StaQ it! Growing neural networks for Policy Mirror Descent

Alena Shilova, Alex Davey, Brahim Driss, Riad Akrour

StaQ it! Growing neural networks for Policy Mirror Descent

Alena Shilova[1], Alex Davey[2], Brahim Driss[2], and Riad Akrour[2]

[1]Inria TAU team, LISN, Université Paris-Saclay, Orsay, France
[1]Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 – CRIStAL, Lille, France

Abstract

In Reinforcement Learning (RL), regularization has emerged as a popular tool both in theory and practice, typically based either on an entropy bonus or a Kullback-Leibler divergence that constrains successive policies. In practice, these approaches have been shown to improve exploration, robustness and stability, giving rise to popular Deep RL algorithms such as SAC and TRPO. Policy Mirror Descent (PMD) is a theoretical framework that solves this general regularized policy optimization problem, however the closed-form solution involves the sum of all past Q-functions, which is intractable in practice. We propose and analyze PMD-like algorithms that only keep the last M Q-functions in memory, and show that for finite and large enough M, a convergent algorithm can be derived, introducing no error in the policy update, unlike prior deep RL PMD implementations. StaQ, the resulting algorithm, enjoys strong theoretical guarantees and is competitive with deep RL baselines, while exhibiting less performance oscillation, paving the way for fully stable deep RL algorithms and providing a testbed for experimentation with Policy Mirror Descent.

1 Introduction

Deep RL has seen rapid development in the past decade, achieving super-human results on several decision making tasks (Mnih et al., 2015; Silver et al., 2016; Wurman et al., 2022). However, the use of neural networks as function approximators exacerbates many challenges of RL, such as the difficulties of exploration and brittleness to hyperparameters (Henderson, 2018). Furthermore, the empirical behavior often poorly aligns with our theoretical understandings (Ilyas et al., 2020; Kumar et al., 2020; van Hasselt et al., 2018). To address these issues, many successful deep RL algorithms consider regularized versions of the original objective, typically either by regularizing the Bellman operators with an entropy bonus e.g. SAC (Haarnoja et al., 2018) or by introducing a KL constraint between successive policies, e.g. TRPO (Schulman et al., 2015).

Policy Mirror Descent (PMD, Abbasi-Yadkori et al. (2019); Lazic et al. (2021); Zhan et al. (2023)) applies Mirror Descent (Nemirovsky & Yudin, 1983; Beck & Teboulle, 2003), a first-order convex optimization method, to the policy improvement step. In the more general entropy-regularized form of PMD, starting with the previous Q-function Q^k, the improved policy at each iteration is the solution of the following optimization problem

$$\pi_{k+1}(s) = \arg\max_{p \in \Delta(A)} \left\{ \mathbb{E}_{a \sim p}[Q^k(s,a)] - \tau h(p) - \eta D_{\mathcal{R}}(p, \pi_k) \right\} \tag{1}$$

for some entropy weight $\tau \geq 0$ and (inverse) step size $\eta > 0$, where h is the negative Shannon entropy and $D_{\mathcal{R}}$ is the Bregman divergence associated with the convex regularizer $\mathcal{R}$ and π_k is the previous policy. This has received a lot of recent theoretical interest as a unifying framework for many regularized policy iteration algorithms (Neu et al., 2017; Geist et al., 2019).

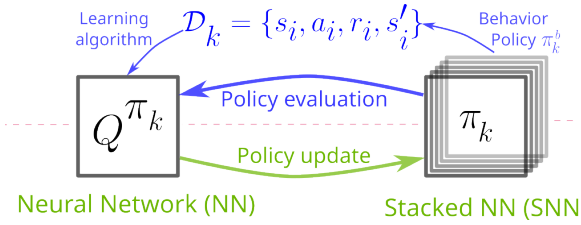


Figure 1: Overview of StaQ, showing the continual training of a Q-function (left), from which we periodically "stack" frozen weight snapshots to form the policy (right). See Sec. 5 for more details. At each iteration k, two steps are performed. i) Policy evaluation, where we generate a dataset $\mathcal{D}_k$ of transitions that are gathered by a behavior policy π_k^b, typically derived from π_k, and then learn Q^{π_k} from $\mathcal{D}_k$. ii) Policy update, performed by "stacking" the NN of Q^{π_k} into the current policy. The policy update is optimization-free and theoretically grounded (Sec. 4), thus only the choice of π_k^b and the policy evaluation algorithm can remain sources of instabilities in this deep RL setting.

When also using the negative entropy as the convex regularizer, the Bregman divergence reduces to the KL-divergence between successive policies, the resulting policy that solves Eq. 1 at each iteration k is given by a weighted average of past Q-functions

$$\pi_k \propto \exp\left(\alpha \sum_{i=0}^{k} \beta^i Q^{k-i}\right), \tag{2}$$

with temperature $\alpha := 1/(\eta + \tau)$ and decay factor $\beta := \eta/(\eta + \tau)$ (see Sec. 3 for more details). The averaging over previous Q-functions induced by the D_{KL} regularizer is known to average out approximation errors over the true Q-functions and has stabilizing properties in practice (Geist et al., 2019; Abbasi-Yadkori et al., 2019).

The sum in Eq. 2 can be computed exactly if the Q-function is a linear function of some fixed feature space, as summing Q-functions is equivalent to summing their weights. Beyond that, for non-linear function approximators such as neural networks, no closed form update for Eq. 2 exists in parameter space, requiring the storage of all past Q-functions, which is intractable. As such, prior work considered several type of approximations to the policy update in Eq. 1, such as following the natural gradient as in TRPO (Schulman et al., 2015) or performing a few gradient steps over Eq. 1 as in MDPO (Tomar et al., 2020).

Instead, we consider a PMD-like algorithm that implements a policy similar to Eq. 2, but where at most M Q-functions are stored. This corresponds to solving Eq. 1 approximately, replacing π_k in the D_{KL} regularization with a slightly altered policy $\tilde{\pi}_k$, for which we have deleted the oldest Q-function (Sec. 4). Abbasi-Yadkori et al. (2019) performed an experiment of the sort on an RL task, keeping in memory the past 10 Q-functions, and noted increased stability and performance over vanilla DQN, but provided no theoretical justification for keeping a finite set of Q-functions, which we adress in this paper. Interestingly, we show that for M large enough, replacing π_k with $\tilde{\pi}_k$ will not hinder the asymptotic convergence to $\pi^\star$.

This paper extends prior work on PMD (Abbasi-Yadkori et al., 2019; Lazic et al., 2021; Zhan et al., 2023) by proposing a provably converging finite-memory PMD-like algorithm (see Fig. 1), that both has strong theoretical guarantees and is fully implementable with promising empirical performance. In detail, our contributions are as follows:

i) We theoretically study the convergence of PMD-like algorithms that store up to M Q-functions, and show that this finite-memory algorithm still converges if M is large enough.

ii) We show that by batching the Q-functions we can efficiently compute the full stack of Q-functions on GPU in parallel.

iii) We show on a large set of tasks that StaQ, the resulting Deep RL algorithm, with its closed-form entropy regularized policy update, is competitive with deep RL baselines on a wide range of MuJoCo (discrete-action setting) and MinAtar environments, while demonstrating stabilized learning, bringing us closer to a completely stable deep RL algorithm.

2 Related Work

Regularization in RL. Regularization has seen widespread usage in RL. It was used with (natural) policy gradient ((N)PG) (Kakade, 2001; Schulman et al., 2015; Yuan et al., 2022), policy search (Deisenroth et al., 2013), policy iteration (Abbasi-Yadkori et al., 2019; Zhan et al., 2023) and value iteration methods (Fox et al., 2016; Vieillard et al., 2020b). Common choices of regularizers include minimizing the D_{KL} between the current and previous policy (Azar et al., 2012; Schulman et al., 2015) or encouraging high Shannon entropy (Fox et al., 2016; Haarnoja et al., 2018), but other regularizers exist (Lee et al., 2019; Alfano et al., 2023). We refer the reader to Neu et al. (2017); Geist et al. (2019) for a broader categorization of entropy regularizers and their relation to existing deep RL methods. In this paper, we use both a D_{KL} penalization w.r.t. the previous policy and a Shannon entropy bonus in a policy iteration context. In Vieillard et al. (2020b), both types of regularizers were used but in a value iteration context. Abbasi-Yadkori et al. (2019); Lazic et al. (2021) are policy iteration methods but only use D_{KL} penalization.

Policy Mirror Descent. Policy Mirror Descent is a family of policy optimization algorithms that can be all characterized by a similar objective functions, where a new policy is found by solving Eq. 1. Prior works on PMD focus mostly on performing a theoretical analysis of convergence speeds or sample complexity for different choices of regularizers (Li et al., 2022; Johnson et al., 2023; Alfano et al., 2023; Zhan et al., 2023; Lan, 2022; Protopapas & Barakat, 2024). As PMD provides a general framework for many regularized RL algorithms, PMD theoretical results can be naturally extended to many policy gradient algorithms like Natural PG (Khodadadian et al., 2021) and TRPO (Schulman et al., 2015) as shown in Neu et al. (2017); Geist et al. (2019). However, the deep RL algorithms from the PMD family generally perform inexact policy updates, adding an additional source of error from the theoretical perspective. For example, TRPO and the more recent MDPO (Tomar et al., 2020) rely on approximate policy updates using policy gradients.

We build on (Abbasi-Yadkori et al., 2019; Lazic et al., 2021; Zhan et al., 2023) by proposing a finite-memory variant, proving the new convergence results and offering a new deep RL algorithm policy update step that does not introduce any additional error, in contrast to prior works.

Growing neural architectures and ensemble methods in RL. Saving past Q-functions has previously been investigated in the context of policy evaluation. In Tosatto et al. (2017), a first Q-function is learned, then frozen and a new network is added, learning the residual error. Shi et al. (2019) uses past Q-functions to apply Anderson acceleration for a value iteration type of algorithm. Anschel et al. (2017) extend DQN by saving the past 10 Q-functions, and using them to compute lower variance target values. Instead of past Q-functions, Chen et al. (2021); Lee et al. (2021); Agarwal et al. (2020); Lan et al. (2020) use an ensemble of independent Q network functions to stabilize Q-function learning in DQN type of algorithms. The aforementioned works are orthogonal to ours, as they are concerned with learning one Q, while policy evaluation in StaQ is a secondary choice. Conversely, both Girgin & Preux (2008) and Della Vecchia et al. (2022) use a special neural architecture called the cascade-correlation network (Fahlman & Lebiere, 1989) to grow neural policies.

The former work studies such policies in combination with LSPI (Lagoudakis & Parr, 2003), without entropy regularization. The latter work is closer to ours, using a D_{KL}-regularizer but without a deletion mechanism. As such the policy grows indefinitely, limiting the scaling of the method. Finally, Abbasi-Yadkori et al. (2019) save the past 10 Q-functions to compute the policy in Eq. 2 for the specific case of $\beta = 1$, but do not study the impact of deleting older Q-functions as we do in this paper. Growing neural architectures are more common in the neuroevolution community (Stanley & Miikkulainen, 2002), and have been used for RL, but are beyond the scope of this paper.

Parallels with Continual Learning. Continual Learning (CL) moves from the usual i.i.d assumption of supervised learning towards a more general assumption that data distributions change through time (Parisi et al., 2019; Lesort et al., 2020; De Lange et al., 2021; Wang et al., 2024). This problem is closely related to that of incrementally computing π_k in Eq. 2, due to the differing data distributions that each Q-function is trained on, and our approach of using a growing architecture to implement a KL-regularized policy update is inspired by parameter isolation methods in the CL literature, which offer some of the best stability-performance trade-offs (see Sec. 6 in De Lange et al. (2021)). Parameter isolation methods were explored in the context of continual RL (Rusu et al., 2016), yet remain understudied in a standard *single-task* RL setting.

3 Preliminaries

Let a Markov Decision Problem (MDP) be defined by the tuple (S, A, R, P, γ), such that S and A are finite state and action spaces, R is a bounded reward function $R : S \times A \mapsto [-R_{\text{x}}, R_{\text{x}}]$ for some positive constant R_{x}, P defines the (Markovian) transition probabilities of the decision process and γ is a discount factor. The algorithms presented in this paper can be extended to more general state spaces. However, the limitation to a finite A is non-trivial to lift due to the sampling from softmax distributions as in Eq. 2. We discuss in Sec. 7 potential ways to address this limitation.

Let $\Delta(A)$ be the space of probability distributions over A, and h be the negative entropy given by $h : \Delta(A) \mapsto \mathbb{R}$, $h(p) = p \cdot \log p$, where $\cdot$ is the dot product and the log is applied element-wise to the vector p. Let $\pi : S \mapsto \Delta(A)$ be a stationary stochastic policy mapping states to distributions over actions. We denote the entropy regularized V-function for policy π and regularization weight $\tau > 0$ as $V_\tau^\pi : S \mapsto \mathbb{R}$, which is defined by $V_\tau^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \{R(s_t, a_t) - \tau h(\pi(s_t))\} | s_0 = s\right]$. In turn, the entropy regularized Q-function is given by $Q_\tau^\pi(s, a) = R(s, a) + \gamma \mathbb{E}_{s'} \left[V_\tau^\pi(s')\right]$. The V-function can be written as the expectation of the Q-function plus the current state entropy, i.e. $V_\tau^\pi(s) = \mathbb{E}_a \left[Q_\tau^\pi(s, a)\right] - \tau h(\pi(s))$ which leads to the Bellman equation $Q_\tau^\pi(s, a) = R(s, a) + \gamma \mathbb{E}_{s', a'} \left[Q_\tau^\pi(s', a') - \tau h(\pi(s'))\right]$. In the following, we will write policies of the form $\pi(s) \propto \exp(Q(s, \cdot))$ for all $s \in S$ more succinctly as $\pi \propto \exp(Q)$. We define optimal V and Q functions where for all $s \in S, a \in A$, $V_\tau^\star(s) := \max_\pi V_\tau^\pi(s)$ and $Q_\tau^\star(s, a) := \max_\pi Q_\tau^\pi(s, a)$.

Moreover, the policy $\pi^\star \propto \exp\left(\frac{Q_\tau^\star}{\tau}\right)$ satisfies $Q_\tau^{\pi^\star} = Q_\tau^\star$ and $V_\tau^{\pi^\star} = V_\tau^\star$ simultaneously for all $s \in S$ (Zhan et al., 2023). In the following, we will overload notations of real functions defined on $S \times A$ and allow them to only take a state input and return a vector in $\mathbb{R}^{|A|}$. For example, $Q_\tau^\pi(s)$ denotes a vector for which the i^{th} entry $i \in \{1, \ldots, |A|\}$ is equal to $Q_\tau^\pi(s, i)$. Finally we define $\bar{R} := \frac{R_{\text{x}} + \gamma\tau \log |A|}{1-\gamma}$, as the finite upper-bound of $\|Q_\tau^\pi\|_\infty$ for any policy π, that can be computed by assuming the agent collects the highest reward and entropy possible at every step.

3.1 Entropy-regularized policy mirror descent

To find $\pi^\star$, we focus on Entropy-regularized Policy Mirror Descent (EPMD) methods (Neu et al., 2017; Abbasi-Yadkori et al., 2019; Lazic et al., 2021) and notably on those that regularize both the policy update and the Q-function (Lan, 2022; Zhan et al., 2023). The PMD setting discussed here is also equivalent to the regularized natural policy gradient algorithm on softmax policies of Cen et al. (2022). Let π_k be the policy at iteration k of EPMD, and $Q_\tau^k := Q_\tau^{\pi_k}$ its Q-function. The next policy in EPMD is the solution of the following optimization problem:

$$\forall s \in S, \quad \pi_{k+1}(s) = \arg\max_{p \in \Delta(A)} \{Q_\tau^k(s) \cdot p - \tau h(p) - \eta D_{\mathrm{KL}}(p; \pi_k(s))\} \tag{3}$$

$$\propto \pi_k(s)^{\frac{\eta}{\eta+\tau}} \exp\left(\frac{Q_\tau^k(s)}{\eta+\tau}\right), \tag{4}$$

where $D_{\mathrm{KL}}(p; p') = p \cdot (\log p - \log p')$ and $\eta > 0$ is the D_{KL} regularization weight. The closed form expression in Eq. 4 is well-known and its proof can be checked in, e.g. Vieillard et al. (2020a). We let $\alpha = \frac{1}{\eta+\tau}$ and $\beta = \frac{\eta}{\eta+\tau}$, hereafter referred to as a step-size and a decay factor respectively.

Let ξ_k be a real function of $S \times A$ for any positive integer k. We assume as the initial condition that $\pi_0 \propto \exp(\xi_0)$ with $\xi_0 = 0$, i.e. π_0 is uniform over the actions. At every iteration of EPMD, the update in Eq. 4 yields the following logits update

$$\pi_{k+1} \propto \exp(\xi_{k+1}), \quad \xi_{k+1} = \beta \xi_k + \alpha Q_\tau^k. \tag{5}$$

From the recursive definition of ξ_{k+1}, it can easily be verified that $\xi_{k+1} = \alpha \sum_{i=0}^{k} \beta^{k-i} Q_\tau^i$. The convergence of EPMD is characterized by the following theorem

Theorem 3.1 (Adapted from Zhan et al. (2023), Thm. 1)**.** *At iteration k of EPMD, the Q-function of π_k satisfies $\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty \leq \gamma d^{k-1} \left(\left\|Q_\tau^\star - Q_\tau^0\right\|_\infty + 2\beta \left\|Q_\tau^\star\right\|_\infty\right)$, with $d = \beta + \gamma(1-\beta) < 1$.*

The above theorem shows that by following EPMD, we have a linear convergence of Q_τ^k towards $Q_\tau^\star$, with a convergence rate of d. In the next section, we will be interested in an approximate version of EPMD, where the Q-function Q_τ^k is computed exactly but where ξ_k is limited to summing at most M Q-functions. We name this setting finite-memory EPMD. In the main paper, we only focus for clarity on the error introduced by this deletion mechanism in the policy update. The theoretical analysis of our algorithm that takes into account errors in policy evaluation is deferred to App. A.

4 Finite-memory policy mirror descent

Let $M > 0$ be a positive integer defining the maximum number of Q-functions we are allowed to store. As a warm-up, we first show in Sec. 4.1 a straightforward implementation of finite-memory EPMD, where we simply truncate the sum of the Q-functions in Eq. 2 to the last M Q-functions.

The main step in this analysis is to quantify the effect of the finite-memory assumption on the policy improvement theorem. As in the class of approximate algorithms analyzed in Zhan et al. (2023), the algorithm in Sec. 4.1 always exhibits an irreducible error for a finite M. To address this issue, we introduce a weight corrected algorithm in Sec. 4.2 that rescales the policy in Eq. 2 to account for its finite-memory nature. This rescaling introduces long range dependencies that complicate the analysis, but can result in convergence to $Q_\tau^\star$, without residual error, provided a large but finite M.

4.1 Vanilla finite-memory EPMD

Consider an approximate EPMD setting where the update to ξ_k is given by

$$\xi_{k+1} = \beta\xi_k + \alpha\left(Q_\tau^k - \beta^M Q_\tau^{k-M}\right), \tag{6}$$

with $Q_\tau^{k-M} := 0$ whenever $k - M < 0$.

Compared to ξ_{k+1} in Eq. 5, we both add the new Q_τ^k and 'delete' an old Q-function by subtracting Q_τ^{k-M} in Eq. 6. As a result, ξ_{k+1} can now be written as $\xi_{k+1} = \alpha\sum_{i=0}^{M-1}\beta^i Q_\tau^{k-i}$, which is a finite-memory EPMD algorithm using at most M Q-functions.

We now want to investigate if we have any convergence guarantees of Q_τ^k towards $Q_\tau^\star$ as for EPMD. Let the policy $\tilde{\pi}_k$ be defined by $\tilde{\pi}_k \propto \exp(\tilde{\xi}_k)$ with $\tilde{\xi}_k = \alpha\sum_{i=0}^{M-2}\beta^i Q_\tau^{k-1-i}$. Here, $\tilde{\xi}_k = \xi_k - \alpha\beta^{M-1}Q_\tau^{k-M}$, i.e. it is obtained by deleting the oldest Q-function from ξ_k and thus is a sum of $M-1$ Q-functions. The update in Eq. 6 can now be rewritten as $\xi_{k+1} = \beta\tilde{\xi}_k + \alpha Q_\tau^k$. From Sec. 3, we recognize this update as the result of the following optimization problem:

$$\text{for all } s \in S, \quad \pi_{k+1}(s) = \arg\max_{p\in\Delta(A)}\{Q_\tau^k(s)\cdot p - \tau h(p) - \eta D_{\mathrm{KL}}(p;\tilde{\pi}_k(s))\}. \tag{7}$$

In this approximate scheme, we compute the D_{KL} regularization w.r.t. $\tilde{\pi}_k$ instead of the previous policy π_k. This can negatively impact the quality of π_{k+1} as it might force π_{k+1} to stay close to the potentially bad policy $\tilde{\pi}_k$. In the following theorem, we provide a form of an approximate policy improvement of π_{k+1} on π_k, that depends on how close $\tilde{\pi}_k$ is to π_k. This theorem applies to any policy $\tilde{\pi}_k$, therefore it can be of interest beyond the scope of this paper.

Theorem 4.1 (Approximate policy improvement)**.** *Let $\pi_k \propto \exp(\xi_k)$ be a policy with associated Q-function Q_τ^k. Let $\tilde{\pi}_k \propto \exp(\tilde{\xi}_k)$ be an arbitrary policy. Let π_{k+1} be the policy optimizing Eq. 7 w.r.t. the hereby defined Q_τ^k and $\tilde{\pi}_k$, then the Q-function Q_τ^{k+1} of π_{k+1} satisfies*

$$Q_\tau^{k+1} \geq Q_\tau^k - \gamma\eta\frac{\max_{s\in S}\left\|(\pi_k - \tilde{\pi}_k)(s)\right\|_1 \left\|\xi_k - \tilde{\xi}_k\right\|_\infty}{1-\gamma}. \tag{8}$$

The proof of Thm. 4.1 and all future proofs are given in App. A. Applying Thm. 4.1 to our setting gives the following policy improvement lower bound

Corollary 4.1.1. *Let $\pi_k \propto \exp(\xi_k)$ be a policy with associated Q-function Q_τ^k, such that $\xi_k = \alpha\sum_{i=0}^{M-1}\beta^i Q_\tau^{k-1-i}$. Let $\tilde{\pi}_k \propto \exp(\tilde{\xi}_k)$ be the policy such that $\tilde{\xi}_k = \alpha\sum_{i=0}^{M-2}\beta^i Q_\tau^{k-1-i}$. Let π_{k+1} be the policy optimizing Eq. 7, then the Q-function Q_τ^{k+1} of π_{k+1} satisfies*

$$Q_\tau^{k+1} \geq Q_\tau^k - \gamma\beta^M\frac{\min\left\{2, \alpha\beta^{M-1}\bar{R}\right\}\bar{R}}{1-\gamma}. \tag{9}$$

In vanilla EPMD, it is guaranteed that $Q_\tau^{k+1} \geq Q_\tau^k$ (Zhan et al., 2023). In this approximate setting, the error is arbitrarily close to 0 through the term β^M by choosing a large enough M, since $\beta < 1$.

Having quantified the error in the policy improvement step, we follow the general steps of the proof of approximate EPMD of Zhan et al. (2023) and come to the following convergence guarantees.

Theorem 4.2 (Convergence of vanilla finite-memory EPMD)**.** *After $k \geq 0$ iterations of Eq. 6, we have that $\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty \leq \gamma d^k \left\|Q_\tau^\star\right\|_\infty + \beta^M C_1$, with $d = \beta + \gamma(1-\beta) < 1$ and $C_1 = \frac{2\gamma\bar{R}}{1-\gamma}\left(1 + \frac{\gamma(1-\beta^M)}{(1-\beta)(1-\gamma)}\right)$.*

Convergence of finite-memory EPMD is still at a rate of d as with exact EPMD. However, we eventually reach an error of $\beta^M C_1$, that does not decrease as k increases, and that we can only control by increasing the memory size M. A problem with the current algorithm is that even if all past Q-functions are equal to $Q_\tau^\star$, then $\tau\xi_k = (1-\beta)\sum_{i=0}^{M-1}\beta^i Q_\tau^\star = (1-\beta^M)Q_\tau^\star$, whereas we know that asymptotically ξ_k should converge to the logits of $\pi^\star$ (Sec. 3) which are $\frac{Q_\tau^\star}{\tau}$. This suggests a slightly modified algorithm that rescales ξ_k by $1-\beta^M$, which we analyze in the next section.

4.2 Weight corrected finite-memory EPMD

Consider now the alternative update to ξ_k given by

$$\xi_{k+1} = \beta\xi_k + \alpha Q_\tau^k + \frac{\alpha\beta^M}{1-\beta^M}(Q_\tau^k - Q_\tau^{k-M}), \tag{10}$$

where $Q_\tau^{k-M} := 0$ whenever $k - M < 0$. In contrast to the vanilla algorithm in Sec. 4.1, we now delete the oldest Q-function in ξ_k and also slightly overweight the most recent Q-function to ensure that the Q-function weights sum to 1. Indeed, assuming that $\xi_0 := 0$, we can show (see App. A.4.1 for a proof) for all $k \geq 0$ that the logits only use the past M Q-functions and are given by

$$\xi_{k+1} = \frac{\alpha}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i Q_\tau^{k-i}. \tag{11}$$

Similar to the previous section, we introduce a policy $\tilde{\pi}_k \propto \exp(\tilde{\xi}_k)$ with $\tilde{\xi}_k = \xi_k + \frac{\alpha\beta^{M-1}}{1-\beta^M}(Q_\tau^k - Q_\tau^{k-M})$ such that logits of π_{k+1} are given by $\xi_{k+1} = \beta\tilde{\xi}_k + \alpha Q_\tau^k$. This form of ξ_{k+1} implies that π_{k+1} satisfies the policy update in Eq. 7, and thus Thm. 4.1 applies and we have

Corollary 4.1.2. *Let $\pi_k \propto \exp(\xi_k)$ be a policy with associated Q-function Q_τ^k, such that $\xi_k = \frac{\alpha}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i Q_\tau^{k-1-i}$. Let $\tilde{\pi}_k \propto \exp(\tilde{\xi}_k)$ be the policy such that $\tilde{\xi}_k = \xi_k + \frac{\alpha\beta^{M-1}}{1-\beta^M}(Q_\tau^k - Q_\tau^{k-M})$. Let π_{k+1} be the policy optimizing Eq. 7 with the hereby defined Q_τ^k and $\tilde{\pi}_k$, then the Q-function Q_τ^{k+1} of π_{k+1} satisfies*

$$Q_\tau^{k+1} \geq Q_\tau^k - 2\gamma\beta^M\frac{\left\|Q_\tau^k - Q_\tau^{k-M}\right\|_\infty}{(1-\gamma)(1-\beta^M)}. \tag{12}$$

Compared to the approximate policy improvement of Sec. 4.1, we see that the lower-bound in Eq. 12 depends on $\left\|Q_\tau^k - Q_\tau^{k-M}\right\|_\infty$ instead of just $\left\|Q_\tau^{k-M}\right\|_\infty$. Thus, we can expect that as the Q-functions converge to $Q_\tau^\star$, we get tighter and tighter guarantees on the policy improvement step, which in turn guarantees convergence to $Q_\tau^\star$ without the residual error of Sec. 4.1. The next two results show that indeed, for M large enough, the finite-memory EPMD scheme defined by Eq. 10 leads to convergence to $Q_\tau^\star$. Lem. 4.3 provides an upper bounding sequence for $\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty$.

Lemma 4.3. *Let $x_{k+1} = d_1 x_k + d_2 x_{k-M}$ be a sequence such that $\forall k < 0$, $x_k = \frac{\|Q_\tau^\star\|_\infty}{\gamma}$, $x_0 = \|Q_\tau^\star\|_\infty + \left\|Q_\tau^0\right\|_\infty$, $d_1 := \beta + \gamma\frac{1-\beta}{1-\beta^M} + \gamma c_2$, $d_2 := \frac{2c_1\gamma^2}{1-\gamma}$, $c_1 := \frac{\beta^M}{1-\beta^M}$, and $c_2 := \left(\frac{1+\gamma}{1-\gamma} - \beta\right)c_1$. After $k \geq 0$ iterations of Eq. 10, we have that $\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty \leq x_k$.*

Then, we compute values of M for which the sequence x_k converges to 0 and characterize the convergence rate of $\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty$ through Thm. 4.4.

Theorem 4.4 (Convergence of weight corrected finite-memory EPMD)**.** *With the definitions of Lemma 4.3, if $M > \log\frac{(1-\gamma)^2(1-\beta)}{\gamma^2(3+\beta)+1-\beta}(\log\beta)^{-1}$ then $\lim_{k\to\infty} x_k = 0$. Moreover, $\forall k \geq 0$,*

$\left\|Q^\star_\tau - Q^k_\tau\right\|_\infty \leq (d_1 + d_2 d_3^{-1})^k \max\left\{\frac{\|Q^\star_\tau\|_\infty}{\gamma}, \|Q^\star_\tau\|_\infty + \left\|Q^0_\tau\right\|_\infty\right\}$, *where* $d_3 := \left(d_1^M + d_2 \frac{1-d_1^M}{1-d_1}\right)$ *and* $\lim_{M\to\infty} d_1 + d_2 d_3^{-1} = \beta + \gamma(1-\beta)$.

Thm. 4.4 defines a minimum memory size that guarantees convergence to $Q^\star_\tau$. This minimum M depends only on β and γ, and is usually within the range of practical values: for example, with $\gamma = 0.99$ and $\beta = 0.95$, the minimum M suggested by Thm. 4.4 is 265, which is reasonable in terms of memory and computation with current GPUs (we used $M = 300$ in all our experiments). As can be expected, these values of M are generally pessimistic and even with higher values of β, we did not observe in practice better performance when using as large M as suggested by Thm. 4.4.

In terms of convergence rate, $d_1 + d_2 d_3^{-1}$ given in Thm 4.4 tends to d—the convergence rate of exact EPMD—as M goes to infinity. Thus, it is slower than exact EPMD, and slower than the algorithm in Sec. 4.1, but unlike the latter it does not have an irreducible error and converges to $Q^\star_\tau$.

5 Practical implementation

Since we only require a finite number of M Q-functions in Thm. 4.4 (for sufficiently large M), we can exactly implement the policy update step by stacked neural networks (SNN, illustrated in Fig. 1). By using batched operations we make efficient use of GPUs and compute multiple Q-values in parallel. We call the resulting algorithm StaQ. After each policy evaluation, we push the weights corresponding to this new Q-function onto the stack. If the stacked NN contains more than M NNs, the oldest NN is deleted in a "first in first out" fashion.

To further reduce the impact of a large M, we pre-compute ξ_k for all entries in the replay buffer[1] at the start of policy evaluation. The logits ξ_k are used to sample on-policy actions when computing the targets for Q^k_τ. As a result of the pre-computation, during policy evaluation, forward and backward passes only operate on the current Q-function and hence the impact of large M is minimized, however rolling out the current behavioural policy π^b_k still requires a full forward pass. Conversely, the policy update consists only of adding the new weights to the stack, and thus, is optimization free and (almost) instantaneous. Table 1 shows the training time of StaQ as a function of M for two environments. Varying M or the state space size has little impact on the runtime of StaQ on GPU, at least for these medium-sized environments.

The NN for Q^0_τ is initialized with an output of zero, so that π_0 is a uniform policy, and for all consecutive iterations the NN for Q^k_τ is initialized at the computed Q^{k-1}_τ (to make the transfer from Q^k_τ to Q^{k-1}_τ smoother). Similarly to SAC (Haarnoja et al., 2018), we learn Q^k_τ by sampling an action from the current policy and using an ensemble of two target Q-functions updated in a hard manner.

While it is natural to use the stochastic policy π_k for the behavioural policy, we find it beneficial to instead consider an ϵ-*softmax* policy over π_k — by analogy with ϵ-greedy policies, mixing the softmax policy π_k and a uniform policy with probabilities $(1-\epsilon)$ and ϵ respectively. This provides a hard minimum sampling probability for all actions, even when the policy π_k learns to suppress some actions. Using only π_k can cause instabilities in the Q-function learning, as discussed in App. B.4 and App. B.2. For further implementation details and the full set of hyperparameters consult App. E.

Table 1: Training times for StaQ (5 million steps), as a function of M, on Hopper-v4 (state dim.=11) and Ant-v4 (state dim. = 105), computed on an NVIDIA Tesla V100 and averaged over 3 seeds.

	Memory size M	1	50	100	300	500
Hopper-v4	Training time (hrs)	9.8	10.1	10.3	10.3	10.9
Ant-v4	Training time (hrs)	10.4	10.7	10.3	11	10.5

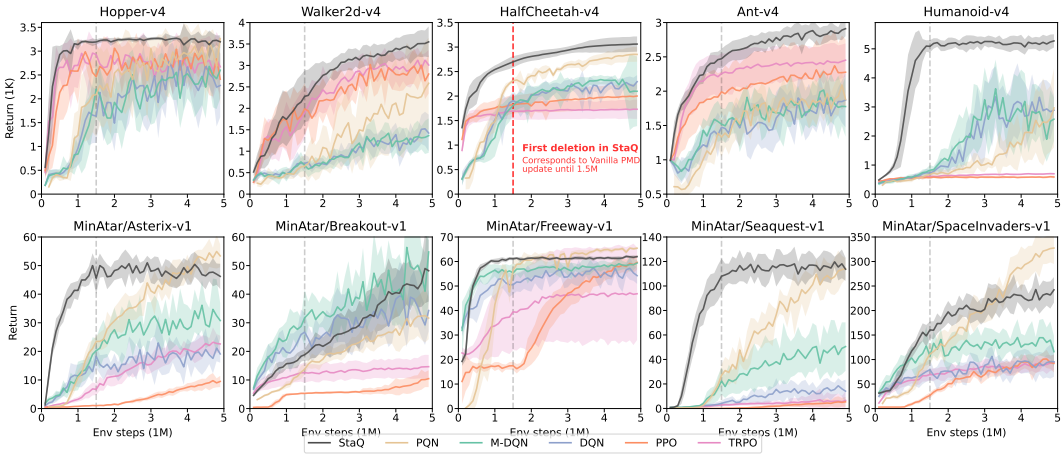


Figure 2: Policy return of StaQ and of deep RL baselines. Plots show mean and one standard deviation computed over 10 seeds. StaQ has consistent performance in both the MuJoCo and MinAtar domains. See App. B.1 for additional results.

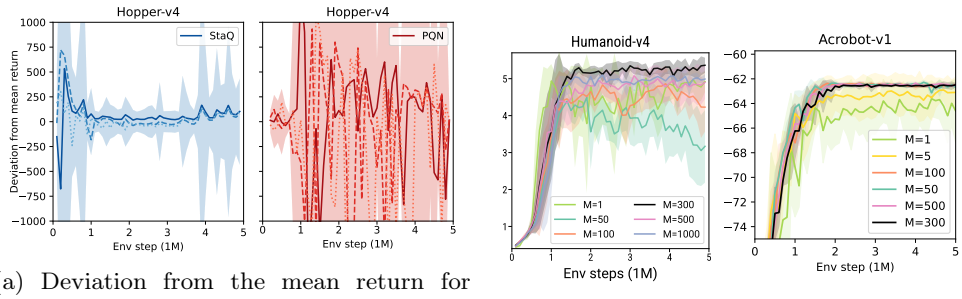
6 Experiments

In this section, we assess the empirical merits of StaQ, paying attention to both the performance and stability, and then discuss some limitations of our algorithms and how they open new perspectives towards a fully reliable deep RL solver.

Environments. We use all 9 environments suggested by Ceron & Castro (2021) for comparing deep RL algorithms with finite action spaces, comprising 4 classic control tasks from Gymnasium (Towers et al., 2023), and all MinAtar tasks (Young & Tian, 2019). To that we add 5 Mujoco tasks (Todorov et al., 2012), adapted to discrete action spaces by considering only extreme actions similarly to (Seyde et al., 2021). To illustrate, the discrete version of a Mujoco task with action space $A = [-1, 1]^d$ consists in several $2d$ dimensional vectors that have zeroes everywhere except at entry $i \in \{1, \dots, d\}$ that can either take a value of 1 or -1; to that we add the zero action, for a total of $2d + 1$ actions.

Baselines. We compare StaQ against the value iteration algorithm DQN (Mnih et al., 2015) and its entropy-regularized variant M-DQN (Vieillard et al., 2020b), the policy gradient algorithm TRPO (Schulman et al., 2015) as it uses a D_{KL} regularizer and PPO (Schulman et al., 2017). StaQ performs entropy regularization on top of a Fitted-Q Iteration (FQI) approach. DQN only uses FQI and is a good baseline to measure the impact of entropy regularization over vanilla FQI, while the other baselines cover a wide range of alternative approaches to entropy regularization in deep RL: through a bonus term (M-DQN), following the natural gradient (TRPO) or with a clipping loss (PPO). SAC (Haarnoja et al., 2018) is another popular deep RL baseline that uses entropy regularization but is not adapted for discrete action environments. However, M-DQN is a close alternative to SAC for discrete

[1]Since we use small replay buffer sizes of 50K transitions, we are likely to process each transition multiple times (25.6 times in expectation in our experiments) making this optimization worthwhile.



(a) Deviation from the mean return for StaQ/PQN.

(b) Impact of memory size M.

Figure 3: **Left:** Deviation from mean policy returns for individual runs on Hopper, comparing StaQ and PQN, the best performing baseline on this environment. Returns are centered at every timestep by subtracting the mean across 10 seeds. In general, individual runs of StaQ have significantly lower variance across timesteps compared to baselines. For clarity, we plot the first three seeds, and one-sided tolerance intervals. See App B.2 for further environments and algorithms. **Right:** Policy returns under different choice of M. On the simpler Acrobot task, $M > 5$ seems sufficient but on Humanoid, even $M = 100$ is insufficient. Plots showing mean and one std computed over 5 seeds.

action spaces as discussed in App. C. Finally, we compare to PQN (Gallici et al., 2025), a recent algorithm that builds on DQN, replacing the target networks with LayerNorm regularisation (Ba et al., 2016), and adding λ-returns (Daley & Amato, 2020). This baseline provides an example of more complex systems which incorporate improvements in policy evaluation orthogonal to our proposed policy update, other examples of such algorithms include Rainbow (Hessel et al., 2017). Comparisons with baselines are averaged over 10 seeds, showing the mean and standard deviation of the return. The return is computed every 100K steps by evaluating the current deterministic policy, averaging 50 rollouts. Hyperparameters for StaQ and the baselines are provided in App. E.

Performance. A comparison of StaQ to deep RL baselines is shown for a selection of environments in Fig. 2, and for all environments in App. B.1. For our setting of $M = 300$, the first deletion occurs at 1.5M timesteps, indicated by a vertical dashed line. Fig. 2 shows that StaQ has consistent performance, and is competitive with the baselines on most MuJoCo and MinAtar environments. In contrast, we see that PPO/TRPO underperforms on MinAtar tasks while M-DQN/DQN/PQN underperform on MuJoCo tasks. While StaQ strongly outperforms PQN in the majority of MuJoCo tasks, it is outperformed by PQN on some MinAtar tasks, especially in SpaceInvaders. PQN builds on DQN by introducing improvements that stabilize Q-function learning, suggesting that a similar direction — further improving the policy evaluation strategy — may similarly improve the performance of StaQ, as discussed in Sec. 7.

Stability. Beyond pure performance, StaQ typically exhibits less performance oscillation when looking at the variability within individual runs, especially in the MuJoCo domain. For example, in Fig. 3 (**Left**), we plot the variation of the return for each seed, centered by subtracting the mean return across all seeds at each evaluation timestep. We see that PQN, while achieving a final performance similar to StaQ on Hopper, exhibits significantly more performance oscillation. Stability comparisons on more environments and all baselines are provided in App. B.2. These experiments confirm the preliminary results of Abbasi-Yadkori et al. (2019) that a policy averaging over multiple Q-functions stabilizes learning. While prior work considered only saving the last 10 Q-functions, we show next that, on more complex tasks, saving an order of magnitude more Q-functions can still have positive effects on stability and performance.

Impact of the memory-size M. According to Sec. 4.2, M is a crucial parameter that should be large enough to guarantee convergence. The M estimation obtained from Thm. 4.4 may be very conservative in practice. In Fig. 3 (**Right**), we present the results for different choices of M for "easy" Acrobot and "difficult" Humanoid. While a low value of $M \leq 100$ ($M \leq 10$ for Acrobot) can still achieve a decent mean performance, stability is negatively affected, which is especially pronounced for more challenging environments such as Humanoid.

Conversely, higher $M = 500, 1000$, while more expensive to compute, does not generally lead to an improvement either in terms of performance or stability. We found $M = 300$ to be a good compromise between stability and compute time. See App B.2 for more environments.

7 Discussion and future work

In this paper, we proposed a policy update rule based on Policy Mirror Descent, that by using a novel re-weighting scheme, results in a convergent policy when storing a finite number of M Q-functions, provided M is sufficiently large. Surprisingly, even when M is large, the final computational burden is small on modern hardware, due to stacking of the Q-functions. The resulting policy update has a solid theoretical foundation and clear empirical benefits as it improves performance and reduces learning instability compared to other entropy regularization methods in the literature.

While the policy update is more stable, some instability in learning the Q-function remains. In App. B.4, we describe an ablation where we compare the ϵ-softmax policy with a pure softmax policy (i.e. $\epsilon = 0$) that hints at Q-learning instability. Even though those issues are mitigated thanks to the averaging over Q-functions of StaQ, they suggest that policy evaluation errors remain significant. Due to its exact policy update, StaQ provides a promising setting for testing more sophisticated forms of policy evaluation, especially recent methods that use normalization techniques to reduce policy evaluation error (Gallici et al., 2025; Bhatt et al., 2024).

Finally, extending StaQ to continuous action domains could be done as in SAC (Haarnoja et al., 2018), using an extra actor network learned by minimizing the D_{KL} to a soft policy. This will lose the optimization-free and exact nature of the policy update but may still result in improved stability if we replace the soft policy $\exp(Q^k_\tau)$ used by SAC with $\exp(\xi_k)$, which stabilizes the target by averaging over a large number of past Q-functions.

Acknowledgements

A. Davey and B. Driss were funded by the project ANR-23-CE23-0006. This work was granted access to the HPC resources of IDRIS under the allocation 2024-AD011015599 made by GENCI.

References

Abbasi-Yadkori, Y., Bartlett, P., Bhatia, K., Lazic, N., Szepesvari, C., and Weisz, G. POLITEX: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, 2019.

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pp. 104–114. PMLR, 2020.

Alfano, C., Yuan, R., and Rebeschini, P. A novel framework for policy mirror descent with general parameterization and linear convergence. In *Advances in Neural Information Processing Systems*, 2023.

Anschel, O., Baram, N., and Shimkin, N. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*, 2017.

Azar, M. G., Gómez, V., and Kappen, H. J. Dynamic policy programming. *Journal of Machine Learning Research*, 2012.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer Normalization, July 2016.

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, May 2003. ISSN 0167-6377. doi: 10.1016/S0167-6377(02)00231-6.

Bhatt, A., Palenicek, D., Belousov, B., Argus, M., Amiranashvili, A., Brox, T., and Peters, J. CrossQ: Batch Normalization in Deep Reinforcement Learning for Greater Sample Efficiency and Simplicity, March 2024.

Cen, S., Cheng, C., Chen, Y., Wei, Y., and Chi, Y. Fast global convergence of natural policy gradient methods with entropy regularization. *Operations Research*, 2022.

Ceron, J. S. O. and Castro, P. S. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, 2021.

Chen, X., Wang, C., Zhou, Z., and Ross, K. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

Daley, B. and Amato, C. Reconciling λ-Returns with Experience Replay, January 2020.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Deisenroth, M. P., Neumann, G., and Peters, J. A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, 2013.

Della Vecchia, R., Shilova, A., Preux, P., and Akrour, R. Entropy regularized reinforcement learning with cascading networks. *arXiv*, 2022.

Fahlman, S. and Lebiere, C. The cascade-correlation learning architecture. *Advances in neural information processing systems*, 2, 1989.

Fox, R., Pakman, A., and Tishby, N. G-learning: Taming the noise in reinforcement learning via soft updates. In *Conference on Uncertainty in Artificial Intelligence*, 2016.

Gallici, M., Fellows, M., Ellis, B., Pou, B., Masmitja, I., Foerster, J. N., and Martin, M. Simplifying Deep Temporal Difference Learning, March 2025.

Geist, M., Scherrer, B., and Pietquin, O. A theory of regularized Markov decision processes. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Girgin, S. and Preux, P. Basis function construction in reinforcement learning using cascade-correlation learning architecture. In *IEEE International Conference on Machine Learning and Applications*, 2008.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft Actor-Critic Algorithms and Applications. In *International Conference on Machine Learning (ICML)*, 2018.

Henderson, P. Reproducibility and reusability in deep reinforcement learning. Master's thesis, McGill University, 2018.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining Improvements in Deep Reinforcement Learning, October 2017.

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, 2022. URL `http://jmlr.org/papers/v23/21-1342.html`.

Ilyas, A., Engstrom, L., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. A closer look at deep policy gradients. In *International Conference on Learning Representations*, 2020.

Johnson, E., Pike-Burke, C., and Rebeschini, P. Optimal convergence rate for exact policy mirror descent in discounted markov decision processes. *Advances in Neural Information Processing Systems*, 36:76496–76524, 2023.

Kakade, S. M. A natural policy gradient. In *Advances in Neural Information Processing Systems*, 2001.

Khodadadian, S., Jhunjhunwala, P. R., Varma, S. M., and Maguluri, S. T. On the linear convergence of natural policy gradient algorithm. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 3794–3799. IEEE, 2021.

Krishnamoorthy, K. and Mathew, T. *Statistical Tolerance Regions: Theory, Applications, and Computation*. John Wiley & Sons, May 2009. ISBN 978-0-470-47389-4.

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

Lagoudakis, M. G. and Parr, R. Least-squares policy iteration. *Journal of Machine Learning Research*, 2003.

Lan, G. Policy mirror descent for reinforcement learning: linear convergence, new sampling complexity, and generalized problem classes. *Mathematical Programming*, 2022.

Lan, Q., Pan, Y., Fyshe, A., and White, M. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

Lazic, N., Yin, D., Abbasi-Yadkori, Y., and Szepesvari, C. Improved regret bound and experience replay in regularized policy iteration. In *International Conference on Machine Learning*, 2021.

Lee, K., Kim, S., Lim, S., Choi, S., and Oh, S. Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning. *arXive*, 2019.

Lee, K., Laskin, M., Srinivas, A., and Abbeel, P. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6131–6141. PMLR, 2021.

Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., and Díaz-Rodríguez, N. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 2020.

Li, Y., Lan, G., and Zhao, T. Homotopic policy mirror descent: Policy convergence, implicit regularization, and improved sample complexity. *arXiv preprint arXiv:2201.09457*, 2022.

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*, 2015.

Nemirovsky, A. S. and Yudin, D. B. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley, 1983.

Neu, G., Jonsson, A., and Gómez, V. A unified view of entropy-regularized markov decision processes. *arXiv*, 2017.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, 2016.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Protopapas, K. and Barakat, A. Policy mirror descent with lookahead. *Advances in Neural Information Processing Systems*, 37:26443–26481, 2024.

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`.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *CoRR*, 2016.

Schulman, J., Levine, S., Jordan, M., and Abbeel, P. Trust Region Policy Optimization. *International Conference on Machine Learning*, 2015.

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

Seyde, T., Gilitschenski, I., Schwarting, W., Stellato, B., Riedmiller, M., Wulfmeier, M., and Rus, D. Is bang-bang control all you need? solving continuous control with bernoulli policies. In *Advances in Neural Information Processing Systems*, 2021.

Shi, W., Song, S., Wu, H., Hsu, Y.-C., Wu, C., and Huang, G. Regularized anderson acceleration for off-policy deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

Stanley, K. O. and Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 2002.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Tomar, M., Shani, L., Efroni, Y., and Ghavamzadeh, M. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020.

Tosatto, S., Pirotta, M., d'Eramo, C., and Restelli, M. Boosted fitted q-iteration. In *International Conference on Machine Learning*, pp. 3434–3443. PMLR, 2017.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., and Younis, O. G. Gymnasium, March 2023. URL `https://zenodo.org/record/8127025`.

van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv*, 2018.

Vieillard, N., Kozuno, T., Scherrer, B., Pietquin, O., Munos, R., and Geist, M. Leverage the average: an analysis of kl regularization in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020a.

Vieillard, N., Pietquin, O., and Geist, M. Munchausen reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020b.

Wang, L., Zhang, X., Su, H., and Zhu, J. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V. R., Lin, H., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M. D., Aghabozorgi, H., Barrett, L., Douglas, R., Whitehead, D., Dürr, P., Stone, P., Spranger, M., and Kitano, H. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 2022.

Young, K. and Tian, T. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Yuan, R., Du, S. S., Gower, R. M., Lazaric, A., and Xiao, L. Linear convergence of natural policy gradient methods with log-linear policies. *arXiv preprint arXiv:2210.01400*, 2022.

Zhan, W., Cen, S., Huang, B., Chen, Y., Lee, J. D., and Chi, Y. Policy mirror descent for regularized reinforcement learning: A generalized framework with linear convergence. *SIAM Journal on Optimization*, 2023.

A Proofs

This section includes proofs of the lemmas and theorems of the main paper.

A.1 Properties of entropy regularized Bellman operators

We first start with a reminder of some basic properties of the (entropy regularized) Bellman operators, as presented in (Geist et al., 2019). Within the MDP setting defined in Sec. 3, let T_τ^π be the operator defined for any map $f : S \times A \mapsto \mathbb{R}$ by

$$\left(T_\tau^\pi f\right)(s,a) = R(s,a) + \gamma \mathbb{E}_{s',a'}[f(s',a') - \tau h(\pi(s'))], \tag{13}$$

For this operator we will need the three following properties.

Proposition A.1 (Contraction)**.** *T_τ^π is a γ-contraction w.r.t. the $\|.\|_\infty$ norm, i.e. $\|T_\tau^\pi f - T_\tau^\pi g\|_\infty \leq \gamma \|f - g\|_\infty$ for any real functions f and g of $S \times A$.*

Proposition A.2 (Fixed point)**.** *Q_τ^π is the unique fixed point of the operator T_τ^π, i.e. $T_\tau^\pi Q_\tau^\pi = Q_\tau^\pi$.*

Let f, g be two real functions of $S \times A$. We say that $f \geq g$ iff $f(s,a) \geq g(s,a)$ for all $(s,a) \in S \times A$.

Proposition A.3 (Monotonicity)**.** *T_τ^π is monotonous, i.e. if $f \geq g$ then $T_\tau^\pi f \geq T_\tau^\pi g$.*

Let the Bellman optimality $T_\tau^\star$ operator be defined by

$$\left(T_\tau^\star f\right)(s,a) = R(s,a) + \gamma \mathbb{E}_{s'}\left[\max_{p \in \Delta(A)} f(s') \cdot p - \tau h(p)\right]. \tag{14}$$

For the Bellman optimality operator we need the following two properties.

Proposition A.4 (Contraction)**.** *$T_\tau^\star$ is a γ-contraction w.r.t. the $\|.\|_\infty$ norm, i.e. $\|T_\tau^\star f - T_\tau^\star g\|_\infty \leq \gamma \|f - g\|_\infty$ for any real functions f and g of $S \times A$.*

Proposition A.5 (Optimal fixed point)**.** *$T_\tau^\star$ admits $Q_\tau^\star$ as a unique fixed point, satisfying $T_\tau^\star Q_\tau^\star = Q_\tau^\star$.*

Finally, we will make use of the well known property that the softmax distribution is entropy maximizing (Geist et al., 2019). Specifically, we know that the policy π_k as defined in Eq. 5 satisfies the following property

$$\text{for all } s \in S, \quad \pi_k(s) = \arg\max_{p \in \Delta(A)} \xi_k(s) \cdot p - h(p), \tag{15}$$

A.2 Proof of Theorem 4.1

We present in this appendix proofs for a more general setting where the Q-functions are inexact. Results with exact policy evaluation of the main paper can be recovered by simply setting the policy evaluation error ϵ_{eval}, as defined below, to zero and by replacing $\tilde{Q}_\tau$ by Q_τ.

Assumption A.1. *We assume that we can only compute Q_τ^k approximately, which is the Q-value function of π_k. We use $\tilde{Q}_\tau^k$ to denote the approximate Q_τ^k and we assume that there exists $\epsilon_{eval} < \infty$ such that the following holds for any k*

$$\left\| Q_\tau^k - \tilde{Q}_\tau^k \right\|_\infty \leq \epsilon_{eval}. \tag{16}$$

Note that Eq. 16 implies that for any s, a,

$$|Q_\tau^k(s,a) - \tilde{Q}_\tau^k(s,a)| \leq \epsilon_{eval} \tag{17}$$

or equivalently

$$-\epsilon_{eval} \leq Q_\tau^k(s,a) - \tilde{Q}_\tau^k(s,a) \leq \epsilon_{eval}. \tag{18}$$

As the exact Q_τ^k is no longer available, the policy update is done in the inexact policy evaluation with $\tilde{Q}_\tau^k$:

$$\text{for all } s \in S, \quad \pi_{k+1}(s) = \arg\max_{p \in \Delta(A)} \left\{ \tilde{Q}_\tau^k(s) \cdot p - \tau h(p) - \eta D_{\mathrm{KL}}(p; \tilde{\pi}_k(s)) \right\}. \tag{19}$$

We restate below the approximate policy improvement theorem in its more general form, and Theorem 4.1 can be recovered for $\epsilon_{eval} = 0$.

Theorem A.1 (Approximate policy improvement with inexact policy evaluation)**.** *Let $\pi_k \propto \exp(\xi_k)$ be a policy with associated evaluated Q-function $\tilde{Q}_\tau^k$. Let $\tilde{\pi}_k \propto \exp(\tilde{\xi}_k)$ be an arbitrary policy. Let π_{k+1} be the policy optimizing Eq. 19 w.r.t. the hereby defined $\tilde{Q}_\tau^k$ and $\tilde{\pi}_k$, then the Q-function Q_τ^{k+1} of π_{k+1} satisfies*

$$Q_\tau^{k+1} \geq \tilde{Q}_\tau^k - \gamma\eta \frac{\max_{s \in S} \|(\pi_k - \tilde{\pi}_k)(s)\|_1 \left\| \xi_k - \tilde{\xi}_k \right\|_\infty}{1-\gamma} - \frac{1+\gamma}{1-\gamma}\epsilon_{eval}. \tag{20}$$

Proof. Let $\pi_k \propto \exp(\xi_k)$ and let $\tilde{\pi}_k \propto \exp(\tilde{\xi}_k)$ with $X := \xi_k - \tilde{\xi}_k$. Define π_{k+1} as in Eq. 19. From Sec. 3, we have that $\pi_{k+1} \propto \exp(\xi_{k+1})$ with the change that now an approximate $\tilde{Q}_\tau^k$ is used in the update:

$$\xi_{k+1} = \beta\tilde{\xi}_k + \alpha\tilde{Q}_\tau^k. \tag{21}$$

From the optimality of π_{k+1} w.r.t. the policy update optimization problem in Eq. 19 we have

$$\begin{aligned} \tilde{Q}_\tau^k(s) \cdot \pi_k(s) - \tau h(\pi_k(s)) &\leq \tilde{Q}_\tau^k(s) \cdot \pi_{k+1}(s) - \tau h(\pi_{k+1}(s)) \\ &\quad - \eta D_{\mathrm{KL}}(\pi_{k+1}(s); \tilde{\pi}_k(s)) + \eta D_{\mathrm{KL}}(\pi_k(s); \tilde{\pi}_k(s)), \qquad (22) \\ &\leq \tilde{Q}_\tau^k(s) \cdot \pi_{k+1}(s) - \tau h(\pi_{k+1}(s)) + \eta D_{\mathrm{KL}}(\pi_k(s); \tilde{\pi}_k(s)), \qquad (23) \end{aligned}$$

where the last inequality is due to the non-negativity of the D_{KL}.

Let us try now to upper bound $D_{\mathrm{KL}}(\pi_k(s); \tilde{\pi}_k(s))$ for any $s \in S$. For clarity, we will drop s from the notations, and only write for e.g. $\xi(a)$ instead of $\xi_k(s,a)$. We define $Z = \sum_a \exp(\xi(a))$ and $\tilde{Z} = \sum_a \exp(\tilde{\xi}(a))$, where the sums are over all $a \in A$.

$$\begin{aligned} D_{\mathrm{KL}}(\pi_k; \tilde{\pi}_k) &= \mathbb{E}_\pi[\log \pi(a) - \log \tilde{\pi}(a)], \qquad (24) \\ &= \mathbb{E}_\pi \left[\log \frac{\exp \xi(a)}{Z} - \log \frac{\exp \tilde{\xi}(a)}{\tilde{Z}} \right], \qquad (25) \\ &= \mathbb{E}_\pi \left[\xi(a) - \tilde{\xi}(a) - \log \frac{Z}{\tilde{Z}} \right], \qquad (26) \\ &= \mathbb{E}_\pi \left[X(a) - \log \frac{\sum_{a'} \exp(X(a')) \exp(\tilde{\xi}(a'))}{\sum_{a'} \exp(\tilde{\xi}(a'))} \right], \qquad (27) \\ &\overset{\text{(i)}}{\leq} \mathbb{E}_\pi \left[X(a) - \mathbb{E}_{\tilde{\pi}} \left[X(a') \right] \right], \qquad (28) \\ &= (\pi - \tilde{\pi}) \cdot X, \qquad (29) \end{aligned}$$

where (i) is due to Jensen's inequality. Replacing Eq. 29 into Eq. 23 yields

$$\tilde{Q}_\tau^k(s) \cdot \pi_k(s) - \tau h(\pi_k(s)) \leq \tilde{Q}_\tau^k(s) \cdot \pi_{k+1}(s) - \tau h(\pi_{k+1}(s)) + \eta(\pi - \tilde{\pi})(s) \cdot X(s). \tag{30}$$

For any $s \in S$, we have that

$$\eta(\pi - \tilde{\pi})(s) \cdot X(s) \leq \eta \max_{s \in S} |(\pi - \tilde{\pi})(s) \cdot X(s)|, \tag{31}$$

$$\overset{(i)}{\leq} \eta \max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1 \|X(s)\|_\infty, \tag{32}$$

$$= \eta \max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1 \|X\|_\infty, \tag{33}$$

$$:= \epsilon, \tag{34}$$

where we applied Hölder's inequality in (i). Combining Eq. 34 with Eq. 30 and using the definition of the operator T_τ^π as in Eq. 13 yields for any $s \in S$ and $a \in A$

$$\begin{aligned} R(s,a) + \gamma \mathbb{E}_{s'} \left[\tilde{Q}_\tau^k(s') \cdot \pi_k - \tau h(\pi_k(s')) \right] \leq R(s,a) + \gamma \mathbb{E}_{s'} [\tilde{Q}_\tau^k(s') \cdot \pi_{k+1}(s') \\ - \tau h(\pi_{k+1}(s')) + \epsilon], \end{aligned} \tag{35}$$

$$\implies (T_\tau^k \tilde{Q}_\tau^k)(s,a) \leq (T_\tau^{k+1} \tilde{Q}_\tau^k)(s,a) + \gamma\epsilon, \tag{36}$$

where $\epsilon = \eta \max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1 \|X\|_\infty$. Since Eq. 36 is valid for any s and a, then

$$T_\tau^k \tilde{Q}_\tau^k \leq T_\tau^{k+1} \tilde{Q}_\tau^k + \gamma\epsilon, \tag{37}$$

Let us have a closer look at $T_\tau^k \tilde{Q}_\tau^k$, if we use Eq. 16 and by the fixed point property of Prop. A.2 we have

$$T_\tau^k \tilde{Q}_\tau^k = T_\tau^k Q_\tau^k + \gamma \mathbb{E}_{s'} \left[(\tilde{Q}_\tau^k - Q_\tau^k) \cdot \pi_k \right] \geq Q_\tau^k - \gamma \epsilon_{eval}. \tag{38}$$

and similarly $T_\tau^{k+1} \tilde{Q}_\tau^k \leq T_\tau^{k+1} Q_\tau^k + \gamma \epsilon_{eval}$. Together, these imply

$$Q_\tau^k \leq T_\tau^{k+1} Q_\tau^k + \gamma\epsilon + 2\gamma\epsilon_{eval}. \tag{39}$$

The addition of $\gamma(\epsilon + 2\epsilon_{eval})$ in the above expression is performed element-wise for all states and actions. Using the monotonicity property of Prop. A.3 on Eq. 39, we have

$$T_\tau^{k+1} Q_\tau^k \leq T_\tau^{k+1} \left(T_\tau^{k+1} Q_\tau^k + \gamma(\epsilon + 2\epsilon_{eval}) \right), \tag{40}$$

$$\leq (T_\tau^{k+1})^2 Q_\tau^k + \gamma^2(\epsilon + 2\epsilon_{eval}), \tag{41}$$

$$\implies Q_\tau^k \leq (T_\tau^{k+1})^2 Q_\tau^k + \gamma(\epsilon + 2\epsilon_{eval}) + \gamma^2(\epsilon + 2\epsilon_{eval}). \tag{42}$$

By repeating the same process one can easily show by induction that

$$Q_\tau^k \leq (T_\tau^{k+1})^n Q_\tau^k + \sum_{i=1}^{n} \gamma^i (\epsilon + 2\epsilon_{eval}). \tag{43}$$

Taking the limit of Eq. 43 for $n \to \infty$ yields by the uniqueness of the fixed point of T_τ^{k+1}

$$Q_\tau^k \leq Q_\tau^{k+1} + \frac{\gamma(\epsilon + 2\epsilon_{eval})}{1 - \gamma}, \tag{44}$$

Finally,

$$\tilde{Q}_\tau^k \leq Q_\tau^{k+1} + \frac{\gamma\epsilon}{1-\gamma} + \frac{1+\gamma}{1-\gamma}\epsilon_{eval} \tag{45}$$

$$= Q_\tau^{k+1} + \gamma\eta \frac{\max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1 \left\| \xi_k - \tilde{\xi}_k \right\|_\infty}{1-\gamma} + \frac{1+\gamma}{1-\gamma}\epsilon_{eval}. \tag{46}$$

□

A.3 Approximate finite-memory EPMD

A.3.1 Proof of Corollary 4.1.1

Cor. 4.1.1 is a direct application of Thm. 4.1 with the specific values for ξ_k and $\tilde{\xi}_k$ of finite-memory EPMD as defined in Sec. 4.1.

Proof. To prove Cor. 4.1.1, we will bound the two terms $\eta \left\| \xi_k - \tilde{\xi}_k \right\|_\infty$ and $\max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1$ individually, using the fact that

$$\xi_k - \tilde{\xi}_k = \alpha \beta^{M-1} \tilde{Q}_\tau^{k-M}. \tag{47}$$

Let us first start with the term

$$\begin{aligned} \eta \left\| \xi_k - \tilde{\xi}_k \right\|_\infty &\overset{(i)}{=} \beta^M \left\| \tilde{Q}_\tau^{k-M} \right\|_\infty && (48) \\ &\leq \beta^M \bar{R} + \beta^M \epsilon_{eval}. && (49) \end{aligned}$$

In (i) we used the fact that $\eta\alpha = \beta$, whereas the second inequality comes from the bounded nature of $\tilde{Q}_\tau$ for any π, where $\bar{R}$ is defined in Sec. 3.

For $\max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1$, we can either upper-bound it by 2, or use the fact that π and $\tilde{\pi}$ are close given large enough M. First, note that the gradient of the negative entropy is given by

$$\begin{aligned} \nabla h(p) &= \nabla(p \cdot \log p), && (50) \\ &= \log p + 1. && (51) \end{aligned}$$

As the negative entropy is 1-strongly convex w.r.t. the $\|.\|_1$ norm (a.k.a. Pinsker's inequality), we have for all $s \in S$, where the s dependency is dropped

$$\begin{aligned} \|\pi_k - \tilde{\pi}_k\|_1^2 &\leq (\pi_k - \tilde{\pi}_k) \cdot (\nabla h(\pi_k) - \nabla h(\tilde{\pi}_k)), && (52) \\ &= (\pi_k - \tilde{\pi}_k) \cdot (\log \pi_k - \log \tilde{\pi}_k), && (53) \\ &\overset{(i)}{=} (\pi_k - \tilde{\pi}_k) \cdot (\xi_k - \tilde{\xi}_k), && (54) \\ &\leq \|\pi_k - \tilde{\pi}_k\|_1 \left\| \xi_k - \tilde{\xi}_k \right\|_\infty, && (55) \\ &= \|\pi_k - \tilde{\pi}_k\|_1 \alpha \beta^{M-1} \left\| \tilde{Q}_\tau^{k-M} \right\|_\infty, && (56) \\ \implies \|\pi_k - \tilde{\pi}_k\|_1 &\leq \alpha \beta^{M-1} \left\| \tilde{Q}_\tau^{k-M} \right\|_\infty, && (57) \\ &\leq \alpha \beta^{M-1} (\bar{R} + \epsilon_{eval}). && (58) \end{aligned}$$

In (i), the normalizing constants $\log Z = \log \sum_a \exp(\xi(a))$ and $\log \tilde{Z} = \log \sum_a \exp(\tilde{\xi}(a))$ do not appear because their dot product with $\pi - \tilde{\pi}_k$ is equal to 0, as they have constant values for all actions. Combining both results, we have

$$\|\pi_k - \tilde{\pi}_k\|_1 \leq \min \left\{ 2, \alpha \beta^{M-1} (\bar{R} + \epsilon_{eval}) \right\}. \tag{59}$$

which holds for all $s \in S$ and thus also for the state $\arg\max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1$. □

In the case of an update

$$\xi_{k+1} = \beta \xi_k + \alpha \left(\tilde{Q}_\tau^k - \beta^M \tilde{Q}_\tau^{k-M} \right), \tag{60}$$

we get that for any $k \geq 0$ holds

$$Q_\tau^{k+1} \geq \tilde{Q}_\tau^k - \min \left\{ 2, \alpha \beta^{M-1} (\bar{R} + \epsilon_{eval}) \right\} \gamma \beta^M \frac{\bar{R} + \epsilon_{eval}}{1 - \gamma} - \frac{1 + \gamma}{1 - \gamma} \epsilon_{eval}. \tag{61}$$

For simplicity, we will further analyse the case of

$$Q_\tau^{k+1} \geq \tilde{Q}_\tau^k - 2\gamma\beta^M \frac{\bar{R} + \epsilon_{eval}}{1-\gamma} - \frac{1+\gamma}{1-\gamma}\epsilon_{eval}. \tag{62}$$

Note that for $k \leq M$, Eq. 62 can be replaced by a stronger $Q_\tau^{k+1} \geq \tilde{Q}_\tau^k - \frac{1+\gamma}{1-\gamma}\epsilon_{eval}$ as $\xi_k - \tilde{\xi}_k = 0$, but for simplicity we only consider Eq. 62.

A.3.2 Proof of Theorem 4.2

To prove Thm. 4.2, we first need the following Lemma, that uses the approximate policy improvement bounds of vanilla finite-memory EPMD in Cor. 4.1.1, to show a relation between the Q-function Q_τ^k and the sum of Q-functions ξ_k. Further, we show the final error that is introduced by having an approximate policy evaluation and how it affects the final convergence results.

Lemma A.2. *After $k \geq 0$ iterations of Eq. 60, we have* $\left\| Q_\tau^\star - \tilde{Q}_\tau^{k+1} \right\|_\infty \leq \gamma \left\| Q_\tau^\star - \tau\xi_{k+1} \right\|_\infty + \frac{1-\beta^M}{1-\beta}\gamma\epsilon + \frac{1-\beta^{M+1}}{1-\beta}\epsilon_{eval} + \gamma\beta^M\bar{R}$, *where* $\epsilon = 2\gamma\beta^M \frac{\bar{R}+\epsilon_{eval}}{1-\gamma} + \frac{1+\gamma}{1-\gamma}\epsilon_{eval}$.

Proof. For all $s \in S$ and $a \in A$

$$(Q_\tau^\star - Q_\tau^{k+1})(s,a) = (T_\tau^\star Q_\tau^\star)(s,a) - \Big(R(s,a) + \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau h(\pi_{k+1}(s'))]\Big) \tag{63}$$

$$\begin{aligned} &= (T_\tau^\star Q_\tau^\star)(s,a) - \Big(R(s,a) + \gamma\mathbb{E}_{s',a'}[\tau\xi_{k+1}(s',a') - \tau h(\pi_{k+1}(s'))] \\ &\quad + \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau\xi_{k+1}(s',a')]\Big). \end{aligned} \tag{64}$$

Looking at the first inner term, using the entropy maximizing nature of π_{k+1} as defined in Eq. 15, and using the definition of the Bellman optimality operator $T_\tau^\star$ gives

$$\begin{aligned} R(s,a) + \gamma\mathbb{E}_{s'}[\tau\xi_{k+1}(s')\cdot\pi_{k+1}(s') - \tau h(\pi_{k+1}(s'))] &= R(s,a) \\ &\quad + \gamma\mathbb{E}_{s'}[\max_{p\in\Delta(A)} \tau\xi_{k+1}(s')\cdot p - \tau h(p)] \end{aligned} \tag{65}$$

$$= (T_\tau^\star \tau\xi_{k+1})(s,a) \tag{66}$$

For the second inner term, using the definition of ξ_{k+1}, the fact that $\tau\alpha = 1-\beta$ and the definition of ϵ, we have for all $s \in S$ and $a \in A$

$$Q_\tau^{k+1} - \tau\xi_{k+1} = Q_\tau^{k+1} - (1-\beta)\sum_{i=0}^{M-1}\beta^i\tilde{Q}_\tau^{k-i} \tag{67}$$

$$= \sum_{i=0}^{M}\beta^i Q_\tau^{k+1-i} - \sum_{i=1}^{M}\beta^i Q_\tau^{k+1-i} + \sum_{i=0}^{M-1}\beta^{i+1}\tilde{Q}_\tau^{k-i} - \sum_{i=0}^{M-1}\beta^i\tilde{Q}_\tau^{k-i} \tag{68}$$

$$= \sum_{i=0}^{M-1}\beta^i(Q_\tau^{k+1-i} - \tilde{Q}_\tau^{k-i}) + \sum_{i=1}^{M}\beta^i(\tilde{Q}_\tau^{k+1-i} - Q_\tau^{k+1-i}) + \beta^M Q_\tau^{k+1-M} \tag{69}$$

$$\geq -\sum_{i=0}^{M-1}\beta^i\epsilon - \beta^M\bar{R} + \sum_{i=1}^{M}\beta^i(\tilde{Q}_\tau^{k+1-i} - Q_\tau^{k+1-i}) \tag{70}$$

$$= -\frac{1-\beta^M}{1-\beta}\epsilon - \beta^M\bar{R} + \sum_{i=1}^{M}\beta^i(\tilde{Q}_\tau^{k+1-i} - Q_\tau^{k+1-i}). \tag{71}$$

Using successively Eq. 66 and Eq. 71 back into Eq. 64 yields

$$\begin{aligned}(Q_\tau^\star - Q_\tau^{k+1})(s,a) &= (T_\tau^\star Q_\tau^\star)(s,a) - (T_\tau^\star \tau\xi_{k+1})(s,a) \\ &\quad - \gamma \mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau\xi_{k+1}(s',a')], \qquad (72)\\ &\leq (T_\tau^\star Q_\tau^\star)(s,a) - (T_\tau^\star \tau\xi_{k+1})(s,a) \\ &\quad + \frac{\gamma(1-\beta^M)}{1-\beta}\epsilon + \gamma\beta^M \bar{R} - \gamma \mathbb{E}_{s',a'} \sum_{i=1}^{M} \beta^i (\tilde{Q}_\tau^{k+1-i} - Q_\tau^{k+1-i}). \qquad (73)\end{aligned}$$

Since $Q_\tau^\star - Q_\tau^{k+1} \geq 0$ and using the triangle inequality, the fact that $\mathbb{E}_{s,a}[X] \leq \|X\|_\infty$ and the contraction property of $T_\tau^\star$ completes the proof

$$\begin{aligned}\left\|Q_\tau^\star - \tilde{Q}_\tau^{k+1}\right\|_\infty &\leq \left\|Q_\tau^\star - Q_\tau^{k+1}\right\|_\infty + \left\|Q_\tau^{k+1} - \tilde{Q}_\tau^{k+1}\right\|_\infty \qquad (74)\\ &\overset{(i)}{\leq} \left\|T_\tau^\star Q_\tau^\star - T_\tau^\star \tau\xi_{k+1}\right\|_\infty + \frac{\gamma(1-\beta^M)}{1-\beta}\epsilon + \gamma\beta^M\bar{R} \\ &\quad + \sum_{i=0}^{M} \beta^i \left\|\tilde{Q}_\tau^{k+1-i} - Q_\tau^{k+1-i}\right\|_\infty, \qquad (75)\\ &\leq \gamma \left\|Q_\tau^\star - \tau\xi_{k+1}\right\|_\infty + \frac{\gamma(1-\beta^M)}{1-\beta}\epsilon + \gamma\beta^M\bar{R} + \frac{1-\beta^{M+1}}{1-\beta}\epsilon_{eval}. \qquad (76)\end{aligned}$$

Here (i) is due to Eq. 73 and $\gamma < 1$. This completes the proof.

□

The next theorem generalizes Thm. 4.2 from the main paper to the case of inexact Q-functions. Thus, the proof for Thm. 4.2 can be retrieved by cancelling ϵ_{eval} terms and replacing $\tilde{Q}_\tau$ by Q_τ.

Theorem A.3 (Convergence of approximate vanilla finite-memory EPMD)**.** *After $k \geq 0$ iterations of Eq. 6, we have that $\left\|Q_\tau^\star - \tilde{Q}_\tau^k\right\|_\infty \leq \gamma d^k \left\|Q_\tau^\star\right\|_\infty + C_1\beta^M + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)^2(1-\beta)}$, with $d = \beta + \gamma(1-\beta) < 1$, $C_1 = \frac{2\gamma\bar{R}}{1-\gamma}\left(1 + \frac{\gamma(1-\beta^M)}{(1-\beta)(1-\gamma)}\right) + \frac{\gamma\epsilon_{eval}}{(1-\gamma)(1-\beta)}$.*

This theorem states that the approximate vanilla finite-memory EPMD algorithm converges to an error that consists of two components: the first one scales with β^M and thus should become negligible for large enough M and the second one fully depends on ϵ_{eval} and is small only if ϵ_{eval} is small too.

Proof. From the definition of ξ_{k+1} in Eq. 60 and the triangle inequality we get

$$\begin{aligned}\left\|Q_\tau^\star - \tau\xi_{k+1}\right\|_\infty &= \left\|Q_\tau^\star - \beta\tau\xi_k - (1-\beta)\tilde{Q}_\tau^k + (1-\beta)\beta^M\tilde{Q}_\tau^{k-M}\right\|_\infty, \qquad (77)\\ &\leq \beta\left\|Q_\tau^\star - \tau\xi_k\right\|_\infty + (1-\beta)\left\|Q_\tau^\star - \tilde{Q}_\tau^k\right\|_\infty + (1-\beta)\beta^M\left\|\tilde{Q}_\tau^{k-M}\right\|_\infty, \qquad (78)\\ &\leq \beta\left\|Q_\tau^\star - \tau\xi_k\right\|_\infty \\ &\quad + (1-\beta)\gamma\left\|Q_\tau^\star - \tau\xi_k\right\|_\infty + \gamma(1-\beta^M)\epsilon + (1-\beta^{M+1})\epsilon_{eval} \qquad (79)\\ &\quad + (1-\beta)\gamma\beta^M\bar{R} + (1-\beta)\beta^M\left\|\tilde{Q}_\tau^{k-M}\right\|_\infty, \\ &\leq (\beta + \gamma(1-\beta))\left\|Q_\tau^\star - \tau\xi_k\right\|_\infty + \gamma(1-\beta^M)\epsilon \\ &\quad + (1+\gamma)(1-\beta)\beta^M\bar{R} + (1+\beta^M)\epsilon_{eval}. \qquad (80)\end{aligned}$$

Where in the last inequality we used the fact that $\left\|\tilde{Q}_\tau^{k-M}\right\|_\infty \leq \bar{R} + \epsilon_{eval}$ and $1 - \beta^{M+1} + (1-\beta)\beta^M = 1 + \beta^M - 2\beta^{M+1} \leq 1 + \beta^M$. Letting

$$d := \beta + \gamma(1-\beta), \qquad (81)$$

one can show by induction, using the fact that $\xi_0 = 0$, that

$$\begin{aligned}
\|Q_\tau^\star - \tau\xi_{k+1}\|_\infty &\leq d^{k+1}\|Q_\tau^\star\|_\infty \\
&\quad + \sum_{i=0}^{k} d^i \left[\gamma(1-\beta^M)\epsilon + (1+\gamma)(1-\beta)\beta^M\bar{R} + (1+\beta^M)\epsilon_{eval}\right], \qquad (82)\\
&\leq d^{k+1}\|Q_\tau^\star\|_\infty + \frac{\gamma(1-\beta^M)\epsilon + (1+\gamma)(1-\beta)\beta^M\bar{R} + (1+\beta^M)\epsilon_{eval}}{1-d}, \qquad (83)\\
&= d^{k+1}\|Q_\tau^\star\|_\infty + \frac{(1+\gamma)\beta^M\bar{R}}{1-\gamma} + \frac{\gamma(1-\beta^M)\epsilon + (1+\beta^M)\epsilon_{eval}}{(1-\gamma)(1-\beta)}. \qquad (84)
\end{aligned}$$

For Eq. 83, we used the fact that $\sum_{i=0}^{k} d^i = \frac{1-d^{k+1}}{1-d} \leq \frac{1}{1-d}$. Using Eq. 84 in Eq. 76 finally gives

$$\begin{aligned}
\left\|Q_\tau^\star - \tilde{Q}_\tau^{k+1}\right\|_\infty &\leq \gamma d^{k+1}\|Q_\tau^\star\|_\infty \\
&\quad + \left[\gamma\beta^M + \frac{(1+\gamma)\gamma\beta^M}{1-\gamma}\right]\bar{R} \\
&\quad + \left[\frac{\gamma^2(1-\beta^M)}{(1-\gamma)(1-\beta)} + \frac{\gamma(1-\beta^M)}{1-\beta}\right]\epsilon \\
&\quad + \left[\frac{\gamma(1+\beta^M)}{(1-\gamma)(1-\beta)} + \frac{1-\beta^{M+1}}{1-\beta}\right]\epsilon_{eval}
\end{aligned} \qquad (85)$$

Now, let us analyse more closely the constants in front of $\bar{R}$ and ϵ_{eval}. First, let us simplify the constant in front of ϵ, we get $\frac{\gamma^2(1-\beta^M)}{(1-\gamma)(1-\beta)} + \frac{\gamma(1-\beta^M)}{1-\beta} = \frac{\gamma(1-\beta^M)}{1-\beta}\left(\frac{\gamma}{1-\gamma} + 1\right) = \frac{\gamma(1-\beta^M)}{(1-\gamma)(1-\beta)}$. By inserting the value of ϵ from Lemma A.2 we obtain the following coefficient for $\bar{R}$

$$\gamma\beta^M + \frac{(1+\gamma)\gamma\beta^M}{1-\gamma} + \frac{\gamma(1-\beta^M)}{(1-\gamma)(1-\beta)}\frac{2\gamma\beta^M}{1-\gamma} = \frac{2\gamma\beta^M}{1-\gamma}\left(1 + \frac{\gamma(1-\beta^M)}{(1-\beta)(1-\gamma)}\right) \qquad (86)$$

and ϵ_{eval}

$$\begin{aligned}
&\frac{\gamma(1+\beta^M)}{(1-\gamma)(1-\beta)} + \frac{1-\beta^{M+1}}{1-\beta} + \frac{\gamma(1-\beta^M)}{(1-\gamma)(1-\beta)}\frac{2\gamma\beta^M + 1 + \gamma}{1-\gamma} \\
&\overset{(i)}{<} \frac{\gamma(1+\beta^M)}{(1-\gamma)(1-\beta)} + \frac{1-\beta^{M+1}}{1-\beta} + \frac{\gamma(1+\gamma)}{(1-\gamma)^2(1-\beta)} \\
&\overset{(ii)}{\leq} \frac{\gamma + \gamma\beta^M + 1 - \gamma}{(1-\gamma)(1-\beta)} + \frac{\gamma(1+\gamma)}{(1-\gamma)^2(1-\beta)} \\
&= \frac{1-\gamma+\gamma+\gamma^2}{(1-\gamma)^2(1-\beta)} + \frac{\gamma\beta^M}{(1-\gamma)(1-\beta)} \\
&= \frac{1+\gamma^2}{(1-\gamma)^2(1-\beta)} + \frac{\gamma\beta^M}{(1-\gamma)(1-\beta)},
\end{aligned} \qquad (87)$$

where in (i) we use $\gamma(1-\beta^M)(2\gamma\beta^M + 1 + \gamma) = 2\gamma^2\beta^M - 2\gamma^2\beta^{2M} + \gamma - \gamma\beta^M + \gamma^2 - \gamma^2\beta^M < \gamma^2\beta^M - \gamma\beta^M + \gamma + \gamma^2 < \gamma(1+\gamma)$ and in (ii) we cancel the negative components. Combining Eq. 85, Eq. 86 and Eq. 87, we complete the proof. □

A.4 Approximate weight-corrected finite-memory EPMD

A.4.1 Proof of the logits expression in Sec. 4.2

Proof. For $k = 0$,

$$\xi_1 = \beta \times 0 + \alpha \tilde{Q}_\tau^0 + \frac{\alpha\beta^M}{1-\beta^M}(\tilde{Q}_\tau^0 - 0), \tag{88}$$

$$= \alpha \left(1 + \frac{\beta^M}{1-\beta^M}\right) \tilde{Q}_\tau^0, \tag{89}$$

$$= \frac{\alpha}{1-\beta^M} \tilde{Q}_\tau^0. \tag{90}$$

If it is true for k, then

$$\xi_{k+1} = \beta \frac{\alpha}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i \tilde{Q}_\tau^{k-1-i} + \alpha \tilde{Q}_\tau^k + \frac{\alpha\beta^M}{1-\beta^M}(\tilde{Q}_\tau^k - \tilde{Q}_\tau^{k-M}), \tag{91}$$

$$= \frac{\alpha}{1-\beta^M} \sum_{i=0}^{M-2} \beta^{i+1} \tilde{Q}_\tau^{k-1-i} + \frac{\alpha\beta^M}{1-\beta^M}(\tilde{Q}_\tau^{k-M} - \tilde{Q}_\tau^{k-M}) + \frac{\alpha}{1-\beta^M} \tilde{Q}_\tau^k, \tag{92}$$

$$= \frac{\alpha}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i \tilde{Q}_\tau^{k-i} \tag{93}$$

□

A.4.2 Proof of Corollary 4.1.2

Proof. The proof is immediate from Thm. 4.1, upper-bounding $\max_{s\in S} \|(\pi_k - \tilde{\pi}_k)(s)\|_1$ by 2, using the definition of $\tilde{\xi}_k - \xi_k = \frac{\alpha\beta^{M-1}}{1-\beta^M}(\tilde{Q}_\tau^k - \tilde{Q}_\tau^{k-M})$ in $\left\|\xi_k - \tilde{\xi}_k\right\|_\infty$ and using the fact that $\alpha\eta = \beta$. □

Note that, as in Cor. 4.1.1, we could have used the expression of the logits of π and $\tilde{\pi}$ to have a bound of $\|(\pi_k - \tilde{\pi}_k)(s)\|_1$ that depends on $\left\|\xi_k - \tilde{\xi}_k\right\|_\infty$ and ultimately on $\left\|\tilde{Q}_\tau^k - \tilde{Q}_\tau^{k-M}\right\|_\infty$. This bound becomes tighter as k goes to infinity for M large enough, since we show below that $\tilde{Q}_\tau^k$ converges to $Q_\tau^\star$ with some error that depends on ϵ_{eval} and thus $\max_{s\in S} \|(\pi_k - \tilde{\pi}_k)(s)\|_1$ converges to 0. Nonetheless, using this tighter bound would introduce quadratic terms $\left\|\tilde{Q}_\tau^k - \tilde{Q}_\tau^{k-M}\right\|_\infty^2$ that would complicate the overall analysis of the algorithm, and thus we use the more crude bound of 2 for $\max_{s\in S} \|(\pi_k - \tilde{\pi}_k)(s)\|_1$ in the remainder of the proofs for Sec. 4.2.

Thus, given Theorem A.1 and Corollary 4.1.2, and in case of an update

$$\xi_{k+1} = \beta\xi_k + \alpha \tilde{Q}_\tau^k + \frac{\alpha\beta^M}{1-\beta^M}(\tilde{Q}_\tau^k - \tilde{Q}_\tau^{k-M}), \tag{94}$$

we get

$$Q_\tau^{k+1} \geq \tilde{Q}_\tau^k - 2\gamma\beta^M \frac{\left\|\tilde{Q}_\tau^k - \tilde{Q}_\tau^{k-M}\right\|_\infty}{(1-\gamma)(1-\beta^M)} - \frac{1+\gamma}{1-\gamma}\epsilon_{eval}. \tag{95}$$

A.5 Proof of Lemma 4.3

As with Thm. 4.2, we first need an intermediary Lemma connecting $\left\|Q_\tau^\star - Q_\tau^{k+1}\right\|_\infty$ and $\|Q_\tau^\star - \tau\xi_{k+1}\|_\infty$ before proving Lem. 4.3.

Lemma A.4. *After $k \geq 0$ iterations of Eq. 94, we have* $\left\|Q_\tau^\star - \tilde{Q}_\tau^{k+1}\right\|_\infty \leq \gamma \left\|Q_\tau^\star - \tau\xi_{k+1}\right\|_\infty + \gamma\beta^{k+1}\left\|Q_\tau^0\right\|_\infty + \gamma\sum_{i=0}^{k}\beta^i\epsilon'_{k-i} + \frac{1+\gamma^2}{(1-\gamma)(1-\beta)}\epsilon_{eval}$, *with* $\epsilon'_k = \left(\frac{1+\gamma}{1-\gamma} - \beta\right)\frac{\beta^M}{1-\beta^M}\left\|\tilde{Q}_\tau^{k-M} - \tilde{Q}_\tau^k\right\|_\infty$.

Proof. Define ϵ_k as

$$\epsilon_k := \frac{2\beta^M}{1-\beta^M}\left\|\tilde{Q}_\tau^{k-M} - \tilde{Q}_\tau^k\right\|_\infty, \tag{96}$$

For all $s \in S$ and $a \in A$

$$(Q_\tau^\star - Q_\tau^{k+1})(s,a) = (T_\tau^\star Q_\tau^\star)(s,a) - \left(R(s,a) + \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau h(\pi_{k+1}(s'))]\right) \tag{97}$$

$$\begin{aligned} &= (T_\tau^\star Q_\tau^\star)(s,a) - \Big(R(s,a) + \gamma\mathbb{E}_{s',a'}[\tau\xi_{k+1}(s',a') - \tau h(\pi_{k+1}(s'))] + \\ &\quad \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau\xi_{k+1}(s',a')]\Big) \end{aligned} \tag{98}$$

Looking at the first inner term and using the entropy maximizing nature of π_{k+1} as defined in Eq. 15 gives

$$R(s,a) + \gamma\mathbb{E}_{s'}[\tau\xi_{k+1}(s')\cdot\pi_{k+1}(s') - \tau h(\pi_{k+1}(s'))] \tag{99}$$

$$= R(s,a) + \gamma\mathbb{E}_{s'}[\max_{p\in\Delta(A)} \tau\xi_{k+1}(s')\cdot p - \tau h(p)] = (T_\tau^\star\tau\xi_{k+1})(s,a) \tag{100}$$

For the second inner term, using the recursive definition of ξ_{k+1} in Eq. 94 gives

$$Q_\tau^{k+1} - \tau\xi_{k+1} = Q_\tau^{k+1} - \left(\beta\tau\xi_k + (1-\beta)\tilde{Q}_\tau^k + \frac{(1-\beta)\beta^M}{1-\beta^M}(\tilde{Q}_\tau^k - \tilde{Q}_\tau^{k-M})\right), \tag{101}$$

$$= \beta(\tilde{Q}_\tau^k - \tau\xi_k) + Q_\tau^{k+1} - \tilde{Q}_\tau^k - \frac{(1-\beta)\beta^M}{1-\beta^M}(\tilde{Q}_\tau^k - \tilde{Q}_\tau^{k-M}), \tag{102}$$

$$\geq \beta(Q_\tau^k - \tau\xi_k) - \frac{\gamma\epsilon_k}{1-\gamma} - \frac{(1-\beta)\epsilon_k}{2} - \frac{1+\gamma}{1-\gamma}\epsilon_{eval} + \beta\left(\tilde{Q}_\tau^k - Q_\tau^k\right). \tag{103}$$

Letting

$$\epsilon'_k := \frac{\gamma\epsilon_k}{1-\gamma} + \frac{(1-\beta)\epsilon_k}{2} = \left(\frac{1+\gamma}{1-\gamma} - \beta\right)\frac{\beta^M}{1-\beta^M}\left\|\tilde{Q}_\tau^{k-M} - \tilde{Q}_\tau^k\right\|_\infty, \tag{104}$$

one can easily show by induction that

$$Q_\tau^{k+1} - \tau\xi_{k+1} \geq \beta^{k+1}Q_\tau^0 - \sum_{i=0}^{k}\beta^i\epsilon'_{k-i} - \sum_{i=0}^{k}\beta^i\frac{1+\gamma}{1-\gamma}\epsilon_{eval} - \sum_{i=1}^{k+1}\beta^i\epsilon_{eval} \tag{105}$$

$$= \beta^{k+1}Q_\tau^0 - \sum_{i=0}^{k}\beta^i\epsilon'_{k-i} - \frac{(1-\beta^{k+1})(1+\gamma)}{(1-\beta)(1-\gamma)}\epsilon_{eval} - \frac{\beta(1-\beta^{k+1})}{1-\beta}\epsilon_{eval} \tag{106}$$

$$\geq \beta^{k+1}Q_\tau^0 - \sum_{i=0}^{k}\beta^i\epsilon'_{k-i} - \frac{(1+\gamma)\epsilon_{eval}}{(1-\gamma)(1-\beta)} - \frac{\beta\epsilon_{eval}}{1-\beta}. \tag{107}$$

The inequality uses the fact that $\xi_0 = 0$. Using successively Eq. 100 and Eq. 107 back into Eq. 98 yields

$$(Q_\tau^\star - Q_\tau^{k+1})(s,a) = (T_\tau^\star Q_\tau^\star)(s,a) - (T_\tau^\star\tau\xi_{k+1})(s,a) - \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau\xi_{k+1}(s',a')], \tag{108}$$

$$\begin{aligned} &\leq (T_\tau^\star Q_\tau^\star)(s,a) - (T_\tau^\star\tau\xi_{k+1})(s,a) - \gamma\mathbb{E}_{s',a'}[\beta^{k+1}Q_\tau^0(s',a')] \\ &\quad + \gamma\sum_{i=0}^{k}\beta^i\epsilon'_{k-i} + \frac{\gamma\beta}{1-\beta}\epsilon_{eval} + \frac{\gamma(1+\gamma)\epsilon_{eval}}{(1-\gamma)(1-\beta)}. \end{aligned} \tag{109}$$

Since $Q^\star_\tau - Q^{k+1}_\tau \geq 0$ and using the triangle inequality, the fact that $\mathbb{E}_{s,a}[Q^0_\tau(s,a)] \leq \left\|Q^0_\tau\right\|_\infty$, and the contraction property of $T^\star_\tau$ gives us

$$\begin{aligned}\left\|Q^\star_\tau - Q^{k+1}_\tau\right\|_\infty &\leq \|T^\star_\tau Q^\star_\tau - T^\star_\tau \tau\xi_{k+1}\|_\infty + \gamma\beta^{k+1}\left\|Q^0_\tau\right\|_\infty \\ &\quad + \gamma\sum_{i=0}^{k}\beta^i\epsilon'_{k-i} + \frac{\gamma\beta}{1-\beta}\epsilon_{eval} + \frac{\gamma(1+\gamma)\epsilon_{eval}}{(1-\gamma)(1-\beta)} \qquad (110)\\ &\leq \gamma\left\|Q^\star_\tau - \tau\xi_{k+1}\right\|_\infty + \gamma\beta^{k+1}\left\|Q^0_\tau\right\|_\infty \\ &\quad + \gamma\sum_{i=0}^{k}\beta^i\epsilon'_{k-i} + \frac{\gamma\beta}{1-\beta}\epsilon_{eval} + \frac{\gamma(1+\gamma)\epsilon_{eval}}{(1-\gamma)(1-\beta)}. \qquad (111)\end{aligned}$$

Finally, using

$$\left\|Q^\star_\tau - \tilde{Q}^{k+1}_\tau\right\|_\infty \leq \left\|Q^\star_\tau - Q^{k+1}_\tau\right\|_\infty + \left\|Q^{k+1}_\tau - \tilde{Q}^{k+1}_\tau\right\|_\infty \leq \left\|Q^\star_\tau - Q^{k+1}_\tau\right\|_\infty + \epsilon_{eval} \qquad (112)$$

and also simplifying the constants $\frac{\gamma\beta}{1-\beta} + 1 = \frac{1-\beta+\gamma\beta}{1-\beta} \leq \frac{1}{1-\beta}$ and $\frac{1}{1-\beta} + \frac{\gamma(1+\gamma)}{(1-\gamma)(1-\beta)} = \frac{1}{1-\beta}(1 + \frac{\gamma(1+\gamma)}{1-\gamma}) = \frac{1+\gamma^2}{(1-\gamma)(1-\beta)}$, we obtain the statement of the lemma. □

We now state a more general form of Lemma 4.3 and prove it.

Lemma A.5. *Let* $x_{k+1} = d_1 x_k + d_2 x_{k-M} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma}$ *be a sequence such that* $\forall k < 0$, $x_k = \frac{\|Q^\star_\tau\|_\infty}{\gamma}$, $x_0 = \|Q^\star_\tau\|_\infty + \left\|\tilde{Q}^0_\tau\right\|_\infty + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-\beta)}$, $d_1 := \beta + \gamma\frac{1-\beta}{1-\beta^M} + \gamma c_2$, $d_2 := \frac{2c_1\gamma^2}{1-\gamma}$, $c_1 := \frac{\beta^M}{1-\beta^M}$, *and* $c_2 := \left(\frac{1+\gamma}{1-\gamma} - \beta\right)c_1$. *After* $k \geq 0$ *iterations of Eq. 10, we have that* $\left\|Q^\star_\tau - \tilde{Q}^k_\tau\right\|_\infty \leq x_k$.

Proof. Define the following constants

$$c_1 := \frac{\beta^M}{1-\beta^M}, \text{ and } c_2 := \left(\frac{1+\gamma}{1-\gamma} - \beta\right)c_1. \qquad (113)$$

Let a sequence x_k defined by $\forall k < 0$, $x_k = \frac{\|Q^\star_\tau\|_\infty}{\gamma}$ and let

$$x_0 = \|Q^\star_\tau\|_\infty + \left\|Q^0_\tau\right\|_\infty + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-\beta)}. \qquad (114)$$

For subsequent terms, we define x_k by the recursive definition, $\forall k \geq 0$

$$x_{k+1} = \gamma\left(\frac{1-\beta}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i x_{k-i} + \beta^{k+1}\frac{\left\|Q^0_\tau\right\|_\infty}{\gamma} + c_2\sum_{i=0}^{k}\beta^i(x_{k-i} + x_{k-i-M})\right) + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-\beta)}. \qquad (115)$$

Note that x_0 can also be recovered by Eq. 115, for $k = -1$. Now, let us simplify Eq. 115.

Using this recursive definition, we have $\forall k \geq 0$

$$x_{k+1} = \gamma \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-i} + \beta^{k+1} \left\|Q_\tau^0\right\|_\infty + c_2\gamma \sum_{i=0}^{k} \beta^i (x_{k-i} + x_{k-i-M}) + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-\beta)}, \tag{116}$$

$$= \beta \left(\frac{\gamma(1-\beta)}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-1-i} + \beta^k \left\|Q_\tau^0\right\|_\infty + \gamma c_2 \sum_{i=0}^{k-1} \beta^i (x_{k-1-i} + x_{k-1-i-M}) + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-\beta)} \right) + \frac{\gamma(1-\beta)}{1-\beta^M} \left(x_k - \beta^M x_{k-M}\right) + \gamma c_2 \left(x_k + x_{k-M}\right) + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma}, \tag{117}$$

$$\overset{(i)}{=} \beta x_k + \gamma \left(\frac{1-\beta}{1-\beta^M} \left(x_k - \beta^M x_{k-M}\right) + c_2 \left(x_k + x_{k-M}\right) \right) + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma}, \tag{118}$$

$$= \left(\beta + \gamma \frac{1-\beta}{1-\beta^M} + \gamma c_2 \right) x_k + \gamma \left(c_2 - \frac{\beta^M(1-\beta)}{1-\beta^M} \right) x_{k-M} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \tag{119}$$

$$= \left(\beta + \gamma \frac{1-\beta}{1-\beta^M} + \gamma c_2 \right) x_k + \frac{2c_1\gamma^2}{1-\gamma} x_{k-M} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \tag{120}$$

In (i) we used the recursive definition of x_k which is also valid for x_0. Letting

$$d_1 := \beta + \gamma \frac{1-\beta}{1-\beta^M} + \gamma c_2, \text{ and } d_2 := \frac{2c_1\gamma^2}{1-\gamma}, \tag{121}$$

x_{k+1} for all $k \geq 0$ can be more compactly defined by

$$x_{k+1} = d_1 x_k + d_2 x_{k-M} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma}. \tag{122}$$

Let us now prove that $\left\|Q_\tau^\star - \tilde{Q}_\tau^k\right\|_\infty \leq x_k$ by induction. For $k = 0$, we have that

$$\left\|Q_\tau^\star - \tilde{Q}_\tau^0\right\|_\infty \leq \left\|Q_\tau^\star - Q_\tau^0\right\|_\infty + \left\|Q_\tau^0 - \tilde{Q}_\tau^0\right\|_\infty \tag{123}$$

$$\leq \left\|Q_\tau^\star\right\|_\infty + \left\|Q_\tau^0\right\|_\infty + \epsilon_{eval}, \tag{124}$$

$$\leq x_0. \tag{125}$$

and for $k < 0$, we have that

$$\left\|Q_\tau^\star - \tilde{Q}_\tau^k\right\|_\infty = \left\|Q_\tau^\star\right\|_\infty, \tag{126}$$

$$\leq \frac{\left\|Q_\tau^\star\right\|_\infty}{\gamma}, \tag{127}$$

$$= x_k. \tag{128}$$

Now assume that $\left\|Q_\tau^\star - \tilde{Q}_\tau^i\right\|_\infty \leq x_i$ is true for all $i \leq k$ and let us prove that $\left\|Q_\tau^\star - \tilde{Q}_\tau^{k+1}\right\|_\infty \leq$

x_{k+1}. First, we note that

$$\|Q^\star_\tau - \tau\xi_{k+1}\|_\infty = \left\| Q^\star_\tau - \tau \frac{\alpha}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i \tilde{Q}^{k-i}_\tau \right\|_\infty, \tag{129}$$

$$= \left\| \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i Q^\star_\tau - \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i \tilde{Q}^{k-i}_\tau \right\|_\infty, \tag{130}$$

$$\leq \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i \left\| Q^\star_\tau - \tilde{Q}^{k-i}_\tau \right\|_\infty, \tag{131}$$

$$\leq \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-i}. \tag{132}$$

We also have that

$$\epsilon'_k = c_2 \left\| \tilde{Q}^{k-M}_\tau - \tilde{Q}^k_\tau \right\|_\infty, \tag{133}$$

$$\leq c_2 \left(\left\| Q^\star_\tau - \tilde{Q}^{k-M}_\tau \right\|_\infty + \left\| Q^\star_\tau - \tilde{Q}^k_\tau \right\|_\infty \right) \tag{134}$$

$$\leq c_2 (x_k + x_{k-M}). \tag{135}$$

Finally, using Eq. 132, Eq. 135 and $\left\|Q^0_\tau\right\|_\infty \leq \frac{\|Q^0_\tau\|_\infty}{\gamma}$ into Lemma A.4 completes the proof

$$\left\| Q^\star_\tau - \tilde{Q}^{k+1}_\tau \right\|_\infty \leq \gamma \left(\|Q^\star_\tau - \tau\xi_{k+1}\|_\infty + \beta^{k+1} \left\|Q^0_\tau\right\|_\infty + \sum_{i=0}^{k} \beta^i \epsilon'_{k-i} \right) + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-\beta)}, \tag{136}$$

$$\leq \gamma \left(\frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-i} + \beta^{k+1} \frac{\left\|Q^0_\tau\right\|_\infty}{\gamma} + c_2 \sum_{i=0}^{k} \beta^i (x_{k-i} + x_{k-i-M}) \right) + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-\beta)}, \tag{137}$$

$$= x_{k+1}. \tag{138}$$

□

A.6 Proof of Theorem 4.4

We state a more general form for Theorem 4.4 that includes policy evaluation error and prove it below.

Theorem A.6 (Convergence of approximate weight corrected finite-memory EPMD)**.** *With the definitions of Lemma 4.3, if* $M > \log \frac{(1-\gamma)^2(1-\beta)}{\gamma^2(3+\beta)+1-\beta} (\log \beta)^{-1}$ *then* $\lim_{k\to\infty} x_k \leq \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-d_1-d_2)}$*. Moreover,* $\forall k \geq 0$, $\left\| Q^\star_\tau - \tilde{Q}^k_\tau \right\|_\infty \leq (d_1 + d_2 d_3^{-1})^k \max\left\{ \frac{\|Q^\star_\tau\|_\infty}{\gamma}, \|Q^\star_\tau\|_\infty + \left\|Q^0_\tau\right\|_\infty \right\} + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-d_1-d_2)}$*, where* $d_3 := \left(d_1^M + d_2 \frac{1-d_1^M}{1-d_1} \right)$ *and* $\lim_{M\to\infty} d_1 + d_2 d_3^{-1} = \beta + \gamma(1-\beta)$.

Proof. Let us find a value of M such that

$$d_1 + d_2 < 1, \tag{139}$$

$$\Leftrightarrow \beta(1-\beta^M) + \gamma(1-\beta) + \gamma\left(\frac{1+\gamma}{1-\gamma} - \beta\right)\beta^M + \frac{2\gamma^2\beta^M}{1-\gamma} < 1 - \beta^M, \tag{140}$$

$$\Leftrightarrow \beta - \beta^{M+1} - \gamma\beta + \gamma\frac{1+\gamma}{1-\gamma}\beta^M - \gamma\beta^{M+1} + \beta^M + \frac{2\gamma^2\beta^M}{1-\gamma} < 1 - \gamma, \tag{141}$$

$$\Leftrightarrow (1-\gamma)\beta + \frac{\gamma^2(3+\beta) + 1 - \beta}{1-\gamma}\beta^M < 1 - \gamma, \tag{142}$$

$$\Leftrightarrow \beta^M < \frac{(1-\gamma)^2(1-\beta)}{\gamma^2(3+\beta) + 1 - \beta}, \tag{143}$$

$$\Leftrightarrow M \log\beta < \log\frac{(1-\gamma)^2(1-\beta)}{\gamma^2(3+\beta) + 1 - \beta}, \tag{144}$$

$$\Leftrightarrow M > \log\frac{(1-\gamma)^2(1-\beta)}{\gamma^2(3+\beta) + 1 - \beta}(\log\beta)^{-1}. \tag{145}$$

We will now show that for the values of M that satisfy Eq. 145, the sequence x_k converges to some finite error that depends on ϵ_{eval} as k goes to infinity. To simplify the analysis of x_k we study a slightly modified version thereof that has the same recursive definition $x_{k+1} = d_1 x_k + d_2 x_{k-M} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma}$ but replaces the terms x_{-k}, $\forall k \geq 0$ with $x_{-k} = \max\left\{\frac{\|Q^\star_\tau\|_\infty}{\gamma}, \|Q^\star_\tau\|_\infty + \left\|Q^0_\tau\right\|_\infty + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-\beta)}\right\}$. Clearly, this modified sequence upper-bounds the previous sequence.

To simplify the analysis, we first analyse another sequence y_k that for $k \leq 0$ is identical to x_k, but for $k \geq 0$ it evolves following the next law $y_{k+1} = d_1 y_k + d_2 y_{k-M}$. Now, if M is such that $d_1 + d_2 < 1$, then the sequence y_k is constant from y_{-M} to y_0 and is strictly decreasing thereafter, since for y_1 we have

$$y_1 = d_1 y_0 + d_2 y_{-M}, \tag{146}$$

$$= (d_1 + d_2) y_0, \tag{147}$$

$$< y_0. \tag{148}$$

Then, $\forall k \geq 1$

$$y_{k+1} = d_1 y_k + d_2 y_{k-M}, \tag{149}$$

$$< d_1 y_{k-1} + d_2 y_{k-M-1}, \tag{150}$$

$$= y_k. \tag{151}$$

Since the sequence is decreasing and lower bounded by 0, it has a limit due to the monotone convergence theorem. Let us study the convergence of a sub-sequence. Let for any integer $a > 0$

$$y_{aM+a} = d_1 y_{aM+a-1} + d_2 y_{aM+a-1-M}, \tag{152}$$

$$< (d_1 + d_2) y_{aM+a-1-M}, \tag{153}$$

$$= (d_1 + d_2) y_{(a-1)M+(a-1)}, \tag{154}$$

$$< (d_1 + d_2)^a y_0. \tag{155}$$

Thus, $\lim_{a\to\infty} y_{aM+a} = 0$, which implies that $\lim_{k\to\infty} y_k = 0$.

Further, let us show that for all k and $C(k) = \sum_{i=0}^{k-1}(d_1 + d_2)^i$,

$$x_k \leq y_k + C(k)\frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma}, \tag{156}$$

and therefore, if we simplify the above expression, then for all k

$$x_k \leq y_k + \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-d_1-d_2)}. \tag{157}$$

We do it by mathematical induction. First,

$$x_1 = d_1 x_0 + d_2 x_{-M} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} = d_1 y_0 + d_2 y_{-M} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} = y_1 + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma}. \tag{158}$$

Then, let us assume that Eq. 156 holds for any $i \leq k$, now we show that it also holds for $k+1$

$$x_{k+1} = d_1 x_k + d_2 x_{k-M} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \tag{159}$$

$$\leq d_1 \left(y_k + C(k)\frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \right) + d_2 \left(y_{k-M} + C(k-M)\frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \right) + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \tag{160}$$

$$\leq y_{k+1} + \max\left\{C(k), C(k-M)\right\}(d_1+d_2)\frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \tag{161}$$

$$\overset{(i)}{=} y_{k+1} + C(k)(d_1+d_2)\frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} + \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \tag{162}$$

$$= y_{k+1} + \sum_{i=0}^{k}(d_1+d_2)^i \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma}. \tag{163}$$

Here, in (i) we use the definition of $C(k)$ from Eq. 156 and its monotonicity that comes out of it. Therefore, we get that $\lim_{k\to\infty} x_k \leq \lim_{k\to\infty} \left(y_k + \sum_{i=0}^{k-1}(d_1+d_2)^i \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} \right) = 0 + \sum_{i=0}^{\infty}(d_1+d_2)^i \frac{(1+\gamma^2)\epsilon_{eval}}{1-\gamma} = \frac{(1+\gamma^2)\epsilon_{eval}}{(1-\gamma)(1-(d_1+d_2))}$, which completes the first part of our proof. Now, let us have a closer look on the convergence speed.

To better characterize the convergence of x_k, we again analyse the sequence y_k. First, we note that the constant $d_1 \geq \beta + \gamma(1-\beta)$ is typically very close to 1, whereas $d_2 \to 0$ as $M \to \infty$. The sequence y_k thus behaves almost as $d_1^k y_0$. A much tighter upper-bounding sequence than that of Eq. 155 can be obtained using the following inequalities

$$y_k = d_1 y_{k-1} + d_2 y_{k-1-M}, \tag{164}$$

$$= d_1^M y_{k-M} + d_2 \sum_{i=0}^{M-1} d_1^i y_{k-1-M-i}, \tag{165}$$

$$\geq \left(d_1^M + d_2 \frac{1-d_1^M}{1-d_1} \right) y_{k-M}, \tag{166}$$

where we have used in the last inequality the fact that y_k is a decreasing sequence. Let

$$d_3 := \left(d_1^M + d_2 \frac{1-d_1^M}{1-d_1} \right), \tag{167}$$

then we can upper bound the sequence y_k by

$$y_{k+1} = \left(d_1 + d_2 d_3^{-1}\right) y_k + d_2(y_{k-M} - d_3^{-1} y_k), \tag{168}$$

$$\leq \left(d_1 + d_2 d_3^{-1}\right) y_k + d_2(y_{k-M} - d_3^{-1} d_3 y_{k-M}), \tag{169}$$

$$= \left(d_1 + d_2 d_3^{-1}\right) y_k, \tag{170}$$

$$\leq \left(d_1 + d_2 d_3^{-1}\right)^{k+1} y_0. \tag{171}$$

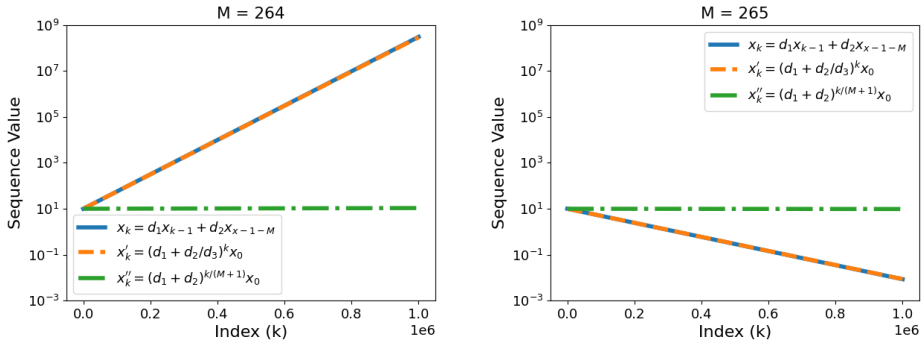


Figure 4: Evolution of x_k for two successive values of M, one being large enough for x_k to converge. The plot additionally shows the sequence x'_k introduced by Thm. 4.4 that closely follows the behavior of x_k. See text for more details.

Now to study the limit $\lim_{M\to\infty} d_1 + d_2 d_3^{-1}$, let us first start with the rightmost term

$$d_2 d_3^{-1} \leq \frac{d_2}{d_1^M}, \tag{172}$$

$$\leq \frac{d_2}{(\beta + \gamma(1-\beta))^M}, \tag{173}$$

$$= \frac{1}{(\beta + \gamma(1-\beta))^M} \frac{2\beta^M \gamma^2}{(1-\gamma)(1-\beta^M)}, \tag{174}$$

$$= \left(\frac{\beta}{\beta + \gamma(1-\beta)}\right)^M \frac{2\gamma^2}{(1-\gamma)(1-\beta^M)}. \tag{175}$$

Since $\beta < \beta + \gamma(1-\beta)$, then clearly $\lim_{M\to\infty} d_2 d_3^{-1} = 0$, and from the definition of d_1 one can see that $\lim_{M\to\infty} d_1 = \beta + \gamma(1-\beta)$. Combining the result above with Eq. 157, we obtain the statement of the theorem. □

To illustrate how close the sequence $\left(d_1 + d_2 d_3^{-1}\right)^k x_0$ is to x_k, let us take a numerical example with $\gamma = 0.99$ and $\beta = 0.95$. In this case, we have that $d_1 + d_2 < 1$ whenever $M \geq 265$. At $M = 265$ we have that $d_1 \approx 0.9997$ and $d_2 \approx 0.0002$. In Fig. 4 we plot the three sequences x_k, $x'_k = \left(d_1 + d_2 d_3^{-1}\right)^k x_0$ and $x''_k = (d_1 + d_2)^{k/(M+1)} x_0$ for $M = 264$ and $M = 265$ and we see that x'_k converges to zero for the same M as x_k and is almost indistinguishable from the latter, whereas x''_k is a much more loose upper-bounding sequence at $M = 265$.

B Additional experimental results

B.1 Comparison with deep RL baselines

We summarize all performance comparisons in Fig. 5 and Table 2. We provide a discussion of the MountainCar environment and some of the challenges of exploration in an entropy-regularized setting in App. B.5.

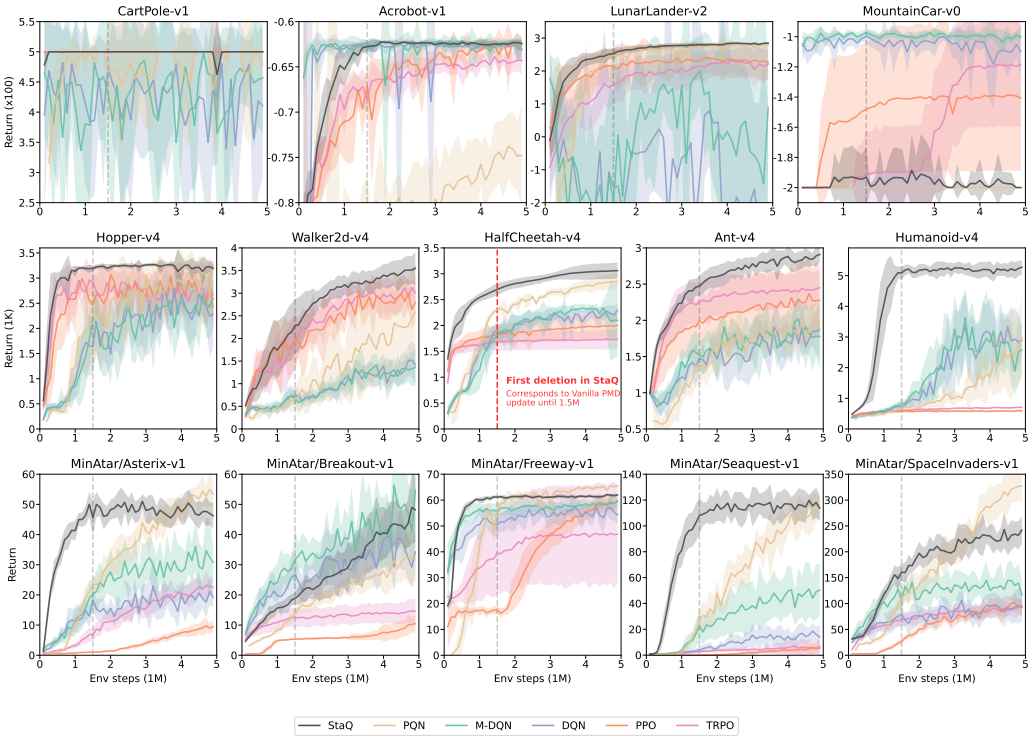


Figure 5: Policy performance across all environments.

	StaQ	PQN	M-DQN	DQN	PPO	TRPO
CartPole-v1	**500**	479	457	411	**500**	**500**
Acrobot-v1	**-62**	-75	-63	-63	-63	-64
LunarLander-v2	**285**	280	88	-317	227	222
MountainCar-v0	-200	-200	**-100**	-110	-141	-118
Hopper-v4	3196	**3263**	2600	2279	2411	2672
Walker2d-v4	**3550**	2585	1364	1424	2799	3010
HalfCheetah-v4	**3061**	2850	2098	2294	2001	1731
Ant-v4	**2910**	1879	1776	1871	2277	2452
Humanoid-v4	**5273**	2965	2580	2887	588	700
MinAtar/Asterix-v1	46	**53**	31	19	9	23
MinAtar/Breakout-v1	48	32	**55**	34	10	15
MinAtar/Freeway-v1	62	**65**	59	54	60	47
MinAtar/Seaquest-v1	**114**	**114**	51	14	5	7
MinAtar/SpaceInvaders-v1	242	**327**	116	95	92	94

Table 2: Final performance on all environments.

B.2 Stability plots (variation within individual runs)

In this section we provide further stability plots to complement Fig. 3 (**Left**). In Fig. 6-8 we plot the returns of the first three seeds of the full results (shown in Fig. 5). At each timestep, the returns for each individual seed are normalised by subtracting and then dividing by the mean across all seeds. In addition to the first three seeds, the shaded regions indicate one-sided tolerance intervals such that at least 95% of the population measurements are bounded by the upper or lower limit, with confidence level 95% (Krishnamoorthy & Mathew, 2009).

We can see from Fig. 6-8 that Approximate Policy Iteration (API) algorithms (StaQ, TRPO, PPO) generally exhibit less variation within runs than Approximate Value Iteration (AVI) ones (DQN, M-DQN, PQN). In simple environments, such as CartPole, all three API algorithms have stable performance, but on higher dimensional tasks, only StaQ retains a similar level of stability while maintaining good performance. This is especially striking on Hopper, where runs show comparatively little variation within iterations while having the highest average performance, as shown in Fig. 5. We attribute this improved stability in the performance of the evaluation policy by the averaging over a very large number of Q-functions ($M = 300$) of StaQ, which reduces the infamous performance oscillation of deep RL algorithms in many cases.

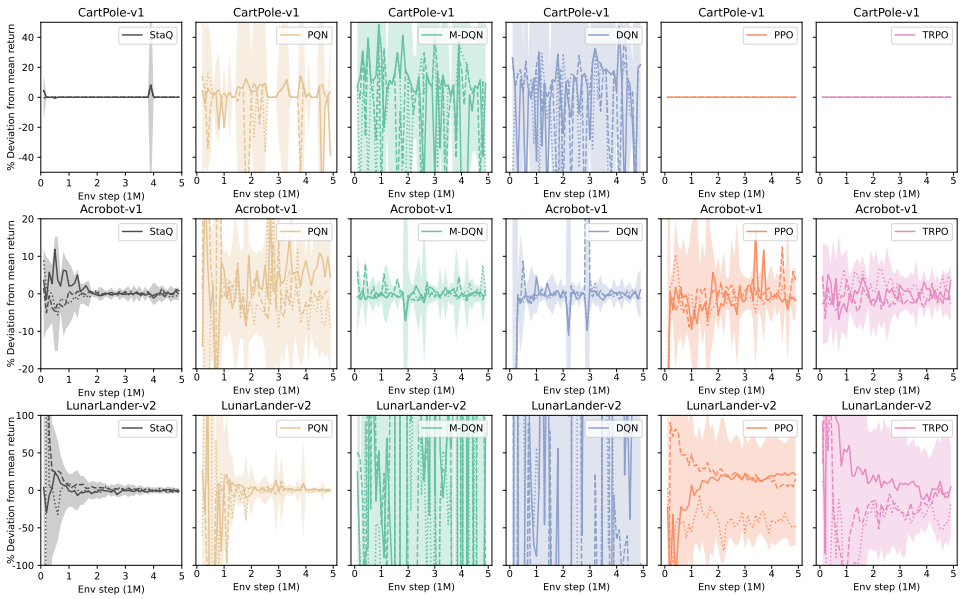


Figure 6: Stability plots for Classic Control environments, plotting normalized performance of the first three individual runs for each algorithm. See text for more details. Figures continue on the next page.

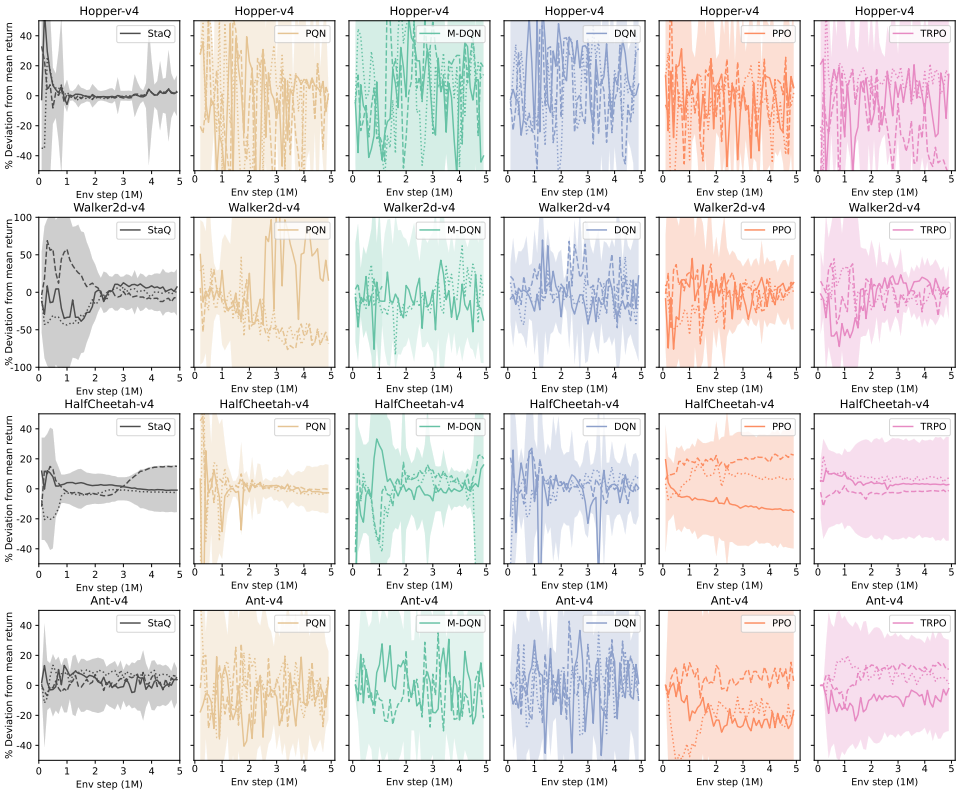


Figure 7: Stability plots for MuJoCo environments, plotting normalized performance of the first three individual runs for each algorithm. See text for more details. Figures continue on the next page.

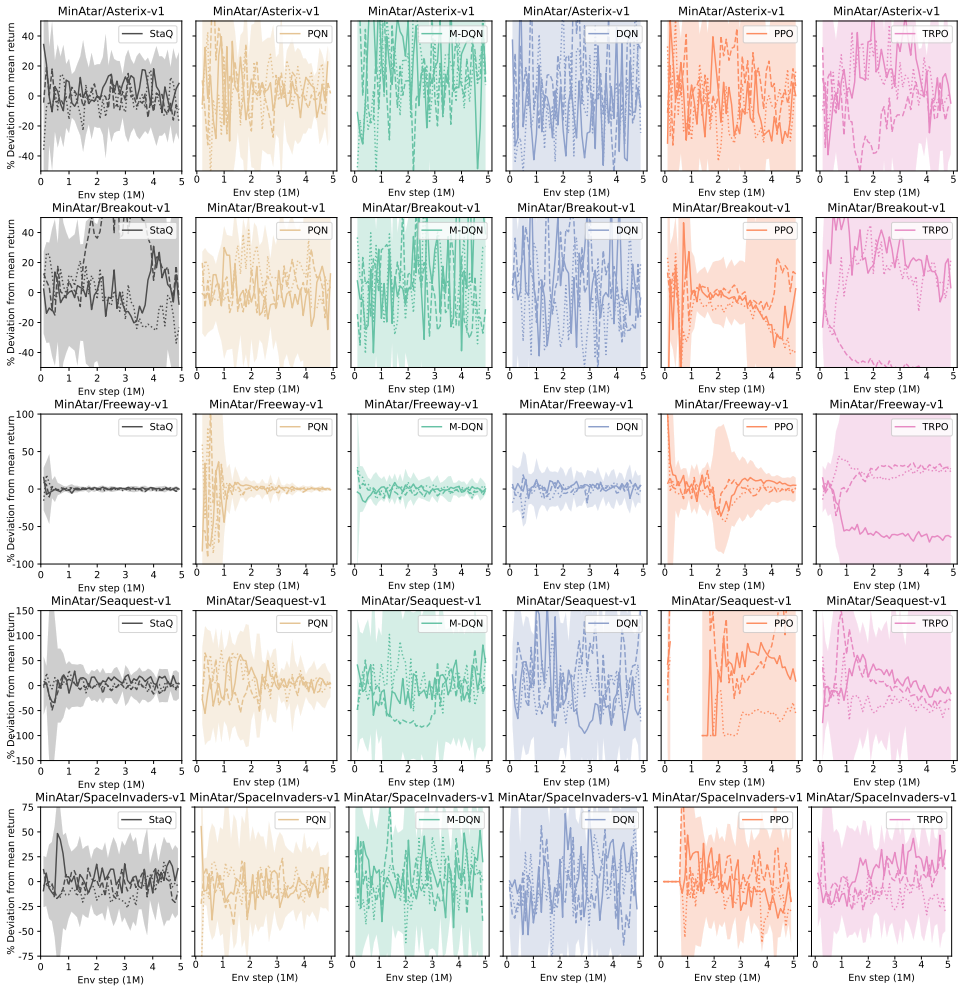


Figure 8: Stability plots for MinAtar environments, plotting normalized performance of the first three individual runs for each algorithm. See text for more details.

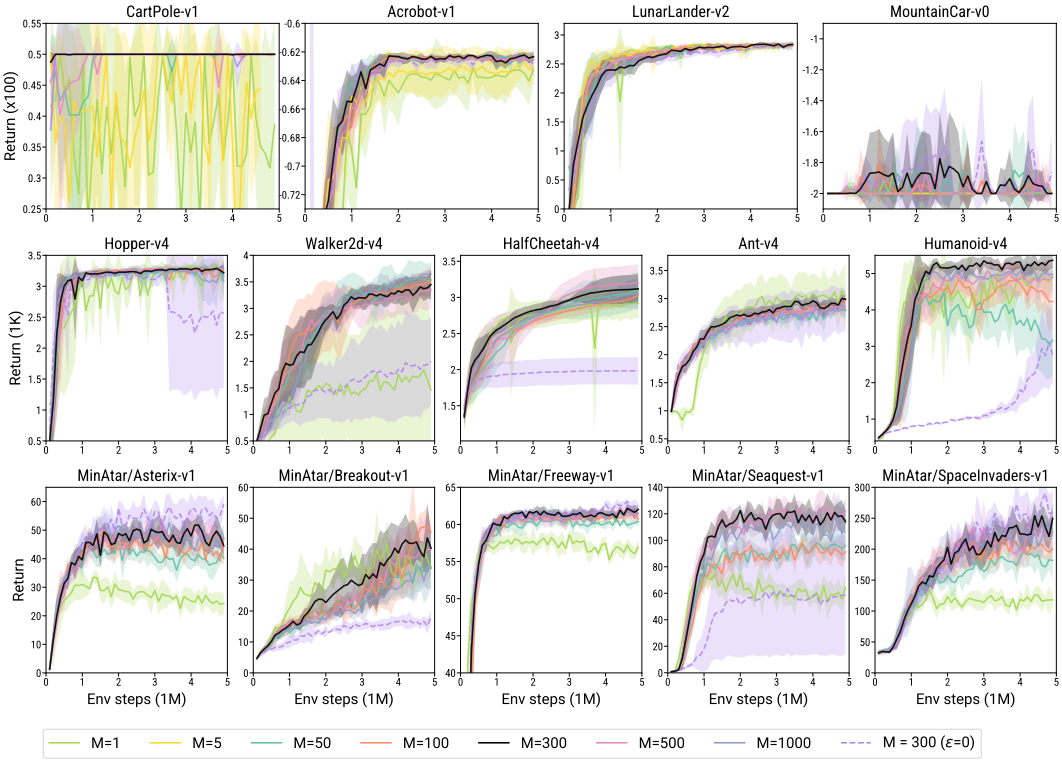


Figure 9: Ablation study for different memory sizes M and for $\epsilon = 0$, on all environments. Results showing the mean and one standard deviation averaged over 5 seeds.

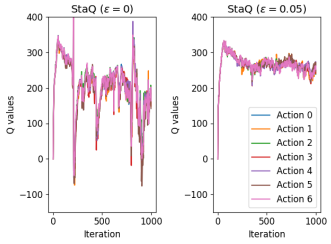
B.3 The impact of the memory-size M and value of ϵ

Figure 9 shows the performance of StaQ for different choices of M and for the hyperparameter $\epsilon = 0$ instead of $\epsilon = 0.05$ on additional MuJoCo tasks. Setting $M = 1$ corresponds to no KL-regularization as discussed in App. C and can be seen as an adaptation of SAC to discrete action spaces. $M = 1$ is unstable on both Hopper and Walker, in addition to Acrobot as shown in Fig. 3 in the main paper. Adding KL-regularization and averaging over at least 50 Q-functions greatly helps to stabilize performance except on the Humanoid task, as shown in Fig. 3, where $M = 50$ was still unstable compared to $M = 300$. Finally, the default setting of $\epsilon = 0.05$ outperforms a pure softmax policy with $\epsilon = 0$ on the Mujoco environments. While being fully on-policy (when $\epsilon = 0$) can benefit some MinAtar environments such as Asterix, Freeway and SpaceInvaders, for many other environments it may result in a very poor performance. We discuss some of the likely reasons for the need of ϵ-softmax exploration in the next section.

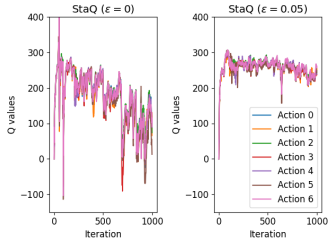
B.4 Instability in learning the Q-function

In certain environments such as Hopper-v4 (leftmost panel of Fig. 9), we observe that when the behavior policy π_k^b is the current softmax policy π_k (i.e. $\epsilon = 0$), there are more performance drops than with the ϵ-softmax behavior policy ($\epsilon > 0$) which StaQ uses by default.

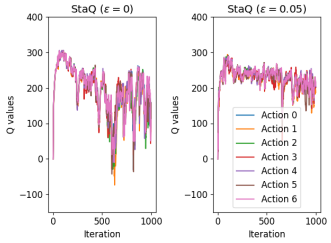
To understand why adding an ϵ-softmax policy on top of the softmax policy π_k stabilizes performance on Hopper-v4 as shown in Fig. 9, we have conducted the following experiment. We first launched two runs of StaQ with an ϵ-softmax policy on top of π_k, with ϵ being either 0.05 or 0. From these two runs, we collected 100 states spread along both training processes. We then launched 5 independent runs for each value of ϵ, and recorded for these



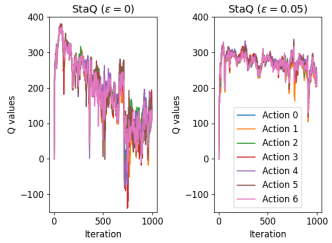
(a) Q-functions recorded at a given state across 1000 iterations for StaQ's behavior policy hyperparameter $\epsilon = 0$ and $\epsilon = 0.05$. Seed 0.



(b) Q-functions recorded at a given state across 1000 iterations for StaQ's behavior policy hyperparameter $\epsilon = 0$ and $\epsilon = 0.05$. Seed 1.



(c) Q-functions recorded at a given state across 1000 iterations for StaQ's behavior policy hyperparameter $\epsilon = 0$ and $\epsilon = 0.05$. Seed 2.



(d) Q-functions recorded at a given state across 1000 iterations for StaQ's behavior policy hyperparameter $\epsilon = 0$ and $\epsilon = 0.05$. Seed 3.

Figure 10: Q-values on four different states across 1000 iterations of StaQ, using an ϵ-softmax behavior policy to collect data in the replay, with $\epsilon = 0.05$ or $\epsilon = 0$. With $\epsilon = 0$, we noticed very large variations in the Q-function between iterations that are reduced when using $\epsilon = 0.05$.

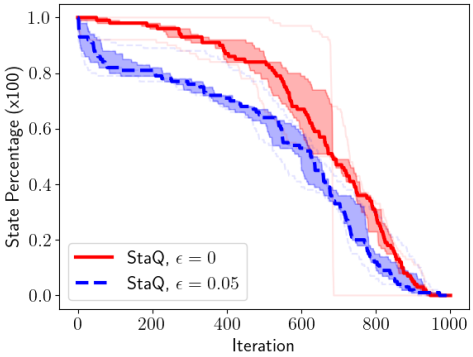


Figure 11: The percentage of states (out of 100 states) in which from iteration k and onward, an action was considered both the best and the worst according to ξ_k of EPMD. The difference of stability in the Q-values between $\epsilon = 0.05$ and $\epsilon = 0$ noted in Fig. 10 causes a difference in stability of policies, where actions switch more frequently from being worst to best when $\epsilon = 0$. The comparison is performed over 5 seeds showing the median and interquartile range.

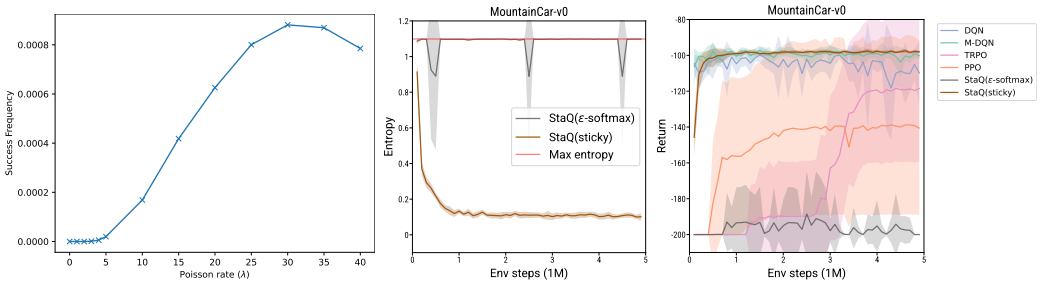


Figure 12: **Left:** Frequency of non-zero rewards of a uniform policy with sticky actions for different choice of Poisson rate λ on MountainCar over 5M timesteps. **Middle:** Entropy of learned policies under different behavior policies. Entropy of the uniform (Max entropy) policy plotted for reference. **Right:** Policy returns for StaQ with different behavior policies and deep RL baselines on MountainCar. Adding sticky actions to StaQ's behavior policy fixes its performance on this task.

100 states the learned Q-values at each iteration. Upon manual inspection of the Q-values, we immediately notice that when $\epsilon = 0$, the Q-values vary more wildly across time for all the actions, which can be seen as an analogue of *catastrophic forgetting* in Q-learning, the phenomenon widely studied in Continual Learning, see App. D. Fig. 10 shows a few examples for four different seeds. To understand whether these variations have any tangible impact on the instability of the policy, we have performed the following test: we compute the logits ξ_k at every iteration following the EPMD formula (Eq. 5) and rank the actions according to ξ_k. At each iteration k, we then compute the proportion of states, out of 100 reference states, in which an action has both the highest and the lowest rank in the next iterations $k' \geq k$. The results are shown in Fig. 11, where we can see that when $\epsilon = 0$, the fraction of states in which an action is considered as either being the best or the worst remains higher than when $\epsilon = 0.05$, which might result in performance drops across iterations. Thus the observed Q-function oscillations that appear more pronounced for $\epsilon = 0$ have a quantifiable impact on the stability of the policy, resulting in more states seeing actions switching from best to worst or vice versa.

It is hard to know exactly what causes the Q-values to oscillate more when $\epsilon = 0$. On the one hand, as these instabilities generally happen after the policy reached its peak performance, they could be because of some actions having very low probability of being selected in some states thus becoming under-represented in the replay buffer $\mathcal{D}_k$. Setting $\epsilon > 0$ ensures that all actions have a non-zero probability of being sampled at any given state. On the other hand, due to the convexity[2] of D_{KL}, i.e. $D_{\mathrm{KL}}\left((1-\epsilon)\pi + \epsilon p, (1-\epsilon)\pi' + \epsilon p'\right) \leq (1-\epsilon)D_{\mathrm{KL}}(\pi, \pi') + \epsilon D_{\mathrm{KL}}(p, p')$, if π_k^b is an ϵ-softmax strategy of π_k, then $D_{\mathrm{KL}}(\pi_k^b, \pi_{k+1}^b) \leq (1-\epsilon)D_{\mathrm{KL}}(\pi_k, \pi_{k+1})$ for any $\epsilon > 0$. This implies that successive replay buffers should be more similar when $\epsilon > 0$, which stabilizes the learning due to smoother transfer from Q_τ^k to Q_τ^{k+1}. Nonetheless, a case of $\epsilon > 0$ is not without its own challenges: ϵ-softmax policies, similarly to ϵ-greedy ones, might prevent deep exploration (Osband et al., 2016) and their off-policy nature might complicate learning (Kumar et al., 2020). We can see in Fig. 10 that $\epsilon = 0.05$ still exhibits sudden changes in the Q-function which might harm stability. While the averaging over past Q-functions of an EPMD policy can stabilize learning, we believe that the catastrophic forgetting in the Q-function itself should be addressed in the future work, which could potentially fix all remaining instabilities in deep RL.

B.5 Entropy regularization does not solve exploration

StaQ achieves competitive performance on all 14 environments except on MountainCar where it fails to learn, as can be seen in Fig. 5. In this section, we perform additional experiments to understand the failure of StaQ on MountainCar.

In short, it appears that the initial uniform policy—which has maximum entropy—acts as a strong (local) attractor for this task: StaQ starts close to the uniform policy, and exploration with this policy does not generate a reward signal in MountainCar. As StaQ does not observe a reward signal in early training, it quickly converges to the uniform policy which has maximum entropy, but also never generates a reward signal. Indeed, if we unroll a pure uniform policy on MountainCar for 5M steps, we will never observe a reward.

However, StaQ is not limited to a specific choice of behavior policy, and choosing a policy that introduces more correlation between adjacent actions, like a simple "sticky" policy allows StaQ to solve MountainCar. This policy samples an action from π_k and applies it for a few consecutive steps, where a number of steps is drawn randomly from Poisson(λ) distribution (in our experiments with StaQ we fix the rate of Poisson distribution at $\lambda = 10$). In Fig. 12, we can see that StaQ with the same hyperparameters for classical environments (see Table 3) and a "sticky" behavior policy manages to find a good policy for MountainCar matching the best baseline. The final policy demonstrates much lower entropy compared to ϵ-softmax policy that fails at learning for this environment, which confirms our statement that entropy maximization cannot be a universal tool when dealing with the exploration problems.

C Comparison with Soft Actor-Critic

In this appendix, we explain the relation between Soft Actor-Critic (SAC, Haarnoja et al. (2018)) and both M-DQN (Vieillard et al., 2020b) and StaQ with $M = 1$. SAC is not directly used as a baseline because SAC is not compatible with discrete action spaces. However, M-DQN can be seen as an adaptation of SAC to discrete action spaces with an additional KL-divergence regularizer. Please see the discussion in Vieillard et al. (2020b) on page 3, between Eq. (1) and (2). Vieillard et al. (2020b) also describe Soft-DQN in Eq. (1) as a straightforward discrete-action version of SAC, that can be obtained from M-DQN by simply setting the KL-divergence regularization weight to zero. Soft-DQN was not included as a baseline because the results of Vieillard et al. (2020b) suggest that M-DQN generally outperforms Soft-DQN.

We also note that by setting $M = 1$ in StaQ, we remove the KL-divergence regularization and only keep the entropy bonus. This baseline can also be seen as an adaptation of SAC to discrete action spaces: indeed, if we set $M = 1$ in Eq. (11) we recover the policy logits

$$\begin{aligned}\xi_{k+1} &= \frac{\alpha}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i Q_\tau^{k-i} \\ &= \frac{\alpha}{1-\beta}Q_\tau^k \\ &= \frac{Q_\tau^k}{\tau},\end{aligned}$$

where the last line is due to $\alpha\tau = 1-\beta$. This results in a policy of the form $\pi_{k+1} \propto \exp\left(\frac{Q_\tau^k}{\tau}\right)$. Meanwhile, for SAC, the actor network is obtained by minimizing the following problem (Eq.

[2]See e.g. `https://statproofbook.github.io/P/kl-conv.html` for the proof.

14 in Haarnoja et al. (2018))

$$\pi_{k+1} = \arg\min \mathrm{KL}\left(\pi \middle| \frac{\exp\left(\frac{Q_\tau^k}{\tau}\right)}{Z_{\text{norm.}}}\right).$$

However, in the discrete action setting, we can sample directly from $\exp\left(\frac{Q_\tau^k}{\tau}\right)$—which is the minimizer of the above KL-divergence term—and we do not need an explicit actor network. As such StaQ with $M = 1$ could be seen as an adaptation of SAC to discrete action spaces.

D A Continual Learning Perspective to Entropy Regularized Deep RL

Reinforcement Learning has strong ties with CL due to the sequential nature in which data arrives. This is true even in this "single-task RL" setting, where we consider only a single MDP, unlike Continual RL (Lesort et al., 2020) where the learner is presented with a sequence of MDPs and one evaluates whether the learner is able learn on the new MDPs while retaining the information of older ones (De Lange et al., 2021; Wang et al., 2024). Drawing a connection with RL is interesting because it opens up a plethora of CL methods that are not well researched in the deep RL context, but are applicable even in a single task setting. Specifically, in this paper we focus on the entropy regularized policy update problem described below (Eq. 3 of the paper)

$$\text{for all } s \in S, \pi_{k+1}(s) = \arg\max_{p \in \Delta(A)} \left\{ Q_\tau^k(s) \cdot p - \tau h(p) - \eta D_{\mathrm{KL}}(p; \pi_k(s)) \right\}.$$

The objective of this update can be seen as CL, as we receive a new "task" which is to find p a maximum entropy distribution over actions that puts its largest mass on actions with high Q-values, yet, through the KL-divergence term above, we do not want to differ too much from π_k, and forget the solution of the previous "task". Because of this similarity with CL, existing methods to solve this problem can be categorized with the CL literature lens, for example: Lazic et al. (2021) used a rehearsal method (replay buffer/experience replay in deep RL terminology) to tackle the above policy update, while Schulman et al. (2015) uses a parameter regularization approach. These methods cover two of the three main classes of CL methods (De Lange et al., 2021), and the novelty of this paper is in investigating a method pertaining to the third class (parameter isolation) to tackle this problem, as this class of methods has strong performance in CL benchmarks (See Sec. 6 of De Lange et al. (2021)), yet remains largely understudied in deep RL.

E Hyperparameters

Here, we provide the full list of hyperparameters used in our experiments. StaQ's hyperparameters are listed in Table 3, while the hyperparameters for our baselines are provided in Tables 4-7. For TRPO and PPO, we use the implementation provided in `stable-baselines4` (Raffin et al., 2021), while we used our in-house PyTorch implementation of (M)-DQN. For PQN, we use the CleanRL implementation (Huang et al., 2022).

Across all environments, we enforce a time limit of 5000 steps. This is particularly useful for Seaquest-v1, since an agent can get stuck performing an infinitely long rollout during data collection. For MinAtar environments, we followed the network architecture used by Young & Tian (2019) consisting of a single convolutional layer with 16 channels and a 3×3 kernel (Conv(16, 3, 3)) followed by a linear layer with 128 neurons.

To account for the different scales of the reward between environments, we apply a different reward scaling to the Classic/MuJoCo environments and MinAtar. Note that this is equivalent to inverse-scaling the entropy weight τ and KL weight η, ensuring that ξ_k is of the same order of magnitude for all environments. To account for the varying action dimension $|A|$ of the environments, we set the *scaled entropy coefficient* $\bar{\tau}$ as a hyperparameter, defined by $\bar{\tau} = \tau \log |A|$, rather than directly setting τ. Furthermore, the entropy weight is linearly annealed from its minimum and maximum values.

Policy evaluation. In all our experiments, we use an ensemble of two neural networks, similarly to e.g. SAC (Haarnoja et al., 2018), to evaluate a Q-function and therefore two SNNs for ξ-logits. In particular, we optimize the current Q-function weights θ to minimize the loss $\mathcal{L}(\theta)$,

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta\left(s,a\right) - \hat{Q}\left(s,a\right)\right)^2\right] \tag{176}$$

$$\hat{Q}(s,a) := R(s,a) + \gamma \mathbb{E}_{s'\sim\mathcal{D}, a\sim\pi(s')}\left[\text{agg}_{i\in\{1,2\}}\, Q_{\hat{\theta}_i}(s',a') - \tau h(\pi(s'))\right] \tag{177}$$

where agg computes either the `min` or `mean` over the target Q-functions with weights $\hat{\theta}_1, \hat{\theta}_2$. We find that using the `min` of the two Q-functions to compute the target values often results in more stable training. `min` gives a more conservative target that is robust to overestimation bias in the Q-functions, and this allows us to reduce the KL weight. However, such a strategy may struggle when reward is not dense enough, e.g. some MinAtar and some classic control environments. Therefore we instead use the `mean` in Classic/MinAtar environments. Future work could use a more sophisticated approach that is both robust to overestimation bias and yet sensitive to weak reward signals.

Hyperparameter	Classic	MuJoCo	MinAtar
Discount (γ)	0.99	0.99	0.99
Memory size (M)	300	300	300
Policy update interval	5000	5000	5000
Ensembling mode	**mean**	**min**	**mean**
Target type	hard	hard	hard
Target update interval	200	200	200
Epsilon	0.05	0.05	0.05
Reward scale	**10**	**10***	**100**
KL weight (η)	**20**	**10**	**20**
Initial scaled ent. weight	2.0	2.0	2.0
Final scaled ent. weight	0.4	0.4	0.4
Ent. weight decay steps	**500K**	**1M**	**1M**
Architecture	256×2	256×2	Conv(16, 3, 3) + 128 MLP
Activation function	ReLU	ReLU	ReLU
Learning rate	0.0001	0.0001	0.0001
Optimizer	Adam	Adam	Adam
Replay capacity	50K	50K	50K
Batch size	256	256	256

Table 3: StaQ hyperparameters, with parameters which vary across environment types in bold. *Hopper-V4 uses a reward scale of 1.

Hyperparameter	Classic	MuJoCo	MinAtar
Discount factor (γ)	0.99	0.99	0.99
Horizon	2048	2048	1024
Num. epochs	10	10	3
Learning starts	5000	20000	20000
GAE parameter	0.95	0.95	0.95
VF coefficient	0.5	0.5	1
Entropy coefficient	0	0	0.01
Clipping parameter	0.2	0.2	$0.1 \times \alpha$
Optimizer	Adam	Adam	Adam
Architecture	64×2	64×2 *	Conv(16, 3, 3) + 128 MLP
Activation function	Tanh	Tanh	Tanh
Learning rate	3×10^{-4}	3×10^{-4}	$2.5 \times 10^{-4} \times \alpha$
Batch size	64	64	256

Table 4: PPO hyperparameters, based on (Schulman et al., 2017). In the MinAtar environments α is linearly annealed from 1 to 0 over the course of learning. *Humanoid-v4 uses a hidden layer size of 256.

Hyperparameter	Classic	MuJoCo	MinAtar
Discount factor (γ)	0.99	0.99	0.99
Horizon	2048	2048	2048
Learning starts	5000	20000	20000
GAE parameter	0.95	0.95	0.95
Stepsize	0.01	0.01	0.01
Optimizer	Adam	Adam	Adam
Architecture	64×2	64×2 *	Conv(16, 3, 3) + 128 MLP
Activation function	Tanh	Tanh	Tanh
Learning rate	3×10^{-4}	3×10^{-4}	2.5×10^{-4}
Batch size	64	64	256

Table 5: TRPO hyperparameters, based on (Schulman et al., 2015). *Humanoid-v4 uses a hidden layer size of 256.

Hyperparameter	Classic	MuJoCo	MinAtar
Discount factor (γ)	0.99	0.99	0.99
Lambda	0.95	0.85	0.65
Epsilon start	1.0	1.0	1.0
Epsilon finish	0.05	0.05	0.05
Decay steps	1M	1M	1M
Num envs	32	32	128
Num steps	64	32	32
Max Grad Norm	10.0	10.0	10.0
Num minibatches	16	32	32
Num epochs	4	4	2
Optimizer	RAdam	RAdam	RAdam
Architecture	128×2	128×2	Conv(16, 3, 3) + 128 MLP
Normalization Type	LayerNorm	LayerNorm	LayerNorm
Input Normalization	None	None	None
Activation function	ReLU	ReLU	ReLU
Learning rate*	1×10^{-4}	1×10^{-4}	1×10^{-4}

Table 6: PQN hyperparameters. Classic and MinAtar hyperparameters are based on the original paper (Gallici et al., 2025), while MuJoCo hyperparameters were found by hyperparameter tuning. * Learning rate linearly annealed to 0 across the course of training.

Hyperparameter	Classic	MuJoCo	MinAtar
Discount factor (γ)	0.99	0.99	0.99
Target update interval	100	8000	8000
Epsilon	0.1	0.1	0.1
Decay steps	20K	20K	20K
M-DQN temperature	0.03	0.03	0.03
M-DQN scaling term	1.0	0.9	0.9
M-DQN clipping value	-1	-1	-1
Architecture	512×2	128×2	Conv(16, 3, 3) + 128 MLP
Activation function	ReLU	ReLU	ReLU
Learning rate	1×10^{-3}	5×10^{-5}	2.5×10^{-4}
Optimizer	Adam	Adam	Adam
Replay capacity	50K	1M	1M
Batch size	128	32	32

Table 7: MDQN and DQN hyperparameters, based on (Vieillard et al., 2020b; Ceron & Castro, 2021)

F Training and Inference Time Comparisons

	Memory size M	1	50	100	300	500
Hopper-v4	Training time (hrs)	9.8	10.1	10.3	10.3	10.9
	Inference speed (steps/s)	610	610	620	640	600
Ant-v4	Training time (hrs)	10.4	10.7	10.3	11	10.5
	Inference speed (steps/s)	540	570	560	540	560

Table 8: Training and inference times for StaQ, as a function of M, on Hopper-v4 (state dim.=11) and Ant-v4 (state dim. = 105), computed on an NVIDIA Tesla V100 and averaged over 3 seeds.

		StaQ	PPO	TRPO	M-DQN	DQN	PQN
Hopper-v4	Training time (hrs)	10.3	3.7	3.2	5.6	4.9	0.6
	Inference speed (steps/s)	640	1040	1020	1550	1460	3740
Ant-v4	Training time (hrs)	11	4.3	3.6	6.1	5.3	0.6
	Inference speed (steps/s)	540	830	850	1110	1040	2180

Table 9: Training and inference times for StaQ ($M = 300$) vs baselines, on the Hopper-v4 and Ant-v4 environments. Timings are computed on an NVIDIA Tesla V100, averaged over 3 seeds.

In this section, we report the training time and inference speed of StaQ, as a function of memory size M and state space dimension. We also compare it to the deep RL baselines. All timing experiments were computed on an NVIDIA Tesla V100, and averaged over 3 seeds. The training time is defined as the time required to train StaQ for 5 million timesteps, not including the time require for data collection, while the inference speed is measured by the number of environment steps per second that can be evaluated during inference. Table 8 shows that memory size and dimension of the state space have a negligible impact on training and inference times, as discussed in Sec 6. Table 9 compares the training and inference time of StaQ ($M = 300$) with the baselines.

G Pseudocode of StaQ

We provide in this section the pseudocode of StaQ in Alg. 1. As an approximate policy iteration algorithm, StaQ comprises three main steps: i) data collection, ii) policy evaluation iii) policy improvement. Data collection (Line 4-5) consist in interacting with the environment to collect transitions of type (state, action, reward, next state) that are stored in a replay buffer. A policy evaluation algorithm (Eq. 176) is then called to evaluate the current Q-function Q_τ^k using the replay buffer. Finally, the policy update is optimization-free and simply consists in stacking the Q-function in the SNN policy as discussed in Sec. 5. After K iterations, the last policy is returned.

Algorithm 1 StaQ (Finite-memory entropy regularized policy mirror descent)

Input: An MDP $\mathcal{M}$, a memory-size M, Number of samples per iteration N, Replay buffer size D, Initial behavior policy π_0^b, entropy weight τ, D_{KL} weight η, ϵ-softmax exploration parameter
Output: Policy $\pi_K \propto \exp(\xi_K)$
for $k = 0$ **to** $K - 1$ **do**
    Interact with $\mathcal{M}$ using the behavior policy π_k^b for N times steps
    Update replay buffer $\mathcal{D}_k$ to contain the last D transitions
    Learn Q_τ^k from $\mathcal{D}_k$ using a policy evaluation algorithm (Eq. 176)
    Obtain logits ξ_{k+1} by stacking the last M Q-functions (see Sec. 5) following the finite-memory EPMD update of Eq. 10.
    Set $\pi_{k+1} \propto \exp(\xi_{k+1})$ and π_{k+1}^b to an ϵ-softmax policy over π_{k+1}
end for