A CONTINUAL LEARNING PERSPECTIVE TO ENTROPY REGULARIZED DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Research on Continual Learning (CL) tackles learning with non-stationary data distributions. The non-stationary nature of data is also one of the challenges of deep Reinforcement Learning (RL), and as a consequence, both CL and deep RL rely on similar approaches to stabilize learning, from the use of replay buffers to the choice of regularization terms. However, while dynamic neural architectures that grow in size to learn new tasks without forgetting older ones are well researched in CL, it remains a largely understudied research direction in RL. In this paper, we argue that Policy Mirror Descent (PMD), a regularized policy iteration RL algorithm, would naturally benefit from dynamic neural architectures as the current policy is a function of the sum of all past Q-functions. To avoid indefinitely increasing the neural architecture, we study PMD-like algorithms that only keep in memory the last M Q-functions, and show that a convergent algorithm can be derived if M is large enough. This theoretical analysis provides insights on how to utilise a fixed budget of Q-functions to reduce catastrophic forgetting in the policy. We implement this algorithm using a new neural architecture that stacks the last M Q-functions as 3-dimensional tensors to allow for fast GPU computations. StaQ, the resulting algorithm, is competitive with state-of-the-art deep RL baselines and typically exhibits lower variance in performance. Beyond its performance, we argue that the simplicity and strong theoretical guarantees of StaQ's policy update makes it an ideal research tool over which we can further build a fully stable deep RL algorithm.

1 INTRODUCTION

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). A CL setting usually goes in pair with memory constraints, limiting the storage of past data. As it cannot access old data, it is desirable for a CL learner to possess i) stability, which is the ability to retain prior knowledge ii) plasticity, which is the ability to learn new knowledge. When training a neural network (NN), these two properties are typically conflicting and we talk of the plasticity-stability dilemma (Grossberg, 1988), which if not handled properly can lead to catastrophic forgetting (McCloskey & Cohen, 1989), i.e. a sudden loss of prior knowledge when trying to acquire new knowledge. To fight catastrophic forgetting, CL applies methods that can be categorized in three groups (De Lange et al., 2021): a) rehearsal methods (Robins, 1995; Rolnick et al., 2019) that keep samples from older data distributions to periodically retrain a learner on them, b) regularization-based methods (Li & Hoiem, 2016; Kirkpatrick et al., 2017; Zenke et al., 2017), that prevent an NN from changing too fast and c) parameter isolation methods (Rusu et al., 2016; Xu & Zhu, 2018; Yoon et al., 2018; Li et al., 2019) that grow an NN, usually while freezing older weights, to increase plasticity without compromising on stability.

Somewhat independently, deep RL has known 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). RL is not always a true CL setting in the sense that an agent may not need to continually adapt to remain optimal, as most environments considered in RL are of a stationary nature, unlike in CL (Abel et al., 2023). However, deep RL and CL both face non-stationary datasets. This non-stationarity coupled with the use of neural networks makes deep RL very sensitive to hyper-parameters (Henderson, 2018) and makes its empirical behavior often poorly align

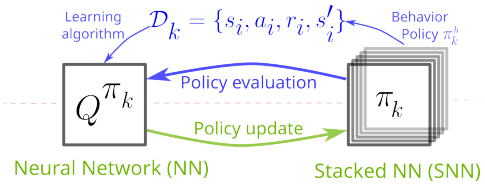


Figure 1: Overview of StaQ, showing a continual training of a Q-function (left), from which we periodically "stack" frozen weight snapshots to form the policy (right). See Sec. 5 for more details. At each iteration k, two steps are performed. i) Policy evaluation, where we generate a dataset $\mathcal{D}_k$ 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 remain sources of instabilities in this deep RL setting.

with our theoretical understandings (Ilyas et al., 2020; Kumar et al., 2020; van Hasselt et al., 2018). As they face similar problems, deep RL and CL often end-up using similar remedies (see discussion in Sec. 2.4.3 of Lesort et al. (2020)). For instance, the rehearsal methods in CL have their counterpart in the use of replay buffers in DQN (Mnih et al., 2015) or SAC (Haarnoja et al., 2018). As for regularization-based CL methods, approaches such as EWC (Kirkpatrick et al., 2017) and its online version (Chaudhry et al., 2018) are extremely similar to TRPO (Schulman et al., 2015)—contrast the regularized cost of Eq. 5 in Chaudhry et al. (2018) with the Kullback-Leibler divergence (further referred to as D_{KL}) in regularized policy update of TRPO, they are identical up to differences in loss functions between supervised learning and RL. However, while parameter isolation methods have been tested in *continual* RL settings (Rusu et al., 2016), their use in *single* task RL problems remains largely understudied.

In this paper, our main inspiration from the CL research community is the use of growing neural architectures to implement our policy update. Such neural architectures belong to the class of 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)). While the policy's growing architecture eliminates catastrophic forgetting in the policy by construction, it still seems to be present in the Q-function network (Sec. 6.3), causing instabilities of the learning process. We provide some ad hoc workarounds in this paper, but we believe several CL approaches—as those discussed in Sec. 7—could be helpful to further stabilize deep RL, and we hope that one side effect of this paper is to further encourage the transfer of ideas between the RL and CL communities.

Our policy update that uses a growing neural architecture is from a class of Policy Mirror Descent algorithms (PMD, Neu et al. (2017); Abbasi-Yadkori et al. (2019); Geist et al. (2019); Zhan et al. (2023)). PMD with D_{KL}-regularization is a policy iteration algorithm where the policy at iteration k is given by a weighted average of past Q-functions, such that

$$\pi_k \propto \exp\left(\alpha \sum_{i=0}^{k} \beta^i Q^{k-i}\right), \tag{1}$$

for temperature $\alpha > 0$ and decay factor $\beta \in [0, 1]$, where β is strictly < 1 if an entropy bonus is used in addition to the D_{KL} regularizer (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 policy in Eq. 1 naturally calls for a parameter isolation approach that would store all past Q-functions, leading to exact entropy-regularized policy update. Abbasi-Yadkori et al. (2019) performed an experiment of the sort on an Atari RL task (Bellemare et al., 2013), keeping in memory the past 10 Q-functions, and noted increased stability and performance over vanilla DQN.

Overall, this paper extends prior work on PMD (Abbasi-Yadkori et al., 2019; Lazic et al., 2021; Zhan et al., 2023) in several ways. i) We propose an NN architecture called Stacked NNs (SNNs) that stacks NN weights through a third dimension (representing the iteration number, see Sec. 5 for more details), allowing us to efficiently compute on a GPU several hundreds of Q-functions, approaching the exact policy formulation in Eq. 1. ii) We study theoretically the convergence of PMD-like algorithms that store up to M most recent Q-functions, replacing the oldest Q-function

with the newest one when $k > M$. These theoretical results extend those of (Zhan et al., 2023) that studied PMD with an inexact policy update. In Eq. 1, when $\beta < 1$, one can clearly see that the weight of old Q-functions decays to 0 as k goes to infinity. Thus, by choosing a large enough M, we can control the error stemming from the use of a finite number of Q-functions. A more subtle result of our theoretical analysis is that Q-functions become increasingly similar as learning progresses. As a result, deleting an old Q-function affects less and less the policy, and we can show that there is a finite M—that depends on β and the discount factor γ of the decision task—large enough to ensure convergence to the optimal policy without residual error, although at a reduced rate than exact PMD. iii) Experiments on a large set of tasks confirm the preliminary results of Abbasi-Yadkori et al. (2019): the averaging over past Q-functions stabilizes learning and improves performance in several cases. The proposed algorithm, which we call StaQ, uses a closed-form policy update that reduces the sources of errors and increase the overall stability. Still, some instabilities may remain coming from the choice of the behavior policy (see Fig. 1) or the policy evaluation algorithm. Yet, we believe that the simplicity of StaQ's policy update, its strong theoretical foundation and its empirical behavior brings us closer to a completely stable deep RL algorithm.

2 Related Work

Entropy-regularization in RL. Entropy regularization has seen widespread usage in RL. It was used with (natural) policy gradient (Kakade, 2001; Schulman et al., 2015), 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., 2020). 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; **?**), but other forms of entropy 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. (2020), 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. Finally, Zhan et al. (2023) is in the same setting as our work but is mostly theoretical. In this paper, we study a family of practical deep RL algorithms that extend the analysis of the approximate algorithms in Zhan et al. (2023), and provide both a theoretical analysis and an extensive empirical evaluation of these algorithms.

Continual Learning (CL) and RL. In CL a learner faces non-stationary data distributions (commonly referred to as *tasks*) and aims at obtaining good performance (e.g. classification accuracy) on the newest task without hindering performance on previous tasks (De Lange et al., 2021; Wang et al., 2024). RL and CL connect in several ways: some continual supervised learning settings can be formulated as RL problems (Kumar et al., 2023), while RL can be extended to the CL setting by having non-stationary rewards or environment dynamics (Rusu et al., 2016; Khetarpal et al., 2020). Even when the rewards and dynamics are stationary—setting which we refer to as single task RL—data in RL can still arrive in a non-stationary fashion as in CL. As such, both RL and CL can end-up using surprisingly similar approaches, as discussed in the previous section. Nonetheless, not all research in CL might be relevant to single task RL and vice versa. For example, in RL, the learner can (indirectly) control the data distribution, providing additional mechanisms to reduce catastrophic forgetting which are not always possible in CL (see Sec. 6.3 for an example).

Growing neural architectures in RL. Saving past Q-functions was 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. Anschel et al. (2017) extended DQN by saving the past 10 Q-functions, and using them to compute lower variance target values. 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, as it uses a D_{KL}-regularizer but does not include 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. 1 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). Such evolutionary approaches were already used for RL tasks but are beyond the scope of this paper.

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 $R : S \times A \mapsto [-R_{\mathrm{x}}, R_{\mathrm{x}}]$ 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 because of the sampling from softmax distributions as in Eq. 1. 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:

$$V_\tau^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \{R(s_t, a_t) - \tau h(\pi(s_t))\} \middle| s_0 = s \right]. \tag{2}$$

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 and Q-function are also related by $V_\tau^\pi(s) = \mathbb{E}_a \left[Q_\tau^\pi(s, a) \right] - \tau h(\pi(s))$ which leads to the Bellman equation:

$$Q_\tau^\pi(s, a) = R(s, a) + \gamma \mathbb{E}_{s', a'} \left[Q_\tau^\pi(s', a') - \tau h(\pi(s')) \right]. \tag{3}$$

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 by:

$$\text{for all } s \in S, a \in A, \quad V_\tau^\star(s) := \max_\pi V_\tau^\pi(s), \quad Q_\tau^\star(s, a) := \max_\pi Q_\tau^\pi(s, a). \tag{4}$$

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 ith entry $i \in \{1, \ldots, |A|\}$ is equal to $Q_\tau^\pi(s, i)$. Finally we define

$$\bar{R} := \frac{R_{\mathrm{x}} + \gamma \tau \log |A|}{1 - \gamma}, \tag{5}$$

as the finite upper-bound of $\|Q_\tau^\pi\|_\infty$ for any policy π.

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 (Zhan et al., 2023; Lan, 2022). 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:

$$\text{for all } s \in S, \pi_{k+1}(s) = \underset{p \in \Delta(A)}{\arg\max} \left\{ Q_\tau^k(s) \cdot p - \tau h(p) - \eta D_{\mathrm{KL}}(p; \pi_k(s)) \right\}, \tag{6}$$

$$\propto \pi_k(s)^{\frac{\eta}{\eta+\tau}} \exp\left(\frac{Q_\tau^k(s)}{\eta + \tau} \right), \tag{7}$$

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. 7 is well-known and its proof can be checked in Zhan et al. (2023) for instance. 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 $\xi_0 = 0$. At every iteration of EPMD, the policy is given by

$$\pi_{k+1} \propto \exp(\xi_{k+1}), \quad \xi_{k+1} = \beta\xi_k + \alpha Q_\tau^k. \tag{8}$$

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 $d = \beta + \gamma(1-\beta) < 1$.*

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. The theoretical study of the finite-memory EPMD where the Q-functions are not exact is left for future work, but investigated experimentally in Sec. 6, where Q-functions are approximated by NNs.

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. 1 to the last M Q-functions. The convergence analysis of this algorithm is largely based on the analysis of the approximate EPMD of Zhan et al. (2023), which assumes a bounded sub-optimality of the policy update procedure (Eq. 6). Compared to the proof from the prior work, the main step in our setting consists in quantifying the effect of the finite-memory assumption to the policy improvement step. 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 that rescales the policy in Eq. 1 to account for its finite-memory nature. This rescaling introduces some long range dependencies that complicate the analysis, but can result in convergence to $Q_\tau^\star$, provided a large enough M.

4.1 Vanilla finite-memory EPMD

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

$$\xi_{k+1} = \beta\xi_k + \alpha\left(Q_\tau^k - \beta^M Q_\tau^{k-M}\right), \tag{9}$$

with $Q_\tau^{k-M} := 0$ whenever $k - M < 0$. Compared to ξ_{k+1} in Eq. 8, we simultaneously add the new Q_τ^k and 'delete' an old Q-function by subtracting Q_τ^{k-M} in Eq. 9. As a result, ξ_{k+1} can now be written as $\xi_{k+1} = \alpha \sum_{i=0}^{M-1} \beta^i Q_\tau^{k-i}$, and thus it 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. 9 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:

$$\text{for all } s \in S, \quad \pi_{k+1}(s) = \underset{p \in \Delta(A)}{\arg\max} \left\{Q_\tau^k(s) \cdot p - \tau h(p) - \eta D_{\mathrm{KL}}(p; \tilde{\pi}_k(s))\right\}. \tag{10}$$

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 a generic 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. 10*

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

$$Q_\tau^{k+1} \geq Q_\tau^k - \gamma\eta \frac{\max_{s\in S} \|(\pi_k - \tilde{\pi}_k)(s)\|_1 \left\|\xi_k - \tilde{\xi}_k\right\|_\infty}{1-\gamma}. \tag{11}$$

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. 10, then the Q-function Q_τ^{k+1} of π_{k+1} satisfies*

$$Q_\tau^{k+1} \geq Q_\tau^k - \gamma\beta^M \frac{\min\left\{2, \alpha\beta^{M-1}\bar{R}\right\}\bar{R}}{1-\gamma}. \tag{12}$$

In vanilla EPMD, it is guaranteed that $Q_\tau^{k+1} \geq Q_\tau^k$ (Zhan et al., 2023). In this approximate setting, we can bring the error 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. 9, we have that $\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty \leq \gamma d^k \left(\|Q_\tau^\star\|_\infty + \left\|Q_\tau^0\right\|_\infty\right) + \gamma\beta^k \left\|Q_\tau^0\right\|_\infty + \beta^M C_1$, with $d = \beta + \gamma(1-\beta) < 1$ and $C_1 = \frac{\gamma\bar{R}}{1-\gamma}\left[\frac{\gamma \min\left\{2, \alpha\beta^{M-1}\bar{R}\right\}}{(1-\beta)(1-\gamma)} + 2\right]$.*

Since $\beta < d < 1$, the slowest term vanishes 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

$$\xi_{k+1} = \beta\xi_k + \alpha Q_\tau^k + \frac{\alpha\beta^M}{1-\beta^M}(Q_\tau^k - Q_\tau^{k-M}), \tag{13}$$

where $Q_\tau^{k-M} := 0$ whenever $k - M < 0$. In contrast to the vanilla algorithm in Sec. 4.1, we now both 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.5 for a proof) for all $k \geq 0$ that the logits only use the past M Q-functions and are given by

$$\xi_{k+1} = \frac{\alpha}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i Q_\tau^{k-i}. \tag{14}$$

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. 10, 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. 10 with the hereby defined Q_τ^k and $\tilde{\pi}_k$, then the Q-function Q_τ^{k+1} of π_{k+1} satisfies*

$$Q_\tau^{k+1} \geq Q_\tau^k - 2\gamma\beta^M \frac{\left\|Q_\tau^k - Q_\tau^{k-M}\right\|_\infty}{(1-\gamma)(1-\beta^M)}. \tag{15}$$

Compared to the approximate policy improvement of Sec. 4.1, we see that the lower-bound in Eq. 15 now 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 should guarantee 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. 13 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. 13, 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, $\forall k \geq 0$, $\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty \leq (d_1 + d_2 d_3^{-1})^k \max\left\{\frac{\|Q_\tau^\star\|_\infty}{\gamma}, \|Q_\tau^\star\|_\infty + \left\|Q_\tau^0\right\|_\infty\right\}$, 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 $Q_\tau^\star$. 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 in practice we did not observe 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 $Q_\tau^\star$.

5 Stacked Neural Networks

To study finite-memory EPMD in the memory regimes suggested by Thm. 4.4, we propose the Stacked NN architecture (SNN, illustrated in Fig. 1) that makes efficient use of GPUs to compute multiple Q-values in parallel, and we call the resulting algorithm StaQ. An SNN is parameterized by M, the maximum number of networks kept in memory. The main operation of an SNN is a `push` function, that "stacks" an input NN into an SNN. The stacking consists in building tensors with one extra dimension for every weight matrix of the NN Q-function. For instance, let A_k be a 256×256 weight matrix of a hidden layer of (the neural approximation of) Q_τ^k. Let $\mathbf{A}_k$ be a $k \times 256 \times 256$ tensor of the SNN that stores all past A_i weight matrices for $i \leq k$. In the forward pass of the SNN, to compute past Q-functions Q_τ^i for $i \leq k$, we use the batch matrix operation of PyTorch (Paszke et al., 2017) on $\mathbf{A}_k$ to make full use of the GPU parallel computing power. Finally, if the SNN contains more than M NNs after a `push` operation, the oldest NN is deleted from the SNN in a "first in first out" fashion. This `push` operation of SNNs currently supports multi-layer Perceptron NN architectures only. We leave the extension to other neural architectures such as CNNs (LeCun et al., 2015) or LSTMs (Hochreiter & Schmidhuber, 1997) to future work.

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_τ^k. 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. Beyond policy evaluation, computing ξ_k is also used in data collection by the behavior policy π_k^b that typically relies on computing π_k (see Sec. 6). Conversely, the policy update consists only of the above `push` operation, and thus, is optimization free and (almost) instantaneous. Table 1 shows the

[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.

	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

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.

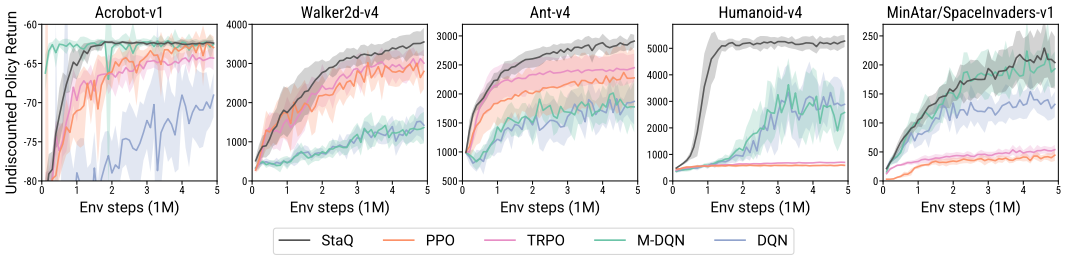


Figure 2: Policy return of StaQ, compared with deep RL baselines. Plots show mean and one standard deviation computed over 10 seeds. StaQ matches or exceeds state-of-the-art on several tasks and has generally lower variance. Even on SpaceInvaders, where the variance appears to be of the same order than the deep RL baselines, the oscillations within runs are generally reduced as show in Fig. 3. See App. B.1 for all results.

training time of StaQ over 5 million steps as a function of M for two environments. Varying M (up to 500) or the size of the state space has little impact on the runtime of StaQ when running on GPU.

6 EXPERIMENTS

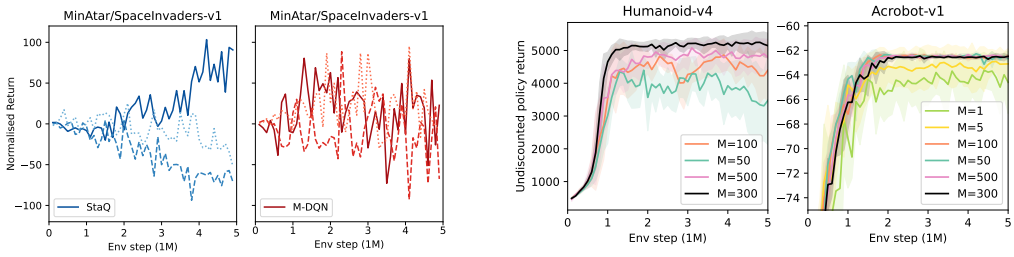
In this section, we assess the empirical merits of StaQ. In particular, we investigate whether the averaging over the large number of Q-functions (several hundreds) done by the SNN architecture improves the stability of the learning process, in terms of reducing performance oscillations across iterations of the same seed. Naturally, we also investigate the hyper-parameter M to determine values for which the above stability might manifest. Finally, we discuss some of the limitations of our algorithm 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 d 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, leading to 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., 2020), the policy gradient algorithm TRPO (Schulman et al., 2015) as it uses a D_{KL} regularizer and PPO (Schulman et al., 2017)[2]. SAC (Haarnoja et al., 2018) is another popular deep RL baseline that uses entropy regularization but is not adapted for discrete action environments, as discussed in App. D. Comparisons with baselines are averaged over 10 seeds and show the mean and standard deviation of the return. The return is computed by evaluating every 100K steps the current deterministic policy by averaging 50 rollouts. Hyperparameters for StaQ and the baselines are provided in App. E.

StaQ's components. We consider the behavior policy π_k^b that is an ϵ-*softmax* policy over π_k — by analogy with ϵ-greedy policies, which mixes the softmax policy π_k and a uniform policy with probabilities $(1 - \epsilon)$ and ϵ respectively. We use the ϵ-softmax policy instead of π_k as we found

[2]For TRPO and PPO, we use the implementation provided in `stable-baselines3` (Raffin et al., 2021), while we implemented our own PyTorch version of (M)-DQN.



(a) Return of individual runs, centered at each step.

(b) Impact of memory size M.

Figure 3: **Left:** Policy returns of individual runs on an environment where overall variance is similar between StaQ and baselines. Returns are centered at every timestep by subtracting the mean across 10 seeds. Individual runs of StaQ show a clearer trend (2 runs plateau and one continues increasing), while M-DQN, the closest performing baseline oscillates more around zero. For clarity, we only plot the first three seeds. 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 $M = 100$ seems insufficient. Plots showing mean and one standard deviation computed over 5 seeds.

that using only π_k can cause catastrophic forgetting in the Q-function, as discussed in Sec. 6.3. For learning Q^k_τ, we used a similar approach to SAC (Haarnoja et al., 2018), sampling an action from the current policy and use an ensemble of two target Q-functions updated in a hard manner. At the start, the NN for Q^0_τ is initialized with the zero output, so that π_0 is a uniform policy, 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). When computing the Q-values in the next states, we aggregate the target values by taking either the `min` or the `mean` of the two Q-functions. We discuss the impact of this choice in App. B.5. For the full set of hyperparameters consult App. E. 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 over the first 1 million timesteps.

6.1 Stability and performance of StaQ

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. In general, the performance matches or exceeds previous state-of-the-art algorithms, with lower variance and less performance oscillation. Even in environments (e.g. MinAtar/SpaceInvaders-v1) where the variance across seeds is comparable to other algorithms, the variability within individual runs of StaQ is typically much lower. An example of this is shown in Fig. 3 (**Left**) where we plot the return for each seed, centered by subtracting the mean return across all seeds at each evaluation. 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.

6.2 The impact of the memory-size M

According to Sec. 4.2, M is a crucial parameter that should be large enough to guarantee the 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 Acrobot and Humanoid. Despite that some low $M \leq 100$ ($M \leq 10$ for Acrobot) can still give us decent average performance, stability is negatively affected, which is especially pronounced for more challenging environment such as Humanoid. Conversely, higher $M = 500$, while being more expensive to compute, does not lead to any improvement either in terms of performance or stability. Thus, $M = 300$ is the best compromise between stability and compute time. More environments are shown in App B.2.

6.3 Catastrophic forgetting in the Q-function

As illustrated in Fig. 4, when the behavior policy π_k^b is the current softmax policy π_k, there are more performance drops than when using an ϵ-softmax policy introduced earlier. To understand this behavior, we have recorded the Q-values across 100 states for two variants of StaQ that only differ in whether or not the behavior policy adds a probability $\epsilon = 0.05$ of picking actions uniformly at random. The results are shown in App. B.4. On several states, the Q-values show larger variability when $\epsilon = 0.0$ than when $\epsilon = 0.05$, which itself creates instability in the policy as discussed in the appendix. While $\epsilon = 0.05$ improves stability, the plot in the appendix still shows occurrences of large and sudden changes in the Q-values, which can still negatively impact performance. Moreover, 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). In this paper, we have focused on the instability of the policy; we believe that proper continual learning treatment of the Q-function, as discussed in Sec. 7, could potentially fix all remaining instabilities in deep RL.

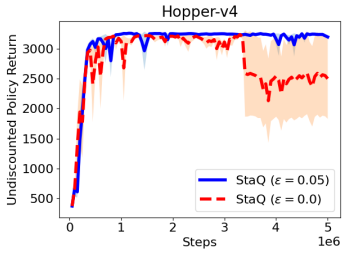


Figure 4: Policy return of StaQ for two different ϵ-softmax behavior policies, with $\epsilon = 0.05$ or $\epsilon = 0$. Plot shows mean and one standard deviation over 5 seeds.

6.4 Entropy regularization does not solve exploration in RL

StaQ is generally competitive with the state-of-the-art, but it might fail on some environments such as MountainCar or a few MinAtar environments. These environments are considered as hard exploration ones by Ceron & Castro (2021), and the combination of a deterministic policy with ϵ-greedy exploration seems to fare better in some tasks than the softmax policies of StaQ, PPO and TRPO, as also discussed by Vieillard et al. (2020), App. B.2. This is especially evident in MountainCar, where the initial maximum-entropy policy that picks an action uniformly at random always observes the same reward and remains close to the uniform policy throughout the learning process. To address this challenge, we have experimented with ϵ-greedy behavior policies in MountainCar as discussed in App. B.5, which improves performance. Although we used a fixed entropy weight for all experiments, we found that on the harder MinAtar environments, tuning the values of the entropy weight can improve performance, as presented in App. B.5. Overall, we believe that exploration is an orthogonal problem to the stability of the policy update we address in this paper, and while entropy regularization is often used as an exploration heuristic, we only use its stabilizing properties here, and leave the development of theoretically sound exploration strategies to future work.

7 Discussion and future work

In this paper, we have proposed to stabilize the policy update of approximate policy iteration by averaging over a very large number M of past Q-functions. Surprisingly, even when M is large, the final computational burden is small on modern hardware mostly thanks to the proposed SNN architecture. The resulting policy update has a solid theoretical foundation and clear empirical benefits as it greatly reduces learning instability on several tasks. Yet, while the policy update is more stable, instability in learning the Q-function remains and would benefit from a more thorough continual learning treatment. To prevent the catastrophic forgetting of low probability actions, one promising idea would be to better manage the replay buffer to keep representative samples, as done in CL by Aljundi et al. (2019) for example. Another interesting line of research from the CL community is the dual architecture based on the complementary learning systems theory (e.g. Lee et al. (2016)). Adapted to RL, the Q-function would be split between a fast learning—but potentially unstable—component that adapts to the most recent samples and a slower learning component that is stabilized using, for example, growing neural architectures as in this work to avoid catastrophic forgetting. If the neural architecture of the Q-function is much larger than the one used in this paper, it might be relevant to compress periodically the policy during learning as done in CL (Schwarz et al., 2018). 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 $\exp(Q_\tau^k)$ used by SAC with $\exp(\xi_k)$ which stabilizes the target by averaging over a large number of past Q-functions.

REPRODUCIBILITY STATEMENT

We provide the code of StaQ with the examples of launching files in the supplementary material. In addition, we carefully report the hyperparameters used for all runs of StaQ and baseline algorithms, see App. E. The code for reproducing the baselines results is withheld for anonymity reasons, as it uses a publicly available in-house library.

REFERENCES

Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellert Weisz. POLITEX: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, 2019.

David Abel, Andre Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

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

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, 2019.

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

Mohammad Gheshlaghi Azar, Vicenç Gómez, and Hilbert J. Kappen. Dynamic policy programming. *Journal of Machine Learning Research*, 2012.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013.

Shicong Cen, Chen Cheng, Yuxin Chen, Yuting Wei, and Yuejie Chi. Fast global convergence of natural policy gradient methods with entropy regularization. *Operations Research*, 2022.

Johan Samir Obando Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, 2021.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision*, 2018.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

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

Riccardo Della Vecchia, Alena Shilova, Philippe Preux, and Riad Akrour. Entropy regularized reinforcement learning with cascading networks. *arXiv*, 2022.

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

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

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

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

S. Grossberg. *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*. Boston studies in the philosophy of science. D. Reidel, 1988.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17. JMLR.org, 2017.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. In *International Conference on Machine Learning (ICML)*, 2018.

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. In *International Conference on Learning Representations*, 2020.

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

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

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

Saurabh Kumar, Henrik Marklund, Ashish Rao, Yifan Zhu, Hong Jun Jeon, Yueyang Liu, and Benjamin Van Roy. Continual learning as computationally constrained reinforcement learning. *arXiv*, 2023.

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

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

Nevena Lazic, Dong Yin, Yasin Abbasi-Yadkori, and Csaba Szepesvari. Improved regret bound and experience replay in regularized policy iteration. In *International Conference on Machine Learning*, 2021.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

Kyungjae Lee, Sungyub Kim, Sungbin Lim, Sungjoon Choi, and Songhwai Oh. Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning. *arXive*, 2019.

Sang-Woo Lee, Chung-Yeon Lee, Dong-Hyun Kwak, Jiwon Kim, Jeonghee Kim, and Byoung-Tak Zhang. Dual-memory deep learning architectures for lifelong learning of everyday human behaviors. In *International Joint Conference on Artificial Intelligence*, 2016.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 2020.

Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, 2016.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of Learning and Motivation. Academic Press, 1989.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv*, 2017.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, 2016.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems Workshop*, 2017.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 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`.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 2019.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, 2016.

John Schulman, Sergey Levine, Michael Jordan, and Pieter Abbeel. Trust Region Policy Optimization. *International Conference on Machine Learning*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Tim Seyde, Igor Gilitschenski, Wilko Schwarting, Bartolomeo Stellato, Martin Riedmiller, Markus Wulfmeier, and Daniela Rus. Is bang-bang control all you need? solving continuous control with bernoulli policies. In *Advances in Neural Information Processing Systems*, 2021.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

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

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

Samuele Tosatto, Matteo Pirotta, Carlo d'Eramo, and Marcello Restelli. Boosted fitted q-iteration. In *International Conference on Machine Learning*, pp. 3434–3443. PMLR, 2017.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL `https://zenodo.org/record/8127025`.

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

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Raj Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 2022.

Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, 2018.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

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

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017.

Wenhao Zhan, Shicong Cen, Baihe Huang, Yuxin Chen, Jason D. Lee, and Yuejie Chi. Policy mirror descent for regularized reinforcement learning: A generalized framework with linear convergence. *SIAM Journal on Optimization*, 2023.

Brian D. Ziebart. Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy. 7 2018.

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

$$(T_\tau^\pi f)(s,a) = R(s,a) + \gamma \mathbb{E}_{s',a'}[f(s',a') - \tau h(\pi(s'))], \tag{16}$$

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

$$(T_\tau^\star f)(s,a) = R(s,a) + \gamma \mathbb{E}_{s'} \left[\max_{p \in \Delta(A)} f(s') \cdot p - \tau h(p) \right]. \tag{17}$$

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. 8 satisfies the following property

$$\text{for all } s \in S, \quad \pi_k(s) = \underset{p \in \Delta(A)}{\arg\max}\, \xi_k(s) \cdot p - h(p), \tag{18}$$

A.2 Proof of Theorem 4.1

We prove here the approximate policy improvement given in Thm. 4.1.

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

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

From Sec. 3, we have that $\pi_{k+1} \propto \exp(\xi_{k+1})$ with

$$\xi_{k+1} = \beta \tilde{\xi}_k + \alpha Q_\tau^k. \tag{20}$$

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

$$\begin{aligned} Q_\tau^k(s) \cdot \pi_k(s) - \tau h(\pi_k(s)) &\leq Q_\tau^k(s) \cdot \pi_{k+1}(s) - \tau h(\pi_{k+1}(s)) \\ &\quad - \eta D_{\mathrm{KL}}(\pi_{k+1}(s); \tilde{\pi}_k(s)) + \eta D_{\mathrm{KL}}(\pi_k(s); \tilde{\pi}_k(s)), \end{aligned} \tag{21}$$

$$\leq Q_\tau^k(s) \cdot \pi_{k+1}(s) - \tau h(\pi_{k+1}(s)) + \eta D_{\mathrm{KL}}(\pi_k(s); \tilde{\pi}_k(s)), \tag{22}$$

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 $a \in A$.

$$D_{\mathrm{KL}}(\pi_k; \tilde{\pi}_k) = \mathbb{E}_\pi[\log \pi(a) - \log \tilde{\pi}(a)], \tag{23}$$

$$= \mathbb{E}_\pi \left[\log \frac{\exp \xi(a)}{Z} - \log \frac{\exp \tilde{\xi}(a)}{\tilde{Z}} \right], \tag{24}$$

$$= \mathbb{E}_\pi \left[\xi(a) - \tilde{\xi}(a) - \log \frac{Z}{\tilde{Z}} \right], \tag{25}$$

$$= \mathbb{E}_\pi \left[X(a) - \log \frac{\sum_{a'} \exp(X(a')) \exp(\tilde{\xi}(a'))}{\sum_{a'} \exp(\tilde{\xi}(a'))} \right], \tag{26}$$

$$\overset{\text{(i)}}{\leq} \mathbb{E}_\pi \left[X(a) - \mathbb{E}_{\tilde{\pi}} \left[X(a') \right] \right], \tag{27}$$

$$= (\pi - \tilde{\pi}) \cdot X, \tag{28}$$

where (i) is due to Jensen's inequality. Replacing Eq. 28 into Eq. 22 yields

$$Q_\tau^k(s) \cdot \pi_k(s) - \tau h(\pi_k(s)) \leq Q_\tau^k(s) \cdot \pi_{k+1}(s) - \tau h(\pi_{k+1}(s)) + \eta(\pi - \tilde{\pi})(s) \cdot X(s). \tag{29}$$

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

$$\eta(\pi - \tilde{\pi})(s) \cdot X(s) \leq \eta \max_{s \in S} |(\pi - \tilde{\pi})(s) \cdot X(s)|, \tag{30}$$

$$\overset{(i)}{\leq} \eta \max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1 \|X(s)\|_\infty, \tag{31}$$

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

$$:= \epsilon, \tag{33}$$

where we applied Hölder's inequality in (i). Combining Eq. 33 with Eq. 29 in the Bellman equation in Eq. 3 yields for any $s \in S$ and $a \in A$

$$\begin{aligned} R(s,a) + \gamma \mathbb{E}_{s'}\left[Q_\tau^k(s') \cdot \pi_k - \tau h(\pi_k(s'))\right] \leq R(s,a) + \gamma \mathbb{E}_{s'}[Q_\tau^k(s') \cdot \pi_{k+1}(s') \\ - \tau h(\pi_{k+1}(s')) + \epsilon], \end{aligned} \tag{34}$$

$$\implies (T_\tau^k Q_\tau^k)(s,a) \leq (T_\tau^{k+1} Q_\tau^k)(s,a) + \gamma\epsilon. \tag{35}$$

Since Eq. 35 is valid for any s and a, and by the fixed point property of Prop. A.2 we have

$$T_\tau^k Q_\tau^k \leq T_\tau^{k+1} Q_\tau^k + \gamma\epsilon, \tag{36}$$

$$\implies Q_\tau^k \leq T_\tau^{k+1} Q_\tau^k + \gamma\epsilon. \tag{37}$$

The addition of $\gamma\epsilon$ in the above expression is performed element-wise for all states and actions. Using the monotonicity property of Prop. A.3 on Eq. 37, we have

$$T_\tau^{k+1} Q_\tau^k \leq T_\tau^{k+1}(T_\tau^{k+1} Q_\tau^k + \gamma\epsilon), \tag{38}$$

$$\leq (T_\tau^{k+1})^2 Q_\tau^k + \gamma^2\epsilon, \tag{39}$$

$$\implies Q_\tau^k \leq (T_\tau^{k+1})^2 Q_\tau^k + \gamma\epsilon + \gamma^2\epsilon. \tag{40}$$

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

$$Q_\tau^k \leq (T_\tau^{k+1})^n Q_\tau^k + \sum_{i=1}^{n} \gamma^i \epsilon. \tag{41}$$

Taking the limit of Eq. 41 for $n \to \infty$ yields by the uniqueness of the fixed point of T_τ^{k+1}

$$Q_\tau^k \leq Q_\tau^{k+1} + \frac{\gamma\epsilon}{1-\gamma}, \tag{42}$$

$$= Q_\tau^{k+1} + \gamma\eta \frac{\max_{s \in S} \|(\pi - \tilde{\pi})(s)\|_1 \left\|\xi_k - \tilde{\xi}_k\right\|_\infty}{1-\gamma}. \tag{43}$$

□

A.3 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

$$\xi_k - \tilde{\xi}_k = \alpha\beta^{M-1} Q_\tau^{k-M}. \tag{44}$$

Let us first start with the term

$$\eta \left\|\xi_k - \tilde{\xi}_k\right\|_\infty \overset{(i)}{=} \beta^M \left\|Q_\tau^{k-M}\right\|_\infty \tag{45}$$

$$\leq \beta^M \bar{R}. \tag{46}$$

In (i) we used the fact that $\eta\alpha = \beta$, whereas the second inequality comes from the bounded nature of Q_τ^π for any π, where $\bar{R}$ is defined in Eq. 5.

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

$$\nabla h(p) = \nabla(p \cdot \log p), \tag{47}$$
$$= \log p + 1. \tag{48}$$

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

$$\|\pi_k - \tilde{\pi}_k\|_1^2 \leq (\pi_k - \tilde{\pi}_k) \cdot (\nabla h(\pi_k) - \nabla h(\tilde{\pi}_k)), \tag{49}$$
$$= (\pi_k - \tilde{\pi}_k) \cdot (\log \pi_k - \log \tilde{\pi}_k), \tag{50}$$
$$\overset{(i)}{=} (\pi_k - \tilde{\pi}_k) \cdot (\xi_k - \tilde{\xi}_k), \tag{51}$$
$$\leq \|\pi_k - \tilde{\pi}_k\|_1 \left\|\xi_k - \tilde{\xi}_k\right\|_\infty, \tag{52}$$
$$= \|\pi_k - \tilde{\pi}_k\|_1 \alpha\beta^{M-1} \left\|Q_\tau^{k-M}\right\|_\infty, \tag{53}$$
$$\implies \|\pi_k - \tilde{\pi}_k\|_1 \leq \alpha\beta^{M-1} \left\|Q_\tau^{k-M}\right\|_\infty, \tag{54}$$
$$\leq \alpha\beta^{M-1}\bar{R}. \tag{55}$$

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

$$\|\pi_k - \tilde{\pi}_k\|_1 \leq \min\left\{2, \alpha\beta^{M-1}\bar{R}\right\}. \tag{56}$$

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

A.4 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.

Lemma A.1. *After $k \geq 0$ iterations of Eq. 9, we have* $\left\|Q_\tau^\star - Q_\tau^{k+1}\right\|_\infty \leq \gamma \|Q_\tau^\star - \tau\xi_{k+1}\|_\infty + \gamma\beta^{k+1}\left\|Q_\tau^0\right\|_\infty + \gamma\beta^M\bar{R}\left[\frac{\gamma \min\left\{2, \alpha\beta^{M-1}\bar{R}\right\}}{(1-\gamma)(1-\beta)} + 1\right]$.

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

$$(Q_\tau^\star - Q_\tau^{k+1})(s,a) = (T_\tau^\star Q_\tau^\star)(s,a) - \Big(R(s,a) + \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau h(\pi_{k+1}(s'))]\Big) \tag{57}$$
$$\begin{aligned} &= (T_\tau^\star Q_\tau^\star)(s,a) - \Big(R(s,a) + \gamma\mathbb{E}_{s',a'}[\tau\xi_{k+1}(s',a') - \tau h(\pi_{k+1}(s'))] \\ &\quad + \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau\xi_{k+1}(s',a')]\Big) \end{aligned} \tag{58}$$

Looking at the first inner term, using the entropy maximizing nature of π_{k+1} as defined in Eq. 18, and using the definition of the Bellman optimality operator $T_\tau^\star$ gives

$$\begin{aligned} R(s,a) + \gamma\mathbb{E}_{s'}[\tau\xi_{k+1}(s') \cdot \pi_{k+1}(s') - \tau h(\pi_{k+1}(s'))] &= R(s,a) \\ &\quad + \gamma\mathbb{E}_{s'}[\max_{p\in\Delta(A)} \tau\xi_{k+1}(s') \cdot p - \tau h(p)] \end{aligned} \tag{59}$$
$$= (T_\tau^\star \tau\xi_{k+1})(s,a) \tag{60}$$

For the second inner term, using the recursive definition of ξ_{k+1} in Eq. 9, the fact that $\tau\alpha = 1-\beta$ and the definition of ϵ in Eq. 33, we have for all $s \in S$ and $a \in A$

$$\begin{aligned} Q_\tau^{k+1}(s,a) - \tau\xi_{k+1}(s,a) &= Q_\tau^{k+1}(s,a) - [\beta\tau\xi_k(s,a) + (1-\beta)Q_\tau^k(s,a) \\ &\quad - (1-\beta)\beta^M Q_\tau^{k-M}(s,a)], \end{aligned} \tag{61}$$

$$\begin{aligned} &= \beta(Q_\tau^k(s,a) - \tau\xi_k(s,a)) + Q_\tau^{k+1}(s,a) \\ &\quad - Q_\tau^k(s,a) + (1-\beta)\beta^M Q_\tau^{k-M}(s,a), \end{aligned} \tag{62}$$

$$\geq \beta(Q_\tau^k(s,a) - \tau\xi_k(s,a)) - \frac{\gamma\epsilon}{1-\gamma} - (1-\beta)\beta^M\bar{R}. \tag{63}$$

Here, the last inequality is due to Eq. 42. Letting

$$\epsilon' := \frac{\gamma\epsilon}{1-\gamma} + (1-\beta)\beta^M\bar{R}, \tag{64}$$

one can easily show by induction that

$$Q_\tau^{k+1}(s,a) - \tau\xi_{k+1}(s,a) \geq \beta^{k+1}Q_\tau^0(s,a) - \sum_{i=0}^{k}\beta^i\epsilon', \tag{65}$$

$$\geq \beta^{k+1}Q_\tau^0(s,a) - \frac{\epsilon'}{1-\beta}. \tag{66}$$

The first inequality uses the fact that $\xi_0 = 0$. Using successively Eq. 60 and Eq. 66 back into Eq. 58 yields

$$(Q_\tau^\star - Q_\tau^{k+1})(s,a) = (T_\tau^\star Q_\tau^\star)(s,a) - (T_\tau^\star\tau\xi_{k+1})(s,a) - \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau\xi_{k+1}(s',a')], \tag{67}$$

$$\leq (T_\tau^\star Q_\tau^\star)(s,a) - (T_\tau^\star\tau\xi_{k+1})(s,a) - \gamma\mathbb{E}_{s',a'}[\beta^{k+1}Q_\tau^0(s',a')] + \frac{\gamma\epsilon'}{1-\beta}. \tag{68}$$

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

$$\left\|Q_\tau^\star - Q_\tau^{k+1}\right\|_\infty \leq \left\|T_\tau^\star Q_\tau^\star - T_\tau^\star\tau\xi_{k+1}\right\|_\infty + \gamma\beta^{k+1}\left\|Q_\tau^0\right\|_\infty + \frac{\gamma\epsilon'}{1-\beta}, \tag{69}$$

$$\leq \gamma\left\|Q_\tau^\star - \tau\xi_{k+1}\right\|_\infty + \gamma\beta^{k+1}\left\|Q_\tau^0\right\|_\infty + \frac{\gamma\epsilon'}{1-\beta}. \tag{70}$$

Replacing ϵ' in Eq. 64 and ϵ by its value from Cor. 4.1.1 gives us the constant

$$\frac{\gamma\epsilon'}{(1-\beta)} = \frac{\gamma^2\epsilon}{(1-\beta)(1-\gamma)} + \gamma\beta^M\bar{R}, \tag{71}$$

$$= \gamma\beta^M\bar{R}\left[\frac{\gamma\min\left\{2, \alpha\beta^{M-1}\bar{R}\right\}}{(1-\beta)(1-\gamma)} + 1\right]. \tag{72}$$

□

With the help of the above Lemma, we are ready to prove Thm. 4.2.

Proof. From the definition of ξ_{k+1} in Eq. 9 and the triangle inequality we get

$$\left\|Q_\tau^\star - \tau\xi_{k+1}\right\|_\infty = \left\|Q_\tau^\star - \beta\tau\xi_k - (1-\beta)Q_\tau^k + (1-\beta)\beta^M Q_\tau^{k-M}\right\|_\infty, \tag{73}$$

$$\leq \beta\left\|Q_\tau^\star - \tau\xi_k\right\|_\infty + (1-\beta)\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty + (1-\beta)\beta^M\left\|Q_\tau^{k-M}\right\|_\infty, \tag{74}$$

$$\begin{aligned} &\leq \beta\left\|Q_\tau^\star - \tau\xi_k\right\|_\infty + (1-\beta)\left(\gamma\left\|Q_\tau^\star - \tau\xi_k\right\|_\infty + \gamma\beta^k\left\|Q_\tau^0\right\|_\infty + \frac{\gamma\epsilon'}{1-\beta}\right) \\ &\quad + (1-\beta)\beta^M\left\|Q_\tau^{k-M}\right\|_\infty, \end{aligned} \tag{75}$$

$$\leq (\beta + \gamma(1-\beta))\left\|Q_\tau^\star - \tau\xi_k\right\|_\infty + (1-\beta)\left(\gamma\beta^k\left\|Q_\tau^0\right\|_\infty + \beta^M\bar{R}\right) + \gamma\epsilon'. \tag{76}$$

Letting

$$d := \beta + \gamma(1-\beta), \tag{77}$$

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

$$\begin{aligned}
\|Q_\tau^\star - \tau\xi_{k+1}\|_\infty &\leq d^{k+1}\|Q_\tau^\star\|_\infty + \sum_{i=0}^{k} d^{k-i}\left[(1-\beta)\left(\gamma\beta^i\left\|Q_\tau^0\right\|_\infty + \beta^M\bar{R}\right) + \gamma\epsilon'\right], && (78)\\
&\leq d^{k+1}\left(\|Q_\tau^\star\|_\infty + \left\|Q_\tau^0\right\|_\infty\right) + \frac{(1-\beta)\beta^M\bar{R} + \gamma\epsilon'}{1-d}, && (79)\\
&= d^{k+1}\left(\|Q_\tau^\star\|_\infty + \left\|Q_\tau^0\right\|_\infty\right) + \frac{\beta^M\bar{R}}{1-\gamma} + \frac{\gamma\epsilon'}{(1-\gamma)(1-\beta)}. && (80)
\end{aligned}$$

For Eq. 79, we used the fact that $\frac{\beta}{d} < 1$, since d interpolates between β and 1, and the following upper bound for the sum

$$\begin{aligned}
\sum_{i=0}^{k}\beta^i d^{k-i} &= d^k\sum_{i=0}^{k}\left(\frac{\beta}{d}\right)^i, && (81)\\
&= \frac{d^k - \beta^{k+1}d^{-1}}{1-\frac{\beta}{d}}, && (82)\\
&\leq \frac{d^k}{1-\frac{\beta}{d}}, && (83)\\
&= \frac{d^{k+1}}{\gamma(1-\beta)}. && (84)
\end{aligned}$$

Using Eq. 80 in Eq. 70 finally gives

$$\left\|Q_\tau^\star - Q_\tau^{k+1}\right\|_\infty \leq \gamma d^{k+1}\left(\|Q_\tau^\star\|_\infty + \left\|Q_\tau^0\right\|_\infty\right) + \gamma\beta^{k+1}\left\|Q_\tau^0\right\|_\infty + \frac{\gamma\beta^M\bar{R}}{1-\gamma} + \frac{\gamma\epsilon'}{(1-\gamma)(1-\beta)} \tag{85}$$

For the constants we have

$$\begin{aligned}
\frac{\gamma\beta^M\bar{R}}{1-\gamma} + \frac{\gamma\epsilon'}{(1-\gamma)(1-\beta)} &= \frac{\gamma\beta^M\bar{R}}{1-\gamma} + \frac{\gamma\beta^M\bar{R}}{1-\gamma}\left[\frac{\gamma\min\left\{2, \alpha\beta^{M-1}\bar{R}\right\}}{(1-\beta)(1-\gamma)} + 1\right], && (86)\\
&= \frac{\gamma\beta^M\bar{R}}{1-\gamma}\left[\frac{\gamma\min\left\{2, \alpha\beta^{M-1}\bar{R}\right\}}{(1-\beta)(1-\gamma)} + 2\right], && (87)
\end{aligned}$$

□

A.5 Poof of the logits expression in Sec. 4.2

Proof. For $k = 0$,

$$\begin{aligned}
\xi_1 &= \beta 0 + \alpha Q_\tau^0 + \frac{\alpha\beta^M}{1-\beta^M}(Q_\tau^0 - 0), && (88)\\
&= \alpha\left(1 + \frac{\beta^M}{1-\beta^M}\right)Q_\tau^0, && (89)\\
&= \frac{\alpha}{1-\beta^M}Q_\tau^0. && (90)
\end{aligned}$$

If it is true for k, then

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

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

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

□

A.6 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 $\tilde{\xi}_k - \xi_k = \frac{\alpha\beta^{M-1}}{1-\beta^M}(Q_\tau^k - Q_\tau^{k-M})$ in $\left\|\xi_k - \tilde{\xi}_k\right\|_\infty$ and using the fact that $\alpha\eta = \beta$. □

Note that, as in the previous section, 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 $\left\|Q_\tau^k - Q_\tau^{k-M}\right\|_\infty$. This bound becomes tighter as k goes to infinity for M large enough, since we show below that Q_τ^k converges to $Q_\tau^\star$ 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 $\left\|Q_\tau^k - Q_\tau^{k-M}\right\|_\infty^2$ 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.

A.7 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.2. *After $k \geq 0$ iterations of Eq. 13, we have $\left\|Q_\tau^\star - Q_\tau^{k+1}\right\|_\infty \leq \gamma \|Q_\tau^\star - \tau\xi_{k+1}\|_\infty + \gamma\beta^{k+1}\left\|Q_\tau^0\right\|_\infty + \gamma\sum_{i=0}^k \beta^i \epsilon'_{k-i}$, with $\epsilon'_k = \left(\frac{1+\gamma}{1-\gamma} - \beta\right)\frac{\beta^M}{1-\beta^M}\left\|Q_\tau^{k-M} - Q_\tau^k\right\|_\infty$.*

Proof. Define ϵ_k as

$$\epsilon_k := \frac{2\beta^M}{1-\beta^M}\left\|Q_\tau^{k-M} - Q_\tau^k\right\|_\infty, \tag{94}$$

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

$$(Q_\tau^\star - Q_\tau^{k+1})(s,a) = (T_\tau^\star Q_\tau^\star)(s,a) - \Big(R(s,a) + \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau h(\pi_{k+1}(s'))]\Big) \tag{95}$$

$$= (T_\tau^\star Q_\tau^\star)(s,a) - \Big(R(s,a) + \gamma\mathbb{E}_{s',a'}[\tau\xi_{k+1}(s',a') - \tau h(\pi_{k+1}(s'))] + \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau\xi_{k+1}(s',a')]\Big) \tag{96}$$

Looking at the first inner term and using the entropy maximizing nature of π_{k+1} as defined in Eq. 18 gives

$$R(s,a) + \gamma\mathbb{E}_{s'}[\tau\xi_{k+1}(s')\cdot\pi_{k+1}(s') - \tau h(\pi_{k+1}(s'))] = R(s,a) + \gamma\mathbb{E}_{s'}[\max_{p\in\Delta(A)} \tau\xi_{k+1}(s')\cdot p - \tau h(p)] \tag{97}$$

$$= (T_\tau^\star \tau\xi_{k+1})(s,a) \tag{98}$$

For the second inner term, using the recursive definition of ξ_{k+1} in Eq. 13 gives

$$Q_\tau^{k+1} - \tau\xi_{k+1} = Q_\tau^{k+1} - \left(\beta\tau\xi_k + (1-\beta)Q_\tau^k + \frac{(1-\beta)\beta^M}{1-\beta^M}(Q_\tau^k - Q_\tau^{k-M})\right), \tag{99}$$

$$= \beta(Q_\tau^k - \tau\xi_k) + Q_\tau^{k+1} - Q_\tau^k + \frac{(1-\beta)\beta^M}{1-\beta^M}(Q_\tau^k - Q_\tau^{k-M}), \tag{100}$$

$$\geq \beta(Q_\tau^k - \tau\xi_k) - \frac{\gamma\epsilon_k}{1-\gamma} - \frac{(1-\beta)\epsilon_k}{2}. \tag{101}$$

Letting

$$\epsilon'_k := \frac{\gamma\epsilon_k}{1-\gamma} + \frac{(1-\beta)\epsilon_k}{2}, \tag{102}$$

one can easily show by induction that

$$Q_\tau^{k+1} - \tau\xi_{k+1} \geq \beta^{k+1}Q_\tau^0 - \sum_{i=0}^{k}\beta^i\epsilon'_{k-i}. \tag{103}$$

The inequality uses the fact that $\xi_0 = 0$. Using successively Eq. 98 and Eq. 103 back into Eq. 96 yields

$$(Q_\tau^\star - Q_\tau^{k+1})(s,a) = (T_\tau^\star Q_\tau^\star)(s,a) - (T_\tau^\star\tau\xi_{k+1})(s,a) - \gamma\mathbb{E}_{s',a'}[Q_\tau^{k+1}(s',a') - \tau\xi_{k+1}(s',a')], \tag{104}$$

$$\leq (T_\tau^\star Q_\tau^\star)(s,a) - (T_\tau^\star\tau\xi_{k+1})(s,a) - \gamma\mathbb{E}_{s',a'}[\beta^{k+1}Q_\tau^0(s',a')] + \gamma\sum_{i=0}^{k}\beta^i\epsilon'_{k-i}. \tag{105}$$

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

$$\left\|Q_\tau^\star - Q_\tau^{k+1}\right\|_\infty \leq \|T_\tau^\star Q_\tau^\star - T_\tau^\star\tau\xi_{k+1}\|_\infty + \gamma\beta^{k+1}\left\|Q_\tau^0\right\|_\infty + \gamma\sum_{i=0}^{k}\beta^i\epsilon'_{k-i} \tag{106}$$

$$\leq \gamma\left\|Q_\tau^\star - \tau\xi_{k+1}\right\|_\infty + \gamma\beta^{k+1}\left\|Q_\tau^0\right\|_\infty + \gamma\sum_{i=0}^{k}\beta^i\epsilon'_{k-i} \tag{107}$$

Using the definition of ϵ in Eq. 94 and ϵ' in Eq. 102 gives for the constant

$$\epsilon'_k = \left(\frac{\gamma}{1-\gamma} + \frac{(1-\beta)}{2}\right)\epsilon_k, \tag{108}$$

$$= \left(\frac{1+\gamma}{1-\gamma} - \beta\right)\frac{\beta^M}{1-\beta^M}\left\|Q_\tau^{k-M} - Q_\tau^k\right\|_\infty \tag{109}$$

□

We can now resume to the proof of Lem. 4.3.

Proof. Define the following constants

$$c_1 := \frac{\beta^M}{1-\beta^M}, \text{ and } c_2 := \left(\frac{1+\gamma}{1-\gamma} - \beta\right)c_1. \tag{110}$$

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

$$x_0 = \|Q_\tau^\star\|_\infty + \left\|Q_\tau^0\right\|_\infty. \tag{111}$$

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

$$x_{k+1} = \gamma \left(\frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-i} + \beta^{k+1} \frac{\left\|Q_\tau^0\right\|_\infty}{\gamma} + c_2 \sum_{i=0}^{k} \beta^i (x_{k-i} + x_{k-i-M}) \right). \tag{112}$$

Note that x_0 can also be recovered by Eq. 112, for $k = -1$. Now, let us simplify Eq. 112. Using this recursive definition, we have $\forall k \geq 0$

$$x_{k+1} = \gamma \left(\frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-i} + \beta^{k+1} \frac{\left\|Q_\tau^0\right\|_\infty}{\gamma} + c_2 \sum_{i=0}^{k} \beta^i (x_{k-i} + x_{k-i-M}) \right), \tag{113}$$

$$= \gamma \Bigg(\beta \left(\frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-1-i} + \beta^k \frac{\left\|Q_\tau^0\right\|_\infty}{\gamma} + c_2 \sum_{i=0}^{k-1} \beta^i (x_{k-1-i} + x_{k-1-i-M}) \right) + \frac{1-\beta}{1-\beta^M} \left(x_k - \beta^M x_{k-M}\right) + c_2 \left(x_k + x_{k-M}\right) \Bigg), \tag{114}$$

$$\overset{(i)}{=} \beta x_k + \gamma \left(\frac{1-\beta}{1-\beta^M} \left(x_k - \beta^M x_{k-M}\right) + c_2 \left(x_k + x_{k-M}\right) \right), \tag{115}$$

$$= \left(\beta + \gamma \frac{1-\beta}{1-\beta^M} + \gamma c_2 \right) x_k + \gamma \left(c_2 - \frac{\beta^M (1-\beta)}{1-\beta^M} \right) x_{k-M} \tag{116}$$

$$= \left(\beta + \gamma \frac{1-\beta}{1-\beta^M} + \gamma c_2 \right) x_k + \frac{2c_1\gamma^2}{1-\gamma} x_{k-M} \tag{117}$$

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

$$d_1 := \beta + \gamma \frac{1-\beta}{1-\beta^M} + \gamma c_2, \text{ and } d_2 := \frac{2c_1\gamma^2}{1-\gamma}, \tag{118}$$

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

$$x_{k+1} = d_1 x_k + d_2 x_{k-M}. \tag{119}$$

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

$$\left\|Q_\tau^\star - Q_\tau^0\right\|_\infty \leq \left\|Q_\tau^\star\right\|_\infty + \left\|Q_\tau^0\right\|_\infty, \tag{120}$$

$$= x_0. \tag{121}$$

and for $k < 0$, we have that

$$\left\|Q_\tau^\star - Q_\tau^k\right\|_\infty = \left\|Q_\tau^\star\right\|_\infty, \tag{122}$$

$$\leq \frac{\left\|Q_\tau^\star\right\|_\infty}{\gamma}, \tag{123}$$

$$= x_k. \tag{124}$$

Now assume that $\left\|Q_\tau^\star - Q_\tau^i\right\|_\infty \leq x_i$ is true for all $i \leq k$ and let us prove that $\left\|Q_\tau^\star - Q_\tau^{k+1}\right\|_\infty \leq x_{k+1}$. First, we note that

$$\left\|Q_\tau^\star - \tau\xi_{k+1}\right\|_\infty = \left\| Q_\tau^\star - \tau \frac{\alpha}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i Q_\tau^{k-i} \right\|_\infty, \tag{125}$$

$$= \left\| \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i Q_\tau^\star - \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i Q_\tau^{k-i} \right\|_\infty, \tag{126}$$

$$\leq \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i \left\|Q_\tau^\star - Q_\tau^{k-i}\right\|_\infty, \tag{127}$$

$$\leq \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-i}. \tag{128}$$

We also have that

$$\epsilon'_k = c_2 \left\| Q_\tau^{k-M} - Q_\tau^k \right\|_\infty , \tag{129}$$
$$\leq c_2 \left(\left\| Q_\tau^\star - Q_\tau^{k-M} \right\|_\infty + \left\| Q_\tau^\star - Q_\tau^k \right\|_\infty \right) \tag{130}$$
$$\leq c_2 (x_k + x_{k-M}). \tag{131}$$

Finally, using Eq. 128, Eq. 131 and $\left\| Q_\tau^0 \right\|_\infty \leq \frac{\|Q_\tau^0\|_\infty}{\gamma}$ into Lemma A.2 completes the proof

$$\left\| Q_\tau^\star - Q_\tau^{k+1} \right\|_\infty \leq \gamma \left(\|Q_\tau^\star - \tau \xi_{k+1}\|_\infty + \beta^{k+1} \left\| Q_\tau^0 \right\|_\infty + \sum_{i=0}^{k} \beta^i \epsilon'_{k-i} \right), \tag{132}$$
$$\leq \gamma \left(\frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i x_{k-i} + \beta^{k+1} \frac{\left\| Q_\tau^0 \right\|_\infty}{\gamma} + c_2 \sum_{i=0}^{k} \beta^i (x_{k-i} + x_{k-i-M}) \right), \tag{133}$$
$$= x_{k+1}. \tag{134}$$

□

A.8 Proof of Theorem 4.4

Proof. Let us find a value of M such that

$$d_1 + d_2 < 1, \tag{135}$$
$$\Leftrightarrow \beta(1-\beta^M) + \gamma(1-\beta) + \gamma \left(\frac{1+\gamma}{1-\gamma} - \beta \right) \beta^M + \frac{2\gamma^2 \beta^M}{1-\gamma} < 1 - \beta^M, \tag{136}$$
$$\Leftrightarrow \beta - \beta^{M+1} - \gamma\beta + \gamma \frac{1+\gamma}{1-\gamma} \beta^M - \gamma\beta^{M+1} + \beta^M + \frac{2\gamma^2\beta^M}{1-\gamma} < 1 - \gamma, \tag{137}$$
$$\Leftrightarrow (1-\gamma)\beta + \frac{\gamma^2(3+\beta) + 1 - \beta}{1-\gamma} \beta^M < 1 - \gamma, \tag{138}$$
$$\Leftrightarrow \beta^M < \frac{(1-\gamma)^2(1-\beta)}{\gamma^2(3+\beta) + 1 - \beta}, \tag{139}$$
$$\Leftrightarrow M \log \beta < \log \frac{(1-\gamma)^2(1-\beta)}{\gamma^2(3+\beta) + 1 - \beta}, \tag{140}$$
$$\Leftrightarrow M > \log \frac{(1-\gamma)^2(1-\beta)}{\gamma^2(3+\beta) + 1 - \beta} (\log \beta)^{-1}. \tag{141}$$

We will now show that for the values of M that satisfy Eq. 141, the sequence x_k converges to 0 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 $x_{k+1} = d_1 x_k + d_2 x_{k-M}$ but replaces the terms x_{-k}, $\forall k \geq 0$ with $x_{-k} = \max\left\{ \frac{\|Q_\tau^\star\|_\infty}{\gamma}, \|Q_\tau^\star\|_\infty + \left\| Q_\tau^0 \right\|_\infty \right\}$. Clearly, this modified sequence upper-bounds the previous sequence and if this new sequence converges to zero then so does the sequence defined in Lemma 4.3.

Now, if M is such that $d_1 + d_2 < 1$, then the new sequence is constant from x_{-M} to x_0 and is strictly decreasing thereafter, since for x_1 we have

$$x_1 = d_1 x_0 + d_2 x_{-M}, \tag{142}$$
$$= (d_1 + d_2) x_0, \tag{143}$$
$$< x_0, \tag{144}$$

and if it is strictly decreasing from x_0 to x_k, $\forall k \geq 1$ then

$$x_{k+1} = d_1 x_k + d_2 x_{k-M}, \tag{145}$$
$$< d_1 x_{k-1} + d_2 x_{k-M-1}, \tag{146}$$
$$= x_k. \tag{147}$$

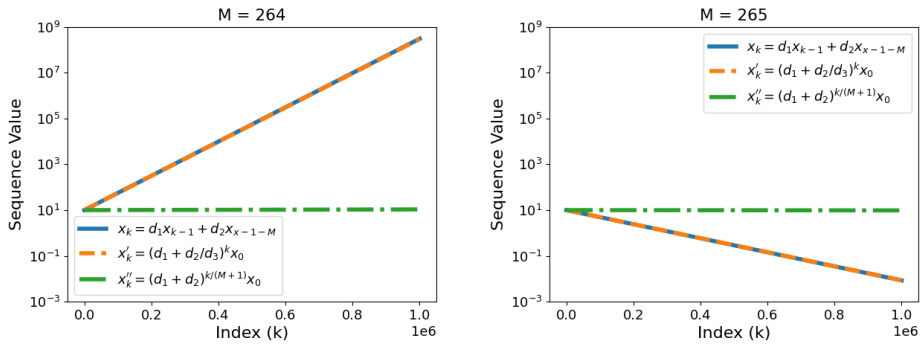


Figure 5: Evolution of x_k for two successive values of M, one being large enough for x_k to converge. The plot additionally shows the sequence x'_k introduced by Thm. 4.4 that closely follows the behavior of x_k. See text for more details.

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 $a > 0$

$$x_{aM+a} = d_1 x_{aM+a-1} + d_2 x_{aM+a-1-M}, \tag{148}$$
$$< (d_1 + d_2) x_{aM+a-1-M}, \tag{149}$$
$$= (d_1 + d_2) x_{(a-1)M+(a-1)}, \tag{150}$$
$$< (d_1 + d_2)^a x_0. \tag{151}$$

Thus, $\lim_{a\to\infty} x_{aM+a} = 0$, which concludes the first part of the proof.

To better characterize the convergence of x_k 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 x_k thus behaves almost as $d_1^k x_0$. A much tighter upper-bounding sequence than that of Eq. 151 can be obtained using the following inequalities

$$x_k = d_1 x_{k-1} + d_2 x_{k-1-M}, \tag{152}$$
$$= d_1^M x_{k-M} + d_2 \sum_{i=0}^{M-1} d_1^i x_{k-1-M-i}, \tag{153}$$
$$\geq \left(d_1^M + d_2 \frac{1 - d_1^M}{1 - d_1} \right) x_{k-M}, \tag{154}$$

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

$$d_3 := \left(d_1^M + d_2 \frac{1 - d_1^M}{1 - d_1} \right), \tag{155}$$

then we can upper bound the sequence x_k by

$$x_{k+1} = \left(d_1 + d_2 d_3^{-1}\right) x_k + d_2 (x_{k-M} - d_3^{-1} x_k), \tag{156}$$
$$\leq \left(d_1 + d_2 d_3^{-1}\right) x_k + d_2 (x_{k-M} - d_3^{-1} d_3 x_{k-M}), \tag{157}$$
$$= \left(d_1 + d_2 d_3^{-1}\right) x_k, \tag{158}$$
$$\leq \left(d_1 + d_2 d_3^{-1}\right)^{k+1} x_0. \tag{159}$$

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

$$d_2 d_3^{-1} \leq \frac{d_2}{d_1^M}, \tag{160}$$

$$\leq \frac{d_2}{(\beta+\gamma(1-\beta))^M}, \tag{161}$$

$$= \frac{1}{(\beta+\gamma(1-\beta))^M} \frac{2\beta^M\gamma^2}{(1-\gamma)(1-\beta^M)}, \tag{162}$$

$$= \left(\frac{\beta}{\beta+\gamma(1-\beta)}\right)^M \frac{2\gamma^2}{(1-\gamma)(1-\beta^M)}. \tag{163}$$

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 $\lim_{M\to\infty} d_1 = \beta + \gamma(1-\beta)$. □

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. 5 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 EXPERIMENT RESULTS

B.1 COMPARISON WITH DEEP RL BASELINES

We summarize all the comparisons in Fig. 6 and Table 2.

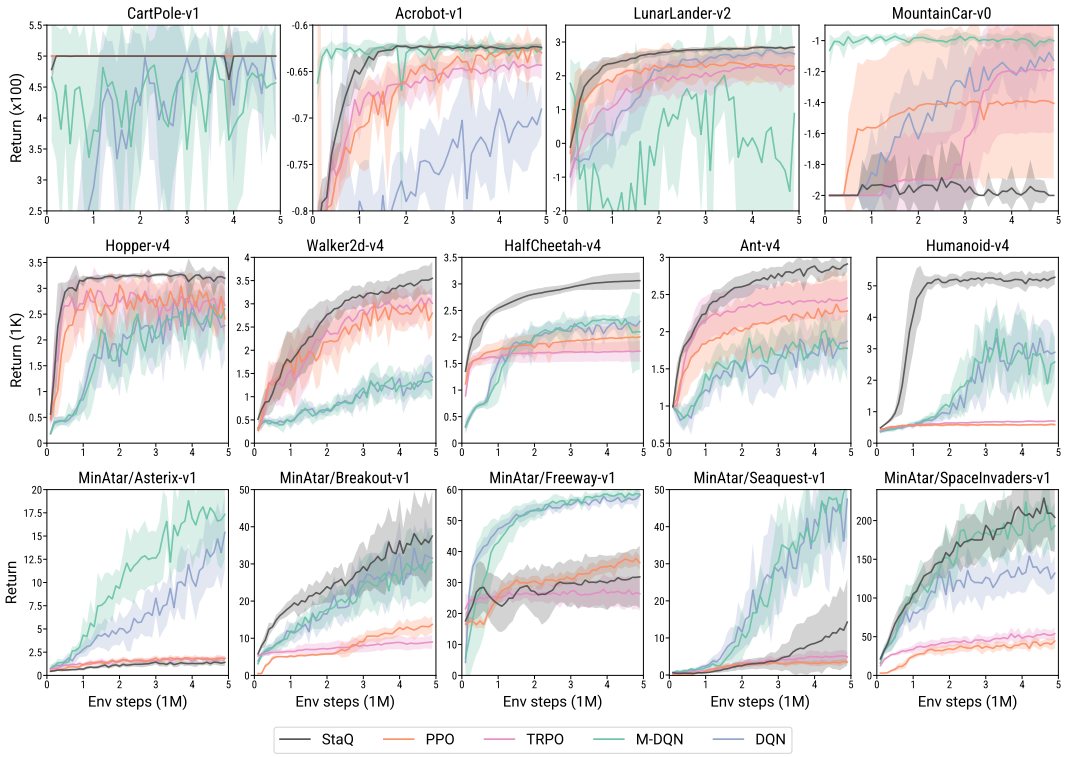


Figure 6: Policy performance across all environments.

	StaQ	PPO	TRPO	M-DQN	DQN
CartPole-v1	**500**	**500**	**500**	457	463
Acrobot-v1	**-62**	-63	-64	-63	-69
LunarLander-v2	**285**	227	222	88	265
MountainCar-v0	-200	-141	-118	**-100**	-113
Hopper-v4	**3196**	2411	2672	2600	2279
Walker2d-v4	**3550**	2799	3010	1364	1424
HalfCheetah-v4	**3061**	2001	1731	2098	2294
Ant-v4	**2910**	2277	2452	1776	1871
Humanoid-v4	**5273**	588	700	2580	2887
MinAtar/Asterix-v1	1	2	2	**17**	15
MinAtar/Breakout-v1	**37**	14	9	30	32
MinAtar/Freeway-v1	32	36	26	**59**	58
MinAtar/Seaquest-v1	14	3	5	**51**	47
MinAtar/SpaceInvaders-v1	**204**	44	54	193	132

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. 7-8 we plot the returns of the first three seeds of the full results (shown in Fig. 6). At each timestep, the returns for

each individual seed are normalised by subtracting and then dividing by the mean across all seeds. We only include environments where all algorithms learn a decent policy to enable a fair comparison. We can see from Fig. 7-8 that Approximate Policy Iteration (API) algorithms (StaQ, TRPO, PPO) generally exhibit less variation within runs than Approximate Value Iteration (AVI) ones (DQN, M-DQN). 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. 6. 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.

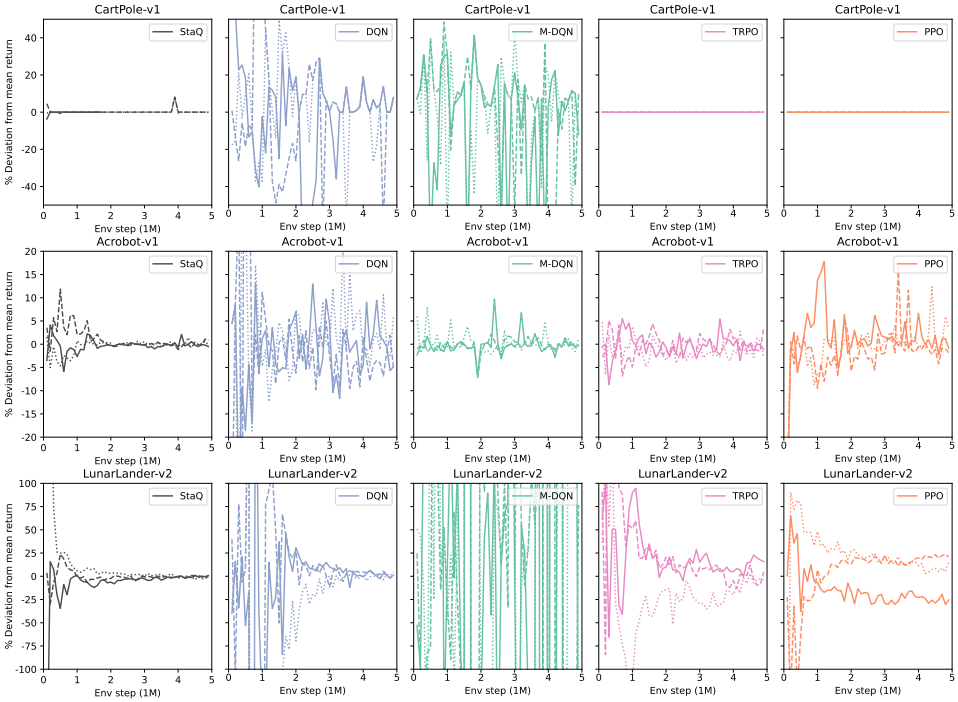


Figure 7: Stability plots showing normalized performance of the first three individual runs for each algorithm. See text for more details. Figures continue on the next page.

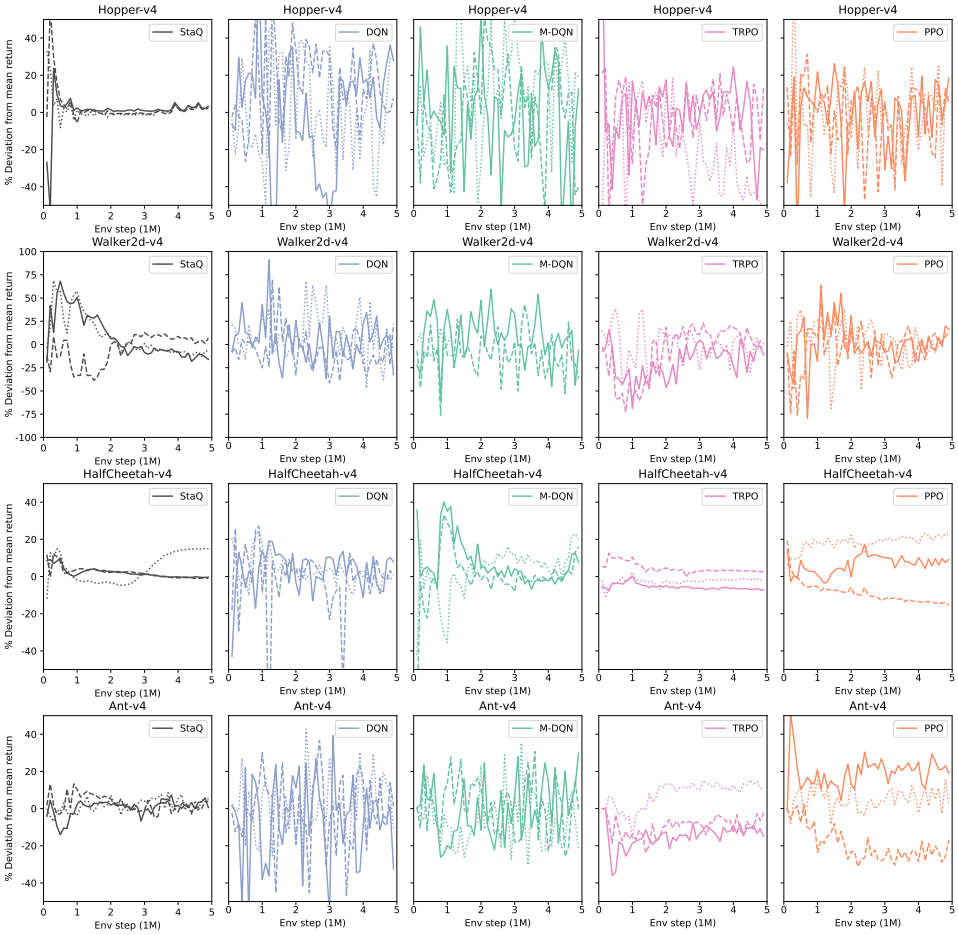


Figure 8: Stability plots showing normalized performance of the first three individual runs for each algorithm. See text for more details.

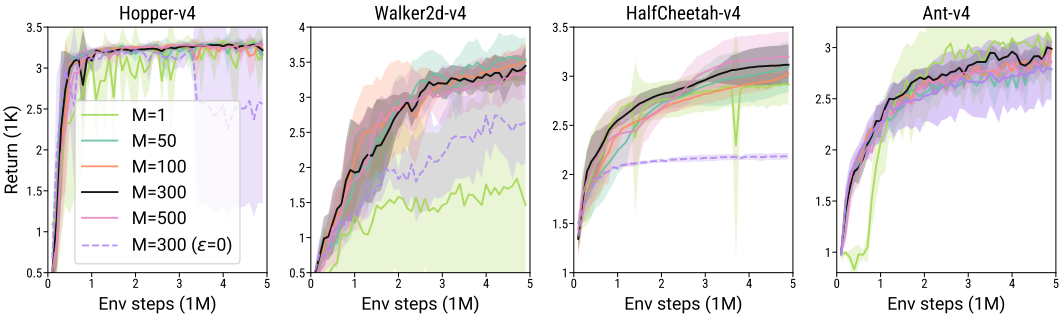
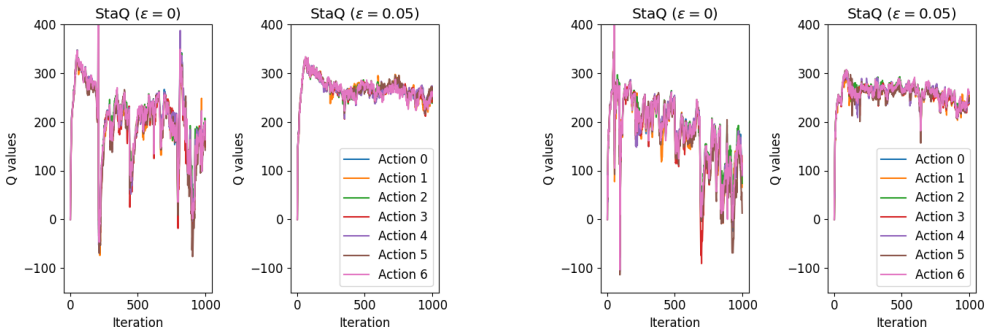


Figure 9: Ablation study for different M values or choice of ϵ. Results showing mean and one standard deviation averaged over 5 seeds.

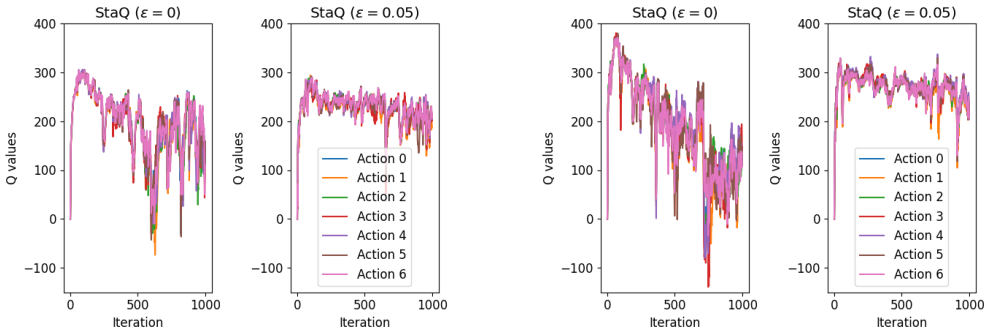
B.3 THE IMPACT OF THE MEMORY-SIZE M AND VALUE OF ϵ

Figure 9 shows the performance of StaQ for different choices of M and for the hyper-parameter $\epsilon = 0$ instead of $\epsilon = 0.05$ on additional MuJoCo tasks. $M = 1$, which corresponds to no KL-regularization as discussed in App. D and which can be seen as an adaptation of SAC to discrete action spaces, 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 all but the Ant environment. We discuss some of the likely reasons for the need of ϵ-softmax exploration in App. B.4.



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

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



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

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

Figure 10: Q-values at some fixed 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 $\epsilon = 0.05$.

B.4 Catastrophic forgetting in the Q-function

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. 4, 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 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. 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. 8) 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

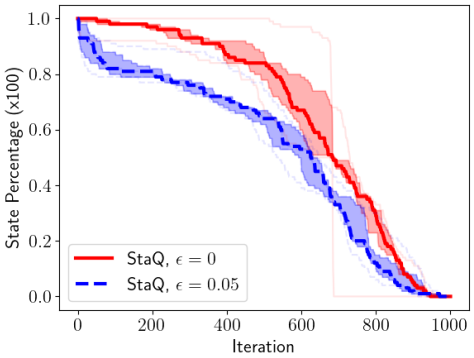


Figure 11: The percentage of states (out of 100 states) in which from iteration k and onward, an action was considered both the best and the worst according to ξ_k of EPMD. The difference of stability in the Q-values between $\epsilon = 0.05$ and $\epsilon = 0$ noted in Fig. 10 causes a difference in 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.

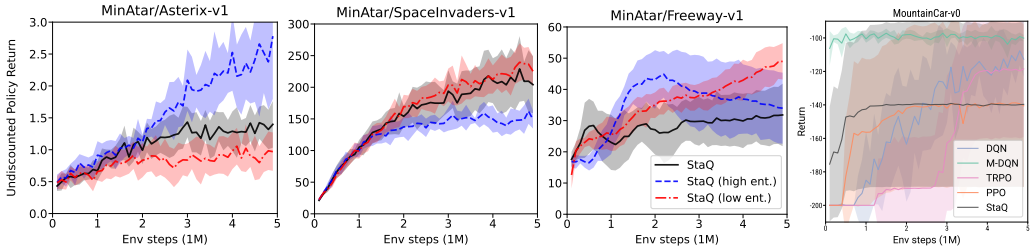


Figure 12: **Left three panels:** Effect of different entropy weights on the MinAtar environments. **Right panel:** Policy returns for StaQ (with additional tweaks to StaQ , see text for details) and deep RL baselines on MountainCar.

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[3] 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 as discussed in Sec. 6.3, and we can see in Fig. 10 that it still exhibits sudden changes in the Q-function which could harm stability. While the averaging over past Q-functions of an EPMD policy can stabilize learning to some extent, we believe that the catastrophic forgetting in the Q-function itself should be addressed to fully stabilize deep RL.

B.5 Entropy regularization does not solve exploration in RL

Entropy regularization is a common technique to improve exploration and robustness of the policy (see e.g. Ziebart (2018); Haarnoja et al. (2017)), however overly-strong regularization can harm learning. In our main experimental results we fix the initial and final (scaled) entropy weight $\bar{\tau}$ across all environments, however we find that certain environments benefit from adjusting the entropy weight. This is illustrated in the left three panels of Fig. 12, where we vary the entropy weights, linearly annealed from 5 to 1 (1 to 0.05) for "high ent" ("low ent") over the first 1 million timesteps. In addition to environments such as Asterix and SpaceInvaders that absolutely prefer high or low entropy, some environments such as Freeway have complex dependence on the entropy that

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

suggests that a constant or linearly-decaying entropy is insufficient. Future work could find best $\bar{\tau}$ automatically, for example as in SAC (Haarnoja et al., 2018).

For MountainCar, we find that having overly stable updates at the start of the training can prevent it from finding the sparse reward signal. StaQ can learn on such an environment (see right panel of Fig. 12), without tuning the entropy or D_{KL} weights, but with additional tweaks, such as increasing a scale of NN weights upon initialization (by $\times 20$), decreasing the policy update interval (from 5000 to 500 steps) and using an ϵ-greedy strategy on top of ξ_k as a behavior policy, rather than an ϵ-softmax policy used for the main experiments. The ϵ-greedy strategy is also used in DQN and M-DQN that outperform other baselines on MountainCar. Note that even though StaQ still performs worse than most other baselines, this is because some runs do not see a reward signal early enough in training and the inherent stability of StaQ prevents them from seeing it after the initial exploration period is over.

Policy evaluation and exploration. We find that using the `min` of the two Q-functions to compute the target values 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 struggles with sparse rewards, and hard exploration problems such as MountainCar-v0 and MinAtar/Seaquest-v1. 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.

C Training and Inference Time Comparisons

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

Table 3: 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.

Table 3 compares the training and inference time of StaQ with the baselines. All timing experiments were computed on an NVIDIA Tesla V100, averaged over 3 seeds. The timings reported for StaQ here are for a memory size $M = 300$, however as discussed in Sec 6 (see also Table 1), the memory size has a negligible impact on the training and inference times.

D COMPARISON WITH SOFT ACTOR-CRITIC

In this appendix, we explain why Soft Actor-Critic (SAC, Haarnoja et al. (2018)) was not used as a baseline and how SAC relates to M-DQN and StaQ with $M = 1$. SAC is not used as a baseline because StaQ is a discrete action algorithm evaluated on discrete action environments, while 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. (2020) on page 3, between Eq. (1) and (2). Vieillard et al. (2020) 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. (2020) 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. (14) 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. 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.

E HYPERPARAMETERS

Here, we provide the full list of hyperparameters used in our experiments. StaQ's hyperparameters are listed in Table 4, while the hyperparameters for our baselines are provided in Tables 5-7. In 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. Furthermore, to account for different scales of reward signals in different environments, we apply different reward scales between 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.

F 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 is then called to evaluation 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.

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***	**200**
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
Decay steps	**500K**	**1M**	**1M**
Architecture	256×2	256×2	256×2
Activation function	ReLu	ReLu	ReLu
Learning rate	0.0001	0.0001	0.0001
Replay capacity	50K	50K	50K
Batch size	256	256	256

Table 4: 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	$0.1 \times \alpha$
Architecture	64×2	64×2 *	256×2
Activation function	Tanh	Tanh	Tanh
Learning rate	3×10^{-4}	3×10^{-4}	$2.5 \times 10^{-4} \times \alpha$
Replay capacity	1M	1M	50K
Batch size	64	64	32

Table 5: 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.

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, 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
  Obtain logits ξ_{k+1} by stacking the last M Q-functions (see Sec. 5) following the finite-memory EPMD update of Eq. 13.
  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

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
Architecture	64×2	64×2 *	256×2
Activation function	Tanh	Tanh	Tanh
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}
Replay capacity	1M	1M	1M
Batch size	64	64	64

Table 6: 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
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	128×2
Activation function	ReLu	ReLu	ReLu
Learning rate	1×10^{-3}	5×10^{-5}	5×10^{-5}
Replay capacity	50K	1M	1M
Batch size	128	32	32

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