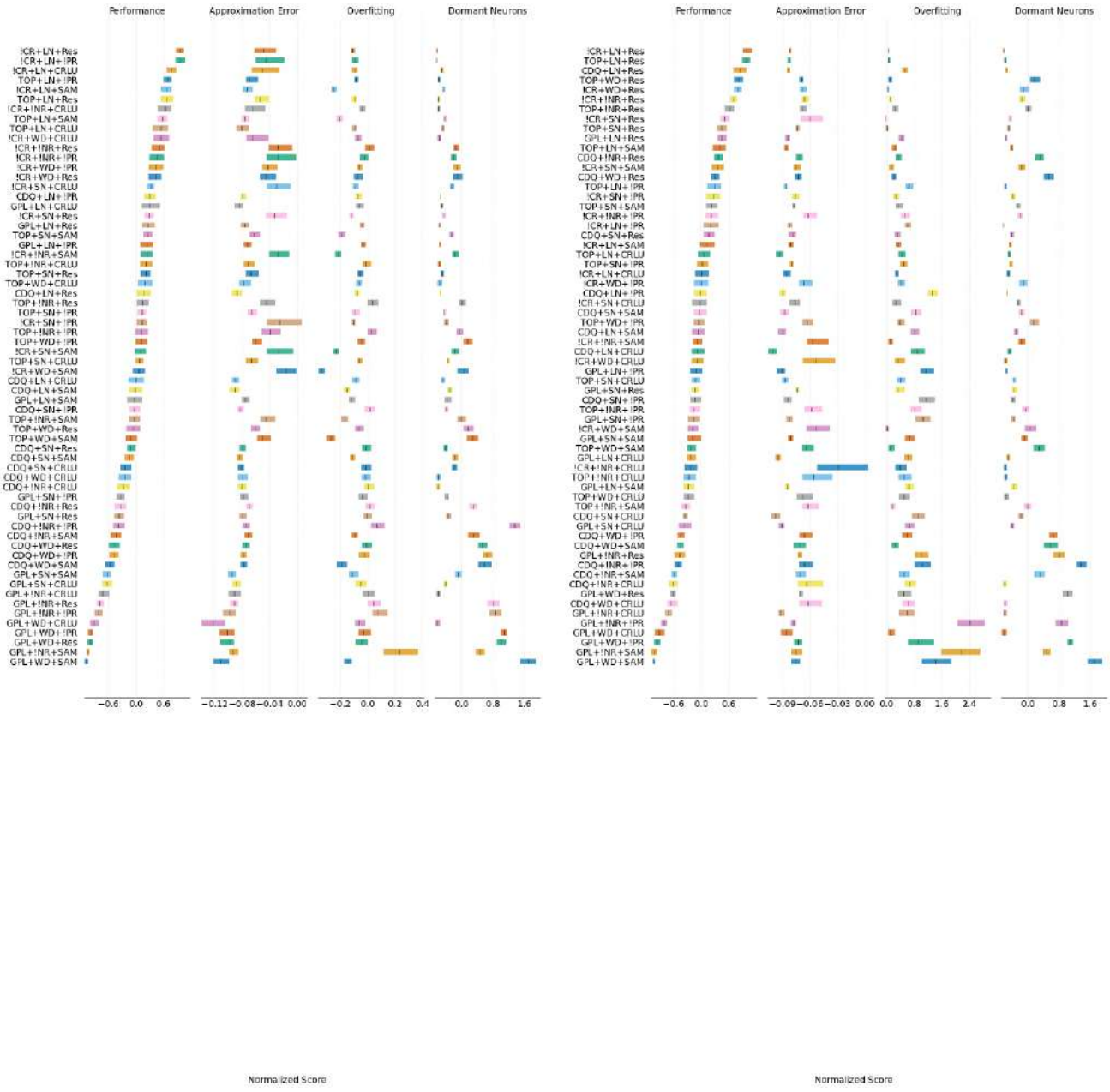


Figure 24. Top performing configuration in the low (left) and high (right) replay regime. 10 seeds per task per algorithm.

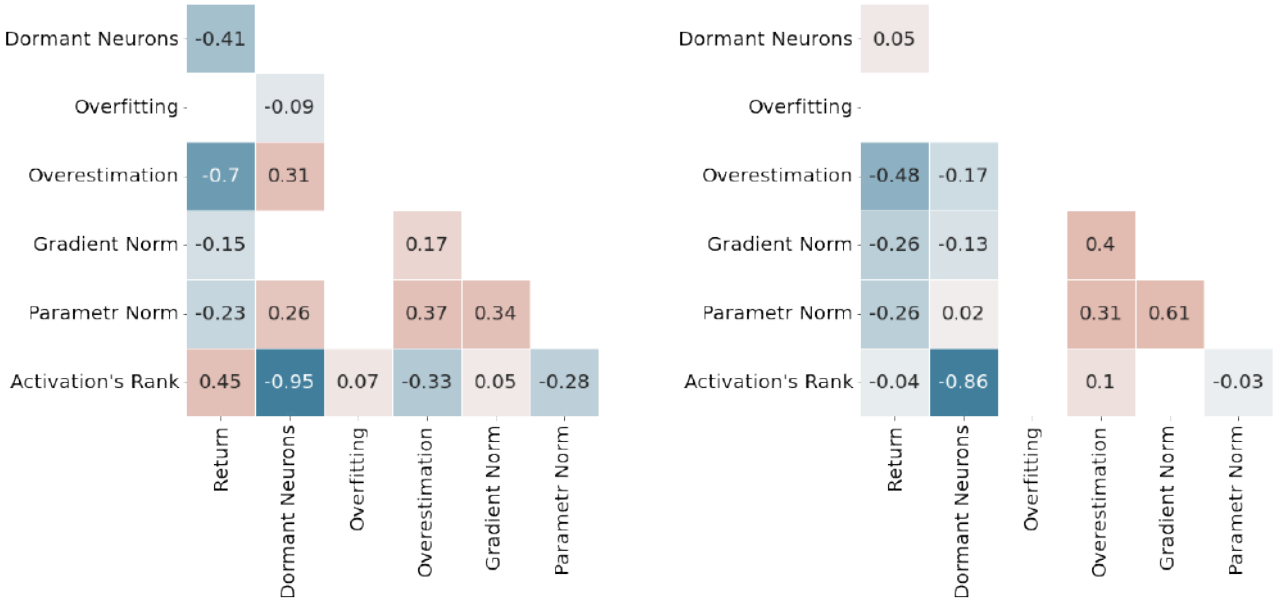


Figure 25. Spearman correlation matrix for explanatory metrics on DMC benchmark (left) and on MetaWorld benchmark (roght plot). Blank spaces are correlations that do not meet the p-value.

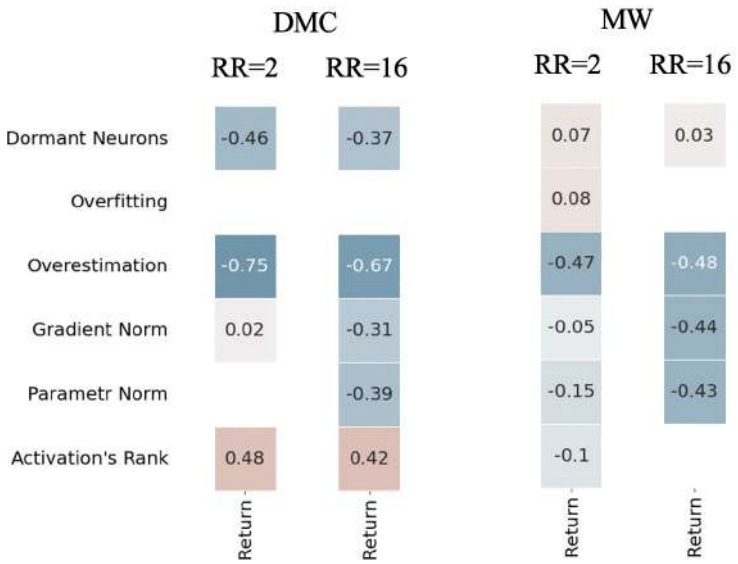


Figure 26. Spearman correlation for different replay ratios.

Decoupled Policy Actor-Critic: Bridging Pessimism and Risk Awareness in Reinforcement Learning

Michal Nauman[1], **Marek Cygan**[1,2]
[1]University of Warsaw
[2]Nomagic
nauman.mic@gmail.com, cygan@mimuw.edu.pl

Abstract

Actor-Critic (AC) algorithms like SAC and TD3 were shown to perform well in a variety of continuous-action tasks. However, the theoretical basis for the pessimistic objectives these algorithms employ remains unestablished, raising questions about the specific class of policies they are implementing. In this work, we apply the expected utility hypothesis, a fundamental concept in economics, to illustrate that both pessimistic and non-pessimistic RL objectives can be interpreted through expected utility maximization using an exponential utility function. This approach reveals that pessimistic policies effectively maximize value certainty equivalent, aligning them with the optimization of risk-aware objectives. Furthermore, we propose Decoupled Policy Actor-Critic (DAC). DAC is a model-free algorithm that features two distinct actor networks: a pessimistic actor for temporal-difference learning and an optimistic actor for exploration. Our evaluations of DAC across various locomotion and manipulation tasks demonstrate improvements in sample efficiency and final performance. Remarkably, DAC, while requiring significantly fewer computational resources, matches the performance of leading model-based methods in the complex dog and humanoid domains.

Introduction

Deep Reinforcement Learning (RL) is still in its infancy, with a variety of tasks unsolved (Sutton and Barto 2018; Hafner et al. 2023) or solved within an unsatisfactory amount of environment interactions (Zawalski et al. 2022; Schwarzer et al. 2023). Whereas increasing the Replay Ratio (RR) (ie. the number of parameter updates per environment interactions step) is a promising general approach for increasing sample efficiency and final performance of RL agents (Janner et al. 2019; Chen et al. 2020; Nikishin et al. 2022), it is characterized by quickly diminishing gains (D'Oro et al. 2022) combined with linearly increasing computational cost (Rumelhart, Hinton, and Williams 1986; Kingma and Ba 2014). Moreover, the limitations of robot hardware and data acquisition frequency constrain the maximum achievable replay ratio (Smith, Kostrikov, and Levine 2022). As such, it is worthwhile to pursue techniques that can be applied across a variety of RR configurations. One continuously researched theme is how a particular algorithm handles the *exploration-exploitation* dilemma (Ciosek et al. 2019; Moskovitz et al. 2021).

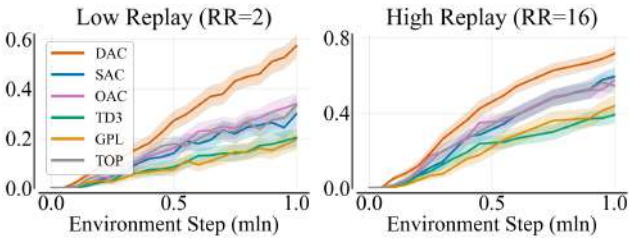


Figure 1: We test the proposed approach (DAC) against various pessimistic and optimistic actor-critic baselines in 30 tasks listed in Table 3. Y-axis reports IQM with 95% CI calculated using 10 seeds, with 1.0 being the maximal score.

Optimism and pessimism of algorithmic agents have been researched in multiple contexts. For instance, optimism in the face of uncertainty was shown to be an effective exploration strategy (Wang et al. 2020; Neu and Pike-Burke 2020). Conversely, pessimistic Q-learning strategies have proven beneficial in counteracting value overestimation caused by temporal difference errors (Hasselt 2010; Fujimoto, Hoof, and Meger 2018). However, there is a disconnect between these strategies and the foundational theories of RL, particularly in how they relate to the goal of value maximization. As a result, despite the empirical success of pessimistic agents like TD3 (Fujimoto, Hoof, and Meger 2018) or SAC (Haarnoja et al. 2018a), the specific class of policies they implement remains ambiguous.

In this paper, we study the reinforcement learning objective aligned with the principles of decision theory. We show that, in contrast to pure value maximization, the decision-theoretic perspective allows for the derivation of both non-pessimistic (e.g., DDPG (Silver et al. 2014)), pessimistic (e.g., TD3 (Fujimoto, Hoof, and Meger 2018)), and optimistic (e.g., OAC (Ciosek et al. 2019)) objectives. As such, we demonstrate that the pessimistic updates in state-of-the-art algorithms such as SAC (Haarnoja et al. 2018a), REDQ (Chen et al. 2020), or TOP (Moskovitz et al. 2021) can be derived from expected utility maximization under an exponential utility function (Fei, Yang, and Wang 2021). Further-

more, we introduce Decoupled Policy Actor-Critic (DAC), an off-policy algorithm with two actors: optimistic and pessimistic. In DAC, each actor is trained independently using gradient backpropagation of a distinct objective: the optimistic actor aims to maximize an upper Q-value bound for exploration, while the pessimistic actor optimizes a lower Q-value bound for temporal-difference (TD) learning (Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018a). DAC also features an automatic adjustment mechanism to balance the divergence between these actors, allowing for adaptability to various tasks without hyperparameter tuning. We evaluate DAC on a diverse set of locomotion and manipulation tasks and find that, despite its simplicity, DAC achieves significant improvements in sample efficiency and performance. Below, we outline our contributions:

1. We consider a generalized utility-based actor-critic objective, capable of formalizing both pessimistic and optimistic actor-critic algorithms. We demonstrate that the policies enacted by Clipped Double Q-Learning and generalizations thereof approximately align with the certainty equivalent of value under an exponential utility, and as such are optimal in the decision-theoretic context.

2. We introduce Decoupled Policy Actor-Critic (DAC), an off-policy actor-critic framework featuring a decoupled actor network configuration. In DAC, each actor is trained through gradient backpropagation stemming from a specific objective that reflects different degrees of risk appetite. We establish the optimistic policy loss function and implement a system for online gradient-based adjustment of optimism hyperparameters. This feature enables DAC to effectively adapt to varying degrees of uncertainties, as well as different reward scales, without the need for manual hyperparameter tuning of the optimism levels.

3. We show that DAC outperforms benchmark algorithms in terms of both sample efficiency and final performance. Notably, DAC solves the dog domain, reaching the performance of significantly more complex model-based methods. To facilitate further research, we perform ablations on various design and hyperparameter choices.

Background

Reinforcement Learning We consider an infinite-horizon Markov Decision Process (MDP) (Puterman 2014) which is described with a tuple (S, A, r, p, γ), where states S and actions A are continuous, $r_{s,a}$ is the transition reward, p is a deterministic transition mapping, with p_0 being the starting state distribution, and $\gamma \in (0, 1]$ is a discount factor. A policy $\pi(a|s)$ is a state-conditioned action distribution with entropy denoted as $\mathcal{H}^\pi(s)$. Soft Value (Haarnoja et al. 2018a) is the sum of expected discounted return and policy entropies from following the policy at a given state $V^\pi(s) = \mathrm{E}_{a\sim\pi}\left(r_{s,a} + \alpha\mathcal{H}^\pi(s) + \gamma V^\pi(s')\right)$, with α denoting the entropy temperature parameter. Q-value is the expected discounted return from performing an action and following the policy thereafter $Q^\pi(s,a) = r_{s,a} + \gamma V^\pi(s')$. A policy is said to be optimal if it maximizes the expected value of the possible starting states s_0, such that $\pi^* = \arg\max_{\pi\in\Pi} \mathrm{E}_{s_0\sim p_0}\, V^\pi(s_0)$, with π^* denoting the optimal policy and Π denoting the considered set of policies (eg. Gaussian). Off-policy actor-critic algorithms perform gradient-based learning of both Q-values (ie. critic or value model, denoted as Q_θ) and the policy (ie. actor, denoted as π_ϕ). The critic parameters θ are updated by minimizing SARSA temporal-difference loss $\mathcal{L}_\theta$ on transitions $T = (s, a, r, s')$ which are sampled from past experiences (Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018a) according to $\mathcal{L}_\theta = \mathrm{E}_{T\sim\mathcal{D}}\ (Q_\theta(s,a) - r_{s,a} - \gamma V_\theta(s'))^2$. Here, $Q_\theta(s,a)$ is the critic output, $V_\theta(s')$ is the bootstrap value derived from a target network, and $\mathcal{D}$ is the experience buffer (Mnih et al. 2015). The policy parameters ϕ are updated to maximize values approximated by the critic (Ciosek and Whiteson 2020): $\mathcal{L}_\phi = \mathrm{E}_{s\sim\mathcal{D}}(\alpha\log\pi_\phi(a|s) - Q_\theta(s,a))$ with $a \sim \pi_\phi(s)$.

Pessimism in Actor-Critic Algorithms like SAC or TD3 calculate actor-critic gradients with respect to the lower bound Q-values (Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018b). Employing the lower-bound has proven effective in mitigating value overestimation in Temporal Difference (TD) learning (Van Hasselt, Guez, and Silver 2016; Fujimoto, Hoof, and Meger 2018). A popular approach is Clipped Double Q-Learning (CDQ), where the lower bound is calculated by taking the minimum value of an ensemble of critics, most often two (Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018b; Ciosek et al. 2019; Hansen, Wang, and Su 2022): $V_\theta(s) \approx \min(Q^1_\theta(s,a), Q^2_\theta(s,a)) - \log\pi_\phi(a|s)$, with $a \sim \pi_\phi(s)$. Here, $Q^1_\theta(s,a)$ and $Q^2_\theta(s,a)$ denote the first and the second critic in the ensemble, and $\log\pi_\phi$ is the state-action entropy. It was shown that using the minimum is equivalent to evaluating the ensemble statistics (Ciosek et al. 2019):

$$\min(Q^1_\theta(s,a), Q^2_\theta(s,a)) = Q^\mu_\theta(s,a) - Q^\sigma_\theta(s,a) \tag{1}$$

Where (μ, σ) denote the mean and standard deviation of the critic model ensemble and are given by:

$$\begin{aligned} Q^\mu_\theta(s,a) &= \frac{Q^1_\theta(s,a) + Q^2_\theta(s,a)}{2} \\ Q^\sigma_\theta(s,a) &= \frac{|Q^1_\theta(s,a) - Q^2_\theta(s,a)|}{2} \end{aligned} \tag{2}$$

This led to generalizations of the lower-bound as to include varying levels of pessimism and optimism (Ciosek et al. 2019; Moskovitz et al. 2021):

$$\begin{aligned} Q^\beta_\theta(s,a) &= Q^\mu_\theta(s,a) + \beta Q^\sigma_\theta(s,a) \\ V^\beta_\theta(s) &= V^\mu_\theta(s) + \beta V^\sigma_\theta(s) \end{aligned} \tag{3}$$

Above, $Q^\beta_\theta(s,a)$ and $V^\beta_\theta(s)$ are the pessimistic Q-values and values respectively, β represents the degree of optimism. The pessimistic values and Q-values are related by $V^\beta(s) = \mathrm{E}_{a\sim\pi}(Q^\beta(s,a) - \log\pi_\theta(a|s))$. In this setup, the level of pessimism is modulated by β. A non-zero β indicates a departure from neutrality: positive β yields an optimistic upper bound, while negative β results in a pessimistic

lower bound. Furthermore, as shown in Equation 1, in case of $\beta = -1$ the objective collapses to CDQ, and it is true that $Q^{\beta}_{\theta} = \min(Q^1_\theta(s,a), Q^2_\theta(s,a))$. In the setup of pessimistic learning the value targets are affected by the uncertainty measured via the value model ensemble disagreement. Thus, given some policy π_ϕ the difference between the pessimistic value (resulting from bootstrapping with Q^β), and neutral values (resulting from bootstrapping with Q^μ) is proportional to the expected values of the discounted sum of critic disagreements (Fujimoto, Hoof, and Meger 2018). As such, it is clear that iterating the pessimistic objective does not result in the optimal policy as long as the critic disagreement is not equal to zero for all state actions (Kumar, Gupta, and Levine 2020). Despite this, numerous effective off-policy algorithms adopt a non-neutral objective, leaning towards either pessimism or optimism (Fujimoto, Hoof, and Meger 2018; Ciosek et al. 2019; Cetin and Celiktutan 2023).

The policy derived from Equation 3 deviates from the optimal policy stemming from pure value maximization. This can be observed by unrolling the Bellman Equation used to train the critic. Whereas a non-pessimistic critic learns the sum of discounted returns and entropies, the pessimistic critic learns the critic disagreements as well:

$$\begin{aligned} &Q_\theta(s,a) \approx r_{s,a} + \gamma V^{\beta}_{\theta}(s') \\ &\approx r_{s,a} + \gamma(r_{s',a'} + \alpha\mathcal{H}^{\pi_\phi}(s) + V^{\sigma}_{\theta}(s') + \gamma V_\theta(s'')) \end{aligned} \tag{4}$$

Expected Utility Theorem The Von Neumann Morgenstern Theorem (Von Neumann and Morgenstern 1947) posits that an agent whose preferences adhere to four axioms (completeness, transitivity, continuity, and independence), has a utility function $\mathcal{U}(x)$ that enables the comparison of the preferences. Expected Utility Hypothesis (EUH) (Von Neumann and Morgenstern 1947; Kahneman and Tversky 1979) provides a framework for modeling decisions where outcomes are uncertain and as such is often the framework of choice to model risk-aware behavior. EUH states that agents choose between risky options by comparing their expected utility. For example, given random variables X_i representing different propositions the agent ought to choose from, it follows that $X_1 \succeq X_2 \iff \mathrm{E}\, U(X_1) \geq \mathrm{E}\, U(X_2)$. In such a setting, the goal of the agent is to optimize the utility rather than its input (Stiglitz 1997): $x^* = \arg\max_x \mathrm{E}_{x_i \sim X}\, \mathcal{U}(x_i)$. Here, the risk stems from the potential decrease in utility due to uncertainty in its input space, making risk preference an attribute of the utility function under consideration. In particular, due to potential non-linearities in the utility, there can be a discrepancy between $\arg\max \mathrm{E}\, U(x_i)$ and $\arg\max \mathrm{E}\, x_i$. This discrepancy is reffered to as risk premium and we denote it by Υ. Certainty equivalent of X, denoted as X_c, measures the impact of uncertainty on utility-optimal choices: $\mathcal{U}(X_c) = \mathrm{E}\, \mathcal{U}(X)$. Certainty equivalent presents a deterministic amount offering the same utility as a random event and varies based on risk preferences. Risk-averse utilities yield a certainty equivalent lower than the expected value while risk-seeking leads to a higher certainty equivalent. The exponential utility, $\mathcal{U}(x,\beta) = e^{\beta x}$, is a simple model for risk awareness, with β determining the risk preference. We show the basic risk-averse and risk-loving utilities, along with their implied risk premium in Figure 5.

Theory of Pessimism in Actor-Critic

Notably, many state-of-the-art algorithms adopt pessimistic update strategies (Moskovitz et al. 2021; Hiraoka et al. 2021; D'Oro et al. 2022). However, the pessimistic correction $V^{\sigma}_{\theta}(s')$, as outlined in the previous Section, is not fully explicated by existing RL theory and does not emerge from pure value maximization problems. To this end, the efficacy of pessimistic or optimistic agents is often attributed to two factors: pessimism is justified by well-documented value overestimation in temporal learning (Hasselt 2010; Fujimoto, Hoof, and Meger 2018; Kumar, Gupta, and Levine 2020) while optimism is supported by lower regret guarantees for exploration (Chen et al. 2017; Ciosek et al. 2019). In this section, we show that off-policy agents that perform updates according to the pessimistic Q-values Q^β presented in Equation 3 (e.g., SAC (Haarnoja et al. 2018a), TD3 (Fujimoto, Hoof, and Meger 2018), OAC (Ciosek et al. 2019), TOP (Moskovitz et al. 2021) or GPL (Cetin and Celiktutan 2023)) can be interpreted as optimizing for the expected utility of values under an exponential utility function. As such, we align the pessimistic and optimistic objectives with risk-averse and risk-loving behaviour as stemming from an exponential utility. To show this, we consider a cycle of policy evaluation and improvement steps and analyze a single step thereof. In every step of iteration, the values are assumed to be samples from the distribution $\mathcal{V}^{\pi}_{\theta}(s)$, which is assumed to have finite moments and expected value denoted by V^{μ}_{θ}, such that $V^{\mu}_{\theta}(s) = \mathrm{E}_{i\sim\mathcal{V}}\, V^{i}_{\theta}(s)$. Assuming that the expected value of $V^{i}_{\theta}(s)$ is an unbiased estimator of the on-policy value, the standard approach requires the policy to optimize for $V^{\mu}_{\theta}(s)$, such that $\pi^* = \arg\max_{\pi\in\Pi} \mathrm{E}_{s\sim\mathcal{D}} V^{\mu}_{\theta}(s)$. However, this is not the approach applied by the risk-aware algorithms, such as SAC or TD3. Given an invertible, increasing utility function $\mathcal{U}$, we define the *certainty equivalent value*, denoted as $V^{c}_{\theta}(s)$:

$$V^{c}_{\theta}(s) = \mathcal{I} \mathop{\mathrm{E}}_{i\sim\mathcal{V}} \mathcal{U}\, V^{i}_{\theta}(s) = V^{\mu}_{\theta}(s) + \Upsilon(s). \tag{5}$$

Above, the inverted utility function is denoted by $\mathcal{I} = \mathcal{U}^{-1}$ and $\Upsilon(s)$ denotes the risk premium. In this context, the certainty equivalent value represents the deterministic amount that amortizes the uncertainties associated with the value approximation stochasticity. As such, if the utility function implies risk-averse behavior, then $\Upsilon(s) < 0$ resulting in a certainty equivalent value that is smaller than $V^{\mu}_{\theta}(s)$. Building on the expected utility objective, we define the *certainty equivalence policy* that seeks to amortize the uncertainty associated with an approximation of values:

$$\pi^{c}_{\phi} = \arg\max_{\pi\in\Pi} \mathop{\mathrm{E}}_{s\sim p} \mathop{\mathrm{E}}_{i\sim\mathcal{V}} \mathcal{U}\, V^{i}_{\theta}(s) = \arg\max_{\pi\in\Pi} \mathop{\mathrm{E}}_{s\sim p} V^{c}_{\theta}(s). \tag{6}$$

Above, π^{c}_{ϕ} represents the policy that optimizes for the expected utility of values, which we term the *certainty equivalent policy*. As follows, the certainty equivalent policy is

greedy with respect to the certainty equivalent value and aligns with the expected utility hypothesis, wherein the agent seeks to maximize the expected utility of returns. This outcome contrasts with achieving the maximal value possible, as is typically sought in standard RL objectives. The certainty equivalent policy π^c_ϕ optimizes for values that are not generally equal to the risk-neutral values. Specifically, the π^c_ϕ and π^* are equivalent only iff $\Upsilon(s) = 0$, thus making $V^c_\theta(s) = V^\mu_\theta(s)$. This highlights the fact that the utility-based objective is a generalization of the traditional RL objective, with the traditional RL objective being understood as utility maximization under a linear utility function. Hence, risk-neutral algorithms like DDPG can be interpreted as linear utility agents optimizing the certainty equivalent.

To explore scenarios where the utility is non-linear and diverges from the traditional objective, one has to evaluate the risk premium. Given that $\mathcal{U}$ is invertible, infinitely differentiable and its Taylor expansion is convergent, then the risk premium can be evaluated via Taylor series:

$$\Upsilon(s) \approx \mathcal{I} \mathop{\mathrm{E}}_{i\sim\mathcal{V}} \sum_{n=1}^{\infty} \frac{\mathcal{U}^n\big(V^\mu_\theta(s)\big)}{n!} \big(V^i_\theta(s) - V^\mu_\theta(s)\big)^n. \quad (7)$$

Above, we denote $\mathcal{U}^n(V^\mu_\theta(s))$ as the nth derivative of $\mathcal{U}$ calculated at $V^\mu_\theta(s)$. As such, the discrepancy between the certainty equivalent value and the risk-neutral value depends on moments of $\mathcal{V}^\pi_\theta(s)$. Using Equation 7 and assuming that the utility is an exponential function $\mathcal{U}(V^i_\theta, \beta) = -e^{2\beta V_i(s)}$ yields a risk premium is equal to:

$$\Upsilon(s) \approx \beta \mathop{\mathrm{E}}_{i\sim\mathcal{V}} \big(V^i_\theta(s) - V^\mu_\theta(s)\big)^2 + \zeta(\beta). \quad (8)$$

Where $\zeta(\beta)$ denotes the sum of Taylor series of higher order than 2. Equation 8 shows that the risk premium is dependent on variance for arbitrary distribution $\mathcal{V}^\pi_\theta(s)$. By assuming the distribution of $\mathcal{V}^\pi_\theta(s)$ to be Gaussian with mean $V^\mu_\theta(s)$ and standard deviation $V^\sigma_\theta(s)$, then:

$$\Upsilon(s) = \beta \mathop{\mathrm{E}}_{i\sim\mathcal{V}} \big(V^i_\theta(s) - V^\mu_\theta(s)\big)^2. \quad (9)$$

Which implies that $\sqrt{\Upsilon(s)} = V^\sigma_\theta(s)$. Then the certainty equivalent value is equal to:

$$V^c(s) \approx \underbrace{\mathop{\mathrm{E}}_{i\sim\mathcal{V}} V_i(s)}_{\text{Ensemble Mean}} + \beta \underbrace{\mathop{\mathrm{E}}_{i\sim\mathcal{V}} \big(V_i(s) - V^\mu(s)\big)^2}_{\text{Ensemble Variance}}. \quad (10)$$

We further detail these results in the Appendix. As a result of these derivations, pessimistic agents like TD3 (Fujimoto, Hoof, and Meger 2018) and SAC (Haarnoja et al. 2018b), as well as optimistic agents like OAC (Ciosek et al. 2019), can be seen as optimizing for the expected utility objective. As such, rather than aiming for expected values, they pursue certainty equivalent values. This insight effectively connects risk-awareness with pessimism and optimism in Actor-Critic methods, as implemented through the objectives in Equation 3. In this context, the pessimistic correction $V^\sigma_\theta(s)$ acts as a risk premium, adjusting the learning to account for errors in the critic network. If $\beta < 0$, then the utility is risk-averse and the agent is pessimistic towards the errors. Conversely, if $\beta > 0$ then the utility is risk-loving, and the agent is optimistic.

Decoupled Policy Actor-Critic

In this section, we propose the Decoupled Policy Actor-Critic (DAC). DAC is a risk-aware, off-policy algorithm that addresses the exploration and temporal-difference (TD) learning dichotomy in actor-critic algorithms. Usually, actor-critic algorithms use a single actor for both exploration (sampling actions for new transitions) and TD learning (calculating TD targets), which requires balancing between optimism for exploration (Wang et al. 2020) and pessimism for avoiding value overestimation (Hasselt 2010). DAC resolves this by employing optimistic and pessimistic actors, with each actor being updated to optimize the certainty equivalent value stemming from utilities with unique risk preferences. Following Soft Actor-Critic, DAC pursues the maximum entropy objective in a policy iteration performed on a dataset of previous experiences (Haarnoja et al. 2018a).

Algorithm 1: DAC training loop

Inputs: π^p_ϕ - pessimistic actor; π^o_η - optimistic actor; Q_θ - critic; $Q_{\bar\theta}$ - target critic; α - temperature; β^o - optimism; τ - KL weight; β^p - pessimism; $\mathcal{KL}^*$ - target KL; m - exploration multiplier

Sample action from the optimistic actor
$s', r = \text{ENV.STEP}(a) \quad a \sim \pi^o_\eta$
Add transition to the replay buffer
$\text{BUFFER.ADD}(s, a, r, s')$
for $i = 1$ **to** ReplayRatio **do**
  Sample batch of transitions
  $s, a, r, s' \sim \text{BUFFER.SAMPLE}$
  Update critic using pessimistic actor (Eq. 13)
  $a' \sim \pi^p_\phi(s')$
  $\theta \leftarrow \nabla_\theta \big(Q_\theta(s,a) - (r + \gamma(Q^{\beta^p}_{\bar\theta}(s',a') - \alpha \log \pi^p_\phi(a'|s'))\big)^2$
  Update pessimistic actor (Eq. 12)
  $\phi \leftarrow \nabla_\phi \big(Q^{\beta^p}_\theta(s,a) - \alpha \log \pi^p_\phi(a|s)\big), \quad a \sim \pi^p_\phi(s)$
  Update optimistic actor (Eq. 11)
  $\mu(\bar\pi^o_\eta) = \mu(\pi^o_\eta), \quad \sigma(\bar\pi^o_\eta) = m\sigma(\pi^o_\eta)$
  $\eta \leftarrow \nabla_\eta \big(Q^{\beta^p}_\theta(s,a) - \tau KL(\pi^p_\phi|\bar\pi^o_\eta)\big), \quad a \sim \pi^o_\eta(s)$
  Update entropy temperature
  $\alpha \leftarrow \alpha - \nabla_\alpha \alpha \big(\mathcal{H}^* - \mathcal{H}(s)\big)$
  Update optimism (Eq. 14)
  $\beta^o \leftarrow \beta^o - \nabla_{\beta^o}(\beta^o - \beta^p)(\frac{1}{|A|} KL(\pi^p_\phi|\pi^o_\eta) - \mathcal{KL}^*)$
  Update KL weight (Eq. 14)
  $\tau \leftarrow \tau + \nabla_\tau \tau(\frac{1}{|A|} KL(\pi^p_\phi|\pi^o_\eta) - \mathcal{KL}^*)$
  Update target network
  $\bar\theta \leftarrow \text{POLYAK}(\theta, \bar\theta)$
end for

Whereas the results presented in the previous Section validate the use of optimistic and pessimistic policies in terms of the optimality of the pursued solutions, we further develop DAC based on the principles of SARSA, off-policy learning, and Optimism in the Face of Uncertainty. Firstly, following SAC, we update the critic network according to

a pessimistic value target sampled from the pessimistic actor. This design choice guarantees that the critic learns the pessimistic values under the pessimistic policy (Van Seijen et al. 2009), while tackling critic value overestimation (Fujimoto, Hoof, and Meger 2018). Secondly, by performing off-policy value updates we allow for exploration via a different policy than the one used for value updates. In particular, we consider a policy that is optimistic and learns to perform actions that yield critic disagreement, thus tackling the issue of pessimistic underexploration (Ciosek et al. 2019). In DAC, the optimistic actor is trained to maximize the upper Q-value bound and is solely used for exploration, while the pessimistic actor, guided by the lower Q-value bound, is used for TD learning and evaluation. This separation allows DAC to explore efficiently without the risk of value overestimation. DAC also addresses the shortcomings of Optimistic Actor-Critic (OAC) (Ciosek et al. 2019). By relaxing the first-order approximation and explicitly modeling the optimistic policy via a neural network DAC can approximate the maximum of arbitrary upper bound (Hornik, Stinchcombe, and White 1989). DAC adjusts the optimism levels associated with the upper bound such that the two policies reach a predefined divergence target, thus alleviating the tandem problem (Ostrovski, Castro, and Dabney 2021) induced by off-policy learning using two actors. DAC is structured around two components: a critic ensemble of k models (following the standard SAC/TD3 implementation we use $k = 2$ models) and the decoupled actor comprising pessimistic and optimistic actors denoted π^p_ϕ and π^o_η respectively. As shown in the previous Section, each actor can be interpreted as optimizing exponential utility function with different levels of risk appetite, denoted as β^p and β^o for the pessimistic and optimistic actors.

Optimistic Actor The optimistic actor π^o_η, pursues an objective function that maximizes a utility expression that accounts for the divergence between the two actors:

$$\mathcal{L}_\eta = \mathop{\mathrm{E}}_{s\sim\mathcal{D}} \tau KL\big(\pi^p_\phi | \bar{\pi}^o_\eta\big) - Q^{\beta^o}_\theta(s,a), \quad a \sim \pi^o_\eta(s). \tag{11}$$

Above, KL represents the empirical Kullback-Leibler divergence between the actors, β^o is the optimism parameter, τ is the penalty weight associated with KL divergence, $Q^{\beta^o}_\theta(s,a)$ is the risk-aware Q-value defined in Equation 3 for β^o, and $\bar{\pi}^o_\eta(s)$ is the transformed pessimistic policy. Incorporating two actors that are used for TD and exploration respectively, it is natural to assign distinct entropy levels to each policy. Our approach involves computing the KL divergence between the pessimistic policy π^p_ϕ and a modified optimistic policy, denoted as $\bar{\pi}^o_\eta$. This modified policy is characterized by a standard deviation that is m-times smaller than that of the optimistic policy π^o_η, with $m \in (0, \infty)$. We refer to m as exploration variance multiplier. We present ablations on different values of m in Figure 3, as well as tests regarding the importance of KL divergence in Table 1. We detail methodology for KL divergence calculation as well as technical details on implementing the optimistic actor via a neural network in the Appendix.

Pessimistic Actor and Critic The pessimistic actor is updated to maximize the utility:

$$\mathcal{L}_\phi = \mathop{\mathrm{E}}_{s\sim\mathcal{D}} \alpha \log \pi^p_\phi(a|s) - Q^{\beta^o}_\theta(s,a), \quad a \sim \pi^p_\phi(s). \tag{12}$$

Here, $\mathcal{L}_\phi$ is the pessimistic actor loss, $\mathcal{D}$ is the experience buffer, α is the entropy temperature, β^p is the pessimism parameter, and $Q^{\beta^o}_\theta(s,a)$ denotes the pessimistic Q-value defined in Equation 3 for β^p. As discussed in the previous Section, optimizing for the expected utility objective under an exponential utility function leads to a risk-aware value. As such, the pessimistic actor convergences to the optimal policy under CDQ assumptions (Fujimoto, Hoof, and Meger 2018). The critic parameters are updated via SARSA under the pessimistic policy:

$$\begin{aligned} \mathcal{L}_\theta &= \mathop{\mathrm{E}}_{T\sim\mathcal{D}} \big(Q_\theta(s,a) - r - \gamma V^{\beta^p}_{\bar{\theta}}(s')\big)^2 \\ V^{\beta^p}_{\bar{\theta}}(s) &\approx Q^{\beta^p}_{\bar{\theta}}(s,a) - \alpha \log \pi^p_\phi(a|s), \quad a \sim \pi^p_\phi(s). \end{aligned} \tag{13}$$

Above, $\mathcal{L}_\theta$ is the critic loss function which is optimized off-policy on the dataset of previous experiences $\mathcal{D}$ with $T = (s, a, r, s')$, and $\bar{\theta}$ denotes the target network parameters which are updated via standard Polyak averaging (Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018a). This setup aligns the objectives of the critic and the pessimistic actor in DAC with the traditional approaches like SAC.

Optimism and KL penalty adjustment The goal of KL regularization used by the optimistic actor is to achieve divergence small enough to ensure good coverage of the pessimistic policy in exploration data, yet large enough to maintain the optimistic policy's exploration effectiveness. In practice, the level of KL is influenced by three factors: the KL penalty weight τ, the difference in risk appetites between the actors $(\beta^o - \beta^p)$, and the critic ensemble disagreement $Q^\sigma_\theta(s,a)$, which varies based on reward scale and environment uncertainty. To achieve a balanced divergence between the two actors, we allow gradient-based adjustments in τ and $(\beta^o - \beta^p)$, while assuming a fixed β^p:

$$\begin{aligned} \mathcal{L}_{\beta^o} &= \mathop{\mathrm{E}}_{s\sim\mathcal{D}} (\beta^o - \beta^p) D(s) \;\text{ for }\; \beta^o \in (\beta^p, \infty), \\ \mathcal{L}_\tau &= \mathop{\mathrm{E}}_{s\sim\mathcal{D}} -\tau D(s) \;\text{ for }\; \tau \in (0, \infty). \end{aligned} \tag{14}$$

Above, $\mathcal{L}_{\beta^o}$ and $\mathcal{L}_\tau$ represent the loss functions of the optimism and the KL penalty weight respectively. Where $D(s)$ represents the discrepancy between the recorded and target KL divergence:

$$D(s) = \frac{1}{|A|} KL\big(\pi^p_\phi(s) | \pi^o_\eta(s)\big) - \mathcal{KL}^*. \tag{15}$$

The mechanism adjusts β^o and τ based on the distance between the empirical and target KL divergences. When the empirical divergence exceeds the target, β^o decreases to a limit of β^p, and τ increases. Conversely, a smaller empirical divergence than the target leads to an increase in β^o and a decrease in τ. This dual adjustment allows DAC to regulate the

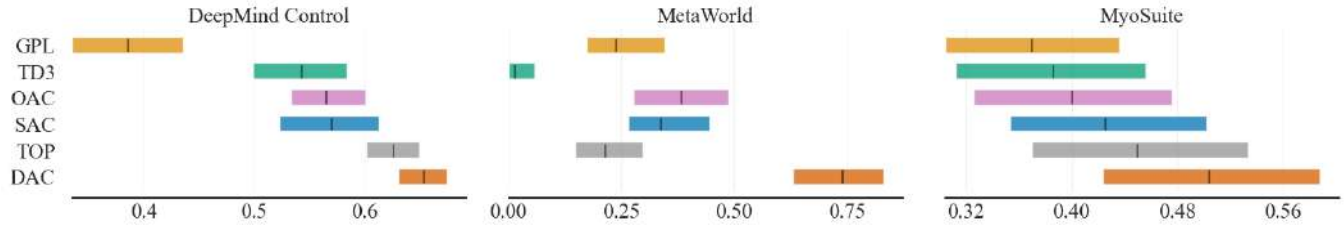


Figure 2: We report final IQM in 30 tasks (Table 3) averaged between low ($RR = 2$) and high replay settings ($RR = 16$). In high replay, all algorithms use resets as proposed by D'Oro et al. (2022). 1.0 denotes the maximal score, 95% CI, and 10 seeds.

divergence between two policy actors even when β^o reaches its lower bound. The optimization objectives for τ and β^o are thus formulated to adapt to varying conditions, ensuring efficient exploration and exploitation balance in different environments. We test the effectiveness of these adjustment mechanisms in Table 1, and depict the different levels of optimism achieved in various tasks in Figure 8. We expand our discussion of DAC in the Appendix.

Experiments

Our framework is based on JaxRL (Kostrikov 2021). We assess the performance of DAC across a diverse set of over 30 locomotion and manipulation tasks listed in Table 3, sourced from the DeepMind Control (DMC) (Tassa et al. 2018), MetaWorld (MW) (Yu et al. 2020) and MyoSuite (MYO) (Caggiano et al. 2022) benchmarks. In DMC we report the returns, whereas in MW and MYO we report the success rates. We calculate evaluation metrics using the RLiable package (Agarwal et al. 2021).

Model-free benchmark We test DAC against an array of both risk-neutral and risk-aware baselines, including TD3 (Fujimoto, Hoof, and Meger 2018), SAC (Haarnoja et al. 2018a), OAC (Ciosek et al. 2019), TOP (Moskovitz et al. 2021), and GPL (Cetin and Celiktutan 2023). We align our experiments with the state-of-the-art SAC implementation, standardizing the common hyperparameters across all algorithms (D'Oro et al. 2022). We employ uniform network architectures and ensemble of two critics, as advocated in previous work (Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018a; Ciosek et al. 2019; Moskovitz et al. 2021; Cetin and Celiktutan 2023). Importantly, by using uniform network architectures and hyperparameters, we ensure that the performance differences between algorithms stem solely from the different risk-preferences in tackling the exploration-exploitation dilemma. Each of 30 tasks is run for 1mln environment steps. We investigate two replay regimes: a compute-efficient regime, involving 2 gradient updates per environment step, and a sample-efficient regime, using 16 gradient updates per step with full-parameter resets every $160k$ environment steps D'Oro et al. (2022). As evidenced by Figure 1, DAC particularly excels in the earlier phases of the training, achieving final performance of the best baseline in only 60% of environment steps. Furthermore, as shown in Figure 2, DAC approach to balancing risk-averse exploitation with risk-loving exploration yields significant performance benefits, particularly visible in the MetaWorld tasks. We present task-dependent training curves in the Appendix.

Design decisions We explore how DAC performance is influenced by variations in its design. We perform evaluations of various DAC modifications in 15 tasks listed in Table 3, where we train for $500k$ environment steps and consider both replay ratios. We assess the performance of the following variations: ONLY π^o_η where we use the optimistic actor for both exploration and exploitation; NO KL REG where we do not use KL regularization for the optimistic actor; DETERMINISTIC π^o_η where we set the optimistic actor to be deterministic; DETERMINISTIC π^p_ϕ where we set the pessimistic actor to be deterministic; NO ADJUSTMENTS where we disable the DAC adjustment mechanisms; NO τ ADJUSTMENT where we disable τ adjustment; and NO β^o ADJUSTMENT where we disable β^o adjustment. As shown in Table 1, we find that all of these design choices bring significant gains to DAC performance. In particular, we observe that the dual policy setup allows effective use of optimism, as evidenced by the poor performance of the ONLY π^o_η agent. Similarly, we find that KL regularization yields significant improvements, most likely due to limiting the tandem problem (Ostrovski, Castro, and Dabney 2021).

	$RR = 2$	$RR = 16$
ONLY π^o_η	0.08	0.17
NO KL REG	0.42	0.77
DETERMINISTIC π^o_η	0.82	0.77
DETERMINISTIC π^p_ϕ	0.92	0.79
NO ADJUSTMENTS	0.91	0.87
NO τ ADJUSTMENT	0.95	0.93
NO β^o ADJUSTMENT	0.84	0.99
BASE DAC	1.00	1.00

Table 1: We evaluate design choices associated with DAC.

Hyperparameter Sensitivity As discussed previously, DAC introduces three tunable hyperparameters: KL target (the target divergence between optimistic and pessimistic policies as introduced in Equation 15), exploration multiplier (parameter that determines how much larger the variance of an optimistic policy is compared to a pessimistic policy as discussed under Equation 11), as well as the learning rates for the adjustment mechanism for β^o and τ (for opti-

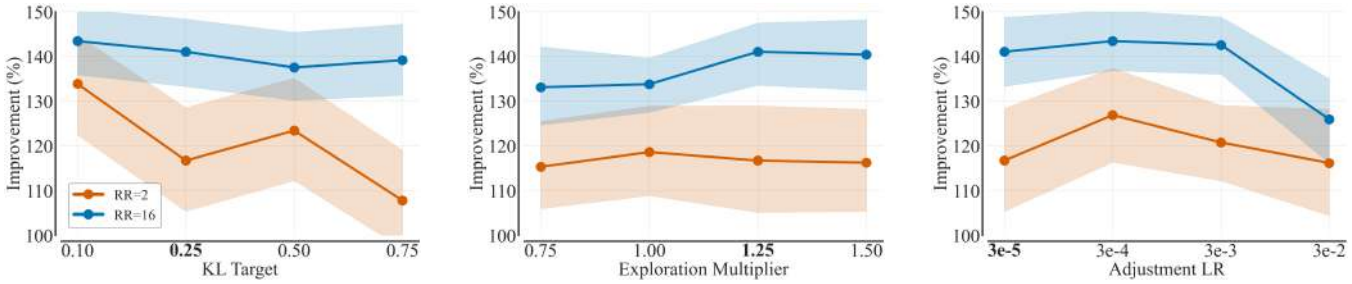


Figure 3: We evaluate DAC when changing the values of its hyperparameters (X-axis) for two replay regimes. The bold value denotes the value used in the main experiment. Y-axis reports the percentage improvement over tuned SAC.

mizing Equation 14). Here, we assess DAC sensitivity when changing the values of these hyperparameters. To this end, we train 12 DAC agents (each with a different hyperparameter setting) on $500k$ steps, in both replay regimes, and on 15 tasks listed in Table 3. We report these results In Figure 3, where we compare the final performance of these agents to SAC with analogical replay ratio. As follows, we find that DAC is stable when changing the values of its hyperparameters, with all variations outperforming the SAC baseline.

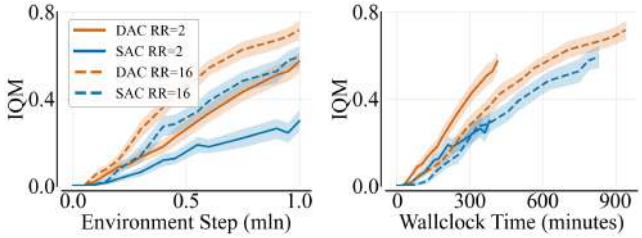


Figure 4: Despite additional compute associated with the decoupled actor, the improvements stemming from DAC lead to a beneficial compute/performance tradeoff. Experiments were conducted on an NVIDIA A100 40GB RAM.

Additional Experiments Firstly, we evaluate the compute costs associated with running a decoupled actor setup proposed by DAC. We find that DAC requires around 15% additional wall clock time to complete 1 mln environment step training compared to SAC. However, as shown in Figure 4, improvements in DAC performance offset additional compute costs, leading to a beneficial trade-off between compute and performance. Furthermore, in Appendix, we detail supplementary experiments conducted to further evaluate DAC. In Figure 9, we compare DAC with model-based TD-MPC when training for 3 mln environment steps in the dog and humanoid domains of DMC. We find that DAC is able to achieve the performance of model-based TD-MPC on these tasks, despite using smaller parameter count. Figure 10 studies the effectiveness of layer normalization when used with DAC. This regularization technique was shown to improve the performance of RL agents on a variety tasks (Hiraoka et al. 2021; Li et al. 2022). We find that using layer normalization further improves DAC performance on locomotion tasks. In Figure 11, we find that the decoupled policy architecture indeed does not increase the value overestimation over the pessimistic baselines. Finally, in Figure 12, we investigate the performance of distributional DAC that explicitly models aleatoric and epistemic uncertainties and is optimistic only with respect to specific types.

Limitations

The main limitation of DAC is its dual-actor framework, which incurs slightly higher memory and computation costs relative to the standard SAC. Furthermore, DAC introduces three additional tunable hyperparameters. Although tests showed DAC robust performance across a variety of hyperparameter values, it is uncertain whether this robustness translates to more complex environments.

Conclusions

In this work, we apply the expected utility theorem to reason about the empirical effectiveness of pessimistic and optimistic actor-critic algorithms. In particular, we demonstrate that policies derived from the commonly used pessimistic objective are approximately utility-optimal when aligned with an exponential risk-aware utility function. This analysis shows that the commonly used pessimistic update rules are, in fact, risk-averse with respect to critic errors. We think that the proposed framework is interesting, as it revisits the foundational decision-theoretic perspective of RL, namely, optimization within the utility space rather than the space of utility inputs. Furthermore, we present DAC, an off-policy actor-critic framework. DAC employs two actors with distinct levels of pessimism to optimize their respective expected utility functions. The pessimistic actor focuses on TD learning and evaluation, while the optimistic actor facilitates exploration. We evaluated DAC in various locomotion and manipulation tasks and compared it with more than ten baseline algorithms. We find that DAC demonstrates significant performance improvements relative to other model-free methods and is competitive with leading model-based approaches. Finally, we investigate the robustness of DAC performance across various hyperparameter configurations and find that the experiments affirm its practical applicability.

Acknowledgments

We would like to thank Łukasz Kuciński for his help in developing the ideas presented in this paper. We also thank Piotr Miłoś and Gracjan Góral for their valuable help and discussions. We also gratefully acknowledge the Polish high-performance computing infrastructure, PLGrid (HPC Center: ACK Cyfronet AGH), for providing computational resources and support under grant no. PLG/2024/017817. Marek Cygan was partially supported by an NCBiR grant POIR.01.01.01-00-0433/20.

References

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

Caggiano, V.; Wang, H.; Durandau, G.; Sartori, M.; and Kumar, V. 2022. MyoSuite–A contact-rich simulation suite for musculoskeletal motor control. *arXiv preprint arXiv:2205.13600*.

Cetin, E.; and Celiktutan, O. 2023. Learning Pessimism for Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 6971–6979.

Chen, R. Y.; Sidor, S.; Abbeel, P.; and Schulman, J. 2017. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*.

Chen, X.; Wang, C.; Zhou, Z.; and Ross, K. W. 2020. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *International Conference on Learning Representations*.

Ciosek, K.; Vuong, Q.; Loftin, R.; and Hofmann, K. 2019. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32.

Ciosek, K.; and Whiteson, S. 2020. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21(2020).

D'Oro, P.; Schwarzer, M.; Nikishin, E.; Bacon, P.-L.; Bellemare, M. G.; and Courville, A. 2022. Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier. In *The Eleventh International Conference on Learning Representations*.

Fei, Y.; Yang, Z.; and Wang, Z. 2021. Risk-sensitive reinforcement learning with function approximation: A debiasing approach. In *International Conference on Machine Learning*, 3198–3207. PMLR.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018a. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018b. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hafner, D.; Pasukonis, J.; Ba, J.; and Lillicrap, T. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Hansen, N.; Wang, X.; and Su, H. 2022. Temporal Difference Learning for Model Predictive Control. In *International Conference on Machine Learning, PMLR*.

Hasselt, H. 2010. Double Q-learning. *Advances in neural information processing systems*, 23.

Hiraoka, T.; Imagawa, T.; Hashimoto, T.; Onishi, T.; and Tsuruoka, Y. 2021. Dropout Q-Functions for Doubly Efficient Reinforcement Learning. In *International Conference on Learning Representations*.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5): 359–366.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32.

Kahneman, D.; and Tversky, A. 1979. Prospect Theory: An Analysis of Decision under Risk. *Econometrica*, 47(2): 263–292.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kostrikov, I. 2021. JAXRL: Implementations of Reinforcement Learning algorithms in JAX.

Kumar, A.; Gupta, A.; and Levine, S. 2020. Discor: Corrective feedback in reinforcement learning via distribution correction. *Advances in Neural Information Processing Systems*, 33: 18560–18572.

Li, Q.; Kumar, A.; Kostrikov, I.; and Levine, S. 2022. Efficient Deep Reinforcement Learning Requires Regulating Overfitting. In *The Eleventh International Conference on Learning Representations*.

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. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Moskovitz, T.; Parker-Holder, J.; Pacchiano, A.; Arbel, M.; and Jordan, M. 2021. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 12849–12863.

Neu, G.; and Pike-Burke, C. 2020. A unifying view of optimism in episodic reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1392–1403.

Nikishin, E.; Schwarzer, M.; D'Oro, P.; Bacon, P.-L.; and Courville, A. 2022. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, 16828–16847. PMLR.

Ostrovski, G.; Castro, P. S.; and Dabney, W. 2021. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 23283–23295.

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

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *nature*, 323(6088): 533–536.

Schwarzer, M.; Ceron, J. S. O.; Courville, A.; Bellemare, M. G.; Agarwal, R.; and Castro, P. S. 2023. Bigger, Better, Faster: Human-level Atari with human-level efficiency. In *International Conference on Machine Learning*, 30365–30380. PMLR.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *International conference on machine learning*, 387–395. PMLR.

Smith, L.; Kostrikov, I.; and Levine, S. 2022. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*.

Stiglitz, J. E. 1997. *Microeconomics*. New York, NY: WW Norton.

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

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

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Van Seijen, H.; Van Hasselt, H.; Whiteson, S.; and Wiering, M. 2009. A theoretical and empirical analysis of Expected Sarsa. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, 177–184. IEEE.

Von Neumann, J.; and Morgenstern, O. 1947. Theory of games and economic behavior, 2nd rev.

Wang, Y.; Wang, R.; Du, S. S.; and Krishnamurthy, A. 2020. Optimism in Reinforcement Learning with Generalized Linear Function Approximation. In *International Conference on Learning Representations*.

Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S. 2020. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, 1094–1100. PMLR.

Zawalski, M.; Tyrolski, M.; Czechowski, K.; Odrzygóźdź, T.; Stachura, D.; Piekos, P.; Wu, Y.; Kuciński, Ł.; and Miłoś, P. 2022. Fast and Precise: Adjusting Planning Horizon with Adaptive Subgoal Search. In *The Eleventh International Conference on Learning Representations*.

A Case for Validation Buffer in Pessimistic Actor-Critic

Michał Nauman[1] , **Mateusz Ostaszewski**[2] , **Marek Cygan**[1,3]
[1]University of Warsaw
[2]Warsaw University of Technology
[3]Nomagic
nauman.mic@gmail.com

Abstract

In this paper, we investigate the issue of error accumulation in critic networks updated via pessimistic temporal difference objectives. We show that the critic approximation error can be approximated via a recursive fixed-point model similar to that of the Bellman value. We use such a recursive definition to retrieve the conditions under which the pessimistic critic is unbiased. Building on these insights, we propose Validation Pessimism Learning (VPL) algorithm. VPL uses a small validation buffer to adjust the levels of pessimism throughout the agent's training, with the pessimism set such that the approximation error of the critic targets is minimized. We investigate the proposed approach on a variety of locomotion and manipulation tasks and report improvements in sample efficiency and performance.

1 Introduction

Approximation errors, although ubiquitous in machine learning, are particularly exaggerated in the context of value-based Reinforcement Learning (RL). Such exaggeration stems from Temporal Difference (TD) in which the critic is supervised via value estimate calculated at a different state [Silver *et al.*, 2014], [Mnih *et al.*, 2015].

Inaccuracies in this estimate lead to propagated errors in state-action updates, and the use of maximization in value estimation inherently promotes overestimation. Addressing such overestimation has proven to be an effective strategy in discrete and continuous action environments [Hasselt, 2010], [Van Hasselt *et al.*, 2016], [Hessel *et al.*, 2018], [Haarnoja *et al.*, 2018]. Clipped Double Q-Learning (CDQL), a common solution to overestimation in continuous action actor-critic algorithms aims to mitigate overestimation by balancing errors against a pessimistic lower bound value approximation [Fujimoto *et al.*, 2018]. However, challenges arise if the lower bound is insufficiently pessimistic, leading to continued overestimation, or overly pessimistic, causing underestimation [Cetin and Celiktutan, 2023]. The latter, though less recognized, can significantly reduce sample efficiency and degrade actor-critic agents' performance in both low and high replay ratio settings which we show in Figure 1.

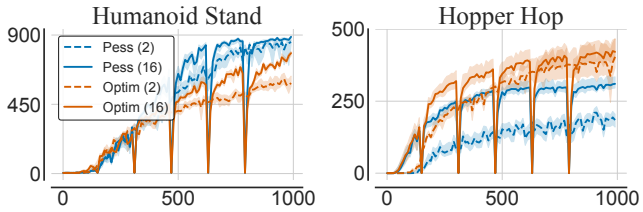


Figure 1: Pessimism adjustment can yield performance benefits exceeding those of increased replay ratio and full-parameter resets. The pessimistic algorithm dominates Humanoid, whereas the optimistic algorithm dominates Hopper. 10 seeds and 95% CI.

In this paper, we investigate the relationship between pessimism in Q-value approximation and error accumulation in critic networks. We start by characterization of existing strategies for pessimism adjustment. Furthermore, we analyze the pessimistic critic approximation error and show that such error can be represented recursively forming a fixed-point model, akin to values. This recursive representation helps us highlight the bias inherent in pessimistic actor-critic algorithms, examine their convergence dynamics, and identify the conditions under which pessimistic critics can achieve zero error. Building on these insights, we propose the Validation Pessimism Learning (VPL) algorithm. VPL employs a small validation replay buffer to adjust the pessimism levels online, aiming to minimize the approximation error of critic targets while preventing overfitting to accumulated experience. We evaluate VPL against existing pessimism adjustment methods on DeepMind control [Tassa *et al.*, 2018] and MetaWorld [Yu *et al.*, 2020]. Our findings demonstrate that VPL not only achieves performance improvements but also exhibits less sensitivity to hyperparameters compared to the baseline algorithms. We list our contributions below:

- We show that critic error can be defined recursively through a fixed-point model. We demonstrate that pessimistic TD learning, a method often used in RL, converges to the true value under strict conditions.
- We present an empirical analysis showing that the performance loss associated with not including every transition in the replay buffer diminishes as training progresses. This observation challenges the traditional belief that every transition must be used in value learning for sample-efficient RL and builds a case for employing

a validation buffer in an online RL setting.

- We propose VPL, an algorithm that uses a small validation buffer for online adjustment of pessimism associated with lower bound Q-value approximation. We test the effectiveness of VPL and other pessimism adjustment strategies in low and high replay regimes. We show that VPL offers performance improvements across a variety of locomotion and manipulation tasks.

2 Background

2.1 Maximum Entropy Reinforcement Learning

We consider an infinite-horizon Markov Decision Process (MDP) [Puterman, 2014], represented by the tuple (S, A, r, p_0, γ), where S and A are continuous state and action spaces, $r_{s,a}$ is the reward, $p_0(s)$ is the initial state distribution, and $\gamma \in (0, 1]$ is the discount factor. The policy $\pi(a|s)$ is a distribution of actions given states. The goal of Maximum Entropy Reinforcement Learning (MaxEnt RL) [Haarnoja *et al.*, 2017] is to maximize the expected cumulative discounted return augmented with an entropy term:

$$\pi^* = \arg\max_{\pi} \mathop{\mathrm{E}}_{p_0,\pi} \sum_{t=0}^{\infty} \gamma^t \big(r_{s_t,a_t} + \alpha \mathcal{H}(\pi(\cdot|s_t))\big), \quad (1)$$

where α controls the balance between reward and entropy [Haarnoja *et al.*, 2018]. The soft Q-value is defined as:

$$Q^\pi(s,a) = r_{s,a} + \gamma \mathop{\mathrm{E}}_{s' \sim p} \big[V^\pi(s')\big], \quad (2)$$

and the soft value is given by:

$$V^\pi(s) = \mathop{\mathrm{E}}_{a \sim \pi} \big[Q^\pi(s,a) - \alpha \log \pi(a|s)\big]. \quad (3)$$

MaxEnt RL algorithms, such as Soft Actor-Critic (SAC), parameterize the policy (actor) π_θ and the Q-value (critic) Q_ϕ, which are optimized iteratively using objectives derived from policy iteration [Haarnoja *et al.*, 2018]. The critic ensemble is often used to address Q-value overestimation via Clipped Double Q-Learning (CDQL) [Fujimoto *et al.*, 2018]. In CDQL, the value lower bound is approximated as:

$$V^{lb}_\phi(s) \approx Q^{lb}_\phi(s,a) - \alpha \log \pi_\theta(a|s), \quad a \sim \pi_\theta, \quad (4)$$

where $Q^{lb}_\phi(s,a) = \min(Q^1_\phi(s,a), Q^2_\phi(s,a))$.

2.2 Pessimism Adjustment

Building upon CDQL, recent works propose parameterizing the lower bound of Q-values as:

$$Q^{lb}_\phi(s,a) = Q^\mu_\phi(s,a) - \beta Q^\sigma_\phi(s,a), \quad (5)$$

where Q^μ_ϕ and Q^σ_ϕ represent the mean and standard deviation of the critic ensemble, respectively [Ciosek *et al.*, 2019], [Nauman and Cygan, 2023]. This formulation allows the adjustment of the pessimism level β, controlling the influence of critic disagreement. Algorithms such as Generalized Pessimism Learning (GPL) [Cetin and Celiktutan, 2023] and On-Policy Pessimism Learning (OPL) [Kuznetsov *et al.*, 2021] optimize β online by aligning pessimism with approximation error. For example, GPL treats this adjustment as a dual optimization problem:

$$\beta = \arg\min_{\beta} \mathop{\mathrm{E}}_{p_0,\pi} \beta\big(Q^\pi(s,a) - r_{s,a} - \mathtt{sg}(\gamma V^{lb}_\phi(s'))\big), \quad (6)$$

where $V^{lb}_\phi(s') \approx Q^\mu_\phi(s,a) - \beta Q^\sigma_\phi(s,a) - \alpha \log \pi_\theta(a|s)$ and $\mathtt{sg}$ denotes the stop-gradient operator. This approach risks overfitting, as the adjustment heavily relies on critic outputs. Alternative methods, such as Tactical Optimism and Pessimism (TOP) [Moskovitz *et al.*, 2021], use external bandit controllers for β, but they can be less effective with continuous pessimism adjustments. We include additional discussion of critic disagreement in the appendix, as well as key comparisons of pessimism adjustment methods in Table 2.

3 Approximation Error and Pessimism

In this section, we focus on the analysis of critic approximation errors within the framework of pessimistic updates. For simplicity, we consider a fixed policy π_θ and use $V(s)$ and $Q(s,a)$ to represent the value and Q-value under this policy. We define the mean and lower bound approximation errors denoted as U^μ_ϕ and U^{lb}_ϕ respectively:

$$\begin{aligned} U^\mu_\phi(s,a) &Q(s,a) - Q^\mu_\phi(s,a), \\ U^{lb}_\phi(s,a) &Q(s,a) - Q^{lb}_\phi(s,a). \end{aligned} \quad (7)$$

Here, $Q(s,a)$ denotes the true Q-value, the term $Q^\mu_\phi(s,a)$ represents the mean Q-value estimated by an ensemble of k critics, calculated as $Q^\mu_\phi(s,a) = \frac{1}{k}\sum^k Q^i_\phi(s,a)$, and $Q^{lb}_\phi(s,a)$ is the lower bound Q-value as defined in Equation 5. Additionally, we introduce the mean and lower bound temporal critic errors, denoted as u^μ_ϕ and u^{lb}_ϕ, respectively:

$$\begin{aligned} u^\mu_\phi(s,a,s') r_{s,a} + \gamma V^\mu_\phi(s') - Q^\mu_\phi(s,a), \\ u^{lb}_\phi(s,a,s') r_{s,a} + \gamma V^{lb}_\phi(s') - Q^\mu_\phi(s,a). \end{aligned} \quad (8)$$

These temporal critic errors quantify the deviation between the Q-values $Q^\mu_\phi(s,a)$ and the mean or lower bound Temporal Difference (TD) targets. The value $V^{lb}_\phi(s)$ is equal to the expected value of $Q^{lb}_\phi(s,a)$ over all state-action pairs under policy π, such that $V^{lb}_\phi(s) = \mathrm{E}_\pi Q^{lb}_\phi(s,a) - \log \pi_\theta(a|s)$.

Lemma 3.1 (Approximation error operator)**.** *Given policy π, k on-policy q-value approximations $Q^1_\phi, Q^2_\phi, ..., Q^k_\phi$, sample mean Q^μ_ϕ and standard deviation Q^σ_ϕ, the mean and lower bound approximation errors follow a recursive formula:*

$$\begin{aligned} U^\mu_\phi(s,a) &= u^\mu_\phi(s,a,s') + \gamma \mathop{\mathrm{E}}_{a' \sim \pi} U^\mu_\phi(s',a'), \\ U^{lb}_\phi(s,a) &= u^{lb}_\phi(s,a,s') + \beta Q^\sigma_\phi(s,a) + \gamma \mathop{\mathrm{E}}_{a' \sim \pi} U^{lb}_\phi(s',a'), \\ U^{lb}_\phi(s,a) &= U^\mu_\phi(s,a) + \beta Q^\sigma_\phi(s,a). \end{aligned}$$

We expand on Lemma 3.1 in Appendix A. The lemma reveals that approximation errors exhibit a recurrent pattern analogous to Q-values. Specifically, the temporal errors function as an immediate signal, akin to rewards, while the future approximation errors serve as the bootstrap signal. Furthermore, this observation formalizes the intuitive concept that

minimizing the lower-bound error necessitates a precise calibration of the pessimistic correction against the temporal error and the approximation errors of subsequent states.

Key Insight

Lemma 3.1 shows that approximation errors propagate through recursive updates, just as Q-values propagate through Bellman equations. The temporal errors, u^{μ}_{ϕ} and u^{lb}_{ϕ}, act as immediate signals, while future approximation errors (weighted by γ) bootstrap over time. This highlights the need to balance immediate errors and long-term disagreement for effective learning.

It can be shown that similarly to the Bellman operator, both mean and lower bound error approximation operators are monotonic contractions:

Theorem 3.2 (Approximation error contraction)**.** *Let $\mathcal{F}$ be the space of functions on domain $S \times A$. We define the mean error and lower bound error operators $\mathcal{U}^{\mu}, \mathcal{U}^{lb} : \mathcal{F} \to \mathcal{F}$ as:*

$$\mathcal{U}^{\mu}\big(f(s,a)\big)u^{\mu}_{\phi}(s,a,s') + \gamma \mathop{\mathrm{E}}_{a' \sim \pi} f(s',a'),$$
$$\mathcal{U}^{lb}\big(f(s,a)\big)u^{lb}_{\phi}(s,a,s') + \beta Q^{\sigma}_{\phi}(s,a) + \gamma \mathop{\mathrm{E}}_{a' \sim \pi} f(s',a').$$

Above, $f(s,a) : S \times A \to R$ represents an estimate of the approximation error, and we assume that $Q^{\sigma}_{\phi}(s,a) \to 0$ as training progresses. Then it follows that both $\mathcal{U}^{\mu}$ and $\mathcal{U}^{lb}$ are monotonic contractions for any f_1 and f_2:

$$||\mathcal{U}(f_1) - \mathcal{U}(f_2)||_{\infty} \leq \gamma ||f_1 - f_2||_{\infty}.$$

We provide the relevant derivations in Appendix A.

Key Insight

Theorem 3.2 formalizes that both mean and lower-bound approximation error operators are contractions. This ensures that repeated applications of these operators converge to a fixed point. Intuitively, this means that as training progresses, the approximation errors stabilize, allowing the model to converge to a consistent value function.

As follows from Theorem 3.2, repeated application of the approximation error operator yields a Cauchy sequence, and therefore leads to a fixed point:

Corollary 3.3 (Approximation error fixed point)**.** *We denote repeated k applications of either approximation error operator to function f as $\mathcal{U}_k(f)$. Then, due to Banach fixed point theorem:*

$$\mathcal{U}^{\infty}(f) = f^* \ \wedge \ \mathcal{U}(f^*) = f^*.$$

The corollary shows that the approximation error of values can be effectively modeled using a fixed-point approach, analogous to treating values themselves. The potential ramifications and applications of this concept are further explored in Appendix A. Principally, the convergence of a pessimistic

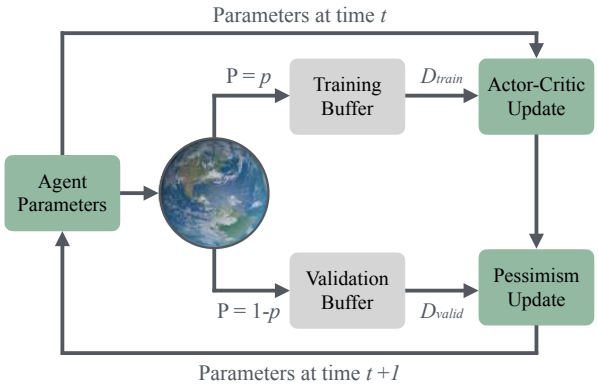


Figure 2: High-level overview of the proposed approach. After each environment step, the transition is stored in either the training buffer (used for updating actor-critic modules) or the validation buffer (used for updating the pessimism module). The pessimism is updated via a "reverse" TD loss, optimisation of which on the training buffer would be prone to overfitting.

value model signifies that the approximation errors converge to zero, implying $U^{\mu}_{\phi} = U^{lb}_{\phi} = 0$. The convergence proof of CDQL indicates that the value model should align with the true on-policy values under the conventional Q-learning convergence assumptions [Watkins and Dayan, 1992], [Fujimoto *et al.*, 2018]. Lemma 3.1 explicitly shows that for all s, a and s', both approximation errors equate to zero iff the following conditions are satisfied:

$$Q^{\mu}_{\phi}(s,a) = r + \gamma V^{\mu}_{\phi}(s') \ \wedge \ \beta Q^{\sigma}_{\phi}(s,a) = 0. \tag{9}$$

The convergence of a pessimistic model necessitates either the absence of critic ensemble disagreement (i.e., $Q^{\sigma}_{\phi}(s,a) = 0$ for all state-action pairs) or an algorithmic ability to diminish the level of pessimism over time, culminating in $\beta = 0$ asymptotically. We emphasize that this pertains specifically to convergence toward the zero-error fixed point defined in Lemma 3.1; an algorithm may still converge to a biased solution even if these conditions are not met, but exact fixed-point convergence with zero approximation error is unattainable otherwise. Figure 10 shows that the critic disagreement does not completely diminish on popular DeepMind Control and MetaWorld benchmarks. Given the improbability of achieving zero critic disagreement in overparameterized deep RL contexts, the adjustment of β emerges as a compelling strategy. Additionally, it can be demonstrated that under the scenario of critic underestimation, the lower-bound error exceeds the mean approximation error:

$$U^{\mu}_{\phi}(s,a) > 0 \implies |U^{\mu}_{\phi}(s,a)| \leq |U^{lb}_{\phi}(s,a)|. \tag{10}$$

The inequality follows from the third formula from Lemma 3.1, where $\beta \geq 0$ and $Q^{\sigma}(s,a) \geq 0$ as the standard deviation of the critic ensemble. We refer the reader to Appendix B for detailed proof.

As follows, pessimistic learning is advantageous only in overestimation, whereas it becomes detrimental in cases of underestimation. To this end, the pessimism levels should be adjusted in tandem with changes in the approximation errors. In practical terms, achieving a zero approximation error for either mean or lower bound is unrealistic. Given that

$U_\phi(s,a) \in R$, one might be interested in optimization of norm of $U^\mu_\phi(s,a)$ or $U^{lb}_\phi(s,a)$. This leads to the possibility of defining an "optimal" level of pessimism, where optimality is considered in relation to minimizing the respective approximation error norm. We note that our analysis yields a different approach to updating pessimism as compared to the method derived from dual optimization [Cetin and Celiktutan, 2023], which we discuss in Section 2.2.

Key Insight

Equation 10 highlights that pessimistic updates are beneficial in situations of overestimation, but they may harm learning when underestimation occurs. Therefore, adjusting the level of pessimism dynamically is critical to balance the benefits of avoiding overestimation with the risks of underestimation.

4 Validation Pessimism Learning Algorithm

Building on the analysis conducted in the previous Section, we propose the Validation Pessimism Learning module (VPL). The goal of the VPL module is to adjust the pessimism parameter such that the critic targets (lower bound Q-value approximation) has the least approximation error. As such, VPL can be used as an alternative to CDQL or GPL in conjuction with any off-policy actor-critic algorithm. For our analysis, we utilize the Soft Actor-Critic (SAC) [Haarnoja *et al.*, 2018] as the backbone algorithm. VPL is based on a simple premise of adjusting pessimism via a TD loss. Given that the critic concurrently optimizes this loss function, such setup is especially prone to overfitting. To mitigate this, the optimization of the pessimism parameter is conducted on a distinct set of *validation* data, which remains unseen by the actor-critic modules. From a theoretical standpoint, VPL can be interpreted as a strategy for pessimism model selection, with the selection process aimed at minimizing the lower bound approximation error delineated in the previous section. A critical aspect of VPL involves conducting the pessimism model selection on validation data. The model selection is achieved through gradient-based optimization of the proposed pessimism loss. The utilization of validation data in this process reduces the probability of overfitting to bootstrapped supervision signals used by TD learning. We summarize VPL approach in Figure 2 and share pseudo-code in Section B.1, where we colour changes wrt. regular SAC.

4.1 Validation Buffer

The employment of validation data is a well-established practice in supervised learning frameworks [Bishop and Nasrabadi, 2006]. It serves a dual purpose: providing an unbiased assessment of model performance trained on the training dataset, and facilitating regularization techniques such as early stopping [Prechelt, 2002] or hyperparameter tuning [Bergstra and Bengio, 2012]. However, the integration of validation data entails a trade-off, notably the reduction of the training set size. In supervised learning, the regret associated with decreasing the training set can be quantitatively evaluated through the lens of neural scaling laws [Rosenfeld *et al.*, 2019]. Such regret is, to the best of our knowledge, a relatively understudied area in the context of online RL. In online RL, the notion of a validation buffer is not popular, primarily due to the requisite sacrifice of actor-critic learning on the validation transitions. Given inherent sample inefficiency of RL, this cost is often deemed as overly burdening. Contrary to supervised learning setup, RL is characterized by a high

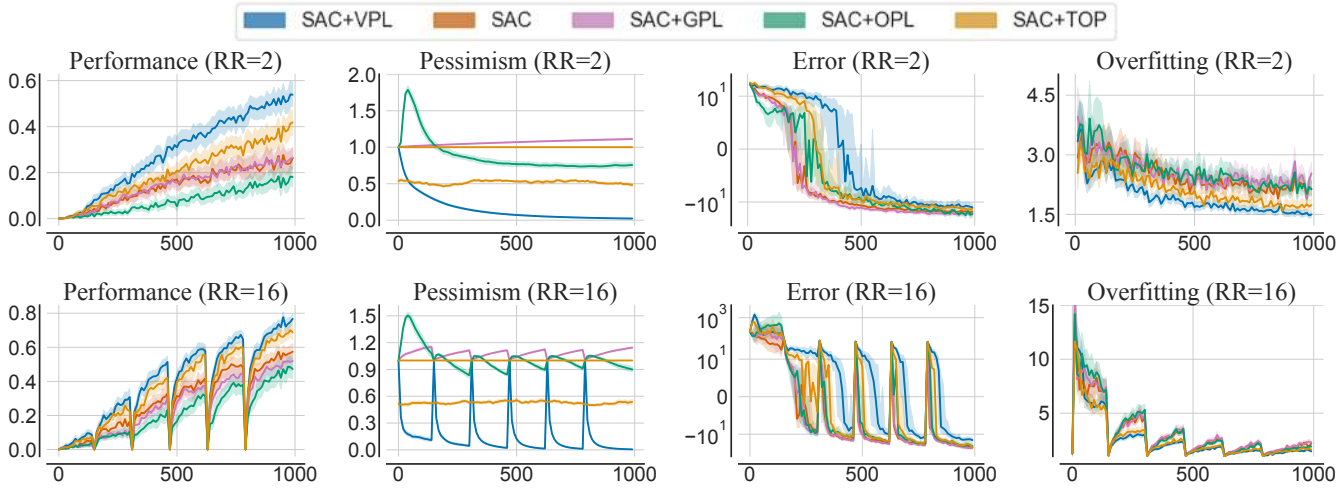


Figure 3: We integrate the Soft Actor-Critic (SAC) and the Scaled-By-Resetting SAC (SR-SAC) with various pessimism adjustment algorithms. Performance is evaluated in both low replay (top row) and high replay (bottom row) regimes. All tested algorithms use the same network architectures and hyperparameter settings, so performance differences arise solely from the pessimism adjustment strategies. Despite similar motivations, each algorithm exhibits different levels of pessimism. Our proposed Validation Pessimism Learning (VPL) module demonstrates the lowest approximation error and mitigates value overfitting more effectively than alternative approaches, leading to improvements in performance and sample efficiency. The experimental setting is detailed in Sections 5 and E. Results are based on 20 tasks with 10 seeds per task, presented as interquartile mean (IQM) and 95% confidence intervals (CI).

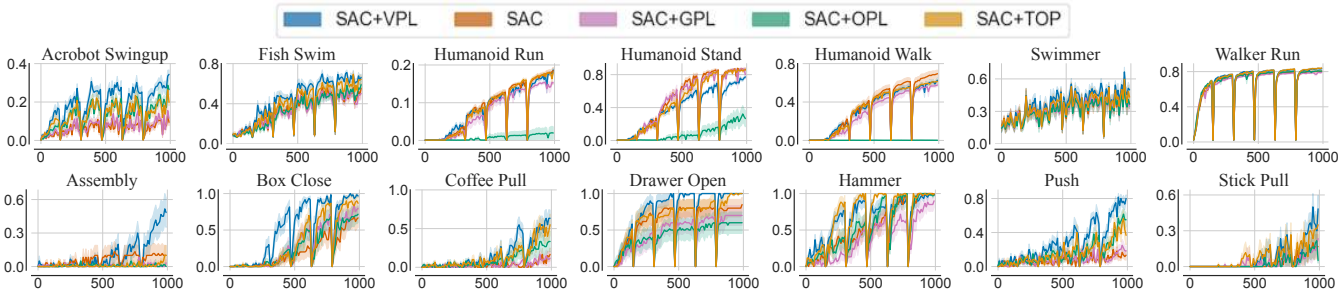


Figure 4: Task-specific performance of high-replay configurations in 14 out of 20 considered tasks. VPL achieves performance improvements, especially in the manipulation tasks. In the case of DMC tasks the y-axis denotes evaluation returns, whereas for MetaWorld tasks it denotes the evaluation success ratio. The x-axis shows environment steps (in thousands). We detail the setting in Section 5.1. 10 seeds per task.

correlation between successive samples, thereby diminishing the marginal utility of processing additional samples from the same trajectory. Consequently, we posit that in online RL, the cost associated with the use of validation data can be counterbalanced, provided the validation data is leveraged to enhance the learning process. In the case of the VPL, we allocate the validation transitions exclusively for the adjustment of the pessimism parameter. This approach presents a novel utilization of validation data in online RL in a manner that is RL-specific, diverging from supervised methodologies.

4.2 Pessimism Update Rule

The persistence of critic disagreement throughout training implies that the standard convergence guarantees of the pessimistic temporal difference update towards on-policy values are not upheld when $\beta \neq 0$. Moreover, in cases where minimizing the mean approximation error is not achievable, particularly in scenarios characterized by strong overestimation, the presence of non-zero critic disagreement can be leveraged to decrease the lower bound approximation error by increasing β. This observation forms the basis for our proposed method of adjusting β. The aim is to minimize the expected lower bound approximation error $U^{lb}_{\phi}(s,a)$, formulated as follows:

$$\beta^* = \arg\min_{\beta} \mathop{\mathrm{E}}_{p_0,\pi} \sum_{t=0}^{\infty} \gamma^t U^{lb}_{\phi}(s,a). \tag{11}$$

Unfortunately, obtaining $U^{lb}_{\phi}(s,a)$ is challenging as it necessitates an estimate of the true on-policy Q-value. Typically, such estimates are derived through methods like Monte-Carlo (MC) rollouts, TD(n), or TD(λ), with MC being the only unbiased method. However, in the context of off-policy learning or non-terminating environments, employing MC rollouts is impractical. Consequently, we leverage the simple approach proposed by [Cetin and Celiktutan, 2023] in which it is assumed that the critic output for prerecorded off-policy actions is unbiased. Therefore, we assume that $Q^{\pi}(s,a) = Q^{\mu}_{\phi}(s,a)$ for actions that do not maximize the output of the policy. Additionally, akin to the approach in off-policy actor-critic algorithms, the policy-induced distribution is approximated using an off-policy replay buffer. This approach leads to the formulation of the following:

$$\beta^* \approx \arg\min_{\beta} \mathop{\mathrm{E}}_{\mathcal{D}_v} \left(Q^{\mu}_{\phi}(s,a) - r_{s,a} - \gamma V^{lb}_{\phi}(s')\right)^2. \tag{12}$$

In this formulation, $\mathcal{D}_V$ represents the validation replay buffer, with s, a, s' denoting transitions sampled from this buffer. In line with other stochastic policy algorithms, we further approximate $V^{lb}_{\phi}(s')$ with the critic output for a single action $a' \sim \pi_{\theta}(a'|s')$. As follows, VPL adjusts the pessimism under the assumption that $Q^{\mu}_{\phi}(s,a)$ is a good representation of $Q^{\pi}(s,a)$. Since the actions at which $Q^{\mu}_{\phi}(s,a)$ is evaluated are sampled from the validation buffer and are off-policy, these actions are likely to produce less overestimation than the adversarial actions sampled from a value-maximizing policy.

The inclusion of a square in the loss function ensures that the optimization remains non-negative and convex with respect to β, focusing the updates on reducing significant errors, which are particularly impactful in reinforcement learning scenarios. Additionally, removing the stop-gradient operator allows β to be updated dynamically based on both the approximation error and the disagreement among critics (Q^{σ}), tightly coupling the level of pessimism to the degree of uncertainty in the critic estimates. This ensures that the level of pessimism is adjusted adaptively during training, aligning with the intuition that higher disagreement indicates greater uncertainty, necessitating increased pessimism.

While we acknowledge that providing a strict mathematical justification for all aspects of the proposed rule is challenging, our empirical results (Figure 3) demonstrate its practical effectiveness in reducing overfitting and improving performance across diverse environments. VPL mitigates the risk of overfitting by computing the pessimism loss exclusively on validation samples that are not used by the actor-critic modules. Unlike GPL, VPL fully incorporates critics' disagreement into the optimization process and dynamically adjusts β, enabling more precise control of the level of pessimism based on the observed uncertainty.

5 Experiments

Our experiments are based on the JaxRL codebase [Kostrikov, 2021]. Since all considered algorithms use SR-SAC [D'Oro *et al.*, 2022] as their backbone, we align the common hyperparameters with those recommended for Scaled-By-Resetting SAC (SR-SAC) as per [D'Oro *et al.*, 2022]. This includes using the same network architectures and a two-critic ensemble in accordance with established practices [Fujimoto *et al.*, 2018], [Haarnoja *et al.*, 2018],

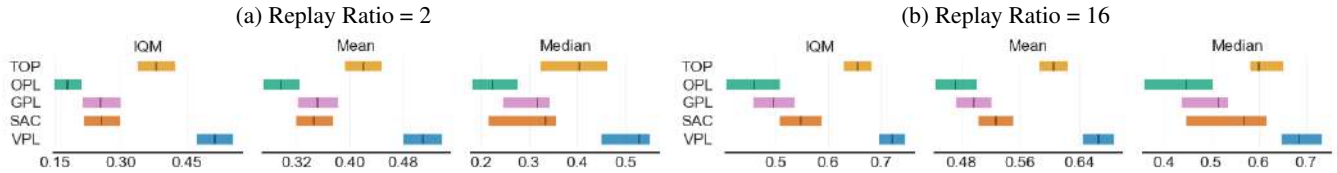


Figure 5: RLiable final performance metrics for the main experiment detailed in Section 5.1. VPL outperforms baseline algorithms in both replay regimes. The performance metrics are calculated on 20 tasks listed in Table 3 with 10 random seeds per task.

[Ciosek *et al.*, 2019], [Moskovitz *et al.*, 2021], [Cetin and Celiktutan, 2023]. We conduct our experiments in two environments: the DeepMind Control (DMC) suite [Tassa *et al.*, 2018] and the single-task MetaWorld [Yu *et al.*, 2020]. Our study encompasses two replay regimes: a compute-efficient setup with 2 gradient steps per environment step without resets, and a sample-efficient setup with 16 gradient steps per environment step, including full-parameter resets every 160k steps, as suggested by [D'Oro *et al.*, 2022]. We provide robust analysis using the RLiable package [Agarwal *et al.*, 2021] and detail the experimental setting in Appendix E.

5.1 Performance and Sample Efficiency

Firstly, we test the performance and sample efficiency of the proposed approach. To this end, we compare SR-SAC [D'Oro *et al.*, 2022] (DMC state of the art) to four algorithms that extend SR-SAC with online pessimism adjustment: GPL [Cetin and Celiktutan, 2023]; OPL [Kuznetsov *et al.*, 2021]; TOP [Moskovitz *et al.*, 2021]; and VPL (the proposed approach). We run the tested algorithms in both replay regimes for 1mln environment steps on 20 medium to hard tasks (10 from DMC and 10 from MetaWorld). We discuss the chosen baselines in Sections 2.2 & C. We discuss hyperparameter selection in Appendix G and the tested tasks in F. We report the results of this experiment in Figures 3, 4 & 5. We find that the proposed approach surpasses baseline algorithms, demonstrating 48% and 27% higher performance than the baseline SR-SAC in low and high replay regimes, respectively. As depicted in Figure 4, VPL exhibits particular effectiveness in MetaWorld manipulation tasks, developing robust policies in environments where other approaches fail, such as the assembly task. To provide further practical context, we evaluate the computational overhead introduced by each of the tested pessimism adjustment methods. Table 1 shows that VPL maintains a low wall-clock overhead compared to SR-SAC, with only a 3.5% increase in the low-replay regime and 3.8% in the high-replay regime. This minimal computational overhead highlights VPL's suitability for large-scale and real-time reinforcement learning applications.

METHOD	GPL	OPL	TOP	VPL
RR$=2$	0.3%	6.3%	0.3%	3.5%
RR$=16$	0.5%	1.1%	0.1%	3.8%

Table 1: We measure runtimes for 2000 runs of each algorithm and find that the pessimism adjustment methods have trivial wall-clock overhead as compared to SAC/SR-SAC.

5.2 Validation Buffer Regret

To understand the impact of a validation buffer on online RL training, we analyze three distinct agent setups: *baseline* SR-SAC, which operates without a validation buffer, thus updating actor-critic modules with all experienced transitions; *regret* SR-SAC, which maintains a validation buffer but does not employ validation transitions for pessimism adjustment; and SR-SAC-VPL, which not only maintains a validation buffer but also utilizes validation transitions for pessimism adjustment (we present these results in Figure 6).

This comparative analysis aims to isolate the performance loss attributable to the presence of a validation buffer and the efficiency gains derived from employing VPL for updating pessimism. We evaluate these agents in high-replay regime on 4 tasks (listed in Table 4) over 1mln environment steps, using varying ratios of validation to training samples, specifically at proportions of $\frac{1}{128}$, $\frac{1}{32}$, $\frac{1}{8}$, and $\frac{1}{2}$. The results for this experiment are presented in Figure 6. We observe that the regret associated with maintaining a validation buffer, and thus not utilizing it for actor-critic updates, diminishes over the course of training. Specifically, the *regret* SR-SAC reaches parity with the SR-SAC in performance for all validation proportions except at $\frac{1}{2}$. We note that the rate of regret reduction correlates with the size of the validation proportion, with smaller proportions converging to baseline performance more rapidly. When examining the effectiveness of pessimism adjustment, we observe its most pronounced impact during the early stages of training. This trend aligns with the expectation of reducing critic disagreement over time. Additionally, the extent of performance gain appears to be influenced by the size of the validation buffer, where larger proportions yield greater improvements. This effect is likely due to the increased diversity of environment transitions available for pessimism adjustment in larger buffers. When considering the combined effects on performance, our findings indicate that, except for the $\frac{1}{2}$ proportion, all validation proportions successfully compensate for the performance loss due to validation buffer maintenance. This result is in line with the broader experimental results presented in Figures 3 & 5.

5.3 Other Experiments

We investigate the sensitivity of VPL to varying pessimism learning rates as compared other pessimism adjustment algorithms. Given the dependency of such learning rate on reward scales and environmental dynamics, determining an optimal rate a priori is challenging, which is a significant restriction for practical applications. To address this, we test the performance of VPL, GPL, and OPL across four environments

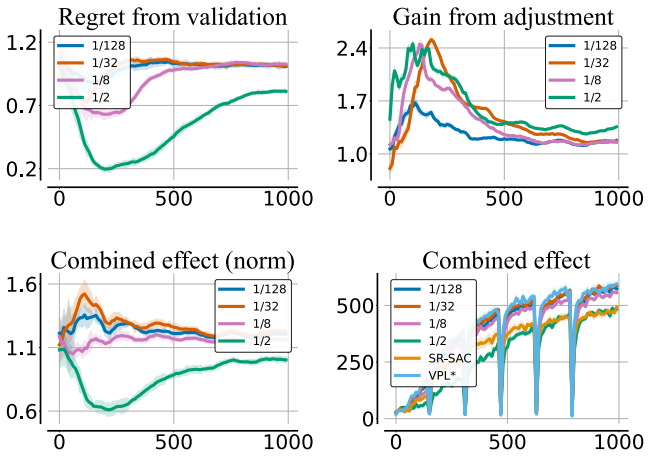


Figure 6: We examine the impact of maintaining a validation buffer on performance distinct from pessimism adjustment across varying proportions of validation samples. Upper-left figure demonstrates whether validation agents can match the performance of their validation-free counterparts without utilizing validation samples for pessimism updates, enabling quantification of the regret associated with allocating samples to a validation buffer. Upper-right figure quantifies the performance gains attributable to pessimism by contrasting agents that do not update pessimism against those that do. Figures in the bottom illustrate the cumulative effect of validation pessimism adjustment for different validation ratios, benchmarking against the baseline performance of SR-SAC and VPL with "free" validation (denoted as VPL*). X-axis denotes environments steps (in thousands) and y-axis denotes performance.

detailed in Table 4 in the high-replay regime. We evaluate agents after 500k environments steps for learning rates of $[5e-5, 5e-4, 5e-3, 5e-2]$. The results, presented in Figure 7, indicate that VPL exhibits less sensitivity to changes in the pessimism learning rate than the other considered algorithms. Furthermore, we investigate the importance of the two proposed design elements: the use of a validation buffer and the VPL pessimism loss as formulated in Equation 12. To this end, we compare the performance of six agents, each employing different combinations of pessimism loss – either the dual optimization pessimism loss or the VPL pessimism loss – along with varying sources for pessimism updates. These sources include samples from the replay buffer, the validation buffer, and the most recent transitions. The results of this analysis are presented in Figure 9. In our final analysis, we focus on validating the premise of VPL: its effectiveness in reducing approximation error and mitigating overfitting compared to baseline algorithms. Our methodology for quantifying approximation error and overfitting are described in Appendix E. We conducted these measurements across both low and high replay regimes, using a selection of 20 tasks from the DMC and MetaWorld as listed in Table 3. The findings, depicted in Figure 3 and Appendix H, confirm that VPL achieves the lowest levels of critic overfitting and approximation error in both replay scenarios.

6 Limitations

The primary challenge of VPL lies in estimating the lower-bound approximation error necessary for the pessimism adjustment mechanism. This estimation currently relies on a simplified assumption, inherited from GPL and discussed in Section 4.2, that Q^{μ} provides a reasonable approximation of Q^{π}. While this assumption works well in many standard benchmarks, it may limit the applicability of VPL in environments with high policy dynamics or entropy, where off-policy actions are less representative of on-policy behavior. Future research could explore alternative estimation methods for the lower-bound approximation error that do not rely on this assumption, potentially leading to more robust algorithms. Surprisingly, our experiments (see Figure 6) reveal that using a validation buffer does not detrimentally impact agent performance in high-replay scenarios, except in extremely sample-scarce environments (e.g., fewer than 250k environment steps). Addressing the limitations of the current error estimation framework may also mitigate this sample-scarcity sensitivity.

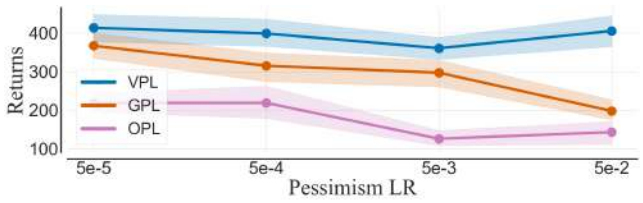


Figure 7: VPL exhibits substantially less sensitivity to the learning rate of the pessimism module. 4 tasks, 10 seeds per task.

7 Conclusions

This paper examined the approximation error in critic networks optimized via temporal difference variants. We introduced a fixed-point model for estimating mean and lower bound errors and used this model to analyze the convergence of pessimistic actor-critic algorithms. We proposed the VPL algorithm, which dynamically adjusts pessimism levels to minimize approximation errors of critic supervision in validation samples. We tested VPL against baseline algorithms in various locomotion and manipulation tasks, showing performance and sample efficiency improvements. We explored the impact of VPL components and their sensitivity to hyperparameter selection. Our results confirm VPLs effectiveness in complex continuous action tasks.

Acknowledgements

Marek Cygan was partially supported by an NCBiR grant POIR.01.01.01-00-0433/20. Mateusz Ostaszewski was partially supported by the National Science Centre, Poland, under a grant 2023/51/D/ST6/01609. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2025/018377

References

[Agarwal *et al.*, 2021] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Belle-

mare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[Ball *et al.*, 2023] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. 2023.

[Bellemare *et al.*, 2017] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[Bergstra and Bengio, 2012] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[Bishop and Nasrabadi, 2006] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[Cetin and Celiktutan, 2023] Edoardo Cetin and Oya Celiktutan. Learning pessimism for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6971–6979, 2023.

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

[Ciosek *et al.*, 2019] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32, 2019.

[D'Oro *et al.*, 2022] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2022.

[Farahmand *et al.*, 2010] Amir-massoud Farahmand, Csaba Szepesvári, and Rémi Munos. Error propagation for approximate policy and value iteration. *Advances in Neural Information Processing Systems*, 23, 2010.

[Foret *et al.*, 2020] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2020.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[Ha and Schmidhuber, 2018] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[Haarnoja *et al.*, 2017] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[Hasselt, 2010] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[Hessel *et al.*, 2018] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Hiraoka *et al.*, 2021] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2021.

[Januszewski *et al.*, 2021] Piotr Januszewski, Mateusz Olko, Michał Królikowski, Jakub Swiatkowski, Marcin Andrychowicz, Łukasz Kuciński, and Piotr Miłoś. Continuous control with ensemble deep deterministic policy gradients. In *Deep RL Workshop NeurIPS 2021*, 2021.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kostrikov, 2021] Ilya Kostrikov. JAXRL: Implementations of Reinforcement Learning algorithms in JAX, 10 2021.

[Kumar *et al.*, 2019] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[Kumar *et al.*, 2020] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *Advances in Neural Information Processing Systems*, 33:18560–18572, 2020.

[Kuznetsov *et al.*, 2020] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pages 5556–5566. PMLR, 2020.

[Kuznetsov *et al.*, 2021] Arsenii Kuznetsov, Alexander Grishin, Artem Tsypin, Arsenii Ashukha, Artur Kadurin, and Dmitry Vetrov. Automating control of overestimation bias for reinforcement learning. *arXiv preprint arXiv:2110.13523*, 2021.

[Lee *et al.*, 2023] Hojoon Lee, Hanseul Cho, Hyunseung Kim, Daehoon Gwak, Joonkee Kim, Jaegul Choo, Se-Young Yun, and Chulhee Yun. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[Li *et al.*, 2022] Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. In *The Eleventh International Conference on Learning Representations*, 2022.

[Lyle *et al.*, 2023] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. *arXiv preprint arXiv:2303.01486*, 2023.

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

[Moskovitz *et al.*, 2021] Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12849–12863, 2021.

[Munos and Szepesvári, 2008] Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5), 2008.

[Munos, 2005] Rémi Munos. Error bounds for approximate value iteration. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1006. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[Nauman and Cygan, 2023] Michal Nauman and Marek Cygan. On the theory of risk-aware agents: Bridging actor-critic and economics. In *ICML 2024 Workshop: Aligning Reinforcement Learning Experimentalists and Theorists*, 2023.

[Nauman *et al.*, 2024] Michal Nauman, Michał Bortkiewicz, Piotr Miłoś, Tomasz Trzcinski, Mateusz Ostaszewski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. PMLR 235:37342-37364.

[Nikishin *et al.*, 2022] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pages 16828–16847. PMLR, 2022.

[Prechelt, 2002] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 2002.

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

[Rosenfeld *et al.*, 2019] Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. In *International Conference on Learning Representations*, 2019.

[Rowland *et al.*, 2023] Mark Rowland, Rémi Munos, Mohammad Gheshlaghi Azar, Yunhao Tang, Georg Ostrovski, Anna Harutyunyan, Karl Tuyls, Marc G Bellemare, and Will Dabney. An analysis of quantile temporal-difference learning. *arXiv preprint arXiv:2301.04462*, 2023.

[Seyde *et al.*, 2022] Tim Seyde, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Learning to plan optimistically: Uncertainty-guided deep exploration via latent model ensembles. In *Conference on Robot Learning*, pages 1156–1167. PMLR, 2022.

[Shang *et al.*, 2016] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning*, pages 2217–2225. PMLR, 2016.

[Silver *et al.*, 2014] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[Tassa *et al.*, 2018] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[Thrun and Schwartz, 2014] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*, pages 255–263. Psychology Press, 2014.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[Wang *et al.*, 2022] Zhihai Wang, Jie Wang, Qi Zhou, Bin Li, and Houqiang Li. Sample-efficient reinforcement learning via conservative model-based actor-critic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8612–8620, 2022.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[Yao *et al.*, 2021] Yao Yao, Li Xiao, Zhicheng An, Wanpeng Zhang, and Dijun Luo. Sample efficient reinforcement learning via model-ensemble exploration and exploitation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4202–4208. IEEE, 2021.

[Yu *et al.*, 2020] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[Zhang *et al.*, 2017] Zongzhang Zhang, Zhiyuan Pan, and Mykel J Kochenderfer. Weighted double q-learning. In *IJCAI*, pages 3455–3461, 2017.

Appendix Contents

We divide the Appendix into the following sections:

1. Derivations (Appendix A) - we present the derivations associated with statements presented in Section 3. Furthermore, we discuss the further implications of our propositions.
2. VPL pseudocode (Appendix B.1 – we present the pseudocode of one step in the Validation Pessimism Learning algorithm.
3. Related Work (Appendix C) - we discuss the works related to the proposed method. In particular, we discuss pessimistic actor-critic algorithms, approaches for online pessimism adjustment and theoretical work on approximation error in TD learning.
4. Future Work (Appendix D) - we discuss avenues for potential further research related to the proposed method.
5. Experimental Details (Appendix E) - we detail all experiments presented throughout the manuscript.
6. Tested Environments (Appendix F) - we list all tested environments from the DeepMind Control and MetaWorld environments.
7. Additional Experimental Results (Appendix H) - we present additional experimental results.
8. Hyperparameters (Appendix G) - we discuss the procedure for hyperparamenter selection for all algorithms and list all used hyperparameters.
9. Learning Curves (Appendix I) - we present learning curves for the experiments.

A Derivations

In this section, we derive statements presented in Section 3. For simplicity, we consider a fixed policy π_θ and use $V(s)$ and $Q(s,a)$ to represent the value and Q-value under this policy. We define the mean and lower bound approximation errors denoted as U^μ_ϕ and U^{lb}_ϕ respectively:

$$\begin{aligned} &U^\mu_\phi(s,a)Q(s,a) - Q^\mu_\phi(s,a) \\ &U^{lb}_\phi(s,a)Q(s,a) - Q^{lb}_\phi(s,a) \end{aligned} \tag{13}$$

$Q(s,a)$ denotes the true Q-value, the term $Q^\mu_\phi(s,a)$ represents the mean Q-value estimated by an ensemble of k critics, calculated as $Q^\mu\phi(s,a) = \frac{1}{k}\sum^k Q^i\phi(s,a)$, and $Q^{lb}\phi(s,a)$ is the lower bound Q-value as defined as follows:

$$Q^{lb}_\phi(s,a) = Q^\mu_\phi(s,a) - \beta Q^\sigma_\phi(s,a) \tag{14}$$

Similarly, we define lower bound value:

$$V^{lb}_\phi(s) = \mathop{\mathrm{E}}_{a\sim\pi_\theta}\big(Q^\mu_\phi(s,a) - \beta Q^\sigma_\phi(s,a) - \alpha\log\pi_\theta(a|s)\big) \tag{15}$$

We also introduce the mean and lower bound temporal critic errors, denoted as u^μ_ϕ and u^{lb}_ϕ, respectively:

$$\begin{aligned} &u^\mu_\phi(s,a,s')r_{s,a} + \gamma V^\mu_\phi(s') - Q^\mu_\phi(s,a) \\ &u^{lb}_\phi(s,a,s')r_{s,a} + \gamma V^{lb}_\phi(s') - Q^\mu_\phi(s,a) \end{aligned} \tag{16}$$

These temporal critic errors quantify the deviation between the Q-values $Q^\mu_\phi(s,a)$ and the mean or lower bound Temporal Difference (TD) targets.

A.1 Approximation Error Operator

Firstly, we note that for the true Q-value the following always holds:

$$\begin{aligned} Q(s,a) &= r_{s,a} + \gamma V(s') \\ &= r_{s,a} + \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta}\big(Q(s',a') - \alpha\log\pi_\theta(a'|s')\big) \end{aligned} \tag{17}$$

Then, using Equations 13, 16 & 17 we write:

$$\begin{aligned} U^\mu_\phi(s,a) &= Q(s,a) - Q^\mu_\phi(s,a) \\ &= r_{s,a} + \gamma V(s') - r_{s,a} - \gamma V^\mu_\phi(s') + u^\mu_\phi(s,a,s') \\ &= u^\mu_\phi(s,a,s') + \gamma\big(V(s') - V^\mu_\phi(s')\big) \\ &= u^\mu_\phi(s,a,s') + \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta}\big(Q(s',a') \\ &- \alpha\log\pi_\theta(a'|s') - Q^\mu_\phi(s',a') + \alpha\log\pi_\theta(a'|s')\big) \\ &= u^\mu_\phi(s,a,s') + \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta} U^\mu_\phi(s',a') \end{aligned} \tag{18}$$

Similarly, we calculate $U^{lb}_\phi(s,a)$:

$$\begin{aligned} U^{lb}_\phi(s,a) &= Q(s,a) - Q^\mu_\phi(s,a) + \beta Q^\sigma_\phi(s,a) \\ &= r_{s,a} + \gamma V(s') + u^{lb}_\phi(s,a,s') \\ &- r_{s,a} - \gamma V^{lb}_\phi(s') + \beta Q^\sigma_\phi(s,a) \\ &= u^{lb}_\phi(s,a,s') + \beta Q^\sigma_\phi(s,a) \\ &+ \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta}\big(Q(s',a') - \alpha\log\pi_\theta(a'|s')\big) \\ &- \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta}\big(Q^\mu_\phi(s',a') - \beta Q^\sigma_\phi(s',a') - \alpha\log\pi_\theta(a'|s')\big) \\ &= u^{lb}_\phi(s,a,s') + \beta Q^\sigma_\phi(s,a) \\ &+ \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta}\big(Q(s',a') - Q^\mu_\phi(s',a') + \beta Q^\sigma_\phi(s',a')\big) \\ &= u^{lb}_\phi(s,a,s') + \beta Q^\sigma_\phi(s,a) \\ &+ \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta}\big(Q(s',a') - Q^{lb}_\phi(s',a')\big) \\ &= u^{lb}_\phi(s,a,s') + \beta Q^\sigma_\phi(s,a) + \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta} U^{lb}_\phi(s',a') \end{aligned} \tag{19}$$

As such, both $U^\mu_\phi(s,a)$ and $U^{lb}_\phi(s,a)$ can be expressed as a function of combination of $U^\mu_\phi(s',a')$ or $U^{lb}_\phi(s',a')$ and $u^\mu_\phi(s,a,s')$ or $u^{lb}_\phi(s,a,s')$.

A.2 Approximation Error Contraction

We show that both approximation error operators are contractions wrt. infinity norm with similar argument to Bellman values [Puterman, 2014].

$$\begin{aligned}||\mathcal{U}(f_1) - \mathcal{U}(f_2)||_\infty &= \sup_{s,a} |u^\mu_\phi(s,a,s') + \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta} f_1(s',a') \\ &\quad - u^\mu_\phi(s,a,s') - \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta} f_2(s',a')| \\ &= \gamma |\mathop{\mathrm{E}}_{a'\sim\pi_\theta} f_1(s',a') - \mathop{\mathrm{E}}_{a'\sim\pi_\theta} f_2(s',a')| \\ &\leq \gamma \mathop{\mathrm{E}}_{a'\sim\pi_\theta} |f_1(s',a') - f_2(s',a')| \\ &\leq \gamma ||f_1 - f_2||_\infty. \end{aligned} \tag{20}$$

B Proof of Approximation Error Relationship

Here, we prove the implication 10 from the main text:

$$U^\mu_\phi(s,a) > 0 \implies |U^\mu_\phi(s,a)| \leq |U^{lb}_\phi(s,a)|.$$

From the definition of the lower-bound approximation error:

$$U^{lb}_\phi(s,a) = U^\mu_\phi(s,a) + \beta Q^\sigma(s,a),$$

where $\beta \geq 0$ and $Q^\sigma(s,a) \geq 0$, it follows that:

$$|U^{lb}_\phi(s,a)| = |U^\mu_\phi(s,a) + \beta Q^\sigma(s,a)|.$$

If $U^\mu_\phi(s,a) > 0$, we have:

$$U^{lb}_\phi(s,a) = U^\mu_\phi(s,a) + \beta Q^\sigma(s,a) \geq U^\mu_\phi(s,a),$$

as both β and $Q^\sigma(s,a)$ are non-negative. Therefore:

$$|U^{lb}_\phi(s,a)| \geq |U^\mu_\phi(s,a)|.$$

This completes the proof.

B.1 VPL pseudocode

Algorithm 1 Validation Pessimism Learning Step

Input: π_θ - actor; Q_ϕ - critic; α - temperature; $\mathcal{D}_T$ - replay buffer; β - pessimism; $\mathcal{D}_V$ - validation buffer
Hyperparameters: B - batch size; v - validation rate
$s', r = \text{ENV.STEP}(a) \quad with \quad a \sim \pi_\theta(a|s)$
$p \sim U(0,1)$
if $p > v$: **then**
  $\mathcal{D}_T.\text{ADD}(s,a,r,s')$
end if
if $p \leq v$: **then**
  $\mathcal{D}_V.\text{ADD}(s,a,r,s')$
end if
for $i = 1$ **to** ReplayRatio **do**
  $s,a,r,s' \sim \mathcal{D}_T.\text{SAMPLE(B)}$
  $s_V, a_V, r_V, s'_V \sim \mathcal{D}_V.\text{SAMPLE(VB)}$
  $\phi \leftarrow \phi - \nabla_\phi \big(Q^\pi_\phi(s,a) - r - \gamma V^{lb}_\phi(s')\big)^2$
  $\theta \leftarrow \theta + \nabla_\theta V^\pi_\theta(s)$
  $\alpha \leftarrow \alpha - \nabla_\alpha \alpha(-\log \pi(a|s) - \mathcal{H}^*)$
  $\beta \leftarrow \beta - \nabla_\beta \big(Q^\pi_\phi(s_V, a_V) - r_V - \gamma V^{lb}_\phi(s'_V)\big)^2$
end for

C Related Work

C.1 Pessimistic Actor-Critic

Recent model-free, off-policy algorithms address the overestimation bias in critic's TD-targets through diverse methods [Thrun and Schwartz, 2014], [Hasselt, 2010]. These include leveraging multiple function approximators to conservatively estimate expected returns [Fujimoto *et al.*, 2018], [Haarnoja *et al.*, 2018], [Ciosek *et al.*, 2019]. Notably, Clipped Double Q-learning (CDQL) employs a pessimistic approach by calculating the critic's TD-targets as the minimum of two action-value model outputs [Fujimoto *et al.*, 2018]. Weighted Double Q-Learning (WDQL) introduces a weighted sum of mean and minimum targets for TD calculations [Zhang *et al.*, 2017]. Furthermore, [Kuznetsov *et al.*, 2020] suggests using a quantile distributional critic with interquartile statistics for TD target computations. An alternative method proposes reducing approximation errors in TD loss by varying batch sample weights as to counteract the negative interaction of approximation bias and the data-collecting distribution [Kumar *et al.*, 2020]. Pessimism was also studied in the context of model-based RL [Ha and Schmidhuber, 2018], [Seyde *et al.*, 2022], [Wang *et al.*, 2022]. A popular approach is to avoid or reduce the impact of simulated trajectories that the dynamics model deems uncertain [Yao *et al.*, 2021]. In this context, similarly to value, model ensemble disagreement is a very popular approach to uncertainty quantification [Yao *et al.*, 2021].

C.2 Pessimism Adjustment

Adjusting pessimism levels has become more dynamic with the development of methods that represent the minimum target as a function of the mean and critic ensemble disagreement [Kuznetsov *et al.*, 2021; **?**], [Cetin and Celiktutan, 2023]. GPL, for instance, modifies pessimism using a dual optimization objective, calculating the loss on replay samples [Cetin and Celiktutan, 2023]. OPL uses an online approach to adjust pessimism levels by comparing critic outputs with on-policy return estimators [Kuznetsov *et al.*, 2021]. TOP uses an auxiliary bandit to select optimal pessimism levels for maximizing online returns [Moskovitz *et al.*, 2021]. These approaches are detailed in Table 2.

C.3 Approximation Error in RL

The regret caused by errors in critic approximation has been explored in approximate value iteration algorithms [Munos, 2005], [Munos and Szepesvári, 2008], [Farahmand *et al.*, 2010]. When a policy is greedy with respect to the critic estimates, value approximation errors can greatly influence the policy and the resulting returns. Therefore, there's been significant work to understand how these approximation errors affect performance [Munos, 2005], [Munos and Szepesvári, 2008], [Farahmand *et al.*, 2010]. These ideas have also been revisited in the area of deep reinforcement learning [Kumar *et al.*, 2019], [Kumar *et al.*, 2020]. In particular, [Kumar *et*

Table 2: Considered algorithms differ by the pessimism domain, strategy for critic error estimation, as well as the pessimism update rule.

	β DOMAIN	CRITIC ERROR ESTIMATED VIA	PESSIMISM UPDATE RULE
TOP	$0, 1$	NA	Auxiliary bandit maximizing episodic returns
GPL	$[0, \infty]$	Critic output on the replay transitions	Dual optimization update
OPL	$[0, \infty]$	Bootstrapped λ-returns on recent transitions	Dual optimization update
VPL	$[0, \infty]$	Critic output on the validation transitions	Minimization of approximation error

al., 2020] examines the detailed patterns in non-pessimistic value approximation errors. Those results remain relevant for off-policy actor-critic algorithms such as SAC, as it can be described as an approximate policy iteration algorithm [Haarnoja *et al.*, 2018].

D Future Work

While our implementation is based on the vanilla SR-SAC algorithm, recent studies have demonstrated that simple regularization methods applied to the critic can significantly enhance performance [Hiraoka *et al.*, 2021], [Li *et al.*, 2022], [Ball *et al.*, 2023]. Consequently, integrating VPL with network regularization appears to be a promising approach. Specifically, layer normalization and spectral normalization have been effective in continuous action off-policy agents [Nauman *et al.*, 2024]. Similarly, it has been observed that deep RL agents experience a reduced ability to learn over time, a phenomenon known as 'plasticity loss'. Addressing this diminishing capacity has been shown to be empirically beneficial [Nikishin *et al.*, 2022], [D'Oro *et al.*, 2022], [Lyle *et al.*, 2023]. Although our approach involves full-parameter resets in the high replay regime, employing multiple techniques to address plasticity loss has proven advantageous [Lee *et al.*, 2023]. Therefore, combining VPL with strategies like CReLU [Shang *et al.*, 2016] or Sharpness Aware Minimization (SAM) [Foret *et al.*, 2020] could potentially lead to further performance improvements. Given that VPL employs a more controlled use of the critic ensemble compared to standard SAC/TD3 methods, increasing the critic ensemble size in VPL may create synergies, potentially surpassing the benefits seen in conventional ensemble AC approaches [Chen *et al.*, 2020], [Januszewski *et al.*, 2021], [Ball *et al.*, 2023]. Additionally, the integration of a distributional critic setup into the pessimism adjustment framework [Moskovitz *et al.*, 2021], which has been shown to enhance RL learning [Bellemare *et al.*, 2017], [Rowland *et al.*, 2023], suggests that incorporating distributional critics into VPL could yield notable performance gains.

E Experimental Details

E.1 Performance and Sample Efficiency

We run the tested algorithms for 1mln environment steps on 20 DMC/MetaWorld tasks listed in Table 3 using hyperparameters described in Section G. All algorithms are evaluated via greedy policies every 10k environment steps. We calculate the final performance presented in Figure 5 by averaging over last 10 policy evaluations (ie. the last 100k environment steps). The results for this setup are presented in Figures 3, 4 and 5, as well as in the final Appendix section.

E.2 Approximation Error and Overfitting

Throughout the manuscripts we present estimates of critic approximation error and overfitting. Here, we discuss our methodology for calculating both.

Approximation Error

We calculate the critic approximation via:

$$U^{\mu}_{\phi}(s,a) = -(Q(s,a) - Q^{\mu}_{\phi}(s,a)) \tag{21}$$

Where we add the negative sign such that the positive approximation error represents overestimation, and negative approximation error represents underestimation. Above, $Q^{\mu}_{\phi}(s,a)$ represents the output of the critic. Estimating the reference Q-value $Q(s,a)$ in our context demands solving two problems. Firstly, as all algorithms use SAC as their backbone, the Q-value is soft, ie. it represents the sum of returns and entropies. This contrasts with regular DDPG-style critic which approximates the returns alone. Secondly, both MetaWorld and DMC are infinite-horizon MDPs. As such, obtaining an unbiased Monte-Carlo rollout value is non-trivial. To this end, we calculate the reference Q-value via:

$$Q(s,a) = \frac{1}{1-\gamma}\big(\hat{R}_{s,a} - \alpha \log \hat{\pi}(a|s)\big) \tag{22}$$

Where $\hat{R}_{s,a}$ denotes the average reward gathered when performing action a at state s and following the policy afterwards, and $\log \hat{\pi}$ denotes the average policy log-probability when following the policy from a given state-action. The term $\frac{1}{1-\gamma}$ stems from the sum of a geometric series, reflecting the infinite-horizon that the critic models. We estimate $\hat{R}$ with using a Monte-Carlo rollout, and use the entropy target to calculate $\log \hat{\pi}$. For each approximation error measurement, we average the error over 5 different starting states.

Overfitting

We calculate the critic overfitting (denoted as $\mathcal{O}(\phi)$) with the following equation:

$$\mathcal{O}(\phi) = \frac{\mathop{\mathrm{E}}_{\mathcal{D}_V} u^{lb}_{\phi}(s_V, a_V, s^{'}_V)}{\mathop{\mathrm{E}}_{\mathcal{D}_T} u^{lb}_{\phi}(s, a, s')} \tag{23}$$

Where $\mathcal{D}_T$ and $\mathcal{D}_V$ denote the training and validation replay buffers respectively. Furthermore, $s_V, a_V, s^{'}_V$ denote the transitions sampled from the validation buffer, and s, a, s' denote the training replay buffer transitions. As such, we calculate our overfitting metric by comparing the temporal difference for unseen validation transitions and the training transitions which were the critic is trained on. For each algo-

rithm, we gather such validation samples during the evaluation rollouts. As such, the validation buffer gathers 5,000 new transitions every 10,000 training transitions. We estimate the expectation on batches of 256 transitions sampled from each buffer. Such definition of overfitting is easily interpretable - when $\mathcal{O}(\phi)$ is close to 1 then both validation and training TD errors are comparable and therefore there is little to none overfitting. When $\mathcal{O}(\phi)$ is greater than 1 then it means that the validation TD errors are relatively larger than the ones in the training, indicating overfitting.

E.3 Validation Buffer Regret

Our study examines three agent configurations: (1) baseline SR-SAC, which updates actor-critic modules with all transitions and lacks a validation buffer; (2) regret SR-SAC, featuring a validation buffer but not using it for pessimism adjustment; and (3) SR-SAC-VPL, which includes a validation buffer and employs validation transitions for pessimism adjustment. The analysis focuses on identifying performance differences caused by the validation buffer and the benefits of using VPL for pessimism updates. We tested these agents across four tasks (see Table 4) for 1 million environment steps, exploring various validation to training sample ratios, namely $\frac{1}{128}$, $\frac{1}{32}$, $\frac{1}{8}$, and $\frac{1}{2}$. The experiments are performed in high replay setting with full parameter resets every 160k environment steps [D'Oro *et al.*, 2022]. Experimental results are detailed in Figure 6.

E.4 Learning Rate Sensitivity

We run GPL, OPL and VPL for 500k environment steps in high replay setup on 4 DMC tasks listed in Table 4. We test each algorithms performance on four pessimism learning rate values: $[5e-5, 5e-4, 5e-3, 5e-2]$. We present the results in Figure 7.

E.5 Design Choices

Finally, we evaluate the impact of each of VPL contributions, namely the VPL update rule and performing updates on the validation buffer. We evaluate six agents, each utilizing different forms of pessimism loss - either dual optimization or VPL pessimism loss - in combination with various sources for pessimism updates. These sources encompass samples from the replay buffer (ie. GPL), the validation buffer (ie. VPL), and the most recent online transitions (ie. OPL). We run the agents for 1mln environment steps on 5 dmc tasks shown in Figure 9.

F Tested Environments

Tables below list tasks from DeepMind Control and Meta-World considered in our experiments.

G Hyperparameters

All considered algorithms use SR-SAC [D'Oro *et al.*, 2022] as their backbonce, we align the common hyperparameters with those recommended for SR-SAC [D'Oro *et al.*, 2022]. This includes using the same network architectures, a two-critic ensemble [Fujimoto *et al.*, 2018], [Haarnoja *et al.*, 2018], [Ciosek *et al.*, 2019], [Moskovitz *et al.*, 2021], [Cetin and Celiktutan, 2023] and ADAM optimizer [Kingma and Ba, 2014]. We choose VPL, GPL, and OPL pessimism adjustment learning rate by performing search over the same domain for all algorithms which we present in Figure 7. We choose the validation ratio for VPL in experiments presented in Figure 6. We choose TOP bandit setting following the best performing configurations presented [Moskovitz *et al.*, 2021]. We use consistent hyperparamenters between all environments and both replay regimes. All experiments are run without any action repeat wrappers (ie. we use an action repeat of 1). The hyperparameters are summarized in Table 5.

H Additional Experimental Results

We evaluate the impact of each of VPL contributions, namely the VPL update rule and performing updates on the validation buffer. We evaluate six agents, each utilizing different forms of pessimism loss - either dual optimization (denoted as "Dual") or VPL pessimism loss (denoted as "VPL") - in combination with one of three sources of data for pessimism updates. These sources encompass samples from the validation buffer (denoted as "Validation"), the replay buffer (denoted as "Replay" and used originally in GPL [Cetin and Celiktutan, 2023]), and the most recent online transitions (denoted as "Online" and used originally in OPL [Kuznetsov *et al.*, 2021]). The performance, pessimism, approximation error and overfitting of these agents is presented in Figures 8 and 9.

We find that the VPL pessimism adjustment loss accounts for the majority of the performance improvements of VPL algorithm. As such, we find that updating the pessimism on validation buffer accounts for minor performance improvements over updates performed on the replay buffer. We still deem this result as significant, since the validation agent skips $\frac{1}{16}$ of training transitions. Furthermore, we find that performing VPL pessimism updated on recent transitions leads to pessimism increases and subpar performance. Interestingly, we find that performing dual optimization pessimism updates on the validation buffer leads to the worse performing agent. In terms of pessimism patterns, we observe that performing pessimism updates on the online transitions leads to more pessimistic agents than then other data sources for pessimism updates. Finally, we find that the VPL pessimism updates performed on the replay data leads to sharper decreases of pessimism that the regular validation VPL.

Finally, we investigate whether the critic disagreement diminishes to zero in a popular training regime of 1mln environment steps. As noted in the paper, the convergence of pessimistic actor-critic depends on the critic disagreement being equal to zero. We find that the critic disagreement indeed does not completely diminish. Interestingly, we observe that the same environments yield the most disagreement in both low and high replay regimes.

I Learning Curves

Finally, we present the detailed training curves.

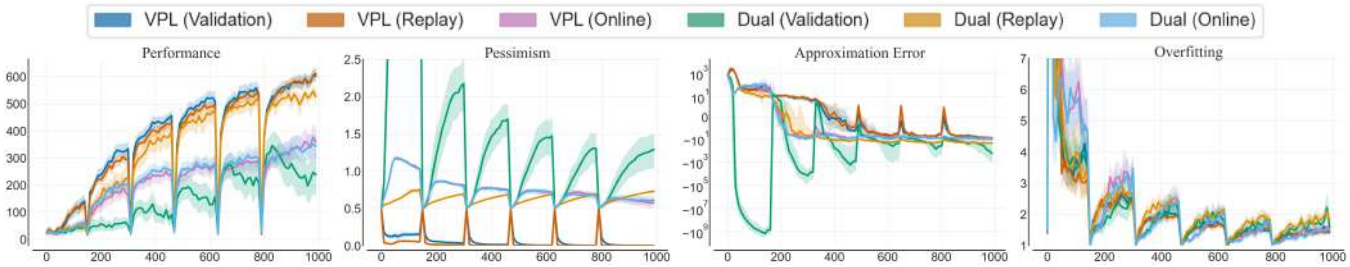


Figure 8: Ablation on design of the pessimism module. 4 tasks, 10 seeds per task.

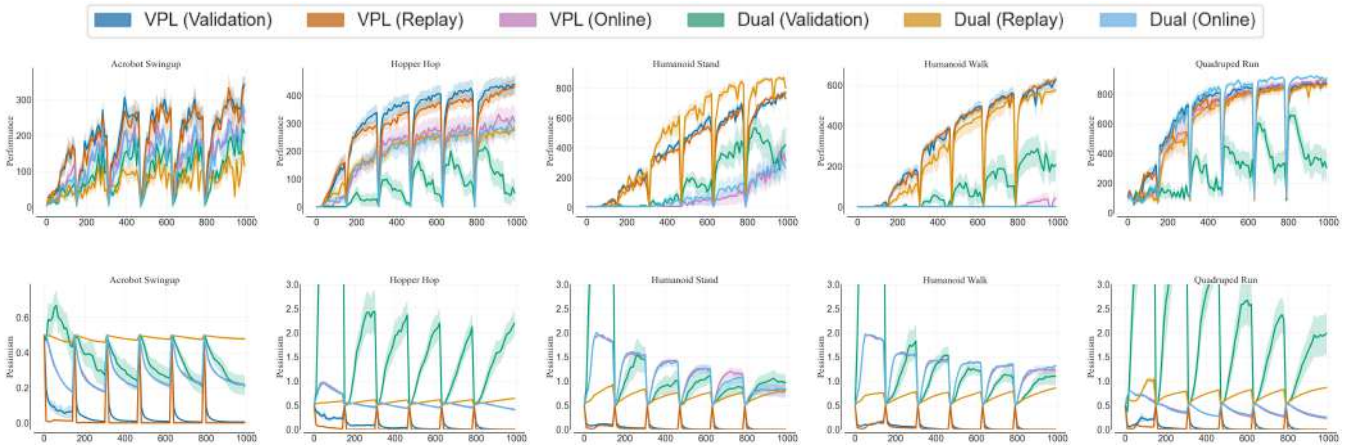


Figure 9: Ablation on design of the pessimism module. 10 seeds per task.

Table 3: 20 DMC and MetaWorld tasks used for the main evaluation.

DeepMind Control	MetaWorld
acrobot-swingup	assembly
fish-swim	box-close
hopper-hop	button-press
hopper-stand	coffee-pull
humanoid-run	coffee-push
humanoid-stand	drawer-open
humanoid-walk	hammer
quadruped-run	push
swimmer-swimmer6	stick-pull
walker-run	sweep

Table 4: 4 DMC tasks used in additional experiments.

DeepMind Control
acrobot-swingup
hopper-hop
humanoid-walk
quadruped-run

Table 5: Hyperparameter values used in the experiments.

Hyperparameter	Notation	Value
Joint		
Network Size	na	$(256, 256)$
Optimizer	na	Adam
Learning Rate	na	$3e-4$
Batch Size	B	256
Discount	γ	0.99
Initial Temperature	α_0	1.0
Initial Steps	na	10000
Target Entropy	$\mathcal{H}^*$	$\lvert\mathcal{A}\rvert/2$
Polyak Weight	τ	0.005
TOP		
Pessimism Values	β	$0, 1$
Bandit Learning Rate	na	0.1
GPL		
Pessimism Learning Rate	na	$5e-5$
Initial Pessimism	β	1.0
OPL		
Pessimism Learning Rate	na	$5e-5$
On-policy Trajectory Length	na	8
TD-λ	na	0.95
Initial Pessimism	β	1.0
VPL		
Pessimism Learning Rate	na	$5e-5$
Validation Proportion	v	$1/32$
Initial Pessimism	β	1.0

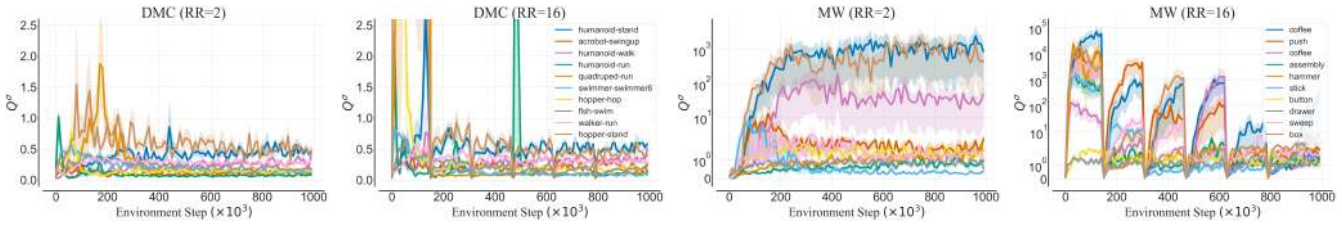


Figure 10: The critic disagreement of SAC algorithm does not completely diminish in the considered training regime of 1 million environment steps, making adjustments to the pessimism a viable strategy. 10 seeds per task.

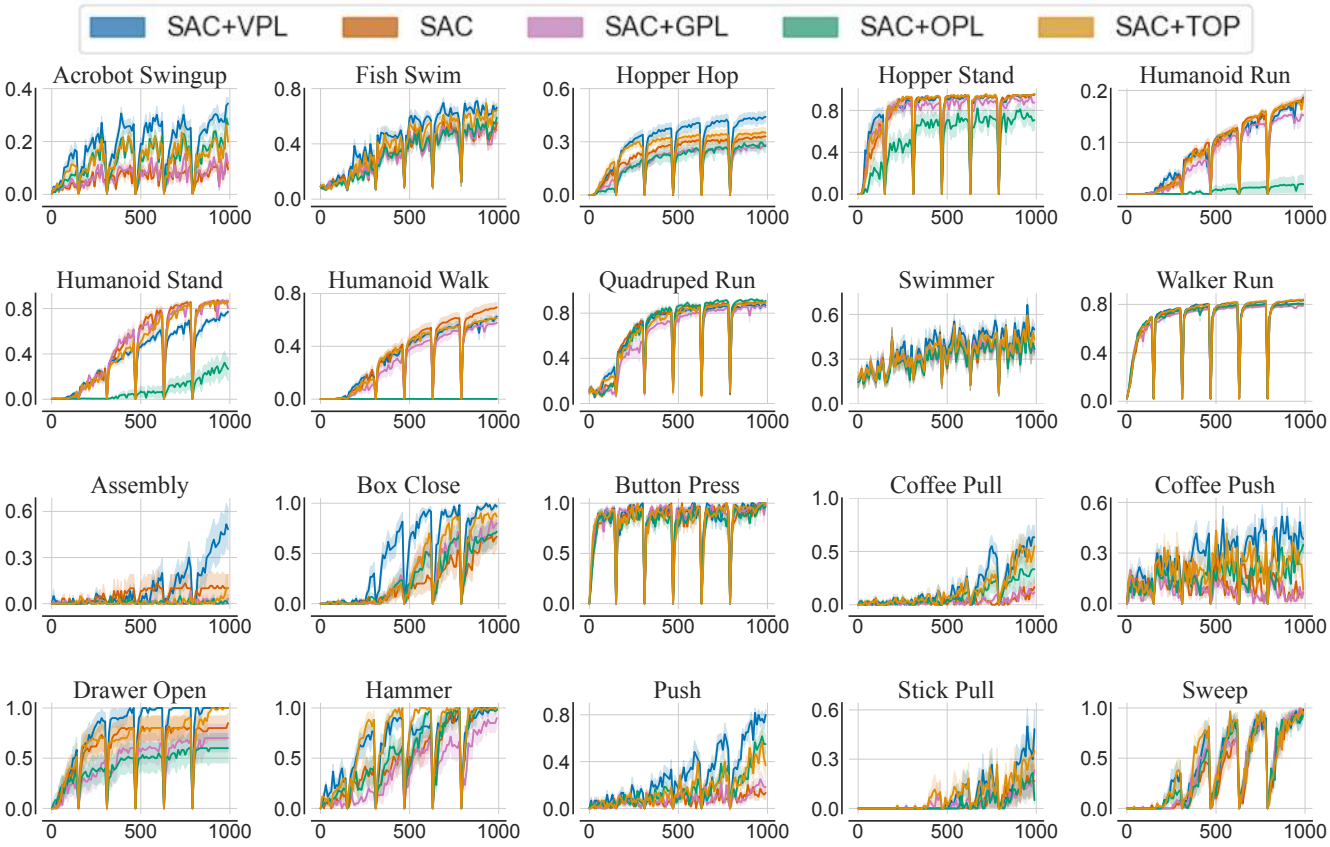


Figure 11: High replay regime results for each considered task. 10 seeds per task, mean and 3 standard deviations.

Bigger, Regularized, Optimistic: scaling for compute and sample-efficient continuous control

Michal Nauman[1,2] **Mateusz Ostaszewski**[3] **Krzysztof Jankowski**[2] **Piotr Miłoś**[1,2,4]

Marek Cygan[2,5]

Abstract

Sample efficiency in Reinforcement Learning (RL) has traditionally been driven by algorithmic enhancements. In this work, we demonstrate that scaling can also lead to substantial improvements. We conduct a thorough investigation into the interplay of scaling model capacity and domain-specific RL enhancements. These empirical findings inform the design choices underlying our proposed BRO (Bigger, Regularized, Optimistic) algorithm. The key insight behind BRO is that strong regularization allows for effective scaling of the critic networks, which, paired with optimistic exploration, leads to superior performance. BRO achieves state-of-the-art results, significantly outperforming the leading model-based and model-free algorithms across 40 complex tasks from the DeepMind Control, MetaWorld, and MyoSuite benchmarks. BRO is the first model-free algorithm to achieve near-optimal policies in the notoriously challenging Dog and Humanoid tasks.

1 Introduction

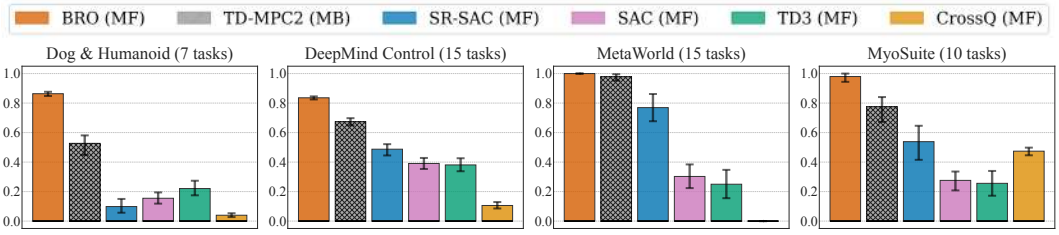


Figure 1: BRO sets new state-of-the-art outperforming model-free (MF) and model-based (MB) algorithms on 40 complex tasks covering 3 benchmark suites. Y-axes report interquartile mean calculated on 10 random seeds, with 1.0 representing the best possible performance in a given benchmark. We use $1M$ environment steps.

Deep learning has seen remarkable advancements in recent years, driven primarily by the development of large neural network models (Devlin et al., 2019; Tan & Le, 2019; Dosovitskiy et al., 2020). These advancements have significantly benefited fields like natural language processing and computer vision and have been percolating to RL as well (Padalkar et al., 2023; Zitkovich et al., 2023). Interestingly, some recent work has shown that the model scaling can be repurposed to achieve sample efficiency in discrete control (Schwarzer et al., 2023; Obando-Ceron et al., 2024), but these approaches cannot be directly translated to continuous action RL. As such, they rely on discrete action representation, whereas many physical control tasks have continuous, real-valued action spaces.

[1]Ideas NCBR; [2]University of Warsaw, [3]Warsaw University of Technology, [4]Polish Academy of Sciences, [5]Nomagic. Correspondence to: Michal Nauman <nauman.mic@gmail.com>

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

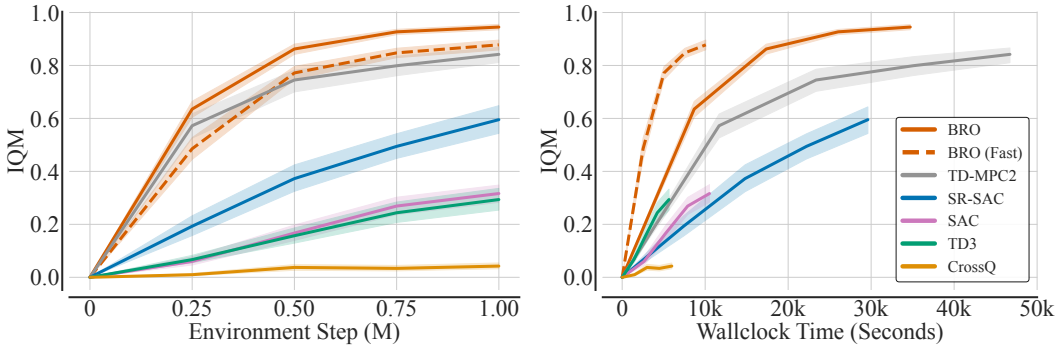


Figure 2: We report sample efficiency (left) and wallclock time (right) for BRO and BRO (Fast) (BRO with reduced replay ratio for increased compute efficiency), as well as baseline algorithms averaged over 40 tasks listed in Table 4. BRO achieves the best sample efficiency, whereas BRO (Fast) matches the sample efficiency of model-based TD-MPC2. In terms of wall clock efficiency, BRO runs approximately 25% faster than TD-MPC2. Remarkably, BRO (Fast) matches the wallclock efficiency of a standard SAC agent while achieving 400% better performance. The Y-axis reports the interquartile mean, with 1.0 representing the maximal possible performance.

Conventional practice in continuous deep RL has relied on small network architectures (Haarnoja et al., 2018; Hiraoka et al., 2021; Raffin et al., 2021; D'Oro et al., 2022), with the primary focus on algorithmic improvements. These enhancements aim to achieve better sample efficiency and address key challenges such as value overestimation (Fujimoto et al., 2018; Moskovitz et al., 2021; Cetin & Celiktutan, 2023), exploration (Chen et al., 2017; Ciosek et al., 2019; Nauman & Cygan, 2023a), and increasing the number of gradient steps (Nikishin et al., 2022; D'Oro et al., 2022). Additionally, evidence suggests that naive model capacity scaling can degrade performance (Andrychowicz et al., 2021; Bjorck et al., 2021). We challenge this status quo by posing a critical question: *Can significant performance improvements in continuous control be achieved by combining parameter and replay scaling with existing algorithmic improvements?*

In this work, we answer this question affirmatively, identifying components essential to successful scaling. Our findings are based on a thorough evaluation of a broad range of design choices, which include batch size (Obando Ceron et al., 2024), distributional Q-values techniques (Bellemare et al., 2017; Dabney et al., 2018), neural network regularizations (Bjorck et al., 2021; Nauman et al., 2024), and optimistic exploration (Moskovitz et al., 2021; Nauman & Cygan, 2023a). Moreover, we carefully investigate the benefits and computational costs stemming from scaling along two axes: the number of parameters and the number of gradient steps. Importantly, we find that the former can lead to more significant performance gains while being more computationally efficient in parallelized setups.

Our work culminates in developing the BRO (Bigger, Regularized, Optimistic) algorithm, a novel sample-efficient model-free approach. BRO significantly outperforms existing model-free and model-based approaches on 40 demanding tasks from the DeepMind Control, MetaWorld, and MyoSuite benchmarks, as illustrated in Figures 1 and 2. Notably, BRO is the first model-free algorithm to achieve near-optimal performance in challenging Dog and Humanoid tasks while being 2.5 times more sample-efficient than the leading model-based algorithm, TD-MPC2. The key BRO innovation is pairing strong regularization with critic model scaling, which, coupled with optimistic exploration, leads to superior performance. We summarize our contributions:

- **Extensive empirical analysis** - we conduct an extensive empirical analysis focusing on critic model scaling in continuous deep RL. By training over $15,000$ agents, we explore the interplay between critic capacity, replay ratio, and a comprehensive list of design choices.
- **BroNet architecture & BRO algorithm** - we introduce the BRO algorithm, a novel model-free approach that combines BroNet architecture for critic scaling with domain-specific RL enhancements. BRO achieves state-of-the-art performance on 40 challenging tasks across diverse domains.
- **Scaling & regularization** - we offer several insights, with the most important being: regularized critic scaling outperforms replay ratio scaling in terms of performance and computational efficiency; the inductive biases introduced by domain-specific RL improvements can be largely substituted by critic scaling, leading to simpler algorithms.

2 Bigger, Regularized, Optimistic (BRO) algorithm

This section presents our novel Big, Regularized, Optimistic (BRO) algorithm and its design principles. The model-free BRO is a conclusion of extensive experimentation presented in Section 3, and significantly outperforms existing state-of-the-art methods on continuous control tasks from proprioceptive states (Figure 1).

2.1 Experimental setup

We compare BRO against a variety of baseline algorithms. Firstly, we consider TD-MPC2 (Hansen et al., 2023), a model-based state-of-the-art that was shown to reliably solve the complex dog domains. Secondly, we consider SR-SAC (D'Oro et al., 2022), a sample-efficient SAC implementation that uses a large replay ratio of 32 and full-parameter resets. For completeness, we also consider CrossQ (Bhatt et al., 2023), a compute-efficient method that was shown to outperform ensemble approaches, as well as standard SAC (Haarnoja et al., 2018) and TD3 (Fujimoto et al., 2018). We run all algorithms with 10 random seeds, except for TD-MPC2, for which we use the results provided by the original manuscript (Hansen et al., 2023). We describe the process of hyperparameter selection for all considered algorithms in Appendix D, and share BRO pseudocode in Appendix (Pseudocode 1). We implement BRO based on the JaxRL (Kostrikov, 2021) and make the code available under the following link: https://github.com/naumix/BiggerRegularizedOptimistic

Environments We consider a wide range of control tasks, encompassing a total of 40 diverse, complex continuous control tasks spanning three simulation domains: DeepMind Control (Tassa et al., 2018), MetaWorld (Yu et al., 2020), and MyoSuite (Caggiano et al., 2022) (a detailed list of environments can be found in Appendix C). These tasks include high-dimensional state and action spaces (with $|S|$ and $|A|$ reaching 223 and 39 dimensions), sparse rewards, complex locomotion tasks, and physiologically accurate musculoskeletal motor control. We run the algorithms for $1M$ environment steps and report the final performance unless explicitly stated otherwise. We calculate the interquartile means and confidence intervals using the RLiable package (Agarwal et al., 2021).

(a) DeepMind Control (DMC) (b) MetaWorld (MW) (c) MyoSuite (MS)

Figure 3: We consider a total of 40 tasks from DeepMind Control (DMC), MetaWorld (MW), and MyoSuite (MS). In particular, we chose the tasks with the biggest optimality gap according to previous evaluations (Hansen et al., 2023). We list all considered tasks in Table 4.

2.2 BRO outline and design choices

The BRO algorithm is based on the well-established Soft Actor-Critic (SAC) (Haarnoja et al., 2018) (see also Appendix A) and is composed of the following key components:

- **Bigger** – BRO uses a scaled critic network with the default of $\approx 5M$ parameters, which is approximately 7 times larger than the average size of SAC models (Haarnoja et al., 2018); as well as scaled training density with a default replay ratio[1] of $RR = 10$, and $RR = 2$ for the BRO (Fast) version.
- **Regularized** – the BroNet architecture, intrinsic to the BRO approach, incorporates strategies for regularization and stability enhancement, including the utilization of Layer Normalization (Ba et al., 2016) after each dense layer, alongside weight decay (Loshchilov & Hutter, 2017) and full-parameter resets (Nikishin et al., 2022).

[1]The replay ratio refers to the number of gradient updates per one environment step.

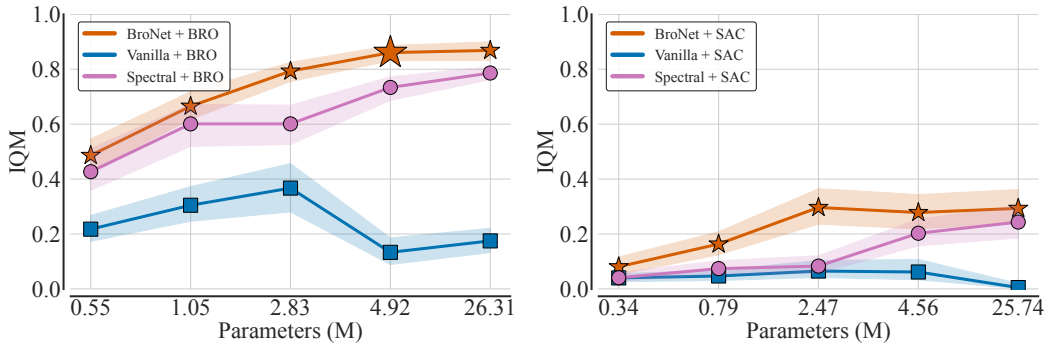


Figure 4: Scaling the critic parameter count for vanilla dense (Fujimoto et al., 2018), spectral normalization ResNet (Bjorck et al., 2021), and our BroNet for BRO (left), and SAC (right). We conclude that to achieve the best performance, we need both the right architecture (BroNet) and the correct algorithmic enhancements encapsulated in BRO. We report interquartile mean performance after $1M$ environment steps in tasks listed in Table 3, with error bars indicating 95% CI from 10 seeds. On the X-axis, we report the approximate parameter count of each configuration.

- **Optimistic** – BRO uses dual policy optimistic exploration (Nauman & Cygan, 2023a) and non-pessimistic (Nauman et al., 2024) quantile Q-value approximation (Dabney et al., 2018; Moskovitz et al., 2021) for balancing exploration and exploitation.

The full details of the algorithm, along with the pseudo-code, are provided in Appendix B. Figure 7 summarizes the impact of removing components of BRO. We observe the biggest impact of scaling the critic capacity (scale) and replay ratio (RR), as well as using non-pessimistic Q-value, i.e. removing Clipped Double Q-learning (CDQ).

Scaling critic network and BroNet architecture The key contribution of this paper is showing how to enable scaling the critic network. We recall that naively increasing the critic capacity does not necessarily lead to performance improvements and that successful scaling depends on a carefully chosen suite of regularization techniques (Bjorck et al., 2021). Figure 5 shows our BroNet architecture, which, up to our knowledge, did not exist previously in the literature. The architecture begins with a dense layer followed by Layer Norm (Ba et al., 2016) and ReLU activation. Subsequently, the network comprises ResNet blocks, each consisting of two dense layers regularized with Layer Norm. Consequently, the ResNet resembles the FFN sub-layer utilized in modern LLM architectures (Xiong et al., 2020), differing primarily in the placement of the Layer Norms. Crucially, we find that BroNet scales more effectively than other architectures (Figure 4 (left)). However, the right choice of architecture and scaling is not a silver bullet. Figure 4 (right) shows that when these are plugged into the standard SAC algorithm naively, the performance is weak. The important elements are additional regularization (weight decay and network resets) and optimistic exploration (see below). Interestingly, we did not find benefits from scaling the actor networks, further discussed in Section 3.

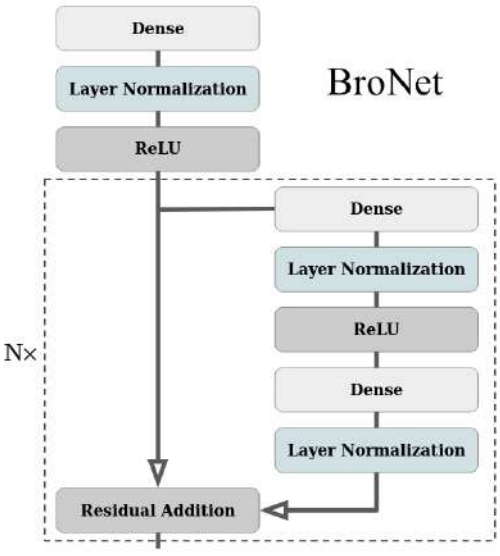


Figure 5: BroNet architecture employed for actor and critic. Each fully connected layer is augmented with Layer Norm, which is essential to unlocking scaling. We use $\approx 5M$ parameters and $N = 2$ in the default setting.

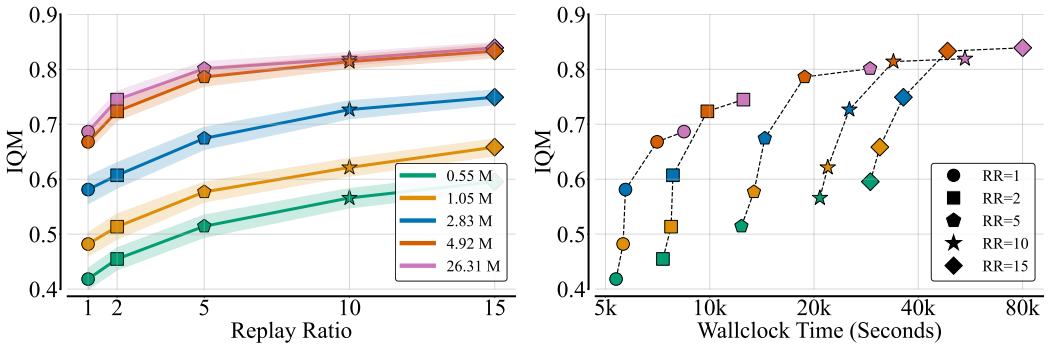


Figure 6: To account for sample efficiency, we report the performance averaged at $250k$, $500k$, $750k$, and $1M$ environment steps across different 5 replay ratios and 5 critic model sizes. All agents were evaluated in tasks listed in Table 3, and 10 random seeds per variant. The left figure shows performance scaling with increasing replay ratios (shapes) and model sizes (colors). The right figure examines the tradeoff between performance and computational cost when scaling replay ratios versus critic model sizes. Increasing model size leads to substantial performance improvements at lower compute costs compared to increasing the replay ratio. We present more scaling results in Appendix E, including a description of model sizes in Table 7.

Scaling replay ratio and relation to model scaling Increasing replay ratio (D'Oro et al., 2022) is another axis of scaling. We investigate mutual interactions by measuring the performance across different model scales (from $0.55M$ to $26M$) and RR settings (from $RR = 1$ to $RR = 15$). Figure 6 reveals that the model scaling has a much stronger impact plateauing at $\approx 5M$ parameters. Increasing the replay ratio also leads to noticeable benefits. For example, a $26M$ model with $RR = 1$ achieves significantly better performance than a small model with $RR = 15$, even though the $26M$ model requires three times less wallclock time. Importantly, model scaling and increasing replay ratio work well in tandem and are interchangeable to some degree. We additionally note that the replay ratio has a bigger impact on wallclock time than the model size. This stems from the fact that scaling replay ratio leads to inherently sequential calculations, whereas scaling model size leads to calculations that can be parallelized. For these reasons, BRO (Fast) with $RR = 2$ and $5M$ network offers an attractive trade-off, being already very sample efficient and fast at the same time.

Optimistic exploration and Q-values BRO utilizes two mechanisms to increase optimism. We observe significant improvements stemming from these techniques in both BRO and BRO (Fast) agents (Figure 7). They are particularly pronounced in the early stages of the training and for smaller models (Figure 9a).

The initial mechanism involves deactivating Clipped Double Q-learning (CDQ) (Fujimoto et al., 2018), a commonly employed technique in reinforcement learning aimed at mitigating Q-value overestimation. For further clarification, refer to Appendix B.6, particularly Eq. 8 where we take the ensemble mean instead of minimum for Q-value calculation. This is surprising, perhaps, as it goes against conventional wisdom. However, some recent work has already suggested that regularization might effectively combat the overestimation (Nauman et al., 2024). We observe a much stronger effect. In Figures 7 & 9a, we compare the performance of BRO with BRO that uses CDQ. This analysis indicates that using risk-neutral Q-value approximation in the presence of network regularization unlocks significant performance improvements without value overestimation (Table 1).

The second mechanism is optimistic exploration. We implement the dual actor setup (Nauman & Cygan, 2023a), which employs separate policies for exploration and temporal difference updates. The exploration policy follows an optimistic upper-bound Q-value approximation, which has been shown to improve the sample efficiency of SAC-based agents (Ciosek et al., 2019; Moskovitz et al., 2021; Nauman & Cygan, 2023a). In particular, we optimize the optimistic actor towards a KL-regularized Q-value upper-bound (Nauman & Cygan, 2023a), with Q-value upper-bound calculated with respect to epistemic uncertainty calculated according to the methodology presented in Moskovitz et al. (2021). As shown in Figure 7, using dual actor optimistic exploration yields around 10% performance improvement in the BRO model.

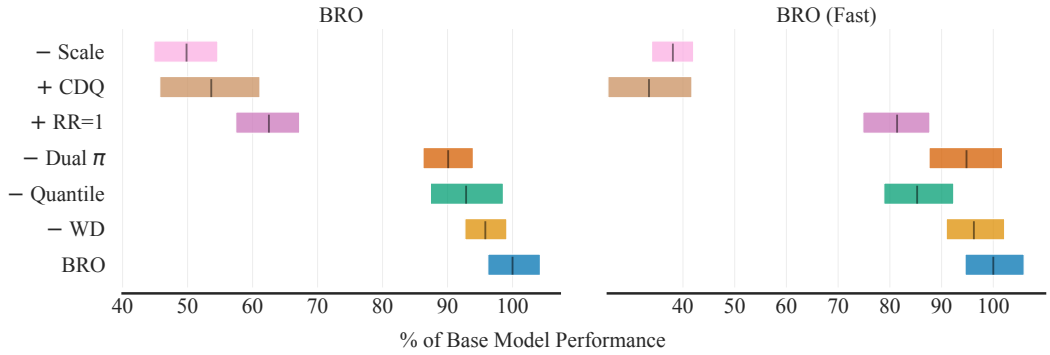


Figure 7: Impact of removing various BRO components on its performance. We report the percentage of the final performance for BRO (left) and BRO (Fast) (right). The y-axis shows the components that are ablated: **-Scale** denotes using a standard-sized network, **+CDQ** denotes using pessimistic Clipped Double Q-learning (which is removed by default in BRO), **+RR=1** uses the standard replay ratio, **-Dual** π removes optimistic exploration, and **-Quantile** and **-WD** stand for removing quantile Q-values and weight decay, respectively. We report the interquartile mean and 95% CIs for tasks in Table 3, with 10 random seeds. The results indicate that the **Scale**, **CDQ**, and **RR=1** components are the most impactful for BRO. Since BRO (Fast) has RR=2 by default, reducing it to one does not significantly affect its performance.

Others We mention two other design choices. First, we use a smaller batch size of 128 than the typical one of 256. This is computationally beneficial while having a marginal impact on performance, which we show in Figure 15. Secondly, we use quantile Q-values (Bellemare et al., 2017; Dabney et al., 2018). We find that quantile critic representation improves performance (Figure 7), particularly for smaller networks. This improvement, however, diminishes for over-parameterized agents (Figure 9a). On top of the performance improvements, the distribution setup allows us to estimate epistemic uncertainties, which we leverage in the optimistic exploration according to the methodology presented in Moskovitz et al. (2021).

3 Analysis

This section summarizes the results of 15,000 experiments, detailed in Table 2, which led us to develop the BRO algorithms. These experiments also provided numerous insights that we believe will be of interest to the community. We adhered to the experimental setup described in Section 2.1. We also present additional experimental results in Appendix E.

Scaling model-free critic allows superior performance We recall that the most important finding is that skillful critic model scaling combined with simple algorithmic improvements can lead to extremely sample-efficient performance and the ability to solve the most challenging environments. We deepen these observations in experiments depicted in Figure 8. Namely, we let the other algorithms, including state-of-the-art model-based TD-MPC2, run for 3M steps on the most challenging tasks in the DMC suite (Dog Stand, Dog Walk, Dog Trot, Dog Run, Humanoid Stand, Humanoid Walk, and Humanoid Run). TD-MPC2 eventually achieves BRO performance levels, but it requires approximately 2.5 more environment steps.

Algorithmic improvements matter less as the scale increases The impact of algorithmic improvements varies with the size of the critic model. As shown in Figure 9a, while techniques like smaller batch sizes, quantile Q-values, and optimistic exploration enhance performance for $1.05M$ and $4.92M$ models, they do not improve performance for the largest $26.3M$ models. We hypothesize this reflects a tradeoff between the inductive bias of domain-specific RL techniques and the overparameterization of large neural networks. Despite this, these techniques still offer performance gains with lower computing costs. Notably, full-parameter resets (Nikishin et al., 2022; D'Oro et al., 2022) are beneficial; the largest model without resets nearly matches the performance of the BRO with resets.

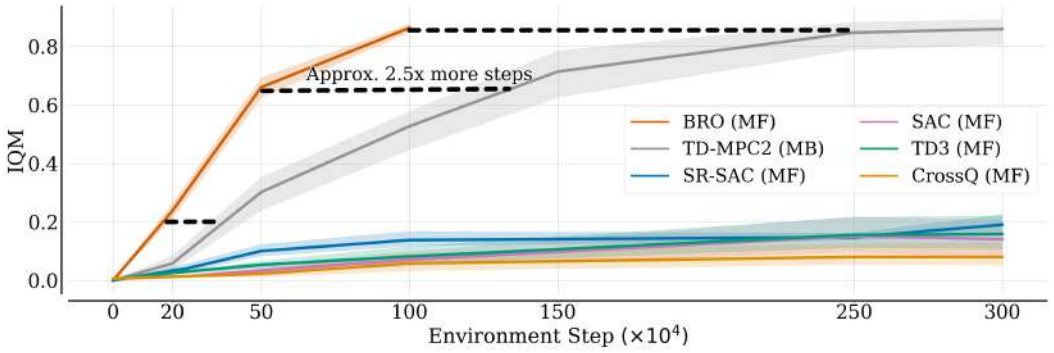


Figure 8: IQM return learning curves for four Dog and three Humanoid environments from the DMC benchmark, plotted against the number of environment steps. Notably, the model-based approach (TD-MPC2) requires approximately 2.5 times more steps to match BRO performance.

Scaling actor is not effective Previous works underscore the relative importance of critic and actor networks in off-policy algorithms like SAC (Fujimoto et al., 2018; D'Oro et al., 2022; Li et al., 2022). For instance, Nikishin et al. (2022) found that critic regularization is significantly more important than actor regularization. We confirm this result by showing that, for off-policy continuous control actor-critic algorithms, increasing critic capacity leads to much better results than increasing the actor model size, which in some cases might be even detrimental (Figure 9a). As such, practitioners can achieve performance improvements by prioritizing critic capacity over actor capacity while adhering to memory limitations.

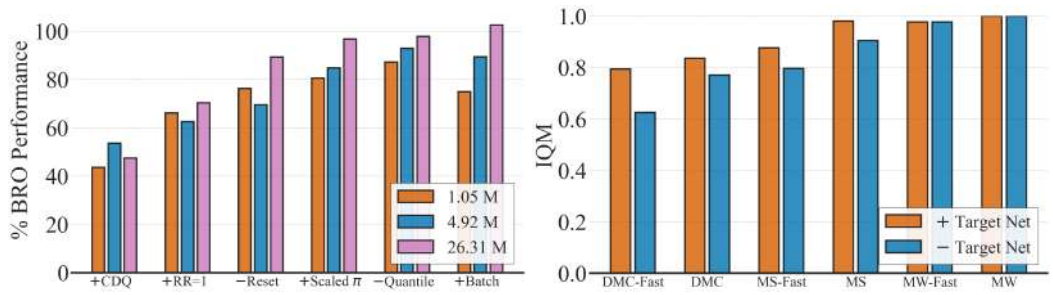
Target networks yield small but noticeable performance benefits Using target networks doubles the memory costs (Schwarzer et al., 2020; Bhatt et al., 2023; Lee et al., 2024), which can be a significant burden for large models. In Figure 9b, we compare the performance of standard BRO and BRO (Fast) agents against their versions without target networks. Consistent with established understanding, we find that using target networks yields a small but significant performance improvement. However, we observe substantial variation in these effects among benchmarks and specific environments (Figure 9b & Figure 16). For example, the majority of performance improvements in DMC and MS environments are attributable to specific tasks.

Architecture matters (especially in complex environments) By breaking down the results from Figure 4 into individual environments, the BroNet architecture achieves better performance in all of them, but the differences are most pronounced in the Dog environments. Therefore, we deepened our analysis with extra metrics to understand these discrepancies better. Table 1 demonstrates that BroNet outperforms the other architectures regarding final performance. The Vanilla MLP exhibits instabilities across all measured metrics, including gradient norm, overestimation, and TD error. While using the Spectral architecture maintains moderate gradient norms and overestimation, it struggles significantly with minimizing the TD error.

Table 1: Comparison of BroNet, Spectral (Bjorck et al., 2021), and Vanilla MLP architectures in notriously hard Dog environments. All metrics except return are averaged over time steps. All architectures are combined with BRO.

	BroNet	Spectral	Vanilla
Final return	763.5	73.5	167.
$\|\|\nabla\|\|_2$	35.5	88.	9.61E+04
Mean Q-values	58.06	153.85	1.20E+05
TD-error	0.38	4.31E+04	6.03E+07

In (Nauman et al., 2024), the authors indicate that the gradient norm and overestimation are strong indicators of poor performance in Dog environments. However, these results suggest that identifying a single cause for the challenges in training a reinforcement learning agent is difficult, highlighting the complexity of these systems and the multifaceted nature of their performance issues.



(a) Design Choices (b) Target Networks

Figure 9: (Left) We analyze the importance of BRO components dependent on the critic model size. Interestingly, most components become less important as the critic capacity grows. (Right) We report the performance of BRO variants with and without a target network. All algorithm variants are run with 10 random seeds.

What did not work? While researching BRO, we tested a variety of techniques that were found to improve the performance of different RL agents; however, they did not work in our evaluations. Firstly, we found that using N-step returns (Sutton & Barto, 2018; Schwarzer et al., 2023) does not improve the performance in the tested environments. We conjecture that the difference between N-step effectiveness in Atari and continuous control benchmarks stems from the sparser reward density in the former. Furthermore, we evaluated categorical RL (Bellemare et al., 2017) and HLGauss (Imani & White, 2018; Farebrother et al., 2024) Q-value representations, but found that these techniques are not directly transferable to a deterministic policy gradient setup and introduce training instabilities when applied naively, resulting in a significant amount of seeds not finishing their training. Finally, we tested a variety of scheduling mechanisms considered by Schwarzer et al. (2023) but found that the performance benefits are marginal and highly task-dependent while introducing much more complexity associated with hyperparameter tuning. A complete list of tested techniques is presented in Appendix B.8.

Are current benchmarks enough? As illustrated in Figure 10, even complex tasks like Dog Walk or Dog Trot can be reliably solved by combining existing algorithmic improvements with critic model scaling within 1 million environment steps. However, some tasks remain unsolved within this limit (e.g., Humanoid Run or Acrobot Swingup). Tailoring algorithms to single tasks risks overfitting to specific issues. Therefore, we advocate for standardized benchmarks that reflect the sample efficiency of modern algorithms. This standardization would facilitate consistent comparison of approaches, accelerate advancements by focusing on a common set of challenging tasks, and promote the development of more robust and generalizable RL algorithms. On that note, in Appendix F, we report BRO performance at earlier stages of the training.

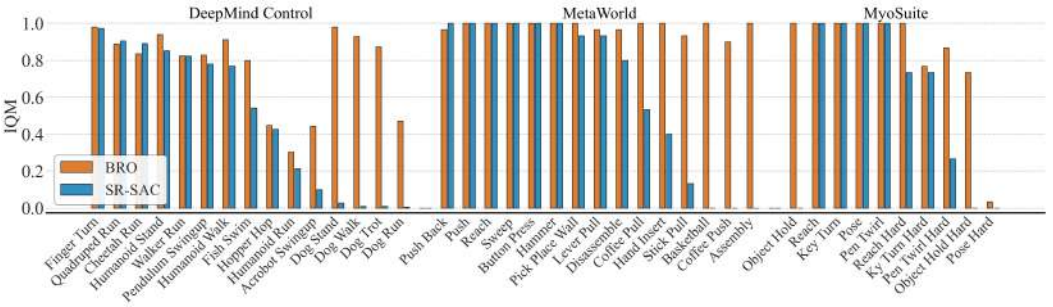


Figure 10: Our experiments cover 40 of the hardest tasks from DMC (locomotion), MW (manipulation), and MS (physiologically accurate musculoskeletal control) considered in previous work (Hansen et al., 2023). In those tasks, the state-of-the-art model-free SR-SAC (D'Oro et al., 2022) achieves more than 80% of maximal performance in 18 out of 40 tasks, whereas our proposed BRO in 33 out of 40 tasks. BRO makes significant progress in the most complex tasks of the benchmarks.

BroNet architecture is useful beyond continuous control We design additional experiments to test the effectiveness of the naive application of BroNet to popular offline reinforcement learning problems in two offline RL benchmarks: AntMaze (6 tasks); and Adroit (9 tasks) (Fu et al., 2020). We run Behavioral Cloning (BC) in pure offline (Sutton & Barto, 2018), Implicit Q-Learning (IQL) offline + fine-tuning (Kostrikov et al., 2022), as well as online reinforcement learning with offline data (Ball et al., 2023). We run all these algorithms with the default MLP network, as well as BroNet backbone. As shown in Figure 11, we find that the naive application of BroNet leads to performance improvements across all tested algorithms.

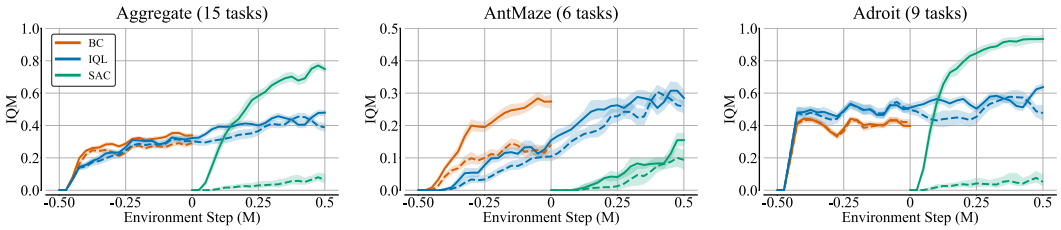


Figure 11: We test three scenarios: offline (comparing vanilla BC to BroNet-based BC), offline fine-tuning (comparing vanilla IQL to BroNet-based IQL), and online with offline data (comparing vanilla SAC to BroNet-based SAC). The solid line represents BRO-based and the dashed line represents vanilla variants. Negative values on the X-axis refer to offline training. 10 seeds per task.

4 Related Work

4.1 Sample efficiency through algorithmic improvements

A significant effort in RL has focused on algorithmic improvements. One recurring theme is controlling value overestimation (Fujimoto et al., 2018; Moskovitz et al., 2021; Cetin & Celiktutan, 2023). For instance, Fujimoto et al. (2018) proposed Clipped Double Q-learning (CDQ), which updates policy and value networks using a lower-bound Q-value approximation. However, since a pessimistic lower-bound can slow down learning, Moskovitz et al. (2021) introduced an approach that tunes pessimism online. Recently, Nauman et al. (2024) showed that layer normalization can improve performance without value overestimation, eliminating the need for pessimistic Q-learning.

A notable effort has also focused on optimistic exploration (Wang et al., 2020; Moskovitz et al., 2021). Various methods have been developed to increase sample efficiency via exploration that is greedy with respect to a Q-value upper bound. These include closed-form transformations of the pessimistic policy (Ciosek et al., 2019) or using a dual actor network dedicated to exploration (Nauman & Cygan, 2023a).

4.2 Sample efficiency through scaling

Recent studies demonstrated the benefits of model scaling when pre-training on large datasets (Driess et al., 2023; Schubert et al., 2023; Taiga et al., 2023) or in pure offline RL setups (Kumar et al., 2023). Additionally, model scaling has proven advantageous for model-based online RL (Hafner et al., 2023; Hansen et al., 2023; Wang et al., 2024). However, in these approaches, most of the model scale is dedicated to world models, leaving the value network small. Notably, Schwarzer et al. (2023) found that increasing the scale of the encoder network improves performance for DQN agents, but did not study increasing the capacity of the value network. Various studies indicate that naive scaling of the value model leads to performance deterioration (Bjorck et al., 2021; Obando-Ceron et al., 2024; Farebrother et al., 2024). For example, Bjorck et al. (2021) demonstrated that spectral normalization enables stable training with relatively large ResNets with performance improvements.

In addition to model size scaling, the community has investigated the effectiveness of replay ratio scaling (i.e., increasing the number of gradient steps for every environment step) (Hiraoka et al., 2021; Nikishin et al., 2022; Li et al., 2022). Recent works have shown that a high replay ratio can improve performance across various algorithms in both continuous and discrete MDPs, provided the neural networks are regularized (Li et al., 2022; D'Oro et al., 2022). In this context, layer normalization

and full-parameter resets have been particularly effective (Schwarzer et al., 2023; Lyle et al., 2024; Nauman et al., 2024).

5 Limitations and Future Work

BRO's larger model size compared to traditional baselines like SAC or TD3 results in higher memory requirements, potentially posing challenges for real-time inference in high-frequency control tasks. Future research could explore techniques such as quantization or distillation to improve inference speed. While BRO is designed for continuous control problems, its effectiveness in discrete settings remains unexplored. Further investigation is needed to assess the applicability and performance of BRO's components in discrete action MDPs. Additionally, our experimentation primarily focuses on continuous control tasks using proprioceptive state representations. Future research is needed to investigate the tradeoff between scaling the critic and the state encoder in image-based RL.

6 Conclusions

Our study underscores the efficacy of scaling a regularized critic model in conjunction with existing algorithmic enhancements, resulting in sample-efficient methods for continuous-action RL. The proposed BRO algorithm achieves markedly superior performance within 1 million environment steps compared to the state-of-the-art model-based TD-MPC2 and other model-free baselines. Notably, it achieves over 90% success rates in MetaWorld and MyoSuite benchmarks, as well as over 85% of maximal returns in the DeepMind Control Suite, and near-optimal policies in the challenging Dog and Humanoid locomotion tasks. While some tasks remain unsolved within 1 million environment steps, our findings underscore the need for new standardized benchmarks focusing on sample efficiency to drive consistent progress in the field. The BRO algorithm establishes a new standard for sample efficiency, providing a solid foundation for future research to build upon and develop even more robust RL algorithms.

Acknowledgements

This research was partially supported by the National Science Centre, Poland, under grant nos. 2020/39/B/ST6/01511 and 2023/51/D/ST6/01609, as well as by the Warsaw University of Technology and the University of Warsaw through the Excellence Initiative: Research University (IDUB) program. We also gratefully acknowledge the Polish high-performance computing infrastructure, PLGrid (HPC Center: ACK Cyfronet AGH), for providing computational resources and support under grant no. PLG/2024/017159. Marek Cygan was partially supported by an NCBiR grant POIR.01.01.01-00-0433/20. Piotr Miłoś research was supported by the National Science Center (Poland) grant number 2019/35/O/ST6/03464.

References

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.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. What matters in on-policy reinforcement learning? a large-scale empirical study. In *ICLR 2021-Ninth International Conference on Learning Representations*, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Ball, P. J., Smith, L., Kostrikov, I., and Levine, S. Efficient online reinforcement learning with offline data. *Proceedings of the 40th International Conference on Machine Learning*, 2023.

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.

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.

Bhatt, A., Palenicek, D., Belousov, B., Argus, M., Amiranashvili, A., Brox, T., and Peters, J. Cross q: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Representations*, 2023.

Bjorck, N., Gomes, C. P., and Weinberger, K. Q. Towards deeper deep reinforcement learning with spectral normalization. *Advances in neural information processing systems*, 34:8242–8255, 2021.

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

Caggiano, V., Wang, H., Durandau, G., Sartori, M., and Kumar, V. Myosuite–a contact-rich simulation suite for musculoskeletal motor control. *arXiv preprint arXiv:2205.13600*, 2022.

Cetin, E. and Celiktutan, O. Learning pessimism for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6971–6979, 2023.

Chen, R. Y., Sidor, S., Abbeel, P., and Schulman, J. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

Ciosek, K. and Whiteson, S. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21(2020), 2020.

Ciosek, K., Vuong, Q., Loftin, R., and Hofmann, K. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32, 2019.

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.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *North American Chapter of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/N19-1423.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2020.

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., and Florence, P. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv: 2303.03378*, 2023.

Farebrother, J., Orbay, J., Vuong, Q., Taïga, A. A., Chebotar, Y., Xiao, T., Irpan, A., Levine, S., Castro, P. S., Faust, A., et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

François-Lavet, V., Fonteneau, R., and Ernst, D. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv: 1512.02011*, 2015.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hansen, N., Su, H., and Wang, X. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv: 2310.16828*, 2023.

Hiraoka, T., Imagawa, T., Hashimoto, T., Onishi, T., and Tsuruoka, Y. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2021.

Hussing, M., Voelcker, C., Gilitschenski, I., Farahmand, A.-m., and Eaton, E. Dissecting deep rl with high update ratios: Combatting value overestimation and divergence. *arXiv preprint arXiv:2403.05996*, 2024.

Imani, E. and White, M. Improving regression performance with distributional losses. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2157–2166. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/imani18a.html.

Kearns, M. and Singh, S. Bias-variance error bounds for temporal difference updates. In *Annual Conference Computational Learning Theory*, 2000. URL https://api.semanticscholar.org/CorpusID:5053575.

Kostrikov, I. JAXRL: Implementations of Reinforcement Learning algorithms in JAX, 10 2021. URL https://github.com/ikostrikov/jaxrl.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. Offline q-learning on diverse multi-task data both scales and generalizes. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=4-k7kUavAj.

Lee, H., Cho, H., Kim, H., Kim, D., Min, D., Choo, J., and Lyle, C. Slow and steady wins the race: Maintaining plasticity with hare and tortoise networks. In *Forty-first International Conference on Machine Learning*, 2024.

Li, Q., Kumar, A., Kostrikov, I., and Levine, S. Efficient deep reinforcement learning requires regulating overfitting. In *The Eleventh International Conference on Learning Representations*, 2022.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2017.

Lyle, C., Zheng, Z., Khetarpal, K., van Hasselt, H., Pascanu, R., Martens, J., and Dabney, W. Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv: 2402.18762*, 2024.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations*, 2018.

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.

Moskovitz, T., Parker-Holder, J., Pacchiano, A., Arbel, M., and Jordan, M. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12849–12863, 2021.

Nauman, M. and Cygan, M. On the theory of risk-aware agents: Bridging actor-critic and economics. In *ICML 2024 Workshop: Aligning Reinforcement Learning Experimentalists and Theorists*, 2023a.

Nauman, M. and Cygan, M. Decoupled actor-critic. *arXiv preprint arXiv:2310.19527*, 2023b. URL https://arxiv.org/pdf/2310.19527v3.

Nauman, M., Bortkiewicz, M., Miłoś, P., Trzcinski, T., Ostaszewski, M., and Cygan, M. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/pdf/2403.00514. PMLR 235:37342-37364.

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.

Obando Ceron, J., Bellemare, M., and Castro, P. S. Small batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Obando-Ceron, J., Sokar, G., Willi, T., Lyle, C., Farebrother, J., Foerster, J., Dziugaite, G. K., Precup, D., and Castro, P. S. Mixtures of experts unlock parameter scaling for deep rl. *arXiv preprint arXiv:2402.08609*, 2024.

Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

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

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Schubert, I., Zhang, J., Bruce, J., Bechtle, S., Parisotto, E., Riedmiller, M., Springenberg, J. T., Byravan, A., Hasenclever, L., and Heess, N. A generalist dynamics model for control. *arXiv preprint arXiv: 2305.10912*, 2023.

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*, 2020.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

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

Tai, J. J., Towers, M., and Tower, E. Shimmy: Gymnasium and PettingZoo Wrappers for Commonly Used Environments. URL https://github.com/Farama-Foundation/shimmy.

Taiga, A. A., Agarwal, R., Farebrother, J., Courville, A., and Bellemare, M. G. Investigating multi-task pretraining and generalization in reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sSt9fROSZRO.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, 2019.

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.

Van Seijen, H., Van Hasselt, H., Whiteson, S., and Wiering, M. A theoretical and empirical analysis of expected sarsa. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, pp. 177–184. IEEE, 2009.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, S., Liu, S., Ye, W., You, J., and Gao, Y. Efficientzero v2: Mastering discrete and continuous control with limited data. *arXiv preprint arXiv:2403.00564*, 2024.

Wang, Y., Wang, R., Du, S. S., and Krishnamurthy, A. Optimism in reinforcement learning with generalized linear function approximation. In *International Conference on Learning Representations*, 2020.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. On layer normalization in the transformer architecture. *International Conference on Machine Learning*, 2020.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

Broader Impact

The work presented in this study, while academic and based on simulated benchmarks, advances the development of more capable autonomous agents. Although our contributions do not directly cause any negative societal impacts, we encourage the community to remain mindful of such potential consequences when extending our research.

A Background

We consider an infinite-horizon Markov Decision Process (MDP) (Puterman, 2014) which is described with a tuple (S, A, r, p, γ), where states S and actions A are continuous, $r(s', s, a)$ is the transition reward, $p(s'|s, a)$ is the transition kernel and $\gamma \in (0, 1]$ is the discount factor.

The policy $\pi(a|s)$ is a state-conditioned action distribution with its entropy denoted as $\mathcal{H}(\pi(s))$. Soft Value (Haarnoja et al., 2018) is the sum of expected discounted return and state entropies from following the policy at a given state

$$V^\pi(s) = \mathrm{E}_{a\sim\pi, s'\sim p}\left[r(s', s, a) + \alpha\mathcal{H}(\pi(s)) + \gamma V^\pi(s')\right], \tag{1}$$

with α denoting the entropy temperature parameter. Q-value is the expected discounted return from performing an action and following the policy thereafter.

$$Q^\pi(s, a) = \mathrm{E}_{s'\sim p}\left[r(s', s, a) + \gamma V^\pi(s')\right] \tag{2}$$

A policy is said to be optimal if it maximizes the expected value of the possible starting states s_0, such that $\dot{\pi} = \arg\max_{\pi\in\Pi} \mathrm{E}_{s_0\sim p} V^\pi(s_0)$, with $\dot{\pi}$ denoting the optimal policy and Π denoting the considered set of policies (e.g., Gaussian). Soft values and soft Q-values are related through the following equation:

$$V^\pi(s) = \mathrm{E}_{a\sim\pi}\left[Q^\pi(s, a) - \log\pi(a|s)\right] \tag{3}$$

This relation is often approximated via a single action sampled according to the policy $a \sim \pi(s)$. In off-policy actor-critic, there is continuous gradient-based learning of both Q-values (critic) and the policy (actor). The critic parameters θ are updated by minimizing SARSA temporal-difference on transitions $T = (s, a, r, s')$, with T being sampled from a replay buffer of transitions (Fujimoto et al., 2018; Haarnoja et al., 2018) according to:

$$\theta = \arg\min_\theta \mathrm{E}_{T\sim\mathcal{D}}\left(Q_\theta(s, a) - r(s', s, a) - \gamma V_{\bar{\theta}}(s)\right), \tag{4}$$

$$V_{\bar{\theta}}(s) = Q_{\bar{\theta}}(s', a') - \alpha\log\pi_\phi(a'|s')), \tag{5}$$

with $a' \sim \pi_\phi$. In this setup, Q_θ is the critic network, $Q_{\bar{\theta}}$ is the value target network, and $\mathcal{D}$ is the replay buffer (Mnih et al., 2015). Q_θ is trained to approximate the Q-value under the policy from which the bootstrap is sampled (Van Seijen et al., 2009; Sutton & Barto, 2018). The policy parameters ϕ are updated to seek locally optimal values approximated by the critic (Ciosek & Whiteson, 2020) according to:

$$\phi = \arg\max_\phi \mathrm{E}_{s\sim\mathcal{D}}\left(Q_\theta(s, a) - \alpha\log\pi_\phi(a|s)\right), \quad \text{with} \quad a \sim \pi_\phi. \tag{6}$$

B BRO additional details

B.1 Base agent

BRO uses the well-established Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as its base. SAC is a stochastic policy, maximum entropy algorithm (see Eq. 1) that employs online entropy temperature adjustment and an ensemble of two critic networks. SAC models the policy via a Tanh-transformed Gaussian distribution whose parameters are modeled by the actor network.

B.2 Architecture details

In the proposed architecture, we adopt the transformer feedforward blueprint from Vaswani et al. (2017) with a novel layer normalization configuration, as shown in Figure 5. Dropout is omitted. All Dense layers in the BRO have a default width of 512 units, with a linear layer at both the input and output stages. To increase the model's depth, we add new residual blocks exclusively. While a similar transformer backbone has been used in previous work (Bjorck et al., 2021), our design choices, detailed in Section E, led us to use layer normalization instead of spectral normalization.

B.3 Scaling

In Figure 4, we examine the scaling capabilities of SAC and BRO agents using a vanilla dense network (Fujimoto et al., 2018; Haarnoja et al., 2018; Moskovitz et al., 2021), a spectral normalization ResNet (Bjorck et al., 2021; Cetin & Celiktutan, 2023), and a layer normalization ResNet inspired by previous work (Nauman et al., 2024). As shown in Figure 4, increasing the model capacity of a vanilla-dense agent can lead to performance degradation beyond a certain model size. However, for the regularized architectures, we observe behavior similar to the empirically observed scaling properties of supervised models, where increasing model size leads to diminishing returns in performance improvements. Furthermore, we find that the layer normalization ResNet achieves better scaling properties than the spectral normalization architecture. Interestingly, the BRO agent consistently outperforms the baseline SAC across all architectures and network sizes, suggesting an interaction between parameter scaling and other algorithmic design choices. The highest performing SAC agent achieves around 25% of maximal performance, whereas our proposed BRO agent achieves more than 90%. Given that the BRO agent performs similarly at 4.92 million and 26.31 million parameters, we use the smaller model to reduce the computational burden.

B.4 Scaling replay ratio

Replay Ratio (RR) scaling in small models leads to diminishing performance increases as RR values rise, eventually plateauing at higher RRs (Nikishin et al., 2022; D'Oro et al., 2022). Unfortunately, increasing RR also results in linear increases in computing costs, as each gradient step must be calculated sequentially. This naturally becomes a burden as the model sizes increase. In Figure 6, we investigate the performance of the BRO agent across different model scales (from 0.55 million to 26.31 million parameters) and RR settings (from RR=1 to RR=15), measuring both performance and wall-clock efficiency. We find that with the BRO regularized critic architecture, critic scaling leads to performance and sample efficiency gains that match those of scaled RR. Scaling both RR and model size produces the best-performing agents. Interestingly, scaling the model size can lead to significant performance improvements even if RR is low while being more computationally efficient due to parallelized workloads (Figure 6). For example, a 5 million parameter BRO model with RR=1 outperforms a 1 million parameter BRO agent with RR=15 despite being five times faster in terms of wall-clock time. This observation challenges the notion that a sample-efficient RL algorithm must use high replay settings.

B.5 Batch Size

Inspired by recent findings that reducing the batch size can result in significant performance gains for discrete action RL (Obando Ceron et al., 2024), we reduce the number of transitions used for gradient calculation from 256 to 128. As shown in Figures 9a & 15, this batch size reduction leads to a marginal improvement in aggregate performance and decreased memory requirements of the algorithm. Interestingly, we find that batch size significantly affects performance, with no single value performing best across all tasks.

B.6 Risk-Neutral Temporal Difference

Using a pessimistic lower-bound Q-value approximation for actor-critic updates, known as Clipped Double Q-learning (CDQ) (Fujimoto et al., 2018; Haarnoja et al., 2018), is a popular method to counteract Q-value overestimation, though it introduces bias. Formally, it modifies the value estimate in Eq. 5 to a lower-bound estimation

$$V_\theta^{lb}(s) \approx Q_\theta^{lb}(s,a) - \alpha \log \pi_\phi(a|s), \quad a \sim \pi_\phi(s), \tag{7}$$

where $Q^{lb}\theta(s,a)$ is a lower-bound Q-value estimation derived from a critic ensemble, often using two networks (Fujimoto et al., 2018; Haarnoja et al., 2018)

$$Q_\theta^{lb}(s,a) = \min(Q_\theta^1(s,a), Q_\theta^2(s,a)). \tag{8}$$

Recent studies have shown that techniques like layer normalization or full-parameter resets can be more effective at combating overestimation than pessimistic Q-value approximation (Nauman et al., 2024). Since our critic architecture leverages multiple regularization techniques, we disable CDQ and use the ensemble mean instead of the minimum to calculate the targets for actor and critic

updates. In Figures 7 & 9a, we compare the performance of the baseline BRO to a BRO agent that uses CDQ. Our findings indicate that using risk-neutral Q-value approximation in the presence of network regularization unlocks significant performance improvements without increasing value overestimation.

B.7 Optimistic Exploration

Optimism is an algorithmic design principle that balances exploration and exploitation (Ciosek et al., 2019; Moskovitz et al., 2021). The dual actor setup (Nauman & Cygan, 2023a,b) employs separate policies for exploration and temporal difference updates, with the exploration policy pursuing an optimistic upper-bound Q-value approximation. This approach has been shown to improve the sample efficiency of SAC-based agents (Nauman & Cygan, 2023a). We implement the optimistic policy such that the Q-value upper bound is calculated based on the epistemic uncertainty estimated via the quantile critic ensemble (Moskovitz et al., 2021). Figure 7 shows that using a dual policy setup leads to performance improvements. We observe that these results are particularly pronounced in the early training stages and for smaller networks (Figure 9a).

Algorithm 2 BRO training step with changes with respect to standard SAC colored red.

Input: π^p_ϕ - pessimistic actor; π^o_η - optimistic actor; $Q^k_{\theta,i}$ - kth quantile of ith critic; $Q^k_{\bar{\theta},i}$ - kth quantile of ith target critic; α - temperature; β^o - optimism; τ - KL weight;
Hyperparameters: $\mathcal{KL}^*$ - target KL; K - number of quantiles

Sample action from the optimistic actor
$s', r = \text{ENV.STEP}(a) \quad a \sim \pi^o_\eta$
Add transition to the replay buffer
$\text{BUFFER.ADD}(s, a, r, s')$
for $i = 1$ **to** ReplayRatio **do**
  Sample batch of transitions
  $s, a, r, s' \sim \text{BUFFER.SAMPLE}$
  Calculate critic target value without CDQ
  $Q^\mu_{\bar{\theta}}(s', a') = \frac{1}{2}(Q^k_{\bar{\theta},1}(s', a') + Q^k_{\bar{\theta},2}(s', a')) \quad \text{with} \quad a' \sim \pi^p_\phi(s')$
  Update critic using pessimistic actor
  $\theta \leftarrow \theta - \nabla_\theta Huber\big(Q_\theta(s, a), (r + \gamma Q^\mu_{\bar{\theta}}(s', a') - \alpha \log \pi^p_\phi(a'|s')\big)$
  Calculate pessimistic actor value without CDQ
  $Q^\mu_\theta(s, a, \pi^p_\phi) = \frac{1}{2K}\sum_{i=1}^{K}(Q^k_{\theta,1}(s, a) + Q^k_{\theta,2}(s, a)) \quad \text{with} \quad a \sim \pi^p_\phi(s)$
  Update pessimistic actor
  $\phi \leftarrow \phi + \nabla_\phi\big(Q^\mu_\theta(s, a, \pi^p_\phi) - \alpha \log \pi^p_\phi(a|s)\big)$
  Calculate optimistic actor value
  $Q^\mu_\theta(s, a, \pi^o_\eta) = \frac{1}{2K}\sum_{i=1}^{K}(Q^k_{\theta,1}(s, a) + Q^k_{\theta,2}(s, a) + \beta^o|Q^k_{\theta,1}(s, a) - Q^k_{\theta,2}(s, a)|), \quad a \sim \pi^o_\eta(s)$
  Update optimistic actor
  $\eta \leftarrow \eta + \nabla_\eta\big(Q^\mu_\theta(s, a, \pi^o_\eta) + \beta^o Q^\sigma_\theta(s, a) - \tau KL\big(\pi^p_\phi(s)|\pi^o_\eta(s)\big)\big), \quad a \sim \pi^o_\eta(s)$
  Update entropy temperature
  $\alpha \leftarrow \alpha - \nabla_\alpha \alpha\big(\mathcal{H}^* - \mathcal{H}(s)\big)$
  Update optimism
  $\beta^o \leftarrow \beta^o - \nabla_{\beta^o}(\beta^o - \beta^p)(\frac{1}{|A|}KL(\pi^p_\phi|\pi^o_\eta) - \mathcal{KL}^*)$
  Update KL weight
  $\tau \leftarrow \tau + \nabla_\tau \tau(\frac{1}{|A|}KL(\pi^p_\phi|\pi^o_\eta) - \mathcal{KL}^*)$
  Update target network
  $\bar{\theta} \leftarrow \text{POLYAK}(\theta, \bar{\theta})$
end for

B.8 Approaches Examined During the Development of BRO

Examined approaches are listed in Table 2. Methods incorporated into BRO include regularization techniques (LayerNorm, Weight Decay, removing CDQL), optimistic exploration, quantile distributional RL, resets and increased replay ratio.

Table 2: Approaches examined during BRO development. Methods incorporated into BRO are highlighted in bold.

Methods Group	Specific Method	Source
Exploration	**DAC**	(Nauman & Cygan, 2023a)
Value Regularization	CDQL **(removed)**	(Fujimoto et al., 2018)
	N-Step Returns	(Sutton & Barto, 2018)
Network Regularization	**LayerNorm**	(Ba et al., 2016)
	Weight Decay	(Loshchilov & Hutter, 2017)
	Spectral Norm	(Miyato et al., 2018)
Scheduling	N-Step Schedule	(Kearns & Singh, 2000)
	Discount Schedule	(François-Lavet et al., 2015)
	Pessimism Schedule	
	Entropy Schedule	
	Learning Rate Schedule	(Andrychowicz et al., 2021)
Distributional RL	HL Gauss	(Imani & White, 2018)
	Categorical	(Bellemare et al., 2017)
	Quantile	(Dabney et al., 2018)
Plasticity Regularization	**Resets**	(Nikishin et al., 2022; D'Oro et al., 2022)
Learning	**Replay Ratio**	(Nikishin et al., 2022; D'Oro et al., 2022)

C Tested Environments

We tested BRO on a variety of 40 tasks from DeepMind Control Suite (Tassa et al., 2018), MyoSuite (Caggiano et al., 2022) and MetaWorld (Yu et al., 2020). Selected tasks cover various challenges, from simple to hard, in locomotion and manipulation. Table 4 presents the environments with specified dimensions of states and actions. BRO is a versatile agent that can successfully perform tasks of different difficulty and various action and state spaces. Our selection of 40 tasks focuses on the most challenging tasks from the DeepMind Control Suite (DMC), MetaWorld (MW), and MyoSuite (MS) benchmarks, as identified in previous studies (Hansen et al., 2023). We chose these hard tasks because many easy tasks from these benchmarks can be solved by modern algorithms within 100k environment steps (Hansen et al., 2023; Wang et al., 2024). In the MetaWorld environment, we follow the TD-MPC2 evaluation protocol (Hansen et al., 2023). As such, the environment issues a truncate signal after 200 environment steps, after which we assess if the agent achieved goal success within the 200th step. We do not implement any changes to how goals are defined in the original MetaWorld and we use V2 environments.

Table 3: List of tasks from DeepMind Control and MetaWorld on which the agents were ablated. The table also contains the dimensions of action and observation space.

Task	**Observation dimension**	**Action dimension**
DEEPMIND CONTROL		
Acrobot-Swingup	6	1
Dog-Trot	223	38
Hopper-Hop	15	4
Humanoid-Run	67	24
Humanoid-Walk	67	24
METAWORLD		
Assembly	39	4
Coffee-Push	39	4
Hand-Insert	39	4
Push	39	4
Stick-Pull	39	4

Table 4: List of tasks from DeepMind Control, MetaWorld, and MyoSuite on which the agents were tested. The table also contains the dimensions of action and observation space.

Task	Observation dimension	Action dimension
DEEPMIND CONTROL		
Acrobot-Swingup	6	1
Cheetah-Run	17	6
Dog-Run	223	38
Dog-Trot	223	38
Dog-Stand	223	38
Dog-Walk	223	38
Finger-Turn-Hard	12	2
Fish-Swim	24	5
Hopper-Hop	15	4
Humanoid-Run	67	24
Humanoid-Stand	67	24
Humanoid-Walk	67	24
Pendulum-Swingup	3	1
Quadruped-Run	78	12
Walker-Run	24	6
METAWORLD		
Basketball	39	4
Assembly	39	4
Button-Press	39	4
Coffee-Pull	39	4
Coffee-Push	39	4
Disassemble	39	4
Hammer	39	4
Hand-Insert	39	4
Push	39	4
Reach	39	4
Stick-Pull	39	4
Sweep	39	4
Lever-Pull	39	4
Pick-Place	39	4
Push-Back	39	4
MYOSUITE		
Key-Turn	93	39
Key-Turn-Hard	93	39
Obj-Hold	91	39
Obj-Hold-Hard	91	39
Pen-Twirl	83	39
Pen-Twirl-Hard	83	39
Pose	108	39
Pose-Hard	108	39
Reach	115	39
Reach-Hard	115	39

D Hyperparameters

Hyperparameters of BRO and other baselines are listed in Table 5. BRO (Fast) shares the same parameters as BRO except replay ratio 2 which significantly speeds the algorithm without sacrificing performance that much. BRO features the BRONet architecture and resets of all parameters done every $250k$ steps until $1M$ steps with additional resets at steps $15k$ and $50k$. The selection of

hyperparameters for BRO was based on the values reported in the main building blocks of BRO and extensive experimentation coupled with ablations studies.

Table 5: Hyperparameter values for actor-critic agents used in the experiments.

Parameter	BRO	SAC	TD3	SR-SAC	CrossQ
Batch size (B)	128	256			
Replay ratio	10	2		32	1
Critic hidden depth	RESIDUAL	2		2	2
Critic hidden size	512	256			2048
Actor depth	RESIDUAL	2		2	2
Actor size	256				
Num quantiles	100	N/A			
KL target	0.05	N/A			
Initial optimism	1.0	N/A			
Std multiplier	0.75	N/A			
Actor learning rate	3e-4				1e-3
Critic learning rate	3e-4				1e-3
Temperature learning rate	3e-4		N/A	3e-4	
Optimizer	ADAMW	ADAM			
Discount (γ)	0.99				
Initial temperature (α_0)	1.0		N/A	1.0	
Exploratory steps	2,500	10,000	25,000	10,000	5,000
Target entropy ($\mathcal{H}^*$)	$\lvert\mathcal{A}\rvert/2$		N/A	$\lvert\mathcal{A}\rvert/2$	$\lvert\mathcal{A}\rvert$
Polyak weight (τ)	0.005				N/A

We compare BRO against official and widely used implementations of CrossQ, SAC, SR-SAC, TD3 and TD-MPC2 with open source repositories listed in Table 6. As the official results do not cover all 40 benchmarking tasks, we ran the baselines independently (except TD-MPC2, where all official results were available). SAC and TD3 are commonly used baselines; therefore, their hyperparameters vary across different implementations. To account for this fact, we ran 2 versions of these baselines: tuned and original. If not specified otherwise, we report the results of the tuned versions with hyperparameters in Table 5. The original versions of SAC and TD3 both feature a replay ratio of 1 and in the case of SAC, target entropy ($\mathcal{H}^*$) equal to the action space dimension $|\mathcal{A}|$. The performance of both variants of the implementations can be observed in Figure 24.

Table 6: Links to the repositories of the used baselines. All are distributed under MIT license.

Baseline	**Source code link**
CrossQ (Bhatt et al., 2023)	`github.com/adityab/CrossQ`
SAC (Haarnoja et al., 2018)	`github.com/denisyarats/pytorch_sac`
SAC (tuned version)	`github.com/ikostrikov/jaxrl`
SR-SAC (D'Oro et al., 2022)	`github.com/proceduralia/high_replay_ratio_continuous_control`
TD3 (Fujimoto et al., 2018)	`github.com/sfujim/TD3`
TD3 (tuned version)	`github.com/ikostrikov/jaxrl`
TD-MPC2 (Hansen et al., 2023)	`github.com/nicklashansen/tdmpc2`

As other baselines were developed and tested on only a subset of our 40 selected tasks, we observed that achieving similar performance on new tasks was challenging. This can be especially observed in the case of CrossQ, which is a state-of-the-art algorithm on selected tasks from OpenAI Gym

(Brockman et al., 2016), but as it was tested only on a fraction of DeepMind Control Suite tasks, its performance does not transfer to our selection of tasks. Originally, CrossQ authors tested their agent on DeepMind Control Suite using Shimmy (Tai et al.) contrary to other agents that use the original codebase (Tassa et al., 2018). We run CrossQ using the DMC wrappers (D'Oro et al., 2022). Comparison between 2 variants of SAC and TD3 (Original and Tuned) is presented in Figure 24. Tuned versions feature a higher value of replay ratio (2 instead of 1) than the original and lower target entropy in the case of SAC ($|\mathcal{A}|/2$ instead of $|\mathcal{A}|$).

Table 7: Description of the considered model sizes.

Size	Number of block	Hidden Size
$0.55M$	1 BroNet block	hidden size of 128
$1.05M$	1 BroNet block	hidden size of 256
$2.83M$	1 BroNet block	hidden size of 512
$4.92M$	2 BroNet blocks	hidden size of 512
$26.31M$	3 BroNet blocks	hidden size of 1024

E Additional Experiments

E.1 Scaling and Time Experiments

The execution time was measured for all agents for each of the 40 tasks averaged over 2 random seeds. We ran each agent for $105k$ steps with initial $5k$ exploration steps, $100k$ training steps, and 1 evaluation. We benchmark all 25 variants of BRO with 5 different model sizes and 5 values of replay ratio. Figure 12 different algorithms performance compared to execution time. Experiments were conducted on an NVIDIA A100 GPU with 10GB of RAM and 8 CPU cores of AMD EPYC 7742 processor. All tasks were run separately so the agents could use all resources independently.

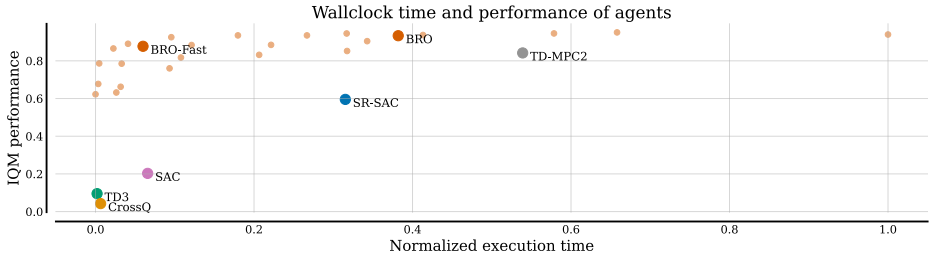


Figure 12: Scatterplot of the performance of agents plotted against normalized execution time.

Increasing the model size and replay ratio both improve the performance. However, the former is more efficient in terms of execution time due to GPU parallelism. For example, the largest BRO variant ($26.31M$ parameters) with replay ratio 5 has similar execution time as the smallest one ($0.55M$ parameters) with replay ratio 15, but the performance is much greater (Figures 13).

E.2 Additional BroNet Analysis

We examine various architectural blueprints on 5 DMC and 5 MetaWorld environments (see Table 3), each with over 10 seeds per task. Our starting point was the transformer-based design by Bjorck et al. (2021), termed `Spectral`. This architecture incorporates recent transformer advancements, moving Layer Norm to the beginning of the residual block to prevent vanishing gradients in deep networks. While `Spectral` performs better than the vanilla MLP, its performance on the DMC benchmark, particularly the Dog environment, is weaker. This aligns with findings from Nauman et al. (2024); Hussing et al. (2024), indicating that Layer Norm is crucial for stability in such complex tasks. To analyze the importance of layer normalization in the BroNet architecture, we replaced spectral norms with Layer Norms in the residual blocks, resulting in the `BRO wo first LN` architecture (Figure 5). This modification improves performance but still lags behind the full BRO architecture. Furthermore,

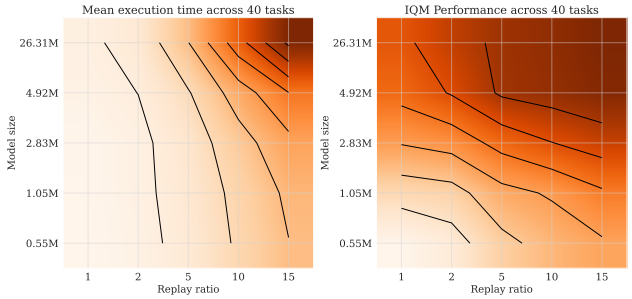


Figure 13: Heatmaps of execution time and IQM performance across 40 tasks of 25 BRO variants with various model sizes and replay ratio values. Black lines connect the same interpolated values.

we examine a simple MLP architecture with Layer Norm before each activation function. Since BRO consists of two residual blocks, we compare it with a 5-layer model, `(Dense + LN) x 5`. Figure 14 shows that Layer Norm after each Dense layer is effective, and in aggregated IQM, this model is comparable to BRO. However, skip connections in BRO are beneficial for managing complex environments like Dog. In conclusion, BroNet architecture uses Layer Norm and residual blocks for superior robustness and performance in challenging tasks.

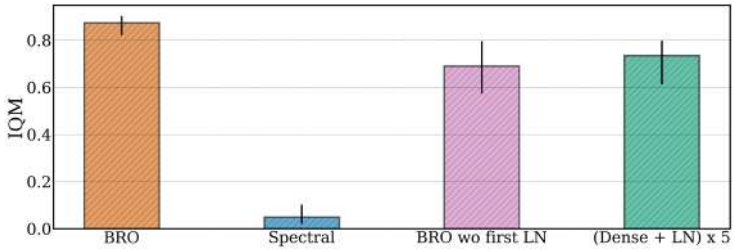


Figure 14: Comparison of five architecture designs across different environments: The top plot shows results on 5 DMC and 5 MetaWorld environments, the middle plot focuses on the 5 DMC environments, and the bottom plot highlights the Dog Trot environment. BRO and Spectral architectures each consist of 2 residual blocks. `(Dense + LN) x 5` represents standard MLP networks with 5 hidden layers, each incorporating Layer Norm before activation. Lastly, `BRO wo first LN` refers to the BRO architecture without Layer Norm in the first Dense block, before the residual connection.

E.3 Additional analysis

Batch sizes We ablate of the minibatch size impact on BRO and BRO (Fast) performance across different benchmarks is depicted in Figure 15. The figure shows that using half or even a quarter of the original minibatch size (256) does not significantly hurt BRO's performance.

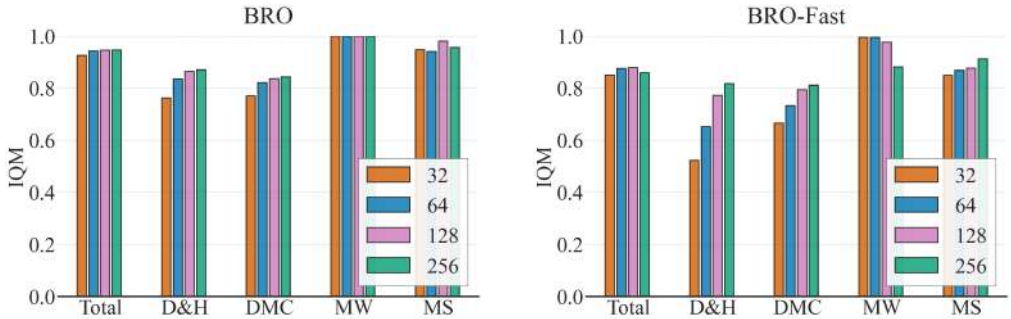


Figure 15: Performance of BRO and BRO (Fast) with different minibatch sizes for: D&H (Dogs and Humanoid), DMC (DeepMind Control), MW (MetaWorld), and MS (MyoSuite).

Target network We investigate the performance benefits stemming from using a target network with the BRO agent. We present these results in Figure 16. Interestingly, we observe that the target network yields performance improvements only in specific environments.

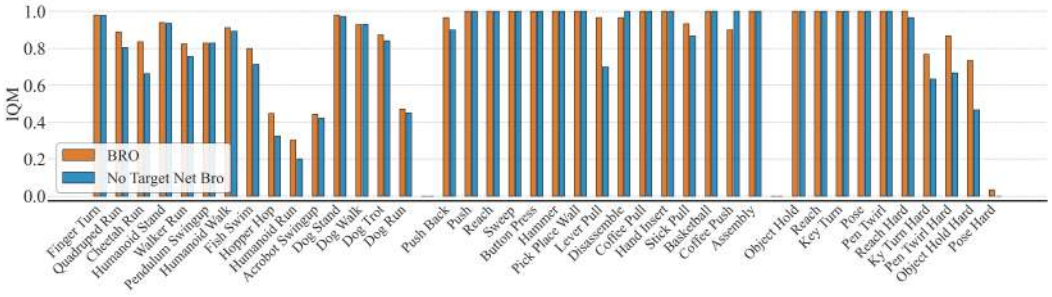


Figure 16: We compare BRO against BRO without target network. 10 seeds per task, $1M$ steps.

More baselines To evaluate BRO performance beyond maximum entropy objectives, we tested BRO and BRO (Fast) with a TD3 backbone. BRO with a SAC backbone slightly outperformed TD3, though TD3 remains a viable option. Furthermore, we compare BRO performance to three additional baselines. These results are presented in Figure 17.

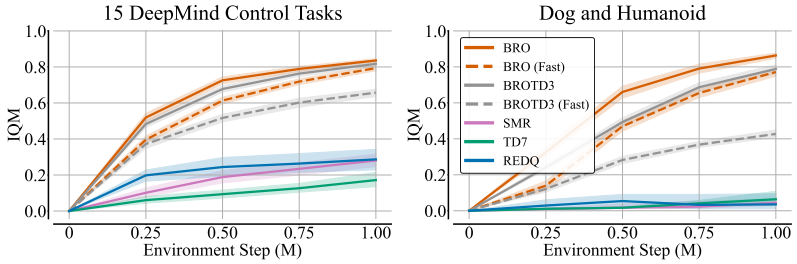


Figure 17: We run BRO and BRO (Fast) with a TD3 backbone (BROTD3). 10 seeds.

Longer training We expanded BRO training beyond 1M environment steps, although in a single-task setup. We trained BRO and BRO (Fast) for 3M and 5M steps respectively on 7 Dog and Humanoid tasks and compared them to TD-MPC2 and SR-SAC. BRO significantly outperforms these baselines and notably almost solves the Dog Run tasks at 5M steps (achieving over 80% of possible returns). We show the 3M results in Figure 18.

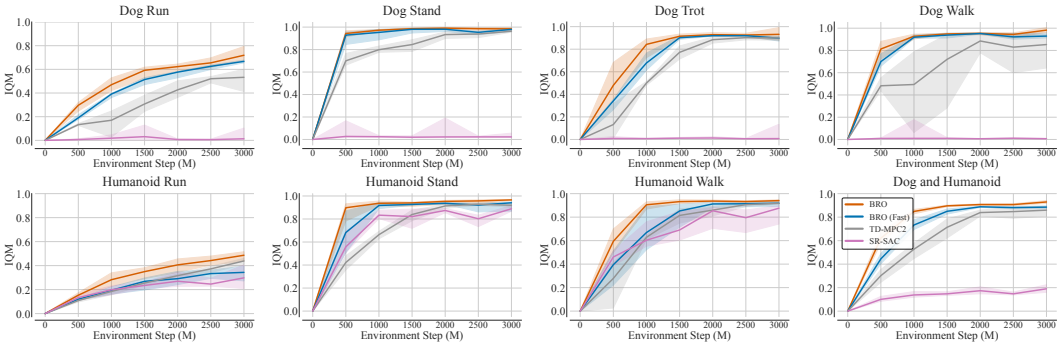


Figure 18: We run BRO on complex tasks for 3M steps. 5 seeds.

Image-based tasks To analyze the impact of BroNet on image-based RL tasks, we experiment with 3 tasks from the Atari 100k (Bellemare et al., 2013) benchmark. Here, we changed the regular

Q-network of the SR-SPR (RR=2) model (D'Oro et al., 2022) to a BroNet, and considered changing the reset schedules to better fit the plasticity of the BroNet model. As depicted in the Table below, applying BroNet to discrete, image-based tasks is a promising avenue for future research.

Table 8: We replace the Q-network in SR-SPR with BroNet, while keeping the standard convolutional encoder. We test two values of reset interval (RI) and shrink-and-perturb (SP) and find that these hyperparameters impact the performance of the BroNet agent. 5 seeds.

Task	**SR-SPR (RI=20k;SP=0.8)**	**SR-SPR-BroNet (RI=20k;SP=0.8)**	**SR-SPR-BroNet (RI=20k;SP=0.5)**	**SR-SPR-BroNet (RI=5k;SP=0.8)**
Pong	-10.5	4.8	10.2	-12.0
Seaquest	714.7	399.0	420.7	782.0
Breakout	24.5	13.5	13.7	33.3

Multi-Task RL Finally, we evaluate whether the increased model size improves the agent's capability in a multi-task setup (Yu et al., 2020; Hansen et al., 2023). To this end, we compare the BRO (Fast) to SAC on the MT50 benchmark from MetaWorld, which comprises 50 different tasks. To accommodate the task diversity, we increase the width of all models twofold and pass a one-hot vector representing the task identifier in the network input. Otherwise, both algorithms are run with the same hyperparameters as the main experiments. We present the results in Figure 19.

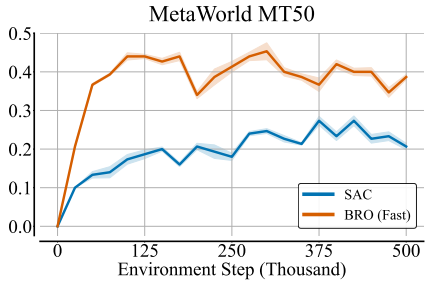


Figure 19: We compare BRO (Fast) with SAC on the multi-task benchmark. 3 seeds

F Training Curves

We present the aggregated performance of BRO compared to other baselines at Dog & Humanoid tasks, DeepMind Control Suite, Metaworld and MyoSuite in Figure 20 together with summarized performance results at $100k$, $200k$, $500k$ and $1M$ steps in Table 9. The performance on each of the 40 individual tasks in shown in Figures 21, 23, 22.

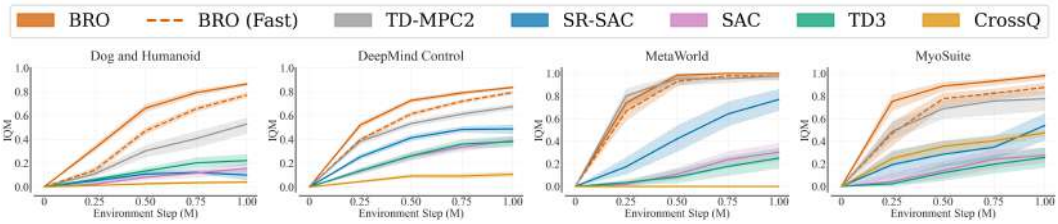


Figure 20: BRO **aggregated** performance over $1M$ steps on 40 **tasks** from DeepMind Control Suite, MetaWorld and MyoSuite. Y-axis represents the IQM of normalized rewards.

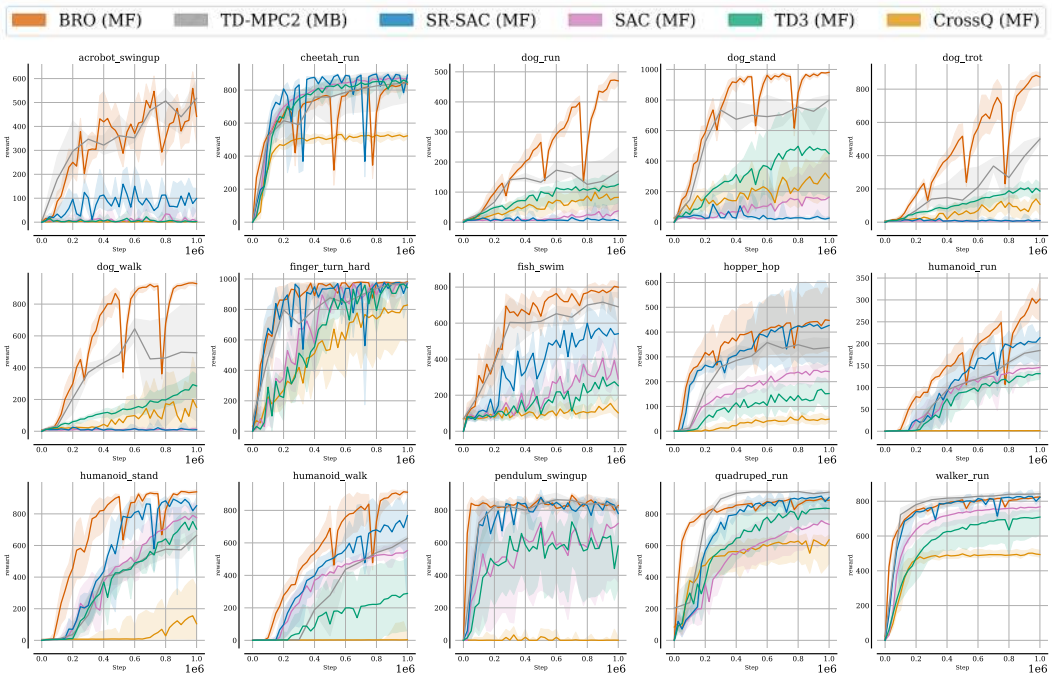


Figure 21: Results of 15 tasks from **DeepMind Control Suite** for BRO and other baselines run for $1M$ steps. We present the IQM of rewards and 95% confidence intervals.

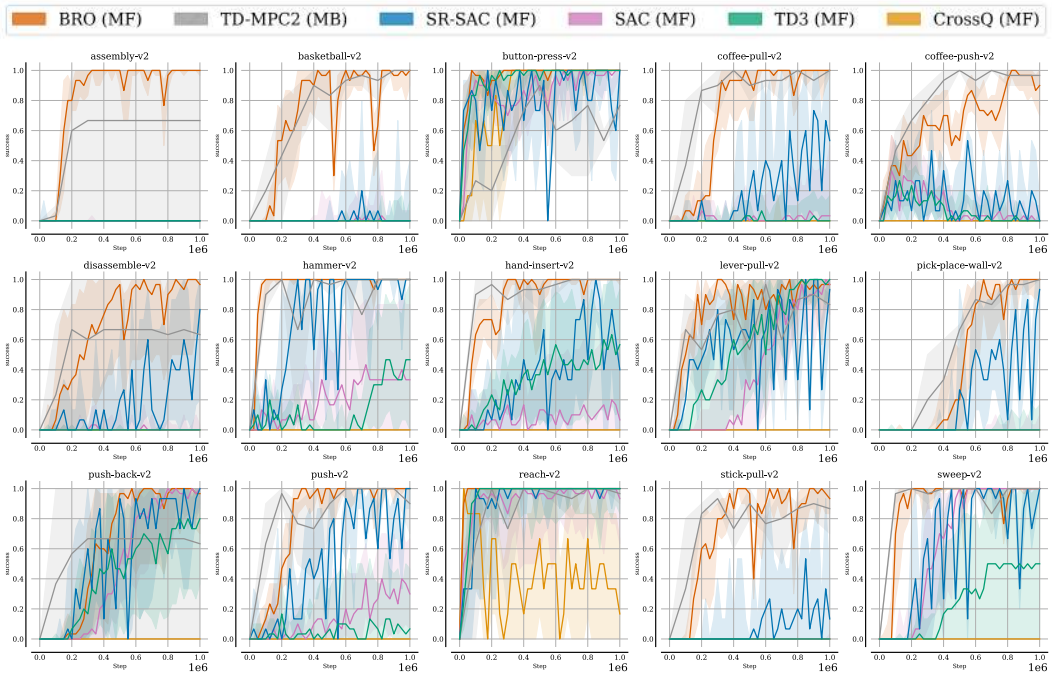


Figure 22: Results of 15 tasks from **MetaWorld** for BRO and other baselines run for $1M$ steps. We present the IQM of success rate and 95% confidence intervals.

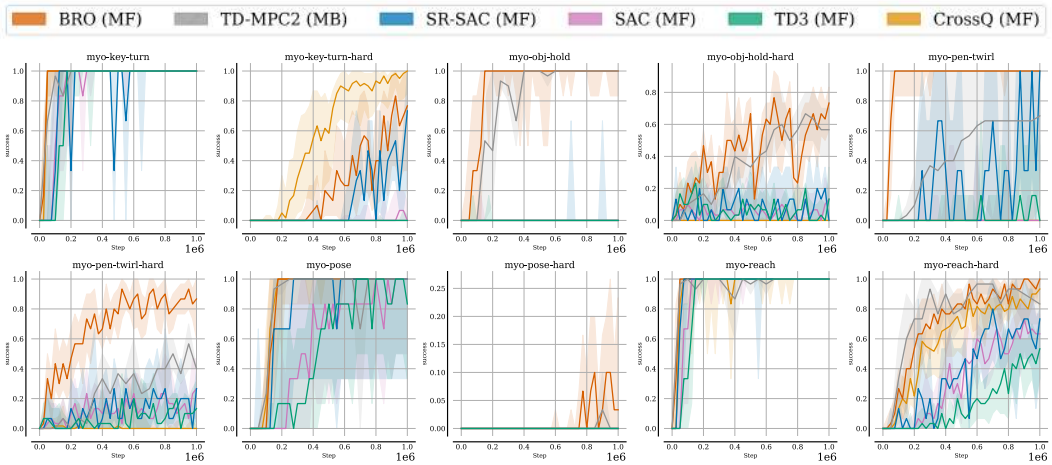


Figure 23: Results of 10 tasks from **MyoSuite** for BRO and other baselines run for $1M$ steps. We present the IQM of success rate and 95% confidence intervals.

Table 9: Summary of IQM results of BRO and other agents evaluated on 40 tasks from DeepMind Control Suite, Metaworld and MyoSuite achieved at $100k$, $200k$, $500k$ and $1M$ steps. BRO achieves **better results** than other state-of-the-art agents (both model-based and model-free) while featuring great sample efficiency.

Step	BRO	BRO (Fast)	TD-MPC2	SR-SAC	SAC	TD3	CrossQ
AGGREGATED 40 TASKS							
$100k$	**0.254**	0.113	0.204	0.046	0.007	0.008	0.004
$200k$	**0.560**	0.384	0.519	0.083	0.043	0.054	0.009
$500k$	**0.862**	0.772	0.745	0.373	0.167	0.157	0.037
$1M$	**0.945**	0.878	0.842	0.595	0.316	0.294	0.042
DEEPMIND CONTROL SUITE							
$100k$	**0.230**	0.123	0.128	0.089	0.030	0.046	0.031
$200k$	**0.442**	0.282	0.332	0.195	0.088	0.091	0.038
$500k$	**0.726**	0.613	0.532	0.412	0.259	0.261	0.092
$1M$	**0.836**	0.794	0.673	0.487	0.391	0.381	0.106
METAWORLD							
$100k$	0.247	0.113	**0.452**	0.046	0.000	0.000	0.000
$200k$	0.642	0.571	**0.835**	0.062	0.018	0.047	0.000
$500k$	**0.984**	0.929	0.952	0.421	0.108	0.084	0.000
$1M$	**1.000**	0.976	0.978	0.769	0.303	0.250	0.000
MYOSUITE							
$100k$	**0.392**	0.124	0.088	0.000	0.000	0.000	0.020
$200k$	**0.748**	0.400	0.394	0.015	0.024	0.008	0.140
$500k$	**0.888**	0.776	0.688	0.285	0.140	0.120	0.354
$1M$	**0.980**	0.876	0.775	0.538	0.276	0.256	0.474
DOG & HUMANOID							
$100k$	**0.038**	0.008	0.014	0.007	0.006	0.013	0.011
$200k$	**0.238**	0.056	0.058	0.024	0.015	0.040	0.013
$500k$	**0.661**	0.469	0.302	0.107	0.080	0.133	0.025
$1M$	**0.864**	0.772	0.527	0.099	0.155	0.221	0.040

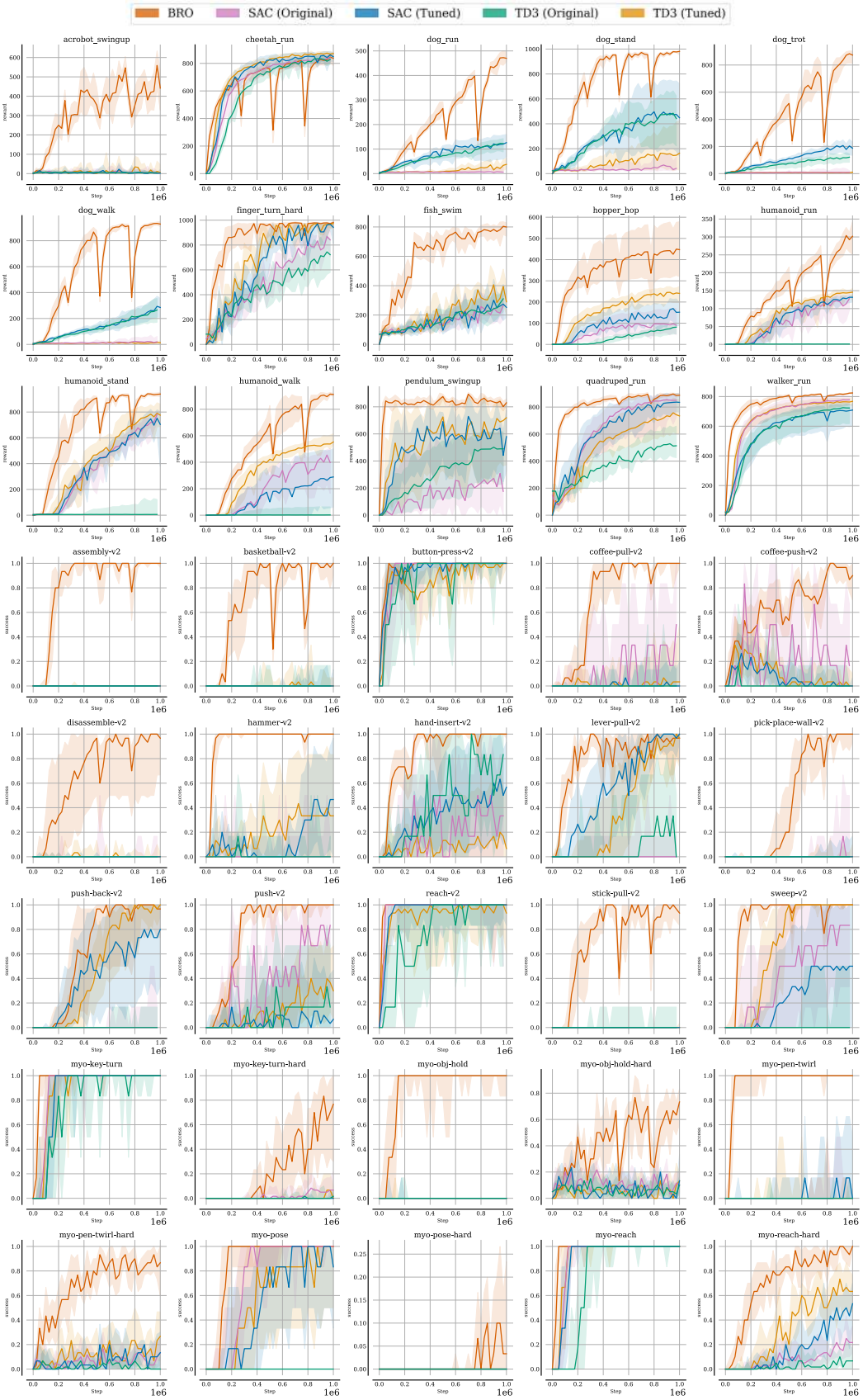


Figure 24: Results of different variants of SAC and TD3 on 40 tasks.

Value-Based Deep RL Scales Predictably

Oleh Rybkin [1] **Michal Nauman** [1 2] **Preston Fu** [1] **Charlie Snell** [1] **Pieter Abbeel** [1] **Sergey Levine** [1] **Aviral Kumar** [3]

Abstract

Abstract: Scaling data and compute is critical to the success of modern ML. However, scaling demands ***predictability***: we want methods to not only perform well with more compute or data, but also have their performance be predictable from small-scale runs, without running the large-scale experiment. In this paper, we show that value-based off-policy RL methods are predictable despite community lore regarding their pathological behavior. First, we show that data and compute requirements to attain a given performance level lie on a *Pareto frontier*, controlled by the updates-to-data (UTD) ratio. By estimating this frontier, we can predict this data requirement when given more compute, and this compute requirement when given more data. Second, we determine the optimal allocation of a total resource *budget* across data and compute for a given performance and use it to determine hyperparameters that maximize performance for a given budget. Third, this scaling is enabled by first estimating predictable relationships between *hyperparameters*, which is used to manage effects of overfitting and plasticity loss unique to RL. We validate our approach using three algorithms: SAC, BRO, and PQL on DeepMind Control, OpenAI gym, and IsaacGym, when extrapolating to higher levels of data, compute, budget, or performance.

1. Introduction

Many latest advances in various areas of machine learning have emerged from training big models on large datasets. In this scaling guided research landscape, successfully executing even one single training run often requires a large amount of data, computational resources, and wall-clock time, such as weeks or months (Achiam et al., 2023; Team et al., 2023; Ramesh et al., 2022; Brooks et al., 2024). To maximize the success of these large-scale runs, the trend in the machine learning (ML) community has shifted toward not just performant, but also more **predictable algorithms that scale reliably** with more computation and training data size, such that downstream performance can be predicted from small-scale experiments, without actually running the large-scale experiment (McCandlish et al., 2018; Kaplan et al., 2020; Hoffmann et al., 2023; Dubey et al., 2024).

In this paper, we study if deep reinforcement learning (RL) is also amenable to such scaling and predictability benefits. We focus on **value-based methods** that train value functions using temporal difference (TD) learning, which are known to be performant at small scales (Mnih et al., 2015; Lillicrap et al., 2016; Haarnoja et al., 2018). Compared to policy gradient (Mnih et al., 2016; Schulman et al., 2017) and search methods (Silver et al., 2016), value-based RL can learn from arbitrary data and require less sampling or search, which can be inefficient or infeasible for open-world problems where environment interaction is costly.

We study scaling properties by predicting relationships between different **resources required for training**. *Data* requirement $\mathcal{D}$ is the amount of data needed to attain a certain level of performance. Likewise, *compute* requirement $\mathcal{C}$ refers to the amount of FLOPs or gradient steps needed to attain a certain level of performance. In RL uniquely, performance can be improved by increasing *either* available data or compute (e.g., training multiple times on the same data), which we capture via a *budget* requirement that combines data and compute $\mathcal{F} = \mathcal{C} + \delta \cdot \mathcal{D}$, where δ is some constant. An additive budget function is useful when the cost of data and compute can be expressed in similar units, such as wall-clock time or required finances.

To establish scaling relationships, we first require a way to predict the **best hyperparameter settings** at each scale. We find that learning rate η, batch size B, and the updates-to-data (UTD) ratio σ are the most crucial hyperparameters for value-based RL. While supervised learning benefits from abundant theory to establish optimal hyperparameters (Krizhevsky, 2014; McCandlish et al., 2018; Yang et al., 2021), value-based RL often does not satisfy assumptions typical of supervised learning. For example, value-based RL needs to account for the non-i.i.d. nature of training data. Distribution shift due to periodic changes in the data col-

[1]UC Berkeley [2]University of Warsaw [3]CMU. Correspondence to: Oleh Rybkin <oleh.rybkin@gmail.com>, Aviral Kumar <aviralku@andrew.cmu.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

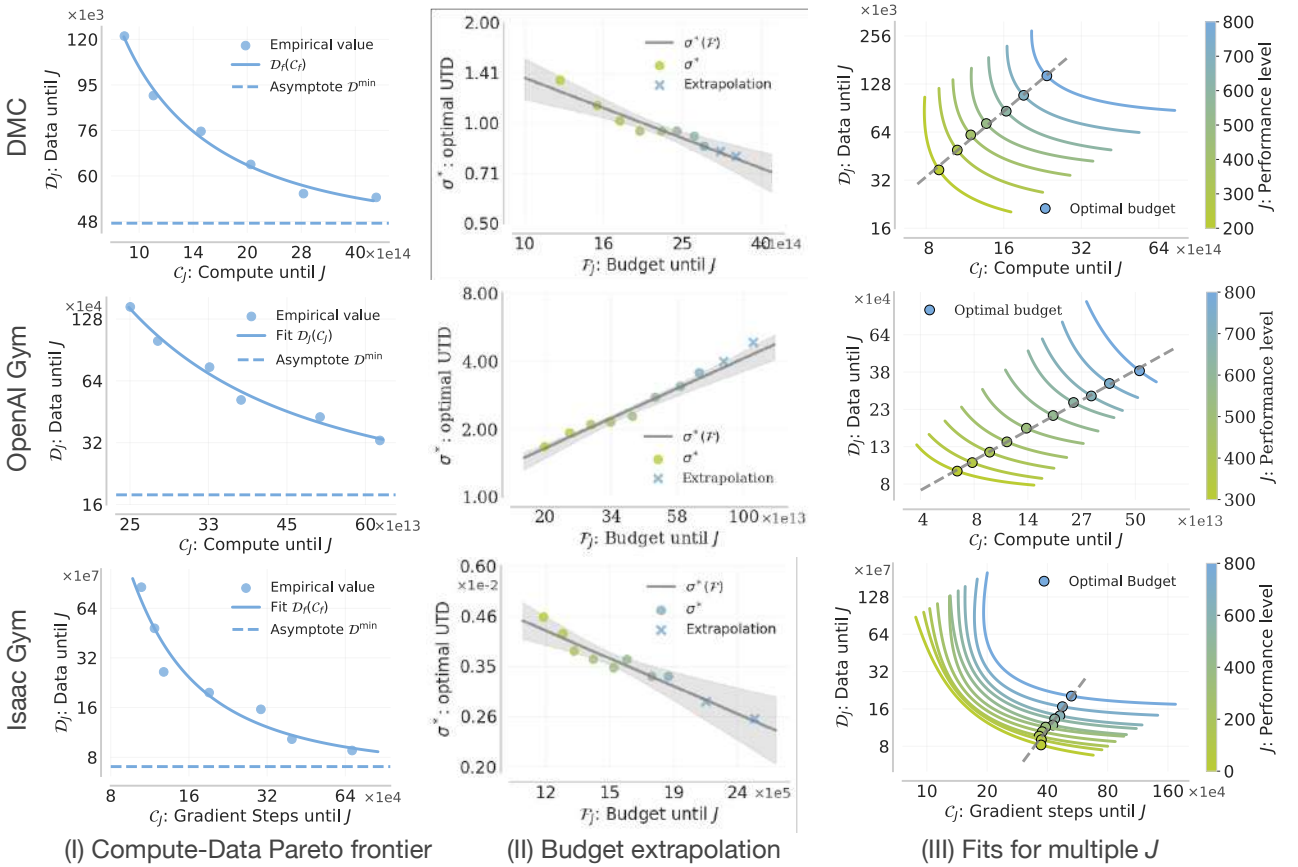


Figure 1: ***Scaling properties when increasing compute $\mathcal{C}$, data $\mathcal{D}$, budget $\mathcal{F}$, or performance J.*** **Left:** Compute versus data requirements Pareto frontier controlled by the UTD ratio σ. We observe that we can trade off data for compute and vice versa, and this relationship is predictable. **Middle:** Extrapolation from low to high performance. We observe that the optimal resource allocation controlled by σ evolves predictably with increasing budget, and can be used to extrapolate from low to high performance. **Right:** Pareto frontiers for several performance levels J.

lection policy (Levine et al., 2020) contributes to a form of overfitting where minimizing training TD error may not result in a low TD error under the data distribution induced by the new policy. In addition, objective shift due to changing target values (Dabney et al., 2021) contributes to "plasticity loss" (D'Oro et al., 2023; Kumar et al., 2021). We show that it is possible to account for the training dynamics unique to value-based RL, and are able to find the best hyperparameters by setting the batch size and learning rate inversely proportional to the UTD ratio. We estimate this dependency using a power law (Kaplan et al., 2020), and observe that this model makes effective predictions.

Using the best predicted hyperparameters, we are now able to establish that data and compute requirements evolve as a predictable function of the UTD ratio σ. Furthermore, σ defines the **tradeoff between data and compute**, which can be visualized as a Pareto frontier (Figure 1, left). Using this model, we are able to extrapolate the resource requirements from low-compute to high-compute setting, as well as from low-data to high-data setting as shown in the figure.

Using the Pareto frontiers, we are now able to **extrapolate from low to high performance levels**. Instead of extrapolating as a function of return, which can be arbitrary and non-smooth, we extrapolate as a function of the allowed budget $\mathcal{F}$. We can define an optimal tradeoff between data and compute, and we observe that such optimal tradeoff value evolves predictably to higher budgets, which also attains a higher performance level (Figure 1, middle). Thus we are able to predict optimal hyperparameters, as well as data and compute allocation, for high-budget runs using only data from low-budget runs.

Our contribution is showing that the behavior of value-based deep RL methods based on TD-learning is predictable in larger data and compute regimes. Specifically, we:

1. establish predictable rules for dependencies between hyperparameters batch size (B), learning rate (η), and UTD ratio (σ) in value-based RL, and show that these rules enable more effective scaling.
2. show that data and compute required to attain a given performance level lie on a Pareto frontier, and are respectively predictable in the higher-compute or higher-data regimes.

3. show the optimal allocation of budget between data and compute, and predict how such allocation evolves with higher budgets for best performance.

Our findings apply to algorithms such as SAC, BRO, and PQL, and domains such as the DeepMind Control Suite (DMC), OpenAI Gym, and IsaacGym. The generality of our conclusions challenges conventional wisdom that value-based deep RL does not scale predictably.

2. RL Preliminaries and Notation

We study standard off-policy online RL, which maximizes the agent's return by training on a replay buffer and periodically collecting new data (Sutton and Barto, 2018). Value-based deep RL methods train a Q-network, Q_θ, to minimize the temporal difference (TD) error:

$$L(\theta) = \mathbb{E}_{\mathcal{P}}\left[\left(r(s,a) + \gamma \bar{Q}(s',a') - Q_\theta(s,a)\right)^2\right], \quad (2.1)$$

where $\mathcal{P}$ is the replay buffer, $\bar{Q}$ is the target Q-network, s denotes a state, and a' is an action drawn from a policy $\pi(\cdot|s)$ that aims to maximize $Q_\theta(s,a)$. We implement this operation by sampling a batch of size B from the buffer and taking a gradient step along the gradient of this loss with a learning rate η. In theory, off-policy algorithms can be made very sample efficient by minimizing the TD error fully over any data batch, which in practice translates to making more update steps to the Q-network per environment step, or higher "updates-to-data" ratio (UTD) (Chen et al., 2020). However, increasing the UTD ratio naïvely can lead to worse performance (Nikishin et al., 2022; Janner et al., 2019). To this end, unlike the standard supervised learning or LLM literature that considers B and η as two main hyperparameters affecting training (Kaplan et al., 2020; Hoffmann et al., 2023), our setting presents another hyperparameter, the UTD ratio σ, that we also study in our paper.

Notation. In this paper, we focus on the following key hyperparameters: the UTD ratio σ, learning rate η, and the batch size B. We will answer questions pertaining to performance of a policy π denoted by $J(\pi)$, the total data utilized by an algorithm to reach a given target level of performance J (denoted by $\mathcal{D}_J$), and the total compute budget utilized by the algorithm to reach performance J (denoted by $\mathcal{C}_J$), which is measured in terms of FLOPs or wall-clock time taken by the algorithm.

3. Problem Statement and Formulation

To demonstrate that the behavior of value-based RL can be predicted reliably at scale, we first post multiple *resource optimization* questions that guide our scaling study. Viewing data and compute as two resources, we answer questions of the form: *what is the minimum value of* `[resource]` *needed to attain a given target performance? And what should the hyperparameters (e.g., B, η, σ) be in such this training run?* We will answer such questions by fitting empirical laws from low data and compute runs to determine relationships between hyperparameters. Doing so, in turn, enables us to determine how to set hyperparameters and allocate resources to maximize performance when provided with a larger data and compute budget. Note that we wish to make these hyperparameter predictions without running the large data and compute budget experiment. While questions of this form have been studied in supervised learning, the answers are different for online RL, because online RL continuously collects its own data, which ties data and compute in a complex manner and breaks i.i.d. nature of datapoints.

Concretely, we study three resource optimization questions: **(1)** maximizing sample efficiency (i.e., minimize the amount of data $\mathcal{D}$ to attain a target performance under a given compute budget), **(2)** conversely, minimizing compute $\mathcal{C}$ (e.g., FLOPs or gradient steps, whichever is more appropriate) to attain a given performance given an upper bound on data that can be collected, and **(3)** maximizing performance given a total bound on data and compute.

Problem 3.1 (Resource optimization problems)**.** Find the best configuration (B, η, σ) for algorithm Alg that minimizes either the data $\mathcal{D}$ or compute $\mathcal{C}$ consumed to obtain performance J_0:

1. *Maximal sample efficiency*:
$$(B^*, \eta^*, \sigma^*) := \arg\min_{(B,\eta,\sigma)} \quad \mathcal{D} \quad \text{s.t. } J\left(\pi_{\text{Alg}}(B,\eta,\sigma)\right) \geq J_0 \quad \mathcal{C} \leq \mathcal{C}_0.$$
2. *Maximal compute efficiency*:
$$(B^*, \eta^*, \sigma^*) := \arg\min_{(B,\eta,\sigma)} \quad \mathcal{C} \quad \text{s.t. } J\left(\pi_{\text{Alg}}(B,\eta,\sigma)\right) \geq J_0 \quad \mathcal{D} \leq D_0.$$

We solve these problems by fitting empirical models of the minimum data and compute needed to attain a target performance for different values of J_0. Doing so allows us to then solve the third setting **(3)** for maximizing performance given a total budget on data and compute as shown below.

Problem 3.2 (Maximize performance at large data and compute budget)**.** Find the best configuration (B, η, σ) and resource allocations for data $\mathcal{D}$ and compute $\mathcal{C}$ that enable Alg to maximize performance at budget $\mathcal{F}_0$
$$(B^*, \eta^*, \sigma^*) := \arg\max_{(B,\eta,\sigma)} \quad J\left(\pi_{\text{Alg}}(B,\eta,\sigma)\right) \quad \text{s.t. } \mathcal{C} + \delta \cdot \mathcal{D} \leq \mathcal{F}_0.$$

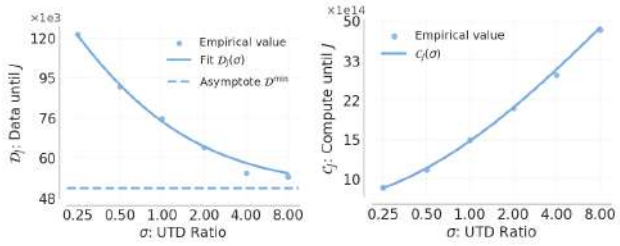


Figure 2: The data-compute tradeoff on DMC. ***Left:*** The minimum required data $\mathcal{D}_J$ scales with the UTD σ as a power law. ***Right:*** The minimum required compute $\mathcal{C}_J$ increases with the UTD σ as a sum of two power laws.

4. Scaling Results For Value-Based Deep RL

We will now present our main results addressing Problem 3.1 under the two settings discussed above. We will then use these results to present results for Problem 3.2. In order to do so, we run several experiments and estimate scaling trends from the results. Although this procedure might appear standard from scaling studies in language modeling, we found that instantiating it for value-based RL requires understanding the interaction of the various hyperparameters appearing in TD updates, and the data and compute efficiency of the algorithm. We will formalize these relationships via empirically estimated *laws* and show that these laws extrapolate reliably to new settings not used to obtain these empirical laws. Therefore, in this section, we present empirical and conceptual arguments to build functional forms of relationships between different hyperparameters. Before doing so, we provide our answers to Problems 3.1 and 3.2.

4.1. Main Scaling Results

We begin by answering Problem 3.1 where we maximize sample efficiency. We wish to estimate the minimal amount of data $\mathcal{D}_J$ needed to attain a given target performance, given an upper bound on compute $\mathcal{C} \leq \mathcal{C}_0$. To do so, we fit $\mathcal{D}_J$ needed to attain the target performance $J = J_0$ parameterized by the UTD ratio σ (Eq. (4.1)). Intuitively, the minimum amount of data needed to attain a given performance is lower as more updates are made per datapoint (i.e., when σ is high), as more "value" could be derived from the same datapoint. In addition, we would expect that even for the best value of σ, there is a minimum number of datapoints $\mathcal{D}^{\min}$ that are needed to learn given the "intrinsic" difficulty of the task at hand. Based on these intuitions, we hypothesize a power law relationship between $\mathcal{D}_J(\sigma)$ and σ, with an offset $\mathcal{D}^{\min}$ and constants α_J and β_J.

$$\mathcal{D}_J(\sigma) \approx \mathcal{D}_J^{\min} + \left(\frac{\beta_J}{\sigma}\right)^{\alpha_J} \tag{4.1}$$

Empirical fits of $\mathcal{D}_J$ and σ on the DMC suite are in Figure 2 and they validate the efficacy of this fit. We also emphasize that the existence of this power law makes $\mathcal{D}_J$ *predictable*, in that we can predict $\mathcal{D}_J$ for larger values of σ that fall outside the range of σ values used to get the fit (Figure 6).

Scaling Observation 1: Data Requirements

The amount of data $\mathcal{D}_J$ needed to reach a given return target J_0 decreases as a predictable function of the UTD σ, and is a power law (Eq. (4.1)).

To answer the optimization questions in Problem 3.1, we also need an expression for required compute until the target return $\mathcal{C}_J$. As σ determines the number of gradient steps per data point, $\mathcal{C}_J$ is a function of σ. In particular, total compute is equal to the number of gradient steps taken multiplied by the parameter count of the model. Our study does not optimize over the model size and treats it as a constant. Thus, we write the compute $\mathcal{C}_J$ as a function of σ as:

$$\mathcal{C}_J(\sigma) \approx 10 \cdot N \cdot B(\sigma) \cdot \sigma \cdot \mathcal{D}_J(\sigma) \tag{4.2}$$

where N denotes the model size, $B(\sigma)$ denotes the "best choice" batch size for a given UTD value σ, and other variables follow definitions from before. Note the additional factor of 10 in Eq. (4.2) emerges from the use of multiple forward passes to compute the loss function for value-based RL and the backward pass, through the Q-network (to contrast with language modeling, the typical multiplier is 6; the gap in our setting comes from the use of multiple forward passes). We plot $\mathcal{C}_J(\sigma)$ for different values of σ and $J = J_0$ in Figure 2. Since $\mathcal{D}_J(\sigma)$ is not a constant and depends itself on σ, we note that this particular relationship between $\mathcal{C}_J(\sigma)$ and σ is not a simple power law unlike Eq. (4.1). Instead, our derivation in Eq. (A.4) shows that $\mathcal{C}_J(\sigma)$ is given by a sum of two different power laws in σ. Similarly to $\mathcal{D}_J$, we also observe that the compute utilized is a *predictable* function of σ: we are able to accurately estimate the compute at larger values of σ using the relationship in Eq. (4.2).

Scaling Observation 2: Compute Requirements

The compute $\mathcal{C}_J$ to attain a given return target J_0 increases as a predictable function of the UTD ratio σ, and is a sum of two power laws (Eq. (4.2)).

We observe that both required compute and data are controlled by the UTD ratio σ, which allows us to define a tradeoff between compute and data controlled by σ. We plot this tradeoff as a curve with compute $\mathcal{C}_J(\sigma)$ as x-axis and $\mathcal{D}_J(\sigma)$ as y-axis in Figure 1 (left). Further, as $\mathcal{D}_J(\sigma)$ is a monotonically decreasing function of σ, this curve defines a Pareto frontier: we can move left on the curve to increase data efficiency as the expense of compute and move right to increase compute efficiency at the expense of data. Also interestingly, due to the compute law being a sum of two power laws, in many environments there is a minimum σ after which compute efficiency no longer improves as seen on OAI Gym in Figure 1.

Solving for maximal data efficiency (Problem 3.1, (1)). We can now solve Problem 3.1 in setting **(1)**. our strategy

to address setting **(1)** is to find the largest σ (say $\sigma_{\max}$) that satisfies the compute constraint $\mathcal{C}_J(\sigma) \leq \mathcal{C}_0$, and then plug this $\sigma_{\max}$ into $\mathcal{D}_J(\sigma)$ to obtain the data estimate. This approach enables us to express $\mathcal{D}_J$ directly as a function of the available compute $\mathcal{C}_0$, as we calculate in Eq. (4.2). This can be visualized as finding the value $\mathcal{D}_J$ corresponding to some value $\mathcal{C}_0$ on the Pareto frontier (Figure 1, left)

Solving for maximal compute efficiency (Problem 3.1, (2)). Likewise, the solution in **(2)** can be obtained by finding the smallest value of σ in the range that satisfies the data constraint $\mathcal{D}_J(\sigma) \leq \mathcal{D}_0$, and computing the corresponding value of $\mathcal{C}_J(\sigma)$. This can similarly be visualized on the Pareto frontier (Figure 1, left). We summarize our observations in terms of the following takeaway.

Solving 3.1: The Compute-Data Pareto frontier

The UTD ratio σ defines a Pareto frontier between data and compute requirements, and estimating this frontier yields predictable solutions to resource optimization problems in settings **(1)** and **(2)**. Theoretically, the optimal $\mathcal{D}_J^*$ for an available compute budget $\mathcal{C}_0$ is:

$$\mathcal{D}_J^*(\mathcal{C}_0) \approx \mathcal{C}_0 \cdot (10 \cdot N \cdot B(\sigma^*) \cdot \sigma^*)^{-1}. \quad (4.3)$$

The optimal $\mathcal{C}_J$ for a given data budget $\mathcal{D}_0$ is:

$$\mathcal{C}_J^*(\mathcal{D}_0) \approx 10 \cdot N \cdot B(\sigma^*) \cdot \sigma^* \cdot \mathcal{D}_0. \quad (4.4)$$

Above, σ^* denotes the minimizing UTD value. Calculation details are in Appendix A.

Maximize return within a budget (Problem 3.2). Finally, we tackle Problem 3.2 in order to extrapolate from low to high return. Here, we do not want to minimize resources, but rather want to maximize performance within a given total "budget" on data and compute. As discussed in Section 3, we consider budget functions linear in both data and compute, i.e., $\mathcal{F} = \mathcal{C} + \delta \cdot \mathcal{D}$, for a given constant δ. Our estimated Pareto frontier in Eq. (4.4) will enable answering this question. To do so, we turn to directly predicting a good UTD value σ^*. This UTD value is one that not only leads to maximal performance, but also stays within the total resource budget $\mathcal{F}_0$. Once the UTD value has been identified, it prescribes a concrete way to partition the total resource budget into good data and compute requirements using the solutions to Problem 3.1.

We plot the data-compute Pareto frontiers for multiple values of J_0 in Figure 3 and in Figure 1 (right), and find that these curves move diagonally to the top-right for larger J_0. Intersecting these curves with iso-budget frontiers over $\mathcal{D}$ and $\mathcal{C}$ prescribed by the budget function, gives us the largest possible J_0 for which there is still a $(\mathcal{D}, \mathcal{C})$ pair that just falls just within the budget $\mathcal{F}_0$ but attains performance J_0 (see Figure 3 for a worked out version of this procedure). Since both $\mathcal{D}$ and $\mathcal{C}$ are explained by σ, we can associate

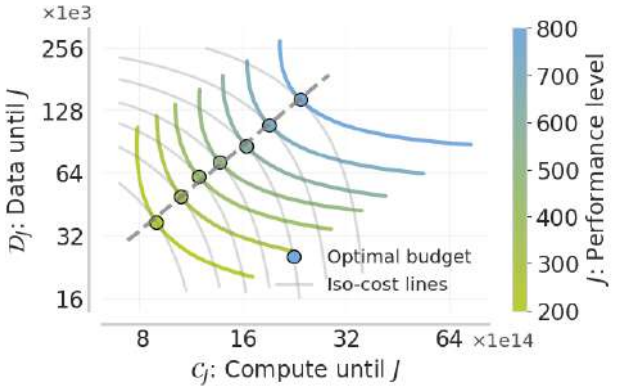


Figure 3: Visualization of the solution to Problem 3.2. Several Pareto frontiers (Figure 1, left) are shown, together with lines of iso-budget $\mathcal{F}$, which define optimal budget points $(\mathcal{D}^*, \mathcal{C}^*)$. Corresponding optimal UTD ratios σ^* are a predictable function of the budgets $\mathcal{F}_0$, trend line shown dashed.

this point with a given σ value. Hence, we can estimate the best value of $\sigma^*(\mathcal{F}_0)$ for a given budget threshold $\mathcal{F}_0$. Concretely, we observe a power law between $\sigma(\mathcal{F}_0)$ and $\mathcal{F}_0$, with constants β_σ and α_σ.

$$\sigma^*(\mathcal{F}_0) \approx \left(\frac{\beta_\sigma}{\mathcal{F}_0}\right)^{\alpha_\sigma}. \quad (4.5)$$

Solving 3.2: Maximize Return Given a Budget

The best UTD value σ that leads to maximal J is a predictable function of the budget $\mathcal{F}_0$ over data and compute, this relationship follows a power law, and also extrapolates to large budgets.

This relationship produces the optimal σ, and as a result, the optimal data and compute allocations to reliably attain maximum performance. As shown in Figure 1, estimating this law from low-budget experiments is sufficient for predicting good σ values for large budget runs. These predicted $\sigma^*(\mathcal{F}_0)$ values extrapolate reliably to budgets outside the range used to fit this law (as shown by $\times$ in Figure 1). This concludes an exposition of our main results.

4.2. Fitting Relationships Between (B, η, σ)

To arrive at these scaling law fits above, we had to set hyperparameters B and η, which we empirically observed to be important. We fit these hyperparameters as a function of σ, the only variable appearing in many of the scaling relationships discussed above. In this section, we will now describe how to estimate good values of B and η in terms of σ. Our analysis here relies crucially on the behavior of TD-learning that is distinct from supervised learning, where the UTD ratio σ does not exist.

To understand relationships between batch size B, learning rate η, and the UTD ratio σ, we ran an extensive grid search. We first attempted to explain the relationship between the B

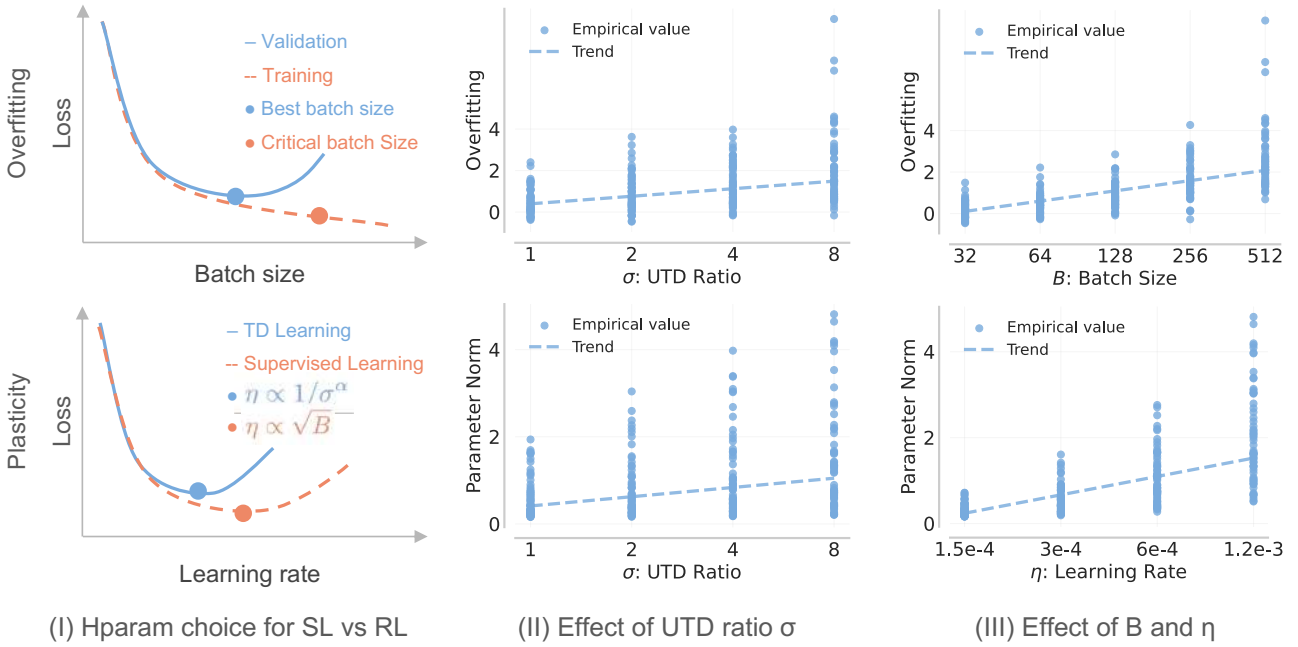


Figure 4: Hyperparameter effects in supervised learning and TD learning on DMC. ***Top:*** Overfitting increases with UTD while batch size can be used to counteract it. ***Bottom:*** Higher UTD leads to poor training dynamics and plasticity loss (D'Oro et al., 2023). Lower learning rates can be used to counteract it. While these relationships are not perfectly predictable, we use them to inform our design choices.

and η values that attain the highest data efficiency (denoted B^*, η^*) using the standard heuristic in supervised learning: *when the batch size is smaller than the critical batch size, B and η are inversely correlated with each other* (McCandlish et al., 2018). However, as shown in Figure 5 (right), we find that without including the UTD ratio σ, best B^* and η^* exhibit very weak correlation. Further, the critical batch size (McCandlish et al., 2018) does not correlate with empirically best batch size as we show in Appendix F. Instead, surprisingly, we observe a strong correlation between B^* and σ, as well as η^* and σ, respectively. Since B^* and η^* exhibit near zero correlation among themselves, we can simply omit their dependency and opt for modeling them independently as a function of the UTD ratio, σ. We conceptually explain relationships between B^* and σ, and η^* and σ below and show that models developed from this understanding enable us to reliably predict good values of B and η, allowing us to fully answer Problem 3.1.

Predicting best choice of B in terms of σ. Our proposed functional form for the best batch size B^* takes the form of a power law in σ, which we also empirically validate in Figure 5 (left). We posit this form because, intuitively, large batch sizes increase the risk of overfitting because they lead to repetitive training on a fixed set of data. Furthermore, a small training loss on the distribution of data in the buffer does not necessarily reflect the behavior policy distribution of a learning agent (Levine et al., 2020). This means that minimizing the training loss to a large extent can result in poor test performance $J(\pi)$, as also seen by prior work (Li et al., 2023a; Nauman et al., 2024a). One way to counteract this form of "overfitting" from a high UTD value σ is to instead reduce the batch size in the run so that the training process sees a given sample fewer times. In fact, for a fixed UTD value σ, we empirically validate this hypothesis that a lower B leads to substantially reduced overfitting on several tasks in Figure 4. Hence, we post an inverse relationship between the best batch size B^* and the UTD value σ. We show in Figure 5 that indeed this inverse relationship can be estimated well by a power law, given formally as:

$$B^*(\sigma) \approx \left(\frac{\beta_B}{\sigma}\right)^{\alpha_B}. \tag{4.6}$$

Predicting best choice of learning rate η as a function of σ. Next we turn to understanding the relationship between η and σ. We start from a simple observation: a very large σ typically leads to worse performance not only due to overfitting but also due to plasticity loss (Kumar et al., 2021; D'Oro et al., 2023; Lyle et al., 2023), defined broadly as the inability of the value network to fit TD targets appearing later in training. Prior work states that plasticity loss is inherently related to the number of gradient steps performed and claims that larger norms of parameters of the Q-network are indicative of plasticity loss (D'Oro et al., 2023; Lyle et al., 2023). We would expect a larger learning rate to make higher magnitude updates against the same TD target, and hence move parameters to a state that suffers from difficulty in fitting subsequent targets (Dabney et al., 2021; Lee et al., 2024). As shown in Figure 4, the parameter norm indeed increases with a high learning rate. Therefore, given a UTD

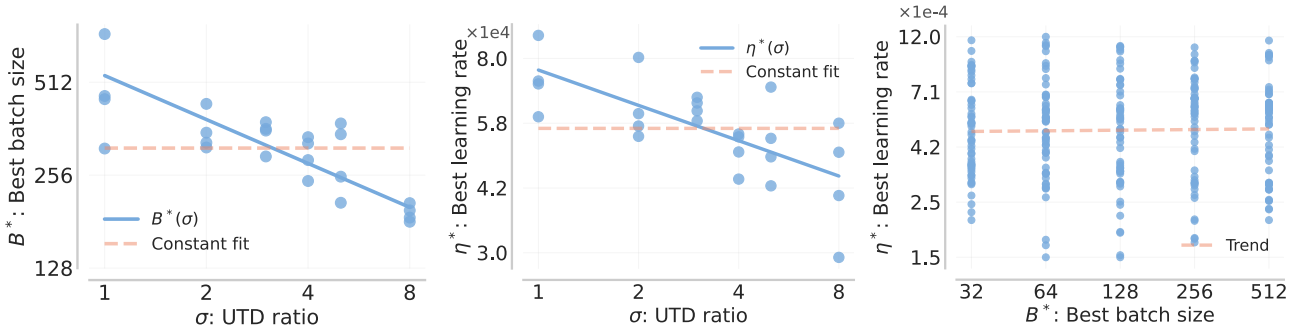


Figure 5: ***Left, middle***: Fitting the best learning rate η^* and batch size B^* given UTD σ on DMC. Modeling the dependency on σ is crucial to obtain good hyperparameters, whereas using constant B, η as is commonly done leads too poor extrapolation. ***Right:*** the best learning rate and batch size are not significantly correlated, a major difference from supervised learning.

value σ, we hypothesize that the best choice of learning rate, $\eta^*(\sigma)$ for a given performance should scale inversely in σ. Empirically we observe that this is indeed the case (Figure 5 (middle)), and we model this relationship:

$$\eta^*(\sigma) \approx \left(\frac{\beta_\eta}{\sigma}\right)^{\alpha_\eta}. \tag{4.7}$$

Scaling Observation 3: Hyperparameter Selection

The best choices for the batch size and learning rate are predictable functions of the UTD σ, and both of these relationships follow a power law.

4.3. Empirical Workflow

Fitting Empirical Relationships

1. Run a sweep for batch size B and learning rate η for several values of UTD σ. Since the batch size and learning rate are independent for the best σ, we can run these sweeps independently.
2. Estimate empirically the best of batch size $\tilde{B}$ and learning rate $\tilde{\eta}$, with statistical bootstrapping.
3. Fit $B^*(\sigma)$ and $\eta^*(\sigma)$ on $\tilde{B}, \tilde{\eta}$ according to Equations (4.6) and (4.7).
4. Using the found fits $B^*(\sigma), \eta^*(\sigma)$, run different values of σ that cover a range spanning an order of magnitude; we use $16\times$, i.e., $\sigma_{\text{max}}/\sigma_{\text{min}} > 16$.
5. Fit $\mathcal{D}_J(\sigma)$ according to Eq. (4.1).
6. Using fits of $\mathcal{D}_J(\sigma)$ for different values of J_0, fit $\sigma^*(\mathcal{F}_0)$ according to Eq. (4.5).
7. Optimal hyperparameters can now be extrapolated to larger data, larger compute, or larger budget settings according to Problem 3.1.

Having presented solutions to Problems 3.1 and 3.2, we now present the workflow we utilize to estimate these empirical fits. Further details are in Section 5 and Appendix D. This workflow can serve as a useful skeletion for scaling law studies with other value-based algorithms as well.

4.4. Evaluating Extrapolation

Evaluating budget extrapolation. Results on all environments are shown in Figure 1 (middle). We estimate several Pareto frontiers corresponding to points with equal changes in budget. We perform the $\sigma^*(\mathcal{F}_0)$ fit, while holding out two largest budgets. The quality of our fit for these two extrapolated budgets can be seen in the figure.

Evaluating Pareto frontier extrapolation. Results on OpenAI Gym are shown in Figure 6. We fit the data efficiency equation $\mathcal{D}_J(\sigma)$ Eq. (4.1) while holding out either two UTD values σ with largest data requirement (left) or two σ values with largest compute requirement (right). The quality of our fit for these two extrapolated σ values can be seen in the figure.

Hyperparameter fit extrapolation. Results on OpenAI Gym are shown in Figure 6 (right). We plot the data efficiency fit when using hyperparameters according to our found dependency $B^*(\sigma), \eta^*(\sigma)$ (shown in red). These fits are estimated from $\sigma = 1, \ldots, 8$ and extrapolated to $\sigma = 0.5$. We compare the typical approach of tuning hyperparameters in online RL, where hyperparameters are tuned for one setting of $\sigma = 2$ and this setting is used for all UTD values (shown in blue). We see that our proposed hyperparameter fits improve results for values other than $\sigma = 2$. Further, this improvement is larger for larger values of σ, showing that accounting for hyperparameter dependency is critical.

5. Experimental Details

Experimental Setup We focus on 12 tasks from 3 domains in our study. On **OpenAI Gym** (Brockman et al., 2016), we use Soft Actor Critic, a commonly used TD-learning algorithm (Haarnoja et al., 2018). We first run a sweep on 5 values of η, then a grid of runs with 4 values of σ and 3 values of B, and then use hyperparameter fits

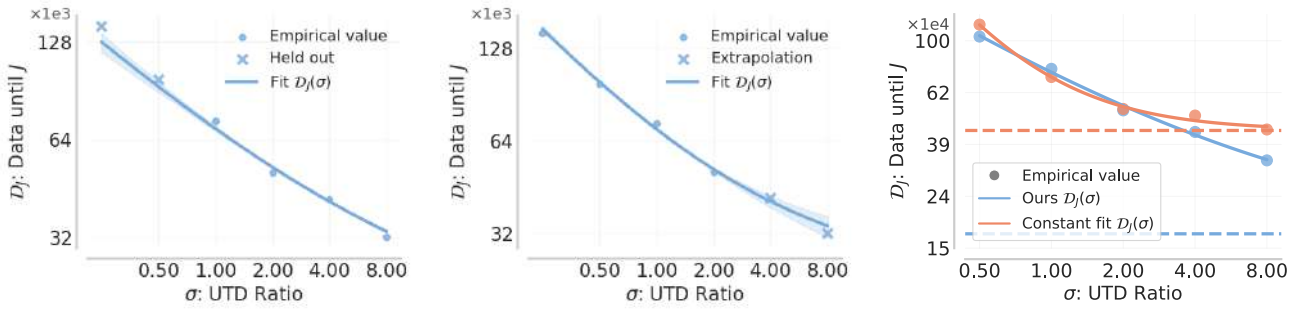


Figure 6: Extrapolation towards unseen values of σ on OpenAI Gym. **Left:** We show Pareto frontier extrapolation towards higher data regime. **Middle:** We show Pareto frontier extrapolation towards higher compute regime. **Right:** We compare the best-performing hyperparameters (red) for $\sigma = 2$ to hyperparameters predicted via our proposed workflow (blue).

to run 2 more value of σ with 8 seeds per task. To test our approach with larger models, we use **DMC** (Tassa et al., 2018), where, we utilize the state-of-the-art Bigger, Regularized, Optimistic (BRO) algorithm (Nauman et al., 2024b) that uses a larger and more modern architecture. We first run 5 values of B, 4 values of η, and 4 σ; and then use hyperparameters fits to run 2 more values of σ, with 10 seeds per task. Finally, we test our approach with more data on **IsaacGym** (Makoviychuk et al., 2021), where we use the Parallel Q-Learning (PQL) algorithm (Li et al., 2023b), which was designed to leverage massively parallel simulation like Isaac Gym that can quickly produce billions of environment samples. Because of computational expense, we only run one IsaacGym task. We first run 4 values of σ, 3 values of η, as well as 5 values of B, with 5 seeds per task, after which we run a second round of grid search with 7 values of σ. Further details are in Appendices B and D and Table 3.

Fitting Functional Forms for Scaling Laws We approximate Eq. (4.1) via brute-force search followed by LBFG-S with a log-MSE loss following (Hoffmann et al., 2023). For Equations (4.6) and (4.7), we fit a line in log space using least squares regression following Kaplan et al. (2020). In our experiments, we run a single fit that is shared across different tasks in a given benchmark. Specifically, we share the slope α_B, α_η and use task-specific intercepts $\sigma_B^{\text{env}}, \sigma_\eta^{\text{env}}$ (as defined in Equations (4.6) and (4.7)) to be different for separate tasks. This technique is standard in ordinary least squares modeling and is referred to as fixed effect regression (Bishop and Nasrabadi, 2006). Sharing this slope serves the goal of variance reduction, which can be important if the granularity of the grid search over various hyperparameters run is coarse. More details are in Appendices B and D.

6. Related Work

Scaling laws and predictability. Prior work has studied scaling laws in the context of supervised learning (Kaplan et al., 2020; Hoffmann et al., 2023), primarily to predict the effect of model size and training data on validation loss, while marginalizing out hyperparameters like batch size (McCandlish et al., 2018) and learning rate (Kaplan et al., 2020). There are several extensions of such scaling laws for language models, such as laws for settings with data repetition (Muennighoff et al., 2023) or mixture-of-experts (Ludziejewski et al., 2024), but most focus on cross-entropy loss, with an exception of Gadre et al. (2024), which focuses on downstream metrics. While scaling laws have guided supervised learning experiments, little work explores this for RL. The closest works are: Hilton et al. (2023) which fits power laws for on-policy RL methods using model size and the number of environment steps; Springenberg et al. (2024) who study model size scaling for offline RL; Jones (2021) which studies the scaling of AlphaZero on board games of increasing complexity; and Gao et al. (2023) which studies reward model overoptimization in RLHF. In contrast, we are the first ones to study predictability off-policy value-based RL methods that are trained via TD-learning. Not only do off-policy methods exhibit training dynamics distinct from supervised learning and on-policy methods (Kumar et al., 2022; Lyle et al., 2023), but we show that this distinction also results in a different functional form for scaling law altogether. We also note that while Hilton et al. (2023) use minimal compute, i.e., $\mathcal{C}_J$ in our notation as a metric of performance, our analysis goes further in several respects: **(1)** we also study the tradeoff between data and compute (Figure 1), **(2)** we can predict the algorithm configuration for best performance (Problem 3.1); **(3)** we study many budget functions ($\mathcal{C} + \delta \cdot \mathcal{D}$ can be any affine function).

Methods for large-scale deep RL. Recent work has scaled deep RL across three axes: model size (Kumar et al., 2023; Schwarzer et al., 2023; Nauman et al., 2024b), data (Kumar et al., 2023; Gallici et al., 2024; Singla et al., 2024), and UTD (Chen et al., 2020; D'Oro et al., 2023). Naïve scaling of model size or UTD often degrades performance or causes divergence (Nikishin et al., 2022; Schwarzer et al., 2023), mitigated by classification losses (Kumar et al., 2023), layer normalization (Nauman et al., 2024a), or feature normaliza-

tion (Kumar et al., 2022). In our work, we use scaled network architectures from Nauman et al. (2024b) (Section 5). In on-policy RL, prior works focus on effective learning from parallelized data streams in a simulator or a world model (Mnih et al., 2016; Silver et al., 2016; Schrittwieser et al., 2020). Follow-up works like IMPALA (Espeholt et al., 2018) and SAPG (Singla et al., 2024) use a centralized learner that collects experience from distributed workers with importance sampling updates. These works differ substantially from our study as we focus exclusively on value-based off-policy RL algorithms that use TD-learning and not on-policy methods. In value-based RL, prior work on data scaling focuses on offline (Yu et al., 2022; Kumar et al., 2023; Park et al., 2024) and multi-task RL (Hafner et al., 2023). In contrast, we study online RL and fit scaling laws to answer resource optimization questions.

7. Discussion, Limitations, and Future Work

In this paper, we show that value-based deep RL algorithms scale predictably. We establish relationships between good values of hyperparameters of value-based RL. We then establish a relationship between required data and required compute for a certain performance. Finally, this allows us to determine an optimal allocation of resources to either data and compute. Although only estimated from small-scale runs, our empirical models reliably *extrapolate* to large compute, data, budget, or performance regimes. Despite folk wisdom to the contrary, we show it is possible to predict behavior of value-based off-policy RL algorithms at larger scale using small-scale experiments.

At the same time, this first study also presents a number of open questions and challenges:

1. While simple power law models work well, an open question remains as to whether such laws are theoretically grounded, and whether there are better and more refined functional forms.
2. Our study only focused on three hyperparameters (B, η, and σ). We do not focus on optimal tradeoff between model size and UTD, which is important for compute scaling. For data efficient RL, it is important to analyze the dependency of weight decay and weight reset frequency on UTD, which are typical tricks employed by many of the most performant methods in literature.
3. While we focus on online RL, it is important to study scaling of offline-to-online and offline RL, which will allow direct applications of scaling law findings to large model training.
4. Finally, while we study relatively small models, future work will focus on verifying our results with larger model scales, larger scale tasks, study the effect of modern architectures, and cover a larger range of compute scales spanning multiple orders of magnitude.

Our work is only one step in studying scaling laws for value-based RL methods. Further research has the potential to improve our understanding of value-based RL at scale, provide researchers with tools to focus innovation on more important components, and eventually provide guidelines towards scaling value-based RL similarly to scaling enjoyed by other modern deep learning approaches.

Acknowledgements

We would like to thank Zhang-Wei Hong, Amrith Setlur, Rishabh Agarwal, Seohong Park, and Max Simchowitz for feedback on an earlier version of this paper. We would like to thank Andrea Zanette, Seohong Park, Kyle Stachowicz, and Qiyang Li for informative discussions. This research was supported by ONR under N00014-24-12206, N00014-22-1-2773, and ONR DURIP grant, with compute support from the Berkeley Research Compute, Polish high-performance computing infrastructure, PLGrid (HPC Center: ACK Cyfronet AGH), that provided computational resources and support under grant no. PLG/2024/017817. Pieter Abbeel holds concurrent appointments as a Professor at UC Berkeley and as an Amazon Scholar. This work was done at UC Berkeley and CMU, and is not associated with Amazon.

Impact Statement

This paper aims to contribute to the advancement of reinforcement learning. While our work may have various societal implications, none warrant specific emphasis here.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint*, 2023.

Richard E Barlow and Hugh D Brunk. The isotonic regression problem and its dual. *Journal of the American Statistical Association*, 1972.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern Recognition and Machine Learning*. Springer, 2006.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024.

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

Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2021.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *International Conference on Learning Representations*, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint*, 2024.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. *International Conference on Machine Learning*, 2018.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, et al. Language models scale reliably with over-training and on downstream tasks. *arXiv preprint*, 2024.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. *arXiv preprint*, 2024.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint*, 2023.

Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning. *arXiv preprint*, 2023.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *Advances in Neural Information Processing Systems*, 2023.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

Andy L. Jones. Scaling scaling laws with board games, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint*, 2020.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint*, 2014.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. DR3: Value-based deep reinforcement learning requires explicit regularization. *International Conference on Learning Representations*, 2022.

Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline Q-learning on diverse multi-task data both scales and generalizes. In *International Conference on Learning Representations*, 2023.

Hojoon Lee, Hanseul Cho, Hyunseung Kim, Daehoon Gwak, Joonkee Kim, Jaegul Choo, Se-Young Yun, and Chulhee Yun. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 2024.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint*, 2020.

Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. In *International Conference on Learning Representations*, 2023a.

Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel Q-learning: Scaling off-policy reinforcement learning under massively parallel simulation. In *International Conference on Machine Learning*, 2023b.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. Scaling laws for fine-grained mixture of experts. In *International Conference on Machine Learning*, 2024.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, 2023.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac Gym: High performance GPU-based physics simulation for robot learning. *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 2023.

Michal Nauman, Michał Bortkiewicz, Piotr Miłoś, Tomasz Trzcinski, Mateusz Ostaszewski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: The bitter lesson of reinforcement learning. In *International Conference on Machine Learning*, 2024a.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: Scaling for compute and sample-efficient continuous control. *Advances in Neural Information Processing Systems*, 2024b.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, 2022.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline RL? *Advances in Neural Information Processing Systems*, 2024.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint*, 2022.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering Atari, Go, chess and Shogi by planning with a learned model. *Nature*, 2020.

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

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In *International Conference on Machine Learning*, 2023.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

Jayesh Singla, Ananye Agarwal, and Deepak Pathak. SAPG: Split and aggregate policy gradients. *International Conference on Machine Learning*, 2024.

Jost Tobias Springenberg, Abbas Abdolmaleki, Jingwei Zhang, Oliver Groth, Michael Bloesch, Thomas Lampe, Philemon Brakel, Sarah Bechtle, Steven Kapturowski, Roland Hafner, et al. Offline actor-critic reinforcement learning scales to large models. *International Conference on Machine Learning*, 2024.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. DeepMind control suite. *arXiv preprint*, 2018.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al.

Gemini: A family of highly capable multimodal models. *arXiv preprint*, 2023.

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

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 2020.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 2021.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Chelsea Finn, and Sergey Levine. How to leverage unlabeled data in offline reinforcement learning. In *International Conference on Machine Learning*, 2022.

Appendices

A. Additional details on derivations

FLOPs calculation. Recall that FLOPs per forward and backward passes are equal to $\mathcal{C}_J^{\text{forward}}(\sigma) \approx 2 \cdot N \cdot B(\sigma) \cdot \sigma \cdot \mathcal{D}_J(\sigma)$ and $\mathcal{C}_J^{\text{backward}}(\sigma) \approx 4 \cdot N \cdot B(\sigma) \cdot \sigma \cdot \mathcal{D}_J(\sigma)$, with σ denoting the number of gradient steps per environment steps. Q-learning methods used in our study use MLP and ResNet architectures, which are well modeled with this approximation. Assuming same size for actor and critic as an approximation, a training iteration of the critic requires three forward passes and one backward pass, totaling $\mathcal{C}_J^{\text{critic}}(\sigma) \approx 10 \cdot N \cdot B(\sigma) \cdot \sigma \cdot \mathcal{D}_J(\sigma)$. A training iteration of the actor requires two forward and two backward passes, totaling $\mathcal{C}_J^{\text{actor}}(\sigma) \approx 12 \cdot N \cdot B(\sigma) \cdot \sigma \cdot \mathcal{D}_J(\sigma)$. Here we follow the standard practice of updating the actor every time a new data point collected, while the critic is updated according to the UTD ratio σ. Since we expect the critic to be updated more then the actor. As such, in this study we assume

$$\mathcal{C}_J(\sigma) \approx \mathcal{C}_J^{\text{critic}}(\sigma) \approx 10 \cdot N \cdot B(\sigma) \cdot \sigma \cdot \mathcal{D}_J(\sigma). \tag{A.1}$$

Compute and sample efficiency. Following Eq. (4.1), the number of data points required to achieve performance J is equal to:

$$\mathcal{D}_J(\sigma) \approx \mathcal{D}_J^{\min} + \left(\frac{\beta_J}{\sigma}\right)^{\alpha_J} \tag{A.2}$$

Given the expressions for required data points, practical batch size, and FLOPs Equations (4.1), (4.6) and (A.1), we can now derive the expression for compute required to reach a particular performance expressed in terms of σ. First, note that the number of parameter updates is

$$\sigma \cdot \mathcal{D}_J(\sigma) \approx \sigma \cdot \mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J}}{\sigma^{\alpha_J - 1}} \tag{A.3}$$

Combining above, Eq. (4.6) with Eq. (A.1) yields:

$$\begin{aligned} \mathcal{C}_J(\sigma) &\approx 10 \cdot N \cdot B(\sigma) \cdot \left(\sigma \cdot \mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J}}{\sigma^{\alpha_J - 1}}\right) \\ &\approx 10 \cdot N \cdot \left(\frac{\beta_B}{\sigma}\right)^{\alpha_B} \cdot \left(\sigma \cdot \mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J}}{\sigma^{\alpha_J - 1}}\right) \\ &\approx 10 \cdot N \cdot \left(\frac{\mathcal{D}_J^{\min} \cdot \beta_B^{\alpha_B}}{\sigma^{\alpha_B - 1}} + \frac{\beta_J^{\alpha_J} \cdot \beta_B^{\alpha_B}}{\sigma^{\alpha_J + \alpha_B - 1}}\right). \end{aligned} \tag{A.4}$$

We observe that the resulting expression is a sum of two power laws. In practice, one of the power laws will dominate the expression and a simple mental model is that compute increases with UTD as a power law with a coefficient < 1 (see Figure 2).

Maximal compute efficiency. Here, we solve the compute optimization problem presented in Section 3. We write the problem:

$$(B^*, \eta^*, \sigma^*) := \arg\min_{(B,\eta,\sigma)} \quad \mathcal{C} \qquad \text{s.t.} \quad J\left(\pi_{\text{Alg}}(B, \eta, \sigma)\right) \geq J_0 \quad \wedge \quad \mathcal{D} \leq D_0. \tag{A.5}$$

Firstly, we formulate the Lagrangian $\mathcal{L}$:

$$\begin{aligned}\mathcal{L}(\sigma, \lambda) &= \mathcal{C}_J(\sigma) + \lambda \cdot (\mathcal{D}_J(\sigma) - D_0) \\ &\approx 10 \cdot N \cdot B(\sigma) \cdot \left(\sigma \cdot \mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J}}{\sigma^{\alpha_J - 1}}\right) + \lambda \cdot \left(\mathcal{D}_J^{\min} + \left(\frac{\beta_J}{\sigma}\right)^{\alpha_J} - \mathcal{D}_0\right)\end{aligned} \tag{A.6}$$

Here, the constrained with respect to performance J_0 is upheld through the use of $\mathcal{C}_J(\sigma)$ and $\mathcal{D}_J(\sigma)$ which are defined such that $J = J_0$. We proceed with calculating the derivative with respect to λ to find the minimal σ that is able to achieve the desired sample efficiency $\mathcal{D}_J$. We denote such optimal UTD as σ^*:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \mathcal{D}_J^{\min} + \left(\frac{\beta_J}{\sigma}\right)^{\alpha_J} - \mathcal{D}_0 = 0 \quad \Longrightarrow \quad \sigma^* = \frac{-\beta_J}{\left(\mathcal{D}_J^{\min} - \mathcal{D}_0\right)^{1/\alpha_J}} \tag{A.7}$$

Then, we substitute the σ^* into the expression defining compute, as well as use Eq. (4.6):

$$\begin{aligned}\mathcal{C}_J(\sigma^*) &\approx 10 \cdot N \cdot \frac{\beta_B^{\alpha_B}}{\sigma^{\alpha_B - 1}} \cdot \left(\mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J}}{\sigma^{\alpha_J}}\right) \\ &\approx 10 \cdot N \cdot \frac{\beta_B^{\alpha_B}}{(\sigma^*)^{\alpha_B - 1}} \cdot \left(\mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J} \cdot \left(\mathcal{D}_J^{\min} - \mathcal{D}_0\right)}{-\beta_J^{\alpha_J}}\right) \\ &\approx 10 \cdot N \cdot \beta_B^{\alpha_B} \cdot (\sigma^*)^{1 - \alpha_B} \cdot \mathcal{D}_0\end{aligned} \tag{A.8}$$

Maximal sample efficiency. Firstly, we note that we treat $B(\sigma)$ as a constant and do not optimize with respect to it. We start with the problem definition:

$$(B^*, \eta^*, \sigma^*) := \arg \min_{(B, \eta, \sigma)} \quad \mathcal{D} \quad \text{s.t.} \quad J\left(\pi_{\text{Alg}}(B, \eta, \sigma)\right) \geq J_0 \quad \wedge \quad \mathcal{C} \leq C_0. \tag{A.9}$$

Similarly to the maximal compute efficiency problem, we formulate the Lagrangian $\mathcal{L}$:

$$\begin{aligned}\mathcal{L}(\sigma, \lambda) &= \mathcal{D}_J(\sigma) + \lambda \cdot (\mathcal{C}_J(\sigma) - C_0) \\ &\approx \mathcal{D}_J^{\min} + \left(\frac{\beta_J}{\sigma}\right)^{\alpha_J} + \lambda \cdot \left(10 \cdot N \cdot B(\sigma) \cdot \sigma \cdot \left(\mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J}}{\sigma^{\alpha_J}}\right) - \mathcal{C}_0\right)\end{aligned} \tag{A.10}$$

Again, we uphold the constraint with respect to the performance through the use of $\mathcal{D}_J(\sigma)$ and $\mathcal{C}_J(\sigma)$. We calculate the derivative with respect to λ:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 10 \cdot N \cdot B(\sigma) \cdot \sigma \cdot \left(\mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J}}{\sigma^{\alpha_J}}\right) - \mathcal{C}_0 = 0 \quad \Longrightarrow \quad \mathcal{D}_J^{\min} + \frac{\beta_J^{\alpha_J}}{\sigma^{\alpha_J}} = \frac{\mathcal{C}_0}{10 \cdot N \cdot B(\sigma) \cdot \sigma} = \mathcal{D}_J \tag{A.11}$$

Since $\mathcal{D}_J$ is monotonic in σ and does not model impact of B on the sample efficiency, the optimization problem can be solved via Weierstrass extreme value theorem. As such, we find the biggest σ and that fulfills the compute constraint, and find the data requirement for such σ.

B. Experimental details

For our experiments, we use a total of 12 tasks from 3 benchmarks (DeepMind Control (Tunyasuvunakool et al., 2020), Isaac Gym (Makoviychuk et al., 2021), and OpenAI Gym (Brockman et al., 2016)). We list all considered tasks in Table 1.

Table 1: Tasks used in presented experiments.

Domain	Task	Optimal π Returns
DeepMind Control	Cartpole-Swingup	1000
	Cheetah-Run	1000
	Dog-Stand	1000
	Finger-Spin	1000
	Humanoid-Stand	1000
	Quadruped-Walk	1000
	Walker-Walk	1000
Isaac Gym	Franka-Push	0.05
OpenAI Gym	HalfCheetah-v4	8500
	Walker2d-v4	4500
	Ant-v4	6625
	Humanoid-v4	6125

Figure 1. We use all available UTD values for the fits, which is 6 for DMC, 5 for OAI Gym, and 7 for Isaac Gym. Given the dependency of compute and data on UTD, we plot the resulting curve. We average the data efficiencies across all tasks in each domain, as described in Appendix D. For plots on the left, we use $J = 800$.

We calculate compute given the model sizes of $N = 4.92\text{e}6$ for DMC, $N = 1.5\text{e}5$ for OAI Gym, and $N = 2\text{e}6$ following standard implementations of the respective algorithms.

For budget extrapolation, we use tradeoff values δ to mimic the wall-clock time of the algorithm. We use $\delta = 1\text{e}10$ for DMC, $\delta = 5\text{e}9$ for OAI Gym, and $\delta = 1\text{e}4$ for Isaac Gym. We exclude runs affected by resets ($\sigma = 8$) for DMC since the returns right after the reset are lower, which adds noise to the results.

Figure 2. We use the same data as for DMC in Figure 1 (left).

Figure 3. We use the same data as for DMC in Figure 1 (right).

Figure 4. **Left**: we show an illustration that reflects our observed empirical results about the dependencies between hyperparameters.

Right, middle: we investigate the correlations between overfitting, parameter norm of the critic network, and σ. We observed the same relationships on all tasks. Here, to avoid clutter, we plot 3 tasks from DMC benchmark: cheetah-run, dog-stand, and quadruped-walk. To measure overfitting, we compare the TD loss calculated on samples randomly sampled from the buffer (corresponding to *training data*) to TD loss calculated on 16 newest transitions (corresponding to *validation data*) according to:

$$\text{Overfitting} = TD^{\text{training}} - TD^{\text{validation}}. \tag{B.1}$$

We fit the linear curves using ordinary least squares with mean absolute error loss.

Figure 5. In the left and central Figures, we evaluate the B^* and η^* models. For each DMC task, we find the best hyperparameters according to our workflow and procedure described in Section 5 and Appendix D. While the intercepts vary across environments, for simplicity we plot data points and fits from all environments in the same figure by shifting them with the corresponding intercept. In the right Figure, we marginalize over σ and visualize best performing pairs of B and η.

Figure 6. Here, we investigate 4 tasks from OpenAI Gym, listed in Table 1, and compare the extrapolation performance of two hyperparameter sets: the best performing hyperparameters for $\sigma = 1$, found by testing 8 different hyperparameter values listed in Table 3 (we refer to this configuration as *baseline*); and hyperparameters predicted by our proposed models of B^* and η^*. We fit our models using $\sigma \in (1, 2, 4, 8)$, and extrapolate to $\sigma \in (0.5, 16)$. The graph shows the data efficiency with threshold as 700, normalized according to the procedure in Appendix D.

Figure 7. The goal of the left Figure is to visualize the effects of isotropic regression fit on a noisy data. We use the SciPy package (Virtanen et al., 2020) to run the isotropic model. In the right Figure we visualize the process of best hyperparameter selection using bootstrapped confidence intervals. We describe the bootstrapping strategy in Appendix D.

C. Resulting Fits

DMC Refer to Table 5 for environment-specific values.

$$\begin{aligned} \eta^* &= \beta_\eta \cdot \sigma^{-0.26} \\ B^* &= \beta_B \cdot \sigma^{-0.47} \\ \mathcal{D}_J &= \mathcal{D}^{\min} \cdot \left(1 + \left(\frac{\sigma}{0.45}\right)^{-0.74}\right) \\ \sigma^* &= 1.4\text{e}8 \cdot \mathcal{F}_0^{-0.53} \end{aligned} \tag{C.1}$$

OpenAI Gym Refer to Table 5 for environment-specific values.

$$\begin{aligned} \eta^* &= \beta_\eta \sigma^{-0.30} \\ B^* &= \beta_B \sigma^{-0.33} \\ \mathcal{D}_J &= \mathcal{D}^{\min} \cdot \left(1 + \left(\frac{\sigma}{4.02}\right)^{-0.69}\right) \\ \sigma^* &= 1.4\text{e}8 \cdot \mathcal{F}_0^{-0.53} \end{aligned} \tag{C.2}$$

Isaac Gym

$$\begin{aligned} \eta^* &= 8.77 \cdot \left(1 + \left(\frac{\sigma}{2.57\text{e-}3}\right)^{-0.26}\right) \\ B^* &= 38.6 \cdot \left(1 + \left(\frac{\sigma}{1.42\text{e-}2}\right)^{-0.68}\right) \\ \mathcal{D}_J &= 6.8\text{e}7 \cdot \left(1 + \left(\frac{\sigma}{1.88}\right)^{-0.87}\right) \\ \sigma^* &= 11.3 \cdot \mathcal{F}_0^{-0.57} \end{aligned} \tag{C.3}$$

Table 2: Coefficients for DMC and OpenAI Gym fits.

Domain	**Task**	β_η	β_B	$\mathcal{D}^{\min}$
DMC	cartpole-swingup	7.55e-4	538.2	2.4e4
	cheetah-run	6.25e-4	564.9	3.5e5
	finger-spin	8.77e-4	608.2	2.9e4
	humanoid-stand	3.86e-4	451.8	3.8e5
	quadruped-walk	8.46e-4	526.4	6.2e4
	walker-walk	9.38e-4	313.3	3.3e4
OpenAI Gym	Ant-v4	1.35e-4	447.0	2.7e5
	HalfCheetah-v4	1.86e-3	415.4	7.8e4
	Humanoid-v4	1.65e-4	351.6	1.8e5
	Walker2d-v4	7.85e-4	399.1	1.7e5

D. Additional details on the fitting procedure

Preprocessing return values. In order to estimate the fits from our laws, we need to track the data and compute needed by a run to hit a target performance level. Due to stochasticity both in training and and evaluation, naïve measurements of this

Table 3: Tested configurations.

Hyperparameters	DeepMind Control	Isaac Gym	OpenAI Gym
Updates-to-data σ	1, 2, 4, 8	$\frac{1}{1024}, \frac{1}{2048}, \frac{1}{4096}, \frac{1}{8192}, \frac{1}{16384}, \frac{1}{32768}, \frac{1}{65536}$	1, 2, 4, 8
Batch size B	32, 64, 128, 256, 512	512, 1024, 2048, 4096, 8192	128, 256, 512
Learning rate η	15e-5, 3e-4, 6e-4, 12e-3	1e-4, 2e-4, 3e-4	1e-4, 2e-4, 5e-4, 1e-3, 2e-3

point can exhibit high variance. This in turn would result in low-quality fits for $\mathcal{D}_J$ and $\mathcal{C}_J$. Thus, we preprocess the return values before estimating the fits by running isotonic regression (Barlow and Brunk, 1972). Isotonic regression transforms return values to the most aligned monotonic sequence of values that can then be used to estimate $\mathcal{D}_J$. While in general return values can decrease with more training after reaching a target value, and this will result in a large deviation between the isotonic fit and true return values, the proposed isotonic transformation still suffices for us as our goal is to simply fit the *minimum* number of samples or compute needed to attain a target return. As we can still make reliable predictions that extrapolate to larger scales, the downstream impact of this error is clearly not substantial. We also average across random seeds before running isotonic regression to further reduce noise. We normalize the returns for all environments to be between 0 and 1000 (Table 1 lists pre-normalized returns), and reserve the points of 700 and 800 for budget extrapolation in Figure 1.

Uncertainty-adjusted optimal hyperparameters. While averaging across seeds and applying isotonic regression reduces noise, we observe that the granularity of our grid search on learning rate and batch size limits the precision of the resulting hyperparameter fits $\tilde{B}$, $\tilde{\eta}$. Noise due to random seed generation makes hyperparameter selection harder as some hyperparameters that appear empirically optimal might simply be so due to noise. We observe that we can correct for this precision loss by constructing a more precise estimate of $\tilde{B}$, $\tilde{\eta}$ adjusted for this uncertainty. Specifically, we run $K = 100$ bootstrap estimates by sampling n random seeds with replacement out of the original n random seeds, applying isotonic regression, and selecting the optimal hyperparameters $\tilde{B}_k$, $\tilde{\eta}_k$. We then use the mean of this bootstrapped estimate to improve the precision:

$$\begin{aligned} \tilde{B}_{\text{bootstrap}} &= \frac{1}{K} \sum_k \tilde{B}_k \\ \tilde{\eta}_{\text{bootstrap}} &= \frac{1}{K} \sum_k \tilde{\eta}_k \end{aligned} \tag{D.1}$$

We have also experimented with more precise laws for learning rate and batchsize by adding an additive offset. In this case, we follow Hoffmann et al. (2023) and fit the data using brute-force search followed by LBFG-S. We use MSE in log space as the error: $\text{MSE}_{\log}(a, b) = (\log a - \log b)^2$.

$$B^*(\sigma) \approx B_{\min} + \frac{\sigma_B}{\sigma^{\alpha_B}} \tag{D.2}$$

$$\eta^*(\sigma) \approx \eta_{\min} + \frac{\sigma_\eta}{\sigma^{\alpha_\eta}}. \tag{D.3}$$

However, we found that this more complex fit did not validate the decrease of degrees of freedom given a limited sweep range, resulting in accuracy of extrapolation.

Independence of B and η. Whereas the optimal choice of B and η is often intertwined as UTD changes, we observe in our experiments that the correlation between them is relatively low (Figure 5). If we ran a cross-product grid search with hyperparameter space $\{B_1, \ldots, B_{n_B}\} \times \{\eta_1, \ldots, \eta_{n_\eta}\}$, we can use this fact to further improve the results by averaging the estimate $\tilde{B}$ over different values of η. That is, we produce the estimate $\tilde{B}^{[\eta=\eta_i]}$ (respectively $\tilde{\eta}^{[B=B_i]}$) by only looking at the runs where $\eta = \eta_i$, and averaging such estimates.

$$\begin{aligned} \tilde{B}_{\text{mean}} &= \frac{1}{n_\eta} \sum_i \tilde{B}^{[\eta=\eta_i]} \\ \tilde{\eta}_{\text{mean}} &= \frac{1}{n_B} \sum_i \tilde{\eta}^{[B=B_i]} \end{aligned} \tag{D.4}$$

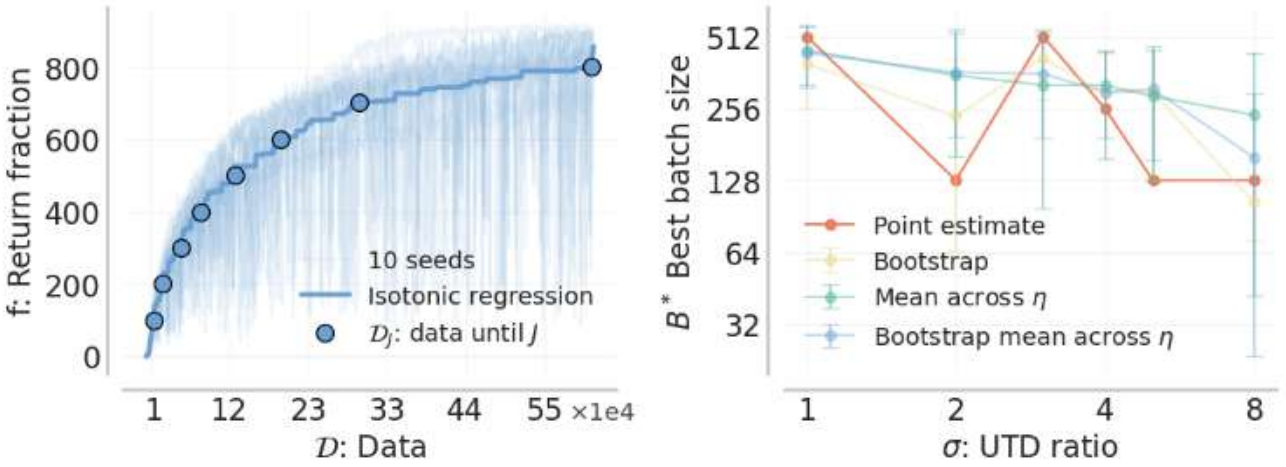


Figure 7: ***Left:*** Determining performance via isotonic regression on DMC. ***Right:*** improving hyperparameter selection with uncertainty adjustment on DMC. Further details are in Appendix D.

Data efficiency. We fit data efficiency of the runs with our found practical hyperparameters B^*, η^* according to Eq. (4.1). We follow Hoffmann et al. (2023) and fit the data using brute-force search followed by LBFG-S. We use MSE in log space as the error: $\text{MSE}_{\log}(a, b) = (\log a - \log b)^2$.

In DeepMind Control Suite, we would like to share the data efficiency fit across different environments env. We normalize the data efficiency $\mathcal{D}$ by the intra-environment median data efficiency medians $\mathcal{D}^{\text{env}}_{\text{med}} = \text{median}\{\mathcal{D}^{\text{env}}_{[\sigma=\sigma_i]} | i = 1..n_\sigma\}$. For interpretability, we further re-normalize D with the overall median $\mathcal{D}_{\text{med}}$: $\mathcal{D}_{\text{norm}} = \mathcal{D} \cdot \mathcal{D}_{\text{med}} / \mathcal{D}^{\text{env}}_{\text{med}}$. We will need to express the data efficiency law alternatively as:

$$D_J(\sigma) \approx \mathcal{D}_J^{\min} \left(1 + \left(\frac{\beta_J}{\sigma}\right)^{\alpha_J}\right). \tag{D.5}$$

This is equivalent to Eq. (4.1) because the coefficient β_J absorbs $\mathcal{D}_J^{\min}$. However, this expression makes explicit an overall multiplicative offset[1] $\mathcal{D}_J^{\min}$. Our median normalization is then equivalent to fitting per-environment coefficients $\mathcal{D}_J^{\min}$, following our procedure for environment-shared hyperparameter fits. However, we further improve robustness by fixing the per-environment coefficients to be the median data efficiency and do not require fitting them.

E. Additional experimental results

Table 4: Correlation coefficients for empirically optimal DMC hyperparameters.

	R
learning rate and batch size	0.04
batch size and UTD	-0.40
learning rate and UTD	-0.46

Table 5: Error of Pareto frontier extrapolation.

	R
toward larger compute	7.8%
toward larger data	10.6%

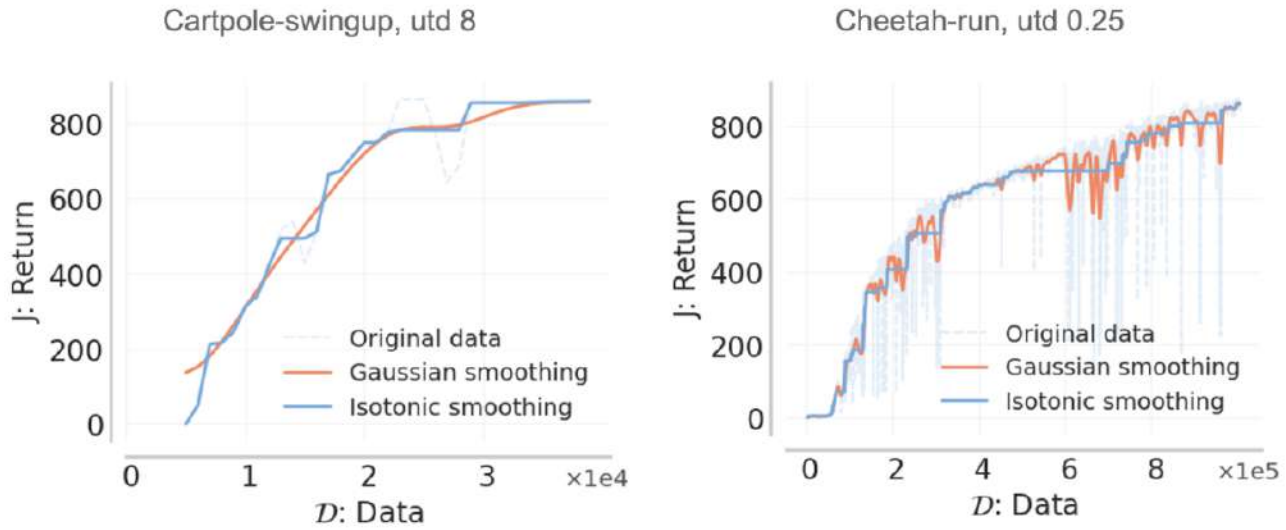


Figure 8: Another example of isotonic regression. Using gaussian smoothing with variance $\sigma = 3$ leads to both oversmoothing (right) and undersmoothing (left).

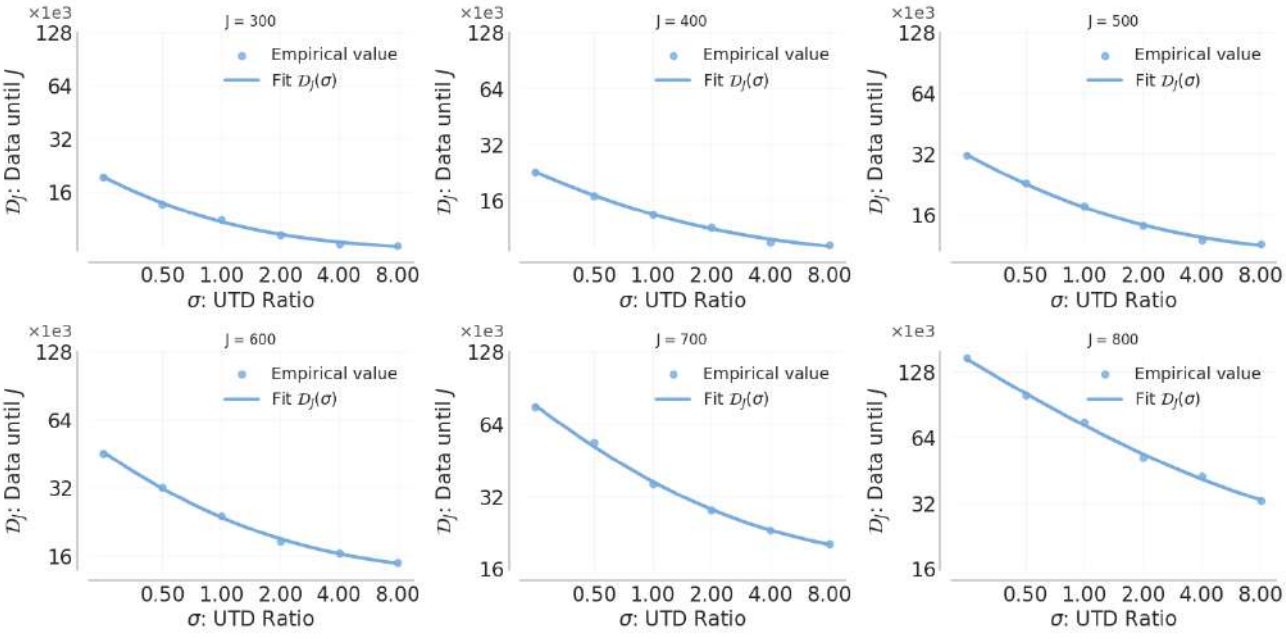


Figure 9: Additional fit results on OpenAI gym for different values of J.

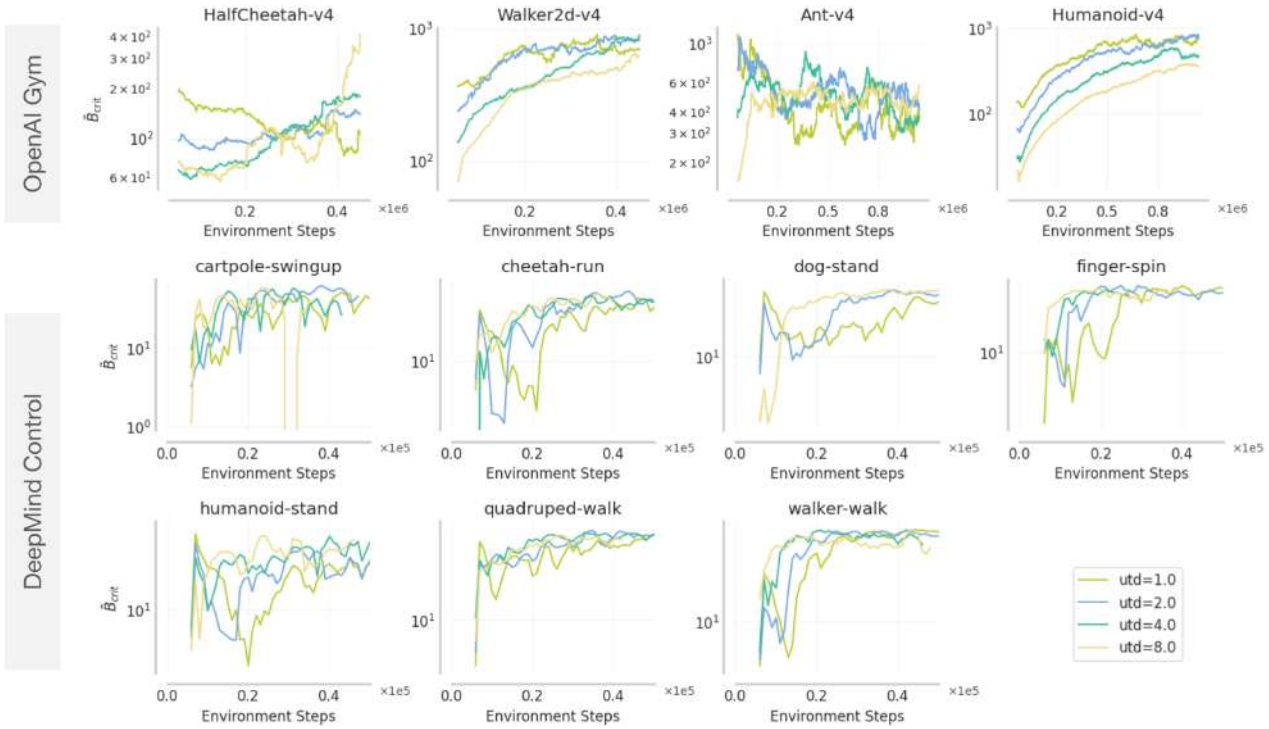


Figure 10: An approximation of the critical batch size over training. Further details are in Appendix F.

F. Critical batch size analysis

Previous work has argued that there is a critical batch size B_{crit} for neural network training in image classification, generative modeling, and reinforcement learning with policy gradient algorithms (McCandlish et al., 2018) — a transition point at which increasing the batch size begins to yield diminishing returns. We follow this work and compute an estimate of the gradient noise scale $B_{\text{noise}} \approx B_{\text{crit}}$ according to the following procedure: throughout training, we compute the gradient norm $|G_B|$ of the critic network for batches of size $B = B_{\text{small}} := 64$ and $B = B_{\text{big}} := 1024$. Then, we evaluate

$$|\mathcal{G}|^2 := \frac{1}{B_{\text{big}} - B_{\text{small}}} \left(B_{\text{big}} |G_{B_{\text{big}}}|^2 - B_{\text{small}} |G_{B_{\text{small}}}|^2 \right)$$

$$\mathcal{S} := \frac{1}{1/B_{\text{small}} - 1/B_{\text{big}}} \left(|G_{B_{\text{small}}}|^2 - |G_{B_{\text{big}}}|^2 \right)$$

and take $\tilde{B}_{\text{crit}} := \mathcal{S}/|\mathcal{G}|^2$. In practice, to account for the noisiness of $|G|^2$, we first take rolling averages of $|G_{B_{\text{small}}}|$ and $|G_{B_{\text{big}}}|$ over training, and tune the window size so that the estimates for $|\mathcal{G}|^2$ and $\mathcal{S}$ are stable.

We show the values of $\tilde{B}_{\text{crit}}$ over training in Figure 10. Unlike policy gradient methods, we find that the critical batch size (averaged over training) has little correlation with the optimal batch size, as shown in Figure 11.

[1]This form enforces that $\mathcal{D}_J^{\min}$ is positive.

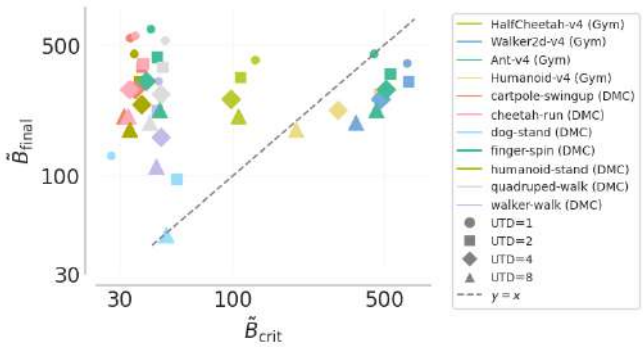


Figure 11: $\tilde{B}_{\text{final}}$ vs. $\tilde{B}_{\text{crit}}$, grouped by task and UTD.

Table 6: Batch size values predicted by the proposed model on DMC.

Task	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 4$	$\sigma = 8$
cartpole-swingup	1040	752	544	384	288	208
cheetah-run	1088	784	560	400	288	208
dog-stand	240	176	128	96	64	48
finger-spin	1168	848	608	432	320	224
humanoid-stand	864	624	448	320	240	176
quadruped-walk	1008	736	528	384	272	192
walker-walk	608	432	320	224	160	112

Table 7: Learning rate values predicted by the proposed model on DMC.

Task	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 4$	$\sigma = 8$
cartpole-swingup	.00108	.000902	.000755	.000631	.000528	.000442
cheetah-run	.000893	.000747	.000625	.000523	.000438	.000366
dog-stand	.000664	.000555	.000465	.000389	.000325	.000272
finger-spin	.00125	.00105	.000877	.000734	.000614	.000514
humanoid-stand	.000551	.000461	.000386	.000323	.00027	.000226
quadruped-walk	.00121	.00101	.000846	.000708	.000592	.000496
walker-walk	.00134	.00112	.000938	.000785	.000657	.000549

Table 8: Batch size values predicted by the proposed model on OpenAI Gym.

Task	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 4$	$\sigma = 8$	$\sigma = 16$
Ant-v4	704	560	448	352	288	224	176
HalfCheetah-v4	672	528	416	336	256	208	160
Humanoid-v4	560	432	352	272	224	176	144
Walker2d-v4	640	496	400	320	256	192	160

Table 9: Learning rate values predicted by the proposed model on OpenAI Gym.

Task	$\sigma = 0.25$	$\sigma = 0.5$	$\sigma = 1$	$\sigma = 2$	$\sigma = 4$	$\sigma = 8$	$\sigma = 16$
Ant-v4	.000206	.000167	.000138	.000109	.000087	.000070	.000060
HalfCheetah-v4	.002820	.002280	.001900	.001510	.001210	.000972	.000827
Humanoid-v4	.000251	.000203	.000169	.000134	.000107	.000086	.000073
Walker2d-v4	.001180	.000958	.000806	.000640	.000512	.000412	.000347

Table 10: Batch size values predicted by the proposed model on IsaacGym.

Task	$\sigma = \frac{1}{65536}$	$\sigma = \frac{1}{32768}$	$\sigma = \frac{1}{16384}$	$\sigma = \frac{1}{8192}$	$\sigma = \frac{1}{4096}$	$\sigma = \frac{1}{2048}$	$\sigma = \frac{1}{1024}$
Franka-Push	7927	5105	3234	2030	1269	791	493

Table 11: Learning rate values predicted by the proposed model on IsaacGym.

Task	$\sigma = \frac{1}{65536}$	$\sigma = \frac{1}{32768}$	$\sigma = \frac{1}{16384}$	$\sigma = \frac{1}{8192}$	$\sigma = \frac{1}{4096}$	$\sigma = \frac{1}{2048}$	$\sigma = \frac{1}{1024}$
Franka-Push	0.000317	0.000265	0.000221	0.000185	0.000154	0.000129	0.000107