Constrained Policy Improvement for Efficient Reinforcement Learning[*]

Elad Sarafian[1] , **Aviv Tamar**[2] and **Sarit Kraus**[1]
[1] Bar-Ilan University, Israel
[2] Technion, Israel
elad.sarafian@gmail.com, avivt@technion.ac.il, sarit@cs.biu.ac.il

Abstract

We propose a policy improvement algorithm for Reinforcement Learning (RL) termed Rerouted Behavior Improvement (RBI). RBI is designed to take into account the evaluation errors of the Q-function. Such errors are common in RL when learning the Q-value from finite experience data. Greedy policies or even constrained policy optimization algorithms that ignore these errors may suffer from an improvement penalty (i.e., a policy impairment). To reduce the penalty, the idea of RBI is to attenuate rapid policy changes to actions that were rarely sampled. This approach is shown to avoid catastrophic performance degradation and reduce regret when learning from a batch of transition samples. Through a two-armed bandit example, we show that it also increases data efficiency when the optimal action has a high variance. We evaluate RBI in two tasks in the Atari Learning Environment: (1) learning from observations of multiple behavior policies and (2) iterative RL. Our results demonstrate the advantage of RBI over greedy policies and other constrained policy optimization algorithms both in learning from observations and in RL tasks.

1 Introduction

While Deep Reinforcement Learning (DRL) is the backbone of many Artificial Intelligence breakthroughs [Silver *et al.*, 2017; OpenAI, 2018 accessed May 2020], factors such as safety and data efficiency may inhibit deployment of RL systems to real-world tasks. DRL is notoriously data and time inefficient – it requires up to billions of history states [Horgan *et al.*, 2018] or weeks of wall-clock time [Hessel *et al.*, 2017] to train to expert level. While it is partially due to the slow training process of deep neural networks, it is also due to inefficient, yet simple to implement, policy improvement routines. For example, an ε-greedy policy improvement is known to have a higher regret than other methods such as Upper Confidence Bound (UCB) [Auer *et al.*, 2002]. However, the latter is much more challenging to adjust to a deep learning framework [Bellemare *et al.*, 2016]. This transformation from the countable state space of bandit and grid-world problems to the uncountable state-space in a DRL framework, calls for efficient improvement methods that fit into existing deep learning frameworks.

For some real-world problems, like autonomous cars [Shalev-Shwartz *et al.*, 2016], safety is a crucial factor. Randomly initialized policies and even an RL algorithm that may suffer from sudden catastrophic performance degradation are both unacceptable in such environments. While policy initialization risks may be avoided with Learning from Demonstrations (LfD) algorithms [Argall *et al.*, 2009], changing the policy to improve performance is still a risky task, mainly since the Q-value of the current policy can only be estimated from the past data. Therefore, for safe RL, it is desirable to design improvement algorithms that model the accuracy of the Q-value evaluation and can mitigate between fast improvement and a safety level [Garcıa and Fernández, 2015; Thomas *et al.*, 2015; Pirotta *et al.*, 2013b].

In this work, we propose a policy improvement method that addresses both the sample efficiency of the learning process and the problem of safe learning from incomplete experiences. We start by analyzing the *improvement penalty* of an arbitrary new policy $\pi(a|s)$ based on an estimated Q-function of a past behavior policy $\beta(a|s)$. We find that under a simplified model of learning the Q-values from i.i.d samples, the variance of a potential improvement penalty is proportional to $\frac{|\beta(a|s)-\pi(a|s)|^2}{\beta(a|s)}$. Therefore, we design a constraint, called reroute, which limits this term. We show that finding the optimal policy under the reroute constraint amounts to solving a simple linear program. Instead of optimizing this policy via a gradient descent optimization, we take a different approach and solve it in the non-parameterized space for every new state the actor encounters. To learn the new, improved policy with a parameterized Neural Network (NN), we store the calculated policy into a replay buffer and imitate the actor's policy with a KL regression.

RBI is designed for safe learning from a batch of experience, yet we show that it also increases data efficiency with respect to a greedy step and other constraints such as the Total Variation (TV) [Kakade and Langford, 2002] and PPO [Schulman *et al.*, 2017]. In fact, it is akin in practice to the forward KL constraint [Vuong *et al.*, 2019], however, unlike the KL constraint, it does not require different scaling for different reward signals, and it is much more intuitive to design. We

[*]This work was supported by the Israel Innovation Authority.

validate our findings both in simple environments such as a two-armed bandit problem with Gaussian distributed reward and also in the Atari Learning Environment.

2 Related Work

Many different algorithms have been suggested to address the problems of efficiency and safety in RL. For safety, [Kakade and Langford, 2002; Pirotta *et al.*, 2013a] introduced the concept of constrained policy optimization in RL for guaranteed monotonic improvement. TRPO [Schulman *et al.*, 2015] adopted it to NN parametrized policies, and its successor PPO [Schulman *et al.*, 2017] established better empirical results with a much simpler algorithm. More recent constrained policy iterations are Smoothing Policies [Papini *et al.*, 2019] and optimization via Importance Sampling [Metelli *et al.*, 2018]. However, these algorithms assume that the Q-function is known, and the safety issue arises due to the step size in the gradient optimization. While they provide improvement guarantees when the Q-value is known, they do not address the problem of imperfect Q-value approximation.

Another line of works, Double Q-learning [Hasselt, 2010; Van Hasselt *et al.*, 2016], and its policy gradient variant TD3 [Fujimoto *et al.*, 2018] addressed the errors in the Q-function from a different perspective. It is known that standard Q-learning [Watkins and Dayan, 1992] introduces a bias towards high-value actions; therefore, they use two different estimators of the Q-function to reduce this bias. Still, they do not address other error elements in the Q-value estimator, which result from learning from incomplete, imperfect data.

3 Preliminaries

We consider a model-free RL paradigm for solving Markov Decision Process (MDP) problems [Puterman, 2014] with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$ and rewards $r \in \mathcal{R}$. Our objective is to learn a policy π which maximizes the expected discounted return, also known as the objective function $J(\pi) = \mathbb{E}[\sum_{k\geq 0} \gamma^k r_k | \pi]$, where $0 < \gamma < 1$ is the discount factor. The value of a state as $V^\pi(s) = \mathbb{E}[\sum_{k\geq 0} \gamma^k r_k | s, \pi]$ is the total discounted reward followed by a visitation at s. Accordingly, the Q-value of a state-action pair is $Q^\pi(s,a) = \mathbb{E}[\sum_{k\geq 0} \gamma^k r_k | s, a, \pi]$. The advantage of an action A^π is the difference between its Q-value and the state's value, i.e. $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. For convenience, we may simply write π_i to denote $\pi(a_i|s)$ (omitting the state's dependency). In the paper, two important distance functions between policies are discussed. The first is the Total Variation (TV) $\delta(\pi, \pi') = \frac{1}{2}\sum_{a_i} |\pi_i - \pi'_i|$ and the second is the KL divergence $D_{KL}(\pi||\pi') = -\sum_{a_i} \pi_i \log \frac{\pi'_i}{\pi_i}$. These metrics are often used to constrain the updates of a learned policy in RL algorithms [Schulman *et al.*, 2015; Schulman *et al.*, 2017].

4 Rerouted Behavior Improvement

Let us start by examining a single improvement step from a batch of experience of a behavior policy. Define by β the behavior policy of a dataset $\mathcal{D}$ and by Q^β, and $\hat{Q}^\beta$ its exact and approximated Q-functions. Theoretically, for an infinite dataset with an infinite number of visitations in each state-action pair, one may calculate the optimal policy in an off-policy fashion [Watkins and Dayan, 1992]. However, practically, one should limit its policy improvement step over β when learning from a realistic finite dataset. To design a proper constraint, we analyze the statistics of the error of our evaluation of $\hat{Q}^\beta$. This leads to an important observation: the Q-value has a higher error for actions that were taken less frequently, thus, to avoid the improvement penalty, we must restrict the ratio of the change in probability $\frac{\pi}{\beta}$. We will use this observation to craft the reroute constraint and show that other well-known monotonic improvement methods (e.g., PPO and TRPO) overlooked this consideration. Hence they do not guarantee improvement when learning from a finite experience.

4.1 Soft Policy Improvement

Before analyzing the error statistics, we identify a set of policies that improve β if our estimation of Q^β is exact. Out of this set, we will pick our new policy π. Recall that the most common improvement method is taking a greedy step, i.e., deterministically acting with the highest Q-value action in each state, which is known by the policy improvement theorem [Sutton and Barto, 2017] to improve the policy performance.

It can be readily shown that the policy improvement theorem may be generalized to also include stochastic policies which satisfy $\sum_a \pi(a|s) A^\beta(s,a) \geq 0 \; \forall s$ where A^β is the advantage function of β. In other words, every policy that increases the probability of taking positive advantage actions over the probability of taking negative advantage actions achieves improvement. Later, we will use the next lemma to show that an RBI iteration improves the previous policy.

Lemma 4.1 (Rank-Based Policy Improvement). *Let $(A_i)_{i=1}^{|\mathcal{A}|}$ be an ordered list of the β advantages in a state s, s.t. $A_{i+1} \geq A_i$, and let $c_i = \pi_i/\beta_i$. If for all states $(c_i)_{i=1}^{|\mathcal{A}|}$ is a monotonic non-decreasing sequence s.t. $c_{i+1} \geq c_i$, then π improves β (proof in the appendix).*

4.2 Standard Error of the Value Estimation

To provide a statistical argument for the expected error of the Q-function, consider learning $\hat{Q}^\beta$ with a tabular representation. The Q-function is the expected value of the random variable $z^\pi(s,a) = \sum_{k\geq 0} \gamma^k r_k | s, a, \pi$. Therefore, the Standard Error (SE) of an approximation $\hat{Q}^\beta(s,a)$ for the Q-value with N i.i.d. MC trajectories[1] is

$$\sigma_{\varepsilon(s,a)} = \frac{\sigma_{z(s,a)}}{\sqrt{N_s \beta(a|s)}}, \tag{1}$$

where N_s is the number of visitations in state s in $\mathcal{D}$, s.t. $N = \beta(a|s) N_s$. Therefore $\sigma_{\varepsilon(s,a)} \propto \frac{1}{\sqrt{\beta(a|s)}}$, and specifically for

[1]Notice that MC rollouts from s, a are indeed independent random variables when averaging only over the first occurrences of the state-action pair s, a in each episode.

low frequency actions such an estimation may suffer from very large errors.[2]

4.3 Policy Improvement in the Presence of Value Estimation Errors

We now turn to the crucial question of what happens when one applies an improvement step with respect to an inaccurate estimation of the Q-function, i.e., $\hat{Q}^\beta$.

Theorem 4.2 (The Improvement Penalty)**.** *Let $\hat{Q}^\beta = \hat{V}^\beta + \hat{A}^\beta$ be an estimator of Q^β with an error $\varepsilon(s,a) = (Q^\beta - \hat{Q}^\beta)(s,a)$ and let π be a policy that satisfies lemma 4.1 with respect to $\hat{A}^\beta$. Then the following holds*

$$V^\pi(s) - V^\beta(s) \geq -\mathcal{E}(s) = \\ -\sum_{s' \in \mathcal{S}} \rho^\pi(s'|s) \sum_{a \in \mathcal{A}} \varepsilon(s',a)\left(\beta(a|s') - \pi(a|s')\right), \quad (2)$$

where $\mathcal{E}(s)$ is called the improvement penalty and $\rho^\pi(s'|s) = \sum_{k\geq 0} \gamma^k P(s \xrightarrow{k} s'|\pi))$ is the unnormalized discounted state distribution induced by policy π (proof in the appendix).

Since $\varepsilon(s',a)$ is a random variable, it is worth considering the variance of $\mathcal{E}(s)$. Define each element in the sum of Eq. (2) as $x(s',a;s) = \rho^\pi(s'|s)\varepsilon(s,a)(\beta(a|s') - \pi(a|s'))$. The variance of each element is therefore

$$\sigma^2_{x(s',a;s)} = (\rho^\pi(s'|s))^2 \sigma^2_{\varepsilon(s',a)} (\beta(a|s') - \pi(a|s'))^2 = \\ \frac{(\rho^\pi(s'|s))^2 \sigma^2_{z(s',a)}}{N_{s'}} \frac{(\beta(a|s') - \pi(a|s'))^2}{\beta(a|s')}.$$

To see the need for the reroute constraint, we can bound the total variance of the improvement penalty

$$\sum_{s',a} \sigma^2_{x(s',a;s)} \leq \sigma^2_{\mathcal{E}(s)} \leq \sum_{s',a,s'',a'} \sqrt{\sigma^2_{x(s',a;s)} \sigma^2_{x(s'',a';s)}},$$

where the upper bound is due to the Cauchy-Schwarz inequality, and the lower bound follows from the fact that $\varepsilon(s,a)$ elements have a positive correlation (as reward trajectories overlap).

Hence, it is evident that the improvement penalty can be enormous when the term $\frac{|\beta-\pi|^2}{\beta}$ is unregulated and even a single mistake along the trajectory, caused by an unregulated element, might severely hurt the performance. However, by using the reroute constraint, which tames each of these terms, we can bound the variance of the improvement penalty.

While we analyzed the error for independent MC trajectories, a similar argument also holds for Temporal Difference (TD) learning [Sutton and Barto, 2017]. [Kearns and Singh, 2000] studied "bias-variance" terms in k-steps TD learning of the value function. Here we present their results for the Q-function error with TD updates. For any $0 < \delta < 1$ with probability at least $1 - \delta$, and a number t of iterations through the data for the TD calculation, the maximal error term abides

$$\varepsilon(s,a) \leq \max_{s,a} |\hat{Q}^\beta(s,a) - Q^\beta(s,a)| \\ \leq \frac{1-\gamma^{kt}}{1-\gamma} \sqrt{\frac{3\log(k/\delta)}{N_s \beta(a|s)}} + \gamma^{kt}. \quad (3)$$

While the "bias", which is the second term in (3), depends on the number of iterations through the dataset, the "variance", which is the square of the first term in (3), is proportional to $\frac{1}{\beta(a|s)N_s}$, therefore bounding the ratio $\frac{|\beta-\pi|^2}{\beta}$ also bounds the improvement penalty for TD learning.

4.4 The Reroute Constraint

In order to confine the ratio $\frac{|\beta-\pi|^2}{\beta}$, we suggest limiting the improvement step to a set of policies based on the following constraint.

Definition 4.1 (Reroute Constraint)**.** *Given a policy β, a policy π is a $reroute(c_{\min}, c_{\max})$ of β if $\pi(a|s) = c(s,a)\beta(a|s)$ where $c(s,a) \in [c_{\min}, c_{\max}]$. Further, note that reroute is a subset of the TV constraint with $\delta = \min(1 - c_{\min}, \max(\frac{c_{\max}-1}{2}, \frac{1-c_{\min}}{2}))$ (proof in the appendix).*

With reroute, each element in the sum of Eq. (2) is proportional to $\sqrt{\beta(a|s)}|1 - c(s,a)|$ where $c(s,a) \in [c_{\min}, c_{\max}]$.

Let us now consider other popular constraints. Clearly, with TV $\delta \geq \frac{1}{2}\sum_{a_i} |\beta_i - \pi_i|$ the ratio $\frac{|\beta-\pi|^2}{\beta}$ is uncontrolled since actions with a zero probability may increase their probability up to δ. For the PPO constraint, we prove in the appendix that it does not force any constraint on the ratio $\frac{|\beta-\pi|^2}{\beta}$ and, moreover, its optimal solution is non-unique.

Next, there are two possible KL constraints, forward KL $D_{KL}(\pi||\beta) \leq \delta$ and backward KL $D_{KL}(\beta||\pi) \leq \delta$. The authors in [Vuong *et al.*, 2019] have shown that the non-parametric solution for the backward constraint, which is implemented in TRPO, is $\pi(a|s) = \frac{\beta(a|s)\lambda(s)}{\lambda'(s) - A^\beta(s,a)}$ where λ, λ' are chosen such that the KL constraint is binding. However, the backward KL does not restrict the ratio $\frac{|\beta-\pi|^2}{\beta}$ simply because actions with zero behavioral probabilities, i.e. $\beta_i = 0$, do not contribute to the constraint (see appendix for a simple example). On the other hand, the forward KL constraint with its non-parameteric solution of $\pi(a|s) = \frac{\beta(a|s)}{Z_\lambda(s)} e^{A^\beta(s,a)/\lambda(s)}$ indeed bounds the $\frac{|\beta-\pi|^2}{\beta}$ ratio since $\beta_i = 0$ forces $\pi_i = 0$. However, this bound depends on the specific size of the advantages and it differs from state to state, hence it requires prior knowledge of the rewards or alternatively inferring λ from the stochastic data (which is another source for errors).

4.5 Maximizing the Improvement Step under the Reroute Constraint

We now turn to the problem of maximizing the objective function $J(\pi)$ under the reroute constraint and whether such maximization improves the previous policy. To do so, we may maximize the advantage of π over β which equals

[2]Note that even for deterministic environments, a stochastic policy inevitably leads to $\sigma_{z(s,a)} > 0$.

Algorithm 1 Max-Reroute

```
Require: $s$, $\beta$, $A^\beta$, $(c_{\min}, c_{\max})$
  $\tilde{\mathcal{A}} \longleftarrow \mathcal{A}$
  $\Delta \longleftarrow 1 - c_{\min}$
  $\pi(a|s) \longleftarrow c_{\min}\beta(a|s)\ \forall a \in \mathcal{A}$
  while $\Delta > 0$ do
    $a = \arg\max_{a\in\tilde{\mathcal{A}}} A^\beta(s,a)$
    $\Delta_a = \min\{\Delta, (c_{\max} - c_{\min})\beta(a|s)\}$
    $\tilde{\mathcal{A}} \longleftarrow \tilde{\mathcal{A}}/a$
    $\Delta \longleftarrow \Delta - \Delta_a$
    $\pi(a|s) \longleftarrow \pi(a|s) + \Delta_a$
  end while
  return $\{\pi(a|s),\ a \in \mathcal{A}\}$
```

to $J^A(\pi,\beta) = \mathbb{E}_{s\sim\pi}[\sum_a \pi(a|s)A^\beta(s,a)]$ [Schulman *et al.*, 2015]. Maximizing J^A without generating new trajectories of π is a hard task since the distribution of states induced by the policy π is unknown. Therefore, one usually resorts to policy gradient techniques which optimize a surrogate off-policy objective function $J^{OP}(\pi,\beta) = \mathbb{E}_{s\sim\beta}[\sum_a \pi(a|s)A^\beta(s,a)]$. However, to avoid large state distribution shifts, this approach can only be applied when the behavior policy is relatively closed to the new policy [Sutton *et al.*, 2000; Schulman *et al.*, 2015].

Here we suggest an alternative: instead of optimizing a parametrized policy that maximizes J^{OP}, the actor (i.e., the agent that interacts with the MDP environment) may ad hoc calculate a non-parametrized policy that maximizes the improvement step $\sum_a \pi(a|s)A^\beta(s,a)$ (i.e., the argument of the $J^A(\pi,\beta)$ objective) for each different state. This method directly maximizes the J^A objective since the improvement step is independent between states. Therefore, it lifts the requirement for a small state distribution shift between β and π.

Non-parametric maximization bears another important advantage. Contrary to policy gradient methods where the optimization may fail to converge to a global maximum (due to sub-optimal optimization routines), it is possible to design algorithms that are guaranteed to maximize the objective under the constraint. For example, for the reroute constraint, solving the non-parametrized problem amounts to solving the following simple linear program for each state

$$\begin{aligned} &\text{Maximize: } (\boldsymbol{A}^\beta)^T\boldsymbol{\pi} \\ &\text{Subject to: } c_{\min}\boldsymbol{\beta} \le \boldsymbol{\pi} \le c_{\max}\boldsymbol{\beta} \\ &\text{And: } \sum \pi_i = 1, \end{aligned} \tag{4}$$

where $\boldsymbol{\pi}$, $\boldsymbol{\beta}$ and $\boldsymbol{A}^\beta$ are vector representations of $(\pi(a_i|s))_{i=1}^{|\mathcal{A}|}$, $(\beta(a_i|s))_{i=1}^{|\mathcal{A}|}$ and $(A^\beta(s,a_i))_{i=1}^{|\mathcal{A}|}$ respectively. While it may be solved with the Simplex algorithm [Vanderbei and others, 2015], we present here a straightforward solution, termed Max-Reroute (Alg. 1). Max-Reroute is a special case of the extended value iteration with the Bernstein bound [Dann and Brunskill, 2015; Fruit *et al.*, 2017]. Similar to Max-Reroute, one may derive other algorithms that maximize other constraints (see Max-TV and Max-PPO in the appendix). We will use these algorithms as baselines for the performance of the reroute constraint.

Notice that Max-Reroute, Max-TV and Max-PPO satisfy the conditions of lemma 4.1. Therefore they always improve the previous policy, and hence, at least for a perfect approximation of Q^β, they are guaranteed to improve the performance. In addition, they all use only the action ranking information to calculate the optimized policy. We postulate that this trait makes them more resilient to value estimation errors. This is in contrast to policy gradient methods, which optimize the policy according to the magnitude of the advantage function.

5 Two-armed Bandit with Gaussian Distributed Rewards

To gain some insight into the nature of the RBI step, we begin by examining it in a simplified model of a two-armed bandit with Gaussian distributed rewards [Krause and Ong, 2011]. To that end, define the reward of taking action a_i as $r_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and let action a_2 be the optimal action s.t. $\mu_2 \ge \mu_1$. Let us start by considering the regret of a single improvement step from a batch of transition samples. In this case, the regret is defined as

$$R^\pi = \mu_2 - V^\pi = \mu_2 - \sum_{i=1,2}\mathbb{E}[\pi(a_i)r_i]$$

Specifically, we are interested in the difference between the behavior policy regret and the modified policy regret, i.e. $R^\beta - R^\pi$, which is equal to

$$R^\beta - R^\pi = P(I)V_I^\pi + (1-P(I))V_{\bar{I}}^\pi - V^\beta$$

where I is an indicator of the clean event when $\hat{r}_2 > \hat{r}_1$ (where $\hat{r}_i$ is the empirical mean of r_i over the batch data). Since $\hat{r}_1$ and $\hat{r}_2$ have Gaussian distributions, we can easily calculate $P(I)$. In Figure 1 we plot the curves for $\mu_1 = -1$ and $\mu_2 = 1$ for a different number of batch sizes and different variances. As a baseline we plot the difference in regret for a greedy step. First, we see that for poor behavior policies s.t. $\beta(a_2) \le \beta(a_1)$, it is better to make a greedy step. This is because the suboptimal action has a sufficiently accurate estimation. However, for good behavior policies s.t. $\beta(a_2) > \beta(a_1)$ (which is generally the case when learning from demonstrations and even in RL after the random initialization phase), the greedy regret grows significantly s.t. it is better to stick with the behavior policy. Intuitively, when the behavior policy reaches the optimum, an overestimation of $\hat{r}_1$ triggers a bad event $\bar{I}$ which leads to a significant degradation. On the other hand, an RBI step is safe s.t. it always has a lower regret than β.[3] In addition, its regret is lower than the greedy regret for good behavior policies.

To find out how RBI performs in an iterative RL process we continue with the bandit example, but now we consider the learning curve of off-policy learning where the behavior policy is mixed with a fixed exploration parameter, i.e. $\beta(a) = \pi(a)(1-\varepsilon) + \frac{\varepsilon}{n_a}$ (where n_a is the number of actions and $\varepsilon = 0.1$). The Q-function is learned with $Q^\pi(a) = (1-\alpha)Q^\pi(a) + \alpha r$, where α is a learning rate, possibly decaying over time. We evaluate several constrained policies: (1) RBI

[3]Note that a choice of too large $c_{\max}$ is not safe, specifically $c_{\min} = 0$, and $c_{\max} \to \infty$ converges to the greedy step.

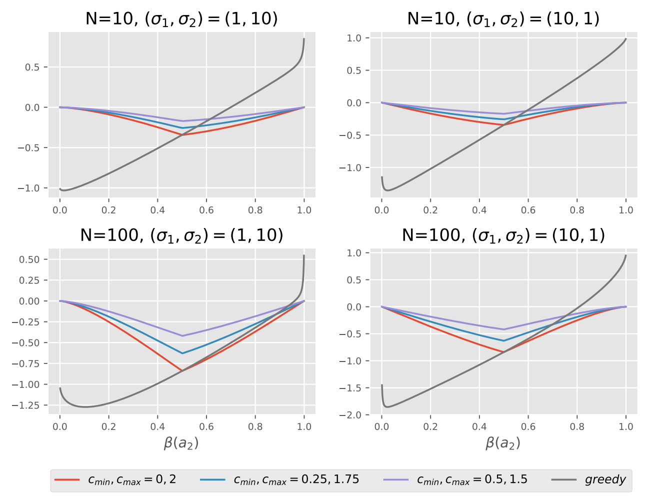

Figure 1: The regret with respect to a past behavior policy β. A positive number means that β has a higher value than π

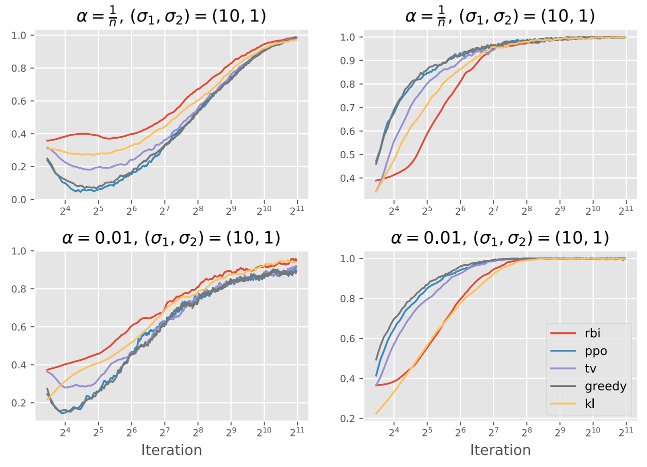

Figure 2: Different constrained policies' performances in a two-armed bandit with Gaussian distributed reward.

with $(c_{\min}, c_{\max}) = (0.5, 1.5)$; (2) PPO with $\varepsilon = 0.5$; (3) TV with $\delta = 0.25$; (4) greedy step; and (5) forward KL with $\lambda = 1$. RBI, TV and PPO were all maximized with our maximization algorithms (without gradient ascent). To avoid absolute zero probability actions, we clipped the policy such that $\pi(a_i) \geq 10^{-3}$. In addition we added 10 random samples at the start of the learning process. The learning curves are plotted in Figure 2.

The learning curves exhibit two different patterns. For the scenario of $\sigma_1 > \sigma_2$, rapid convergence of all policies was obtained (note the x-axis logarithmic scale). Intuitively, when better action has low variance, it is easy to discard the worse action by choosing it and rapidly improving its value estimation and then switching to the favorable action. On the other hand, for the case of $\sigma_1 < \sigma_2$, it is much harder for the policy to improve the estimation of the favorable action after committing to the worse action. We see that RBI defers early commitment, and while it slightly reduces the rate of convergence in the easier scenario, it significantly increases the data efficiency in the harder one. We postulate that the harder scenario is also more prevalent in real-world problems where, usually, one must take a risk to get a big reward (where avoiding risks significantly reduces not only the reward variance but also the expected reward).

An interesting distinction is between the ideal learning rate (LR) of $\alpha = \frac{1}{n}$ and a constant rate of $\alpha = 0.01$. In the ideal LR case, the advantage of RBI and KL is reduced over time. This is obvious since an LR of $\alpha = \frac{1}{n}$ takes into account the entire history. As such, for a large history and after a large number of iterations, there is no need for a policy that learns well from a finite dataset. On the other hand, there is a stable advantage of RBI and KL for a fixed LR as a fixed LR does not correctly weight all the transition samples. Notice that in a larger than 1-step MDP, it is unusual to use an LR of $\frac{1}{n}$ since the policy changes as the learning progresses, therefore, usually, the LR is fixed or decays over time (but not over state-visitations). Hence, we expect a positive advantage for RBI over greedy steps during the entire training process.

6 Learning to Play Atari by Observing Human Players

In section 4, the reroute constraint was derived for learning from a batch of transition samples of a single behavior policy β and a tabular value and policy parametrization. In this section, we empirically extend our results to a NN parametric form (both for value and policy) and a dataset generated by multiple policies $\{\beta_i\}$. To that end, we use a crowdsourced dataset of 4 Atari games (SpaceInvaders, MsPacman, Qbert, and Montezuma's Revenge) [Kurin *et al.*, 2017], where each game has roughly 1000 recorded episodes. Such a dataset of observations is of particular interest since it is often easier to collect observations than expert demonstrations. Therefore, it may be practically easier to initialize an RL agent with learning from observations than to generate trajectories of a single expert demonstrator.

For this purpose, we aggregated all trajectories into a single dataset and estimated the average policy β as the average behavior of the dataset. Practically, when using a policy network, this sums to minimizing the Cross-Entropy loss between the $\beta_\theta(\cdot|s)$ and the empirical evidence (s, a) as done in classification problems. To estimate the Q-value of β we employed a Q-network in the form of the Dueling DQN architecture [Mnih *et al.*, 2015; Wang *et al.*, 2015] and minimized the MSE loss between Monte-Carlo reward trajectories $\sum_{k \geq 0} \gamma^k r_k$ and the $Q^\beta(s, a)$ estimation. Note that this does not necessarily converge to the true Q-value of β (since the trajectories were generated by the behavior policies $\{\beta_i\}$), but it empirically provides a close estimation with a small error.

Given our estimations of β and Q^β, we applied several improvement steps and evaluated their performance. For the RBI, TV, and greedy steps, we calculated the improved policy π in the non-parametric space (i.e., during evaluation) with Max-Reroute and Max-TV. For the PPO step we applied an additional gradient ascent optimization step and calculated the PPO policy. As an additional baseline, we implemented the DQfD algorithm [Hester *et al.*, 2018].

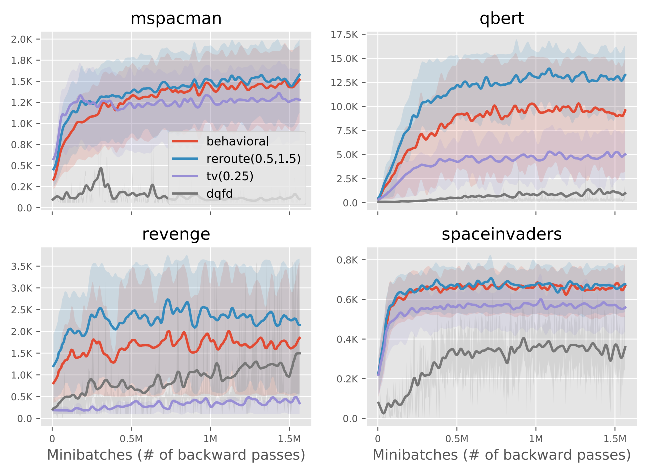

Figure 3: Learning to play Atari from a dataset of human players: Learning curves.

Method	Pacman	Qbert	Revenge	SI
Humans	3024	3401	1436	634
Behavioral	1519	8562	1761	678
RBI$(0.5, 1.5)$	1550	12123	2379	**709**
RBI$(0.25, 1.75)$	**1638**	13254	**2431**	682
RBI$(0, 2)$	1565	**14494**	473	675
TV(0.25)	1352	5089	390	573
PPO(0.5)	1528	14089	388	547
DQfD	83	1404	1315	402

Table 1: Learning to play Atari from a dataset of human players: Final scores table

First, we found out that behavioral cloning, i.e., merely playing with the calculated average behavior β, generally yielded good results with the exception of MsPacman, which is known to be a harder game (see a comparison of human and DQN scores in [Mnih *et al.*, 2015]). For Qbert, the behavioral score was much better than the average score, and we assume that this is because good episodes tend to be longer in Qbert. Thus, their weight in the average behavior calculation is higher.

We evaluated Max-TV with $\delta = 0.25$ since it encapsulates the $reroute(0.5, 1.5)$ region. However, unlike $reroute(0.5, 1.5)$, a TV constrained update obtained lower performance than the behavioral cloning in all games. At first glance, this may be surprising evidence, but it is expected after analyzing Eq. (2). Only a single state with a bad estimation along the trajectory is required to reduce the overall score significantly. On the other hand, $reroute(0.5, 1.5)$ always increased the behavioral score and provided the overall best performance.

While PPO with $\varepsilon = 0.5$ generally scored better than TV, we noticed that in two games, it reduced the behavioral score. The final results (see Table 1) showed similarity between $PPO(0.5)$ and $reroute(0, 2)$. This indicates that PPO tends to settle negative advantage action to zero probability in order to avoid the negative PPO penalty. Empirically, this also indicates that it is important to set $c_{\min} > 0$ to avoid *catastrophic forgetting* which occurs when the probability of a good action is reduced to zero due to value estimation errors. Finally, we observed a significantly lower score of the DQfD algorithm. DQfD uses the demonstrations data to learn a policy with an almost completely off-policy algorithm (it adds a regularization term that should tie the learned policy to the expert behavior, but its effectiveness depends on the Q-value magnitude). This strengthens our assertion that when learning from a fixed size of transition samples, the calculated policy must be constrained to the behavior policy presented in the data.

7 Iterative RL with RBI

We now turn to the implementation of RBI learning in an iterative RL setting in the Atari environment without utilizing any prior human demonstrations. We adopted a distributed learning setting with multiple actors and a single learner process, similar to the setting of Ape-X [Horgan *et al.*, 2018]. However, contrary to the Ape-X algorithm, which learns only a single Q-value network (and infers the policy with a greedy action selection), we learned side-by-side two different networks: (1) a Q-value network, termed Q-net and denoted Q^{π}_{ϕ}; and (2) a policy network termed π-net and denoted π_θ.

In this experiment we attempt to verify: (1) whether RBI is a good method in DRL also for a better final performance (a tabular example of RL with RBI was discussed in section 5); and (2) whether our approach of solving for the optimal policy in the non-parametrized space as part of the actor's routine can be generalized to iterative RL. Recently, [Vuong *et al.*, 2019] developed a framework of constrained RL where they generate samples with a policy π_{θ_k} (where θ_k is the set of network parameters in the k-th learning step), and in the learner process, they calculate an improved non-parameterized policy π under a given constraint and learn the next policy $\pi_{\theta_{k+1}}$ by minimizing the KL distance $D_{KL}(\pi_{\theta_{k+1}}, \pi)$.

We adopted a slightly different approach. Instead of calculating the non-parameterized policy in the learner process, each actor loads a stored policy π_{θ_k} and calculates the non-parameterized optimal policy π as part of the actor routine while interacting with the environment and generating its trajectories. The optimized policy for each new state is then stored to a replay buffer and executed by the actor (after mixing with a small random exploration). The learner process learns to imitate the optimized policy by minimizing the loss $D_{KL}(\pi, \pi_{\theta_{k+1}})$. This approach has several additional benefits:

1. The learner is not required to calculate the optimized policy; therefore, the rate of processing mini-batches of transitions increases. This rate is generally the bottleneck of the wall clock time of the process.
2. The Q-net estimates the value of the π-net policy (i.e., the past policy), contrary to [Vuong *et al.*, 2019] where the Q-net estimates the value of the past policy, but the π-net is the future policy. This improves the value estimation since π-net is used to calculate the target value.
3. The executed policy π is the optimal solution of the constrained optimization in Eq. (4), unlike policy gradient iteration, which gradually approaches the optimal solution.

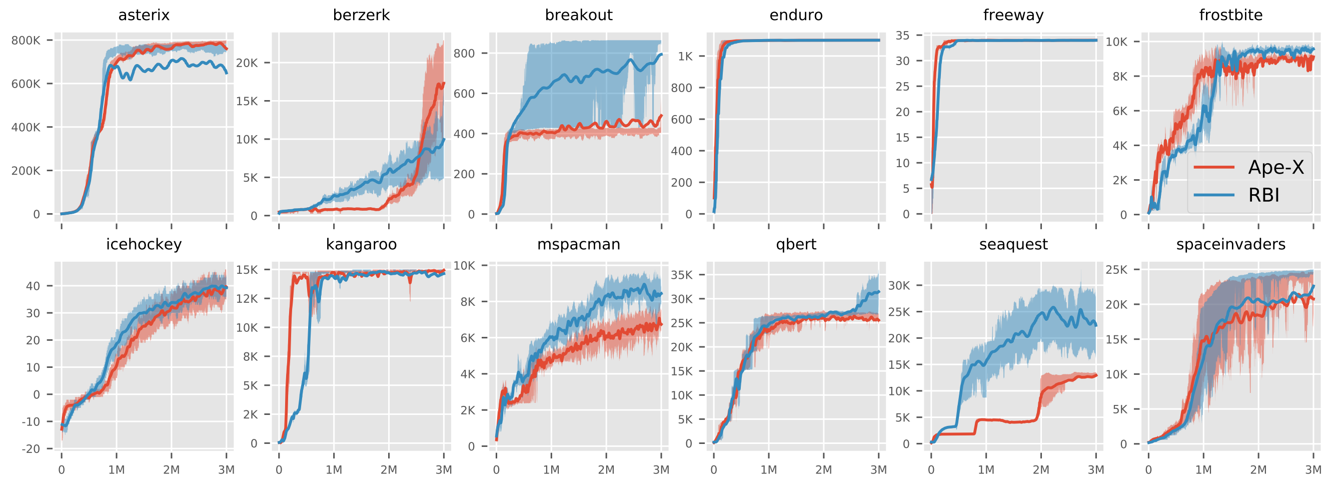

Figure 4: Performance curves of 12 Atari games over 3.125M training steps. The second and third quartiles are shadowed.

Algorithm 2 RBI learner

```
Require: $\mathcal{D} = \{(s_i, a_i, \pi_i, r_i), ...\}$
  repeat
    $B \longleftarrow$ PriorityBatchSample$(\mathcal{D}, N)$
    $\pi_{\theta_k} = \pi$-net$(s; \theta_k)$
    $Q^{\pi}_{\phi_k} = Q$-net$(s, a; \phi_k)$
    Calculate a target value with a target network:
    $R_i = \sum_{t=1}^{n} \gamma^t r_{i+t} + \gamma^n \sum_{a_j} \pi_{\bar{\theta}}(a_j|s_{i+n}) Q^{\pi}_{\bar{\phi}}(s_{i+n}, a_j)$

    $\mathcal{L}^{\pi}(\pi_{\theta_k}) = \frac{1}{N}\sum_i D_{KL}(\pi_i, \pi_{\theta_k,i})$
    $\mathcal{L}^{q}(Q^{\pi}_{\phi_k}) = \frac{1}{N}\sum_i (R_i - Q^{\pi}_{\phi_k,i})^2$
    $\theta_{k+1} = \theta_k - \alpha \nabla_{\theta} \mathcal{L}^{\pi}(\pi_{\theta_k})$
    $\phi_{k+1} = \phi_k - \alpha \nabla_{\phi} \mathcal{L}^{q}(Q^{\pi}_{\phi_k})$
  until termination
  return $\theta, \phi$
```

Algorithm 3 RBI actor

```
  reset the MDP and generate $s_0$
  repeat
    $\pi_{\theta} = \pi$-net$(s_i; \theta)$
    $Q^{\pi}_{\phi} = Q$-net$(s_i, \cdot; \phi)$
    $\pi =$ MaxReroute$(\pi_{\theta}, s_i, Q^{\pi}_{\phi})$
    Add exploration: $\beta = \pi(1 - \varepsilon) + \frac{\varepsilon}{n_a}$
    $a_i =$ Sample$(\beta)$
    executed $a_i$ and obtain a new state $s_{i+1}$ and reward $r_i$
    Add to dataset $\mathcal{D} += \{(s_i, a_i, \pi_i, r_i), ...\})$
  until termination
```

As stated in the previous experiment, setting $c_{\min} > 0$ is important in avoiding catastrophic forgetting, and choosing too high a $c_{\max}$ can lead to greedy-like improvement steps (see also Fig. 1). Following these observations, we set $(c_{\min}, c_{\max}) = (0.1, 2)$, which we found to balance between safety and efficiency. We also found that it is better to mix the RBI solution with a small amount of a greedy policy for a faster recovery of actions with near-zero probability. Therefore, the actor policy was

$$\pi = (1 - c_{greedy})\pi^{rbi} + c_{greedy}\pi^{greedy},$$

with $c_{greedy} = 0.1$. The final algorithms for the learner and actors are presented in Algorithms 2 and 3.

We evaluated RBI and the Ape-X baseline over a set of 12 Atari games with a varying range of difficulty. As an external baseline, we compared our results to the Rainbow score [Hessel *et al.*, 2017]. For a fair comparison, we used a batch size of 128 and capped the learning process to 3.125M backward passes. This is identical to the number of different states that were processed by the network in Rainbow (12.5M backward passes with a batch of 32). Altogether, RBI provided better policies in 7 out of 11 environments with respect to our Ape-X implementation and in 9 out of 11 environments with respect to the Rainbow score (all tied in Freeway). See Table 2 and the extended results section in the appendix. In some games (e.g., Qbert), RBI improved its policy while Ape-X converged to a steady policy. We postulate that with NN parametrization, taking greedy steps may lead to faster convergence of the nets since the diversity of actions is much lower. Generally, high diversity, i.e., a higher entropy policy, is known to be beneficial for exploration [Haarnoja *et al.*, 2018], and indeed, we noticed that RBI learns hard exploration tasks faster, like oxygen replenish in Seaquest. Another interesting game is Berzerk; in this game, while RBI was more efficient during the majority of the learning process and had lower regret (i.e., minus the area under the curve), the Q-learning achieved a higher final score. We found out that due to the deterministic nature of the ALE, the deterministic greedy policy found a better strategy near the end of the learning process.

8 Conclusions

We introduced a constrained policy improvement method, termed RBI, designed to generate safe policy updates in the presence of common estimation errors of the Q-function. In

Learning from Observation settings, where one has no access to new samples and off-policy RL fails, the RBI update can safely improve upon naive behavioral cloning. In addition to safety, we found out that the RBI updates are more data-efficient than greedy and other constrained policies in training RL agents. We validated our findings both in a simple two-armed bandit problem and in the Atari domain. To train parametrized policies with the RBI updates, we designed a two-phase method: an actor solves a non-parametrized constrained optimization problem (Eq. (4)) while a learner imitates the actor's policy with a parametrized network.

Game	RBI	Ape-X	Rainbow
Asterix	838,018.3	**926,429.2**	428,200.3
Berzerk	12,159.3	**43,867.7**	2545.6
Breakout	**780.4**	515.2	417.5
Enduro	2260.9	**2276.1**	2,125.9
Freeway	34.0	34.0	34.0
Frostbite	**9,655.0**	8,917.3	9,590.5
IceHockey	**40.4**	39.9	1.1
Kangaroo	14,426.7	**14,897.5**	14,637.5
MsPacman	**8,193.9**	6,401.8	5,380.4
Qbert	32,894.2	26,127.5	**33,817.5**
Seaquest	**26,963.7**	12,920.7	15,898.9
SpaceInvaders	**45,155.8**	40,923.4	18,729.0

Table 2: Final scores table

A Supplementary Material

The appendix and the source code for the Atari experiments are found at `github.com/eladsar/rbi/tree/rbi`.

References

[Argall *et al.*, 2009] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[Auer *et al.*, 2002] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[Barreto *et al.*, 2017] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[Bellemare *et al.*, 2016] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[Dann and Brunskill, 2015] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015.

[Fruit *et al.*, 2017] Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Emma Brunskill. Regret minimization in mdps with options without prior knowledge. In *Advances in Neural Information Processing Systems*, pages 3166–3176, 2017.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[Garcıa and Fernández, 2015] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[Hasselt, 2010] Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[Henderson *et al.*, 2017] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

[Hessel *et al.*, 2017] 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. *arXiv preprint arXiv:1710.02298*, 2017.

[Hester *et al.*, 2018] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Horgan *et al.*, 2018] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

[Kakade and Langford, 2002] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

[Kearns and Singh, 2000] Michael J Kearns and Satinder P Singh. Bias-variance error bounds for temporal difference updates. In *COLT*, pages 142–147. Citeseer, 2000.

[Krause and Ong, 2011] Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

[Kurin *et al.*, 2017] Vitaly Kurin, Sebastian Nowozin, Katja Hofmann, Lucas Beyer, and Bastian Leibe. The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*, 2017.

[Metelli *et al.*, 2018] Alberto Maria Metelli, Matteo Papini, Francesco Faccio, and Marcello Restelli. Policy optimization via importance sampling. In *Advances in Neural Information Processing Systems*, pages 5442–5454, 2018.

[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.

[OpenAI, 2018 accessed May 2020] OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018 - accessed May 2020.

[Papini *et al.*, 2019] Matteo Papini, Matteo Pirotta, and Marcello Restelli. Smoothing policies and safe policy gradients. *arXiv preprint arXiv:1905.03231*, 2019.

[Pirotta *et al.*, 2013a] Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Adaptive step-size for policy gradient methods. In *Advances in Neural Information Processing Systems*, pages 1394–1402, 2013.

[Pirotta *et al.*, 2013b] Matteo Pirotta, Marcello Restelli, Alessio Pecorino, and Daniele Calandriello. Safe policy iteration. In *International Conference on Machine Learning*, pages 307–315, 2013.

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

[Racine, 2000] Jeff Racine. Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. *Journal of econometrics*, 99(1):39–61, 2000.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

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

[Shalev-Shwartz *et al.*, 2016] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[Silver *et al.*, 2017] 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, 2017.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Sutton and Barto, 2017] RS Sutton and AG Barto. Reinforcement learning: An introduction, (complete draft), 2017.

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

[Thomas *et al.*, 2015] Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High confidence policy improvement. In *International Conference on Machine Learning*, pages 2380–2388, 2015.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

[Vanderbei and others, 2015] Robert J Vanderbei et al. *Linear programming*. Springer, 2015.

[Vuong *et al.*, 2019] Quan Vuong, Yiming Zhang, and Keith W. Ross. SUPERVISED POLICY UPDATE. In *International Conference on Learning Representations*, 2019.

[Wang *et al.*, 2015] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

A Operator Notation

We use vector and operator notation, where v^π and q^π represent the value and Q-value functions as vectors in $\mathcal{V} \equiv \mathbb{R}^{|\mathcal{S}|}$ and $\mathcal{Q} \equiv \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$. We denote the partial order $x \geq y \iff x(s,a) \geq y(s,a) \ \forall s,a \in \mathcal{S}\times\mathcal{A}$, and the norm $\|x\| = \max_{s,a}|x|$. Let us define two mappings between $\mathcal{V}$ and $\mathcal{Q}$:

1. $\mathcal{Q} \to \mathcal{V}$ mapping Π^π: $\left(\Pi^\pi q^\pi\right)(s) = \sum_{a\in\mathcal{A}} \pi(a|s) q^\pi(s,a) = v^\pi(s)$.
   This mapping is used to backup state-action-values to state value.
2. $\mathcal{V} \to \mathcal{Q}$ mapping $\tilde{\mathcal{P}}$: $\left(\tilde{\mathcal{P}} v^\pi\right)(s,a) = \sum_{s'\in\mathcal{S}} P(s'|s,a) v^\pi(s')$.
   This mapping is used to backup state-action-values based on future values. Note that this mapping is not dependent on a specific policy.

Let us further define two probability operators:

1. $\mathcal{V} \to \mathcal{V}$ state to state transition probability P_v^π:
   $\left(P_v^\pi v^\pi\right)(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}} P(s'|s,a) v^\pi(s')$.
2. $\mathcal{Q} \to \mathcal{Q}$ state-action to state-action transition probability P_q^π:
   $\left(P_q^\pi q^\pi\right)(s,a) = \sum_{s'\in\mathcal{S}} P(s'|s,a) \sum_{a'\in\mathcal{A}} \pi(a'|s') q^\pi(s',a')$.

Note that the probability operators are bounded linear transformations with spectral radius $\sigma(P_v^\pi) \leq 1$ and $\sigma(P_q^\pi) \leq 1$ [Puterman, 2014]. Prominent operators in the MDP theory are the recursive value operator $T_v^\pi v = \Pi^\pi r + \gamma P_v^\pi v$ and recursive Q-value operator $T_q^\pi q = r + \gamma P_q^\pi q$. Both have a unique solution for the singular value equations $v = T_v^\pi v$ and $q = T_q^\pi q$. These solutions are v^π and q^π respectively [Sutton and Barto, 2017; Puterman, 2014]. In addition, in the proofs we will use the following easy to prove properties:

1. $P_q^\pi = \tilde{\mathcal{P}}\Pi^\pi$
2. $P_v^\pi = \Pi^\pi \tilde{\mathcal{P}}$
3. $x \geq y, \ \ x,y \in \mathcal{V}, \ \Rightarrow \tilde{\mathcal{P}}x \geq \tilde{\mathcal{P}}y$
4. The probability of transforming from s to s' in k steps is $P(s \xrightarrow{k} s'|\pi) = (P_v^\pi)^k(s,s')$.
5. $[(P_v^\pi)^k x](s) = \mathbb{E}_{s_k\sim\pi|s}\left[x(s_k)\right]$

B Rerouted Behavior Improvement Proofs

B.1 Soft Policy Improvement

The soft policy improvement rule states that every policy π that satisfies

$$\sum_a \pi(a|s) A^\beta(s,a) \geq 0 \ \ \forall s \in \tilde{\mathcal{S}}$$

improves the policy β s.t. $V^\pi \geq V^\beta \ \forall s$. To avoid stagnation we demand that the inequality be strict for at least a single state. In operator notation, the last equation can be written as

$$\Pi^\pi a^\beta \geq 0$$

Proof. Plugging in $a^\beta = q^\beta - v^\beta$ and using $v^\beta = \Pi^\pi v^\beta$, we get that

$$v^\beta \leq \Pi^\pi q^\beta = \Pi^\pi(r + \gamma\tilde{\mathcal{P}} v^\beta) = r^\pi + \gamma P_v^\pi v^\beta. \tag{5}$$

Then, by applying (5) recursively we get

$$v^\beta \leq r^\pi + \gamma P_v^\pi v^\beta = r^\pi + \gamma P_v^\pi (r^\pi + \gamma P_v^\pi v^\beta) \leq r^\pi + \gamma P_v^\pi r^\pi + \gamma^2 (P_v^\pi)^2 v^\beta \leq r^\pi + \gamma P_v^\pi r^\pi + \gamma^2 (P_v^\pi)^2 r^\pi + \gamma^3 (P_v^\pi)^3 v^\beta \leq \ldots \leq \sum_{k\geq 0} \gamma^k (P_v^\pi)^k r^\pi = (\mathcal{I} - \gamma P_v^\pi)^{-1} r^\pi = v^\pi.$$

□

[Schulman *et al.*, 2015] have shown that

$$J(\pi) - J(\beta) = \mathbb{E}_{T\sim\pi}\left[\sum_{k\geq 0} \gamma^k A^\beta(s_k, a_k)\right].$$

This equation can also be written as

$$J(\pi) - J(\beta) = \sum_{k\geq 0} \gamma^k \mathbb{E}_{s_k \sim \pi|s_0} \left[\sum_{a\in\mathcal{A}} \pi(a|s_k) A^\beta(s_k, a_k) \right] =$$

$$= \sum_{k\geq 0} \gamma^k \sum_{s\in\mathcal{S}} P(s_0 \xrightarrow{k} s|\pi) \sum_{a\in\mathcal{A}} \pi(a|s) A^\beta(s,a) =$$

$$\sum_{s\in\mathcal{S}} \left(\sum_{k\geq 0} \gamma^k P(s_0 \xrightarrow{k} s|\pi) \right) \sum_{a\in\mathcal{A}} \pi(a|s) A^\beta(s_k, a).$$

Recall the definition of the discounted distribution of states

$$\rho^\pi(s) = \sum_{k\geq 0} \gamma^k P(s_0 \xrightarrow{k} s|\pi),$$

we conclude that

$$J(\pi) - J(\beta) = \sum_{s\in\mathcal{S}} \rho^\pi(s) \sum_{a\in\mathcal{A}} \pi(a|s) A^\beta(s,a)$$

B.2 Rank-Based Policy Improvement

Proof. Denote by i the index of the advantage ordered list $\{A_i\}$. Since $\sum_{a_i} \beta_i A_i^\beta = 0$, we can write it as an equation of the positive and negative advantage components

$$\sum_{a=i_p}^{N} \beta_i A_i^\beta = \sum_{a=0}^{i_n} \beta_i (-A_i^\beta),$$

where i_p is the minimal index of positive advantages and i_n is the maximal index of negative advantages. Since all probability ratios are non-negative, it is sufficient to show that

$$\sum_{a=i_p}^{N} c_i \beta_i A_i^\beta \geq \sum_{a=0}^{i_n} c_i \beta_i (-A_i^\beta).$$

But clearly

$$\sum_{a=i_p}^{N} c_i \beta_i A_i^\beta \geq c_{i_p} \sum_{a=i_p}^{N} \beta_i A_i^\beta \geq c_{i_n} \sum_{a=0}^{i_n} \beta_i (-A_i^\beta) \geq \sum_{a=0}^{i_n} c_i \beta_i (-A_i^\beta).$$

□

B.3 Policy Improvement Penalty

Let $\hat{Q}^\beta$ be an approximation of Q^β with an error $\varepsilon(s,a) = (Q^\beta - \hat{Q}^\beta)(s,a)$ and let π be a policy that satisfies the soft policy improvement theorem with respect to $\hat{Q}^\beta$. Then the following holds:

$$V^\pi(s) - V^\beta(s) \geq - \sum_{s'\in\mathcal{S}} \left(\sum_{k\geq 0} \gamma^k P(s \xrightarrow{k} s'|\pi) \right) \sum_{a\in\mathcal{A}} \varepsilon(s',a) \left(\beta(a|s') - \pi(a|s') \right). \tag{6}$$

The proof resembles the Generalized Policy improvement theorem [Barreto *et al.*, 2017].

Proof. We will use the equivalent operator notation. Define the vector $\varepsilon = q^\beta - \hat{q}^\beta$

$$\begin{aligned} T_q^\pi \hat{q}^\beta = r + \gamma\tilde{\mathcal{P}}\Pi^\pi \hat{q}^\beta &\geq r + \gamma\tilde{\mathcal{P}}\Pi^\beta \hat{q}^\beta \\ &= r + \gamma\tilde{\mathcal{P}}\Pi^\beta q^\beta - \gamma\tilde{\mathcal{P}}\Pi^\beta \varepsilon \\ &= T_q^\beta q^\beta - \gamma P_q^\beta \varepsilon \\ &= q^\beta - \gamma P_q^\beta \varepsilon \\ &= \hat{q}^\beta + (1 - \gamma P_q^\beta)\varepsilon \end{aligned}$$

.

Note that the inequality is valid, since $\Pi^\pi \hat{q}^\beta \geq \Pi^\beta \hat{q}^\beta$ (by the theorem assumptions), and if $v \geq u$ then $\tilde{\mathcal{P}}v \geq \tilde{\mathcal{P}}u$. Set $y = (1 - \gamma P_q^\beta)\varepsilon$ and notice that $T_q^\pi(\hat{q}^\beta + y) = T_q^\pi \hat{q}^\beta + \gamma P_q^\pi y$. By induction, we show that

$$(T_q^\pi)^n \hat{q}^\beta \geq \hat{q}^\beta + \sum_{k=0}^{n} \gamma^k (P_q^\pi)^k y.$$

We showed it for $n = 1$, assume it holds for n, then for $n+1$ we obtain

$$(T_q^\pi)^{n+1}\hat{q}^\beta = T_q^\pi (T_q^\pi)^n \hat{q}^\beta \geq T_q^\pi \left(\hat{q}^\beta + \sum_{k=0}^{n} \gamma^k (P_q^\pi)^k y \right) =$$
$$= T_q^\pi \hat{q}^\beta + \sum_{k=0}^{n} \gamma^{k+1} (P_q^\pi)^{k+1} y \geq \hat{q}^\beta + y + \sum_{k=0}^{n} \gamma^{k+1} (P_q^\pi)^{k+1} = \hat{q}^\beta + \sum_{k=0}^{n+1} \gamma^k (P_q^\pi)^k y.$$

Using the contraction properties of T_q^π, s.t. $\lim_{k\to\infty} (T_q^\pi)^k x = q^\pi, \forall x \in \mathcal{Q}$ and plugging back $(1 - \gamma P_q^\beta)\varepsilon = y$, we obtain

$$\begin{aligned} q^\pi = \lim_{n\to\infty} (T_q^\pi)^n (\hat{q}^\beta) &\geq \hat{q}^\beta + \sum_{k\geq 0} \gamma^k (P_q^\pi)^k (1 - \gamma P_q^\beta)\varepsilon \\ &= \hat{q}^\beta + \varepsilon + \sum_{k\geq 0} \gamma^{k+1} (P_q^\pi)^k (P_q^\pi - P_q^\beta)\varepsilon. \end{aligned}$$

Applying Π^π we transform back into $\mathcal{V}$ space

$$\begin{aligned} v^\pi &\geq \Pi^\pi \hat{q}^\beta + \Pi^\pi \varepsilon + \sum_{k\geq 0} \gamma^{k+1} \Pi^\pi (P_q^\pi)^k (P_q^\pi - P_q^\beta)\varepsilon \\ &\geq \Pi^\beta \hat{q}^\beta + \Pi^\pi \varepsilon + \sum_{k\geq 0} \gamma^{k+1} \Pi^\pi (P_q^\pi)^k (P_q^\pi - P_q^\beta)\varepsilon \\ &= v^\beta + (\Pi^\pi - \Pi^\beta)\varepsilon + \sum_{k\geq 0} \gamma^{k+1} \Pi^\pi (P_q^\pi)^k (P_q^\pi - P_q^\beta)\varepsilon \end{aligned}$$

.

Notice that $\Pi^\pi (P_q^\pi)^k P_q^\pi = \Pi^\pi (\tilde{\mathcal{P}}\Pi^\pi)^k \tilde{\mathcal{P}}\Pi^\pi = (\Pi^\pi \tilde{\mathcal{P}})^{k+1}\Pi^\pi = (P_v^\pi)^{k+1}\Pi^\pi$, and in the same manner, $\Pi^\pi (P_q^\pi)^k P_q^\beta = (P_v^\pi)^{k+1}\Pi^\beta$. Therefore, we can write

$$v^\pi \geq v^\beta - \sum_{k\geq 0} \gamma^k (P_v^\pi)^k (\Pi^\beta - \Pi^\pi)\varepsilon,$$

which may also be written as (6). □

B.4 Reroute is a subset of TV

The set of reroute policies with $[c_{\min}, c_{\max}]$ is a subset of the set of δ-TV policies, where $\delta = \min(1 - c_{\min}, \max(\frac{c_{\max}-1}{2}, \frac{1-c_{\min}}{2}))$.

Proof. For a reroute policy, the TV distance is $\frac{1}{2}\sum_a \beta(a_i|s)|1 - c_i| \leq \sum_a \beta(a_i|s)\delta = \delta$. For the second upper bound $1 - c_{\min}$, notice that a rerouted policy $\{\pi_i\}$ may be written as $\pi_i = c_{\min}\beta_i + \delta_i$, where $\delta_i \geq 0$. Therefore, the TV distance is

$$TV = \frac{1}{2}\sum_i |\pi_i - \beta_i| = \frac{1}{2}\sum_i |c_{\min}\beta_i + \delta_i - \beta_i| \leq \frac{1}{2}\sum_i \left((1 - c_{\min})\beta_i + \delta_i\right),$$

but $1 = \sum_i \pi_i = \sum_i (c_{\min}\beta_i + \delta_i) = c_{\min} + \sum_i \delta_i$. Thus, $\sum_i \delta_i = 1 - c_{\min}$ and we may write

$$TV \leq \frac{1}{2}\left((1 - c_{\min}) + (1 - c_{\min})\right) = 1 - c_{\min}.$$

To show that the TV region is not in reroute, take any policy with a zero probability action and another action with a probability of $\beta(a_i|s) \leq \delta$. The policy which switches probabilities between these two actions is in $TV(\delta)$ but it is not in reroute for any finite $c_{\max}$. □

B.5 Unbounded Probability Ratios in the TV and KL constraints

To verify that TV and KL do not necessarily regulate the probability ratios, let's consider a tiny example of maximizing the objective function J^{PPO} for a single state MDP with two actions $\{a_0, a_1\}$. Assume a behavior policy $\beta = \{1, 0\}$ and an estimated advantage $A^\beta = \{0, 1\}$. We search for a policy $\pi = \{1-\alpha, \alpha\}$ that Maximize improvement step under the TV or KL constraints.

For a δ-TV constraint, $\frac{1}{2}(1 - (1-\alpha) + \alpha) = \alpha \le \delta$. The improvement step in this case is $\sum_{a_i} A_i^\beta \pi_i = \alpha$. Hence the solution is $\alpha = \delta$ and the probability ratio $\pi(a_1)/\beta(a_1)$ is unconstrained (and undefined). Similarly for a δ-KL constraint we get $-\log \alpha \le \delta$ and the improvement step is identical. Hence $\alpha = e^{-\delta}$ and again, no constraint is posed on the probability ratio.

B.6 Max-TV

Algorithm 4 Max-TV

```
Require: $s$, $\beta$, $A^{\mathcal{D}}$, $\delta$
  $\pi(a|s) \longleftarrow \beta(a|s), \forall a \in \mathcal{A}$
  $a = \arg\max_{a \in \tilde{\mathcal{A}}} A^{\mathcal{D}}(s,a)$
  $\Delta = \min\{\delta, 1 - \beta(a|s)\}$
  $\pi(a|s) \longleftarrow \pi(a|s) + \Delta$
  $\tilde{\mathcal{A}} \longleftarrow \mathcal{A}$
  while $\Delta > 0$ do
    $a = \arg\min_{a \in \tilde{\mathcal{A}}} A^{\mathcal{D}}(s,a)$
    $\Delta_a = \min\{\Delta, \beta(a|s)\}$
    $\tilde{\mathcal{A}} \longleftarrow \tilde{\mathcal{A}}/a$
    $\Delta \longleftarrow \Delta - \Delta_a$
    $\pi(a|s) \longleftarrow \pi(a|s) - \Delta_a$
  end while
  return $\{\pi(a|s),\ a \in \mathcal{A}\}$
```

B.7 The PPO Objective Function

[Schulman *et al.*, 2017] suggested a surrogate objective function, termed *Proximal Policy Optimization* (PPO), that heuristically should provide a reliable performance as in TRPO without the complexity of a TRPO implementation:

$$J^{PPO}(\pi) = \mathbb{E}_{s\sim\beta}\left[\sum_{a\in\mathcal{A}} \beta(a|s) \min\left(A^\beta(s,a)\frac{\pi(a|s)}{\beta(a|s)}, A^\beta(s,a)\,\text{clip}\left(\frac{\pi(a|s)}{\beta(a|s)}, 1-\varepsilon, 1+\varepsilon\right)\right)\right],$$

where ε is a hyperparameter. PPO tries to ground the probability ratio by penalizing negative advantage actions with probability ratios above $1-\varepsilon$. In addition, it clips the objective for probability ratios above $1+\varepsilon$ so there is no incentive to move the probability ratio outside the interval $[1-\varepsilon, 1+\varepsilon]$. However, we show in the following that the solution of PPO is not unique and is dependent on the initial conditions, parametric form and the specific optimization implementation. This was also experimentally found in [Henderson *et al.*, 2017]. The effect of all of these factors on the search result is hard to design or predict. Moreover, some solutions may have unbounded probability ratios, in this sense, J^{PPO} is not safe.

First, notice that PPO maximization can be achieved by ad hoc maximizing each state since for each state the objective argument is independent and there are no additional constraints. Now, for state s, let's divide $\mathcal{A}$ into two sets: the set of positive advantage actions, denoted $\mathcal{A}^+$, and the set of negative advantage actions, $\mathcal{A}^-$. For convenience, denote $c_i = \frac{\pi(a_i|s)}{\beta(a_i|s)}$ and $\beta_i = \beta(a_i|s)$. Then, we can write the PPO objective of state s as

$$J^{PPO}(\pi, s) = \sum_{a_i\in\mathcal{A}^+} \beta_i A_i^\beta \min(c_i, 1+\varepsilon) - \sum_{a_i\in\mathcal{A}^-} \beta_i(-A_i^\beta)\max(c_i, 1-\varepsilon).$$

Clearly maximization is possible (yet, still not unique) when setting all $c_i = 0$ for $a_i \in \mathcal{A}^-$, namely, discarding negative advantage actions. This translates into a reroute maximization with parameters $(c_{\min}, c_{\max}) = (0, 1+\varepsilon)$

$$\arg\max_{c_i}[J^{PPO}(\pi,s)] = \arg\max_{c_i}\left[\sum_{a_i\in\mathcal{A}^+}\beta_i A_i^\beta \min(c_i, 1+\varepsilon)\right] = \arg\max_{c_i}\left[\sum_{a_i\in\mathcal{A}^+} c_i\beta_i A_i^\beta\right]$$

for $c_i \le 1+\varepsilon$. The only difference is that the sum $\sum_i c_i\beta_i = 1-\Delta$ may be less then 1. In this case, let us take the unsafe approach and dispense Δ to the highest ranked advantage. It is clear that partition of Δ is not unique, and even negative advantage actions may receive part of it as long as their total probability is less than $(1-\varepsilon)\beta_i$. We summarize this procedure in the following algorithm.

Algorithm 5 Max-PPO

```
Require: $s$, $\beta$, $A^\beta$, $\varepsilon$
  $\tilde{A} = \{\tilde{A}^+, \tilde{A}^-\}$
  $\tilde{A} \longleftarrow \mathcal{A}^+$
  $\Delta \longleftarrow 1$
  $\pi(a|s) \longleftarrow 0 \; \forall a \in \mathcal{A}$
  while $\Delta > 0$ and $|\tilde{A}| > 0$ do
    $a = \arg\max_{a \in \tilde{A}} A^\beta(s,a)$
    $\Delta_a = \min\{\Delta, (1+\varepsilon)\beta(a|s)\}$
    $\tilde{A} \longleftarrow \tilde{A}/a$
    $\Delta \longleftarrow \Delta - \Delta_a$
    $\pi(a|s) \longleftarrow \pi(a|s) + \Delta_a$
  end while
  $a = \arg\max_{a \in \mathcal{A}} A^\beta(s,a)$
  $\pi(a|s) \longleftarrow \pi(a|s) + \Delta$
  return $\{\pi(a|s), \; a \in \mathcal{A}\}$
```

C Learning from Observations of Human players: Technical Details

C.1 Dataset Preprocessing

The dataset [Kurin *et al.*, 2017] (Version 2) does not contain an end-of-life signal. Therefore, we tracked the changes of the life icons in order to reconstruct the end-of-life signal. The only problem with this approach was in Qbert where the last life period has no apparent icon but we could not match a blank screen since, during the episode, life icons are flashed on and off. Thus, the last end-of-life signal in Qbert is missing. Further, the dataset contained some defective episodes in which the screen frames did not match the trajectories. We found them by comparing the last score in the trajectory file to the score that appears in the last or near last frame.

We also found some discrepancies between the Javatari simulator used to collect the dataset and the Stella simulator used by the atari_py package:

- The Javatari reward signal has an offset of -2 frames. We corrected this shift in the preprocessing phase.
- The Javatari actions signal has an offset of ~ -2 frames (depending on the action type), where it is sometimes recorded non-deterministically s.t. the character executed an action in a frame before the action appeared in the trajectory file. We corrected this shift in the preprocessing phase.
- The Javatari simulator is not deterministic, while Stella can be defined as deterministic. This has the effect that icons and characters move in different mod_4 order, which is crucial when learning with frame skipping of 4. Thus, we evaluated the best offset (in terms of score) for each game and sampled frames according to this offset.
- There is a minor difference in colors/hues and objects location between the two simulators.

C.2 Network Architecture

A finite dataset introduces overfitting issues. To avoid overfitting, DQN constantly updates a replay buffer with new samples. In a finite dataset this is not possible, but, contrary to many Deep Learning tasks, partitioning into training and validation sets is also problematic: random partitions introduce a high correlation between training and testing, and blocking partitioning [Racine, 2000] might miss capturing parts of the action-states space. Fortunately, we found that Dropout [Srivastava *et al.*, 2014], as a source of regularization, improves the network's resiliency to overfitting. In addition, it has the benefit of better generalization to new unseen states since the network learns to classify based on a partial set of features. We added two sources of dropout: (1) in the input layer (25%) and (2) on the latent features i.e. before the last layer (50%).

We also found that for a finite dataset with a constant statistics, Batch Normalization (BN) can increase the learning rate. Therefore, we used BN layer before the ReLu nonlinearities. Note that for an iterative RL algorithm, BN may sometimes impede learning because the statistics may change during the learning process. To summarize, we used two DQN style networks: one for β and the second for Q^β. We used the same network architecture as in [Mnih *et al.*, 2015] except for the following add-ons:

- A batch normalization layer before nonlinearities.
- A 25% Dropout in the input layer.

- A 50% Dropout in the latent features layer. To compensate for the dropped features we expanded the latent layer size to 1024, instead of 512 as in the original DQN architecture.
- An additional output that represents the state's value.

The advantage for the PPO objective was estimated with $A_i = Q_i - \sum_i \beta_i Q_i$.

C.3 Hyperparameters tables

Table 3: Policy and Q-value networks Hyperparameters

Name	Value
Last linear layer size	1024
Optimizer	Adam
Learning Rate	0.00025
Dropout [first layer]	0.25
Dropout [last layer]	0.5
Minibatch	32
Iterations	1562500
Frame skip	4
Reward clip	-1,1

Table 4: DQfD Hyperparameters

Name	Value
Optimizer	Adam
Learning Rate	0.0000625
n-steps	10
Target update period	16384
l_{a_E}	0.8
Priority Replay exponent	0.4
Priority Replay IS exponent	0.6
Priority Replay constant	0.001
Other parameters	As in policy/value networks (except Dropout)

Table 5: Final scores table

Method	MsPacman	Qbert	Revenge	SpaceInvaders
Humans	3024	3401	1436	634
Behavioral cloning	1519	8562	1761	678
Reroute-$(0.5, 1.5)$	1550	12123	2379	**709**
Reroute-$(0.25, 1.75)$	**1638**	13254	**2431**	682
Reroute-$(0, 2)$	1565	**14494**	473	675
TV(0.25)	1352	5089	390	573
PPO(0.5)	1528	14089	388	547
DQfD	83	1404	1315	402

D RL with RBI

The Q-net architecture is identical to Duel DQN [Wang *et al.*, 2015] and the π-net architecture is identical to the advantage branch in Duel DQN with an additional final softmax layer. Q-net and π-net are saved to ram-disk every 100 batches and each actor process loads a new snapshot every 1K environment steps (this is slightly different from Ape-X where policies are loaded from the learner every 400 environment frames). The target-Q is updated every 2500 training batches (as in the original paper). The actors' samples are stored in the dataset only at episode termination and they are sampled uniformly by the learner ones every 5K batches. This design was chosen in order to facilitate the implementation and it clearly generates lags in the training process. Nevertheless, we found that those lags are not necessarily negative since they serve as a momentum for the learning process where past networks are used to generate samples and past samples are used in the learning process. Such learning momentum can prevent the network from converging to a local optimum, however, we leave the trade-off between fast updates and learning momentum for future research. Another parameter that may generate lags and momentum is the pace of samples stored in the dataset by the parallel actors. We tried to control this parameter to an average of 240-280 new states for a single training batch in the learner. Learning rate and optimization parameters are taken from [Hessel *et al.*, 2017], i.e. Adam optimization with a learning rate of $0.00025/4$ for both networks. We do not clip gradient norm.

For training, as in Ape-X, we also capped the episode length to 50K frames. This helps hard exploration games such as Berzerk but it limits the score of easier games such as Spaceinvaders, Enduro and Asterix. The latter games may be considered as solved, converged into near optimal policy which rarely loses life and its score is limited only by the allowed episode length. For final score evaluation we capped the episode to 108K frames as in Rainbow and started the environment with the no-op initialization method. In the table below we list the final score across the 12 environments. The column of Ape-X represents the Ape-X baseline implementation as described in the experimental setup. The Rainbow score is taken as a reference from [Hessel *et al.*, 2017]. Surprisingly, our Ape-X baseline provided better results from the original Ape-X paper [Horgan *et al.*, 2018] in 5 games: Asterix, Enduro, Freeway, IceHockey and Kangaroo, in much less wall clock time (roughly 24 hours) and significantly lower computer resources (a single machine with 48 cores and 4 Nvidia 1080Ti GPUs). We hypothesize that the different learning momentum may be the staple reason for that.

Table 6: Hyperparameters table

Name	Value
$c_{\max}$	2
$c_{\min}$	0.1
c_{greedy}	0.1
c_{mix}	$\frac{m}{43.7+m}$
priority exponent α	0.5
Batch	128
N players	256

Note that m is the number of actions.