Value Improved Actor Critic Algorithms

Yaniv Oren
Delft University of Technology
2628 CD Delft, The Netherlands
y.oren@tudelft.nl

Moritz A. Zanger
Delft University of Technology
2628 CD Delft, The Netherlands
m.a.zanger@tudelft.nl

Pascal R. van der Vaart
Delft University of Technology
2628 CD Delft, The Netherlands
p.r.vandervaart-1@tudelft.nl

Matthijs T. J. Spaan
Delft University of Technology
2628 CD Delft, The Netherlands
m.t.j.spaan@tudelft.nl

Wendelin Böhmer
Delft University of Technology
2628 CD Delft, The Netherlands
j.w.bohmer@tudelft.nl

Abstract

Many modern reinforcement learning algorithms build on the actor-critic (AC) framework: iterative improvement of a policy (the actor) using *policy improvement operators* and iterative approximation of the policy's value (the critic). In contrast, the popular value-based algorithm family employs improvement operators in the value update, to iteratively improve the value function directly. In this work, we propose a general extension to the AC framework that employs two separate improvement operators: one applied to the policy in the spirit of policy-based algorithms and one applied to the value in the spirit of value-based algorithms, which we dub Value-Improved AC (VI-AC). We design two practical VI-AC algorithms based in the popular online off-policy AC algorithms TD3 and DDPG. We evaluate VI-TD3 and VI-DDPG in the Mujoco benchmark and find that both improve upon or match the performance of their respective baselines in all environments tested.

1 Introduction

Modern reinforcement learning (RL) methods can be roughly divided into two algorithmic families: value-based methods and policy-and-value-based methods. Most policy-and-value-based methods rely on the Actor Critic (AC) underlying scheme: iteratively improving a policy π (the actor) and iteratively estimating the value function of the improved policy (the critic). To improve the policy, AC methods rely on *policy improvement operators* (PIOs) $\mathcal{I}$, perhaps the most common of which is the *policy gradient* (Sutton et al., 1999). Value based methods, on the other hand, rely on PIOs such as the $\arg\max$ to iteratively improve the value function directly, rather than estimating the value of an improved policy. In this work we extend the AC scheme with an additional improvement step, applied directly to the value update in the spirit of value-based methods.

An abstracted point of view on the connection between value and policy based methods is captured in the Optimistic Policy Iteration (OPI) Dynamic Programming (DP) algorithm (Tsitsiklis, 2002). OPI maintains a value prediction and a policy. At each iteration OPI updates the policy with the $\arg\max$ operator and updates the value with k Bellman-updates. When $k = 1$, OPI coincides with Value Iteration, and when $k \to \infty$ OPI coincides with Policy Iteration (Sutton & Barto, 2018).

To facilitate theoretical analysis of an RL framework with two improvement operators, we begin this work by generalizing OPI to arbitrary improvement operators with possibly stochastic policies (Generalized OPI, GOPI). Using this framework, we describe a novel DP algorithm that employs two separate improvement steps – one to the policy and one to the value (Value-Improved Generalized OPI, VI-GOPI). We prove that both GOPI and VI-GOPI converge in the finite horizon MDP setting.

We believe that the convergence result of GOPI covers a missing link in current RL theory: Many modern AC(-like) methods such as Alpha/MuZero (Silver et al., 2018; Schrittwieser et al., 2020), MPO (Abdolmaleki et al., 2018) and GreedyAC (Neumann et al., 2023) do not directly rely on the policy gradient for improvement but on other PIOs, as well as specifically stochastic policies. One of the motivations for the choice to use different PIOs – other than practical benefits – is in the *Policy Improvement Theorem* (Sutton & Barto, 2018) which guarantees convergence for Policy Iteration with deterministic policies and PIOs. In practice, however, in addition to using stochastic policies, many modern AC algorithms rely on a small number of value updates k per policy update. These design choices do not not coincide with the assumptions of Policy Iteration, but do coincide with those of GOPI.

Building on VI-GOPI, this paper designs a class of practical off-policy online-RL algorithms which we call Value-Improved Actor-Critics (VI-ACs). We propose two variations of VI-AC, built upon two popular off-policy AC algorithms, TD3 (Fujimoto et al., 2018) and DDPG (Lillicrap et al., 2015). We evaluate the performance of VI-TD3 and VI-DDPG with different PIOs on the Mujoco benchmark. In our experiments, VI-TD3 and VI-DDPG outperform or match the baselines TD3 and DDPG respectively in all environments. Our contribution provides motivation for future AC algorithms to be designed with multiple improvement steps in mind.

2 Background

We formulate the reinforcement learning problem as an agent interacting with a finite-horizon Markov Decision Process (MDP) $\mathcal{M}(\mathcal{S}, \mathcal{A}, P, R, \rho, H)$, where $\mathcal{S}$ a discrete state space, $\mathcal{A}$ a discrete action space, $P : \mathcal{S} \times \mathcal{A} \to \mathscr{P}(\mathcal{S})$ is a conditional probability measure over the state space that defines the transition probability $P(\cdot|s, a)$. The immediate reward $R(s, a)$ is a state-action dependent random variable with mean function which is assumed to be bounded with $\max_{s,a} |\mathbb{E}[R|s, a]| < \infty$. Initial states are sampled from the start-state distribution ρ. The horizon H specifies the length of a trajectory in the environment. The objective of the agent is to find a stochastic policy $\pi : \mathcal{S} \to \mathscr{P}(\mathcal{A})$, a distribution over actions at each state, that maximizes the expected return J. This quantity can also be written as the expected state value V^π with respect to starting states s_0:

$$J(\pi) = \mathbb{E}_{s_0\sim\rho}\big[V^\pi(s_0)\big] = \mathbb{E}\Bigg[\sum_{t=0}^{H-1} \gamma^t r_t \,\Bigg|\, r_t \sim R(s_t, a_t), s_{t+1} \sim P(\cdot|s_t, a_t), a_t \sim \pi(\cdot|s_t), s_0 \sim \rho\Bigg].$$

The state value V^π can also be used to define a state-action Q value:

$$Q^\pi(s, a) = \mathbb{E}\Big[r + \gamma V^\pi(s') \,\Big|\, r \sim R(s, a), s' \sim P(s, a)\Big].$$

The discount factor $0 < \gamma \leq 1$ is traditionally set to 1 in finite horizon MDPs.

2.1 Dynamic Programming (DP)

It is a common choice to formulate DP for MDPs with operators over the state-action space. We use $\mathcal{R}$ to denote the mean-reward vector $\mathcal{R} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, where $\mathcal{R}_{s,a} = \mathbb{E}[R|s, a]$. We use $\mathcal{P}^\pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|}$ to denote the matrix of transition probabilities multiplied by a policy, indexed as follows: $\mathcal{P}^\pi_{s,a,s',a'} = P(s'|s, a)\pi(a'|s')$. We denote the state-action value q and the policy π as vectors in the state-action space s.t. $q, \pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. The set $\Pi \subset R^{|\mathcal{S}||\mathcal{A}|}$ contains all admissible policies that define a probability distribution over the action space for every state. For convenience, we denote $q(s, a), \pi(s, a)$ as a specific entry in the vector indexed by s, a and $q(s), \pi(s)$ as the appropriate $|\mathcal{A}|$ dimensional vectors for index s. In this notation, we can write expectations as the dot product $q(s) \cdot \pi(s) = \mathbb{E}_{a\sim\pi(s)}[q(s, a)] = v(s)$. With slight abuse of notation, we use $q \cdot \pi = v$, $v \in \mathbb{R}^{|\mathcal{S}|}$ to denote the vector with entries $v(s)$. We use $\max_a q \in \mathbb{R}^{|\mathcal{S}|}$ to denote the vector with entries $\max_a q(s) = \max_a q(s, a)$. Finally, we use q^π to denote the vector with entries that are the true Q values of the policy π. The optimal policy π^* that maximizes $J(\pi)$ satisfies $V^{\pi^*}(s) = \max_\pi V^\pi(s)$, $\forall s \in \mathcal{S}$ and has state-action Q value q^*.

We can write the *Bellman operator* $\mathcal{T}^\pi$ as: $(\mathcal{T}^\pi q)_{s,a} := \mathcal{R}_{s,a} + \gamma(\mathcal{P}^\pi q)_{s,a}$. It is a well known property of the Bellman operator that $\lim_{k\to\infty}(\mathcal{T}^\pi)^k q = q^\pi, \forall q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, i.e. repeated applications of the operator to any starting vector q will converge to Q^π (Bertsekas, 2007). Similarly, the *Bellman optimality operator* $\mathcal{T}^*$ can be written as follows: $(\mathcal{T}^* q)_{s,a} := \mathcal{R}_{s,a} + \gamma \sum_{s'} P(s'|s,a) \max_{a'} q_{s',a'}$ and is known to converge to the value of the optimal policy, that is: $\lim_{k\to\infty} \|(\mathcal{T}^*)^k q - q^*\| = 0, \forall q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. The convergence of $\mathcal{T}^\pi$ and $\mathcal{T}^*$ is usually proven in the infinity norm $\|\cdot\| := \|\cdot\|_\infty$, which is the norm we will use for the rest of the paper. Many DP and RL algorithms rely on *policy improvement operators* (PIOs), which are traditionally defined as a mapping $\mathcal{I} : \Pi \to \Pi$ (Sutton & Barto, 2018) satisfying the following:

Definition 1 (Policy Improvement Operator)**.** *An operator $\mathcal{I} : \Pi \to \Pi$ is a policy improvement operator if $V^{\mathcal{I}(\pi)}(s) \geq V^\pi(s), \forall s \in \mathcal{S}$, and $\exists s \in \mathcal{S}$ such that $V^{\mathcal{I}(\pi)}(s) > V^\pi(s)$, unless the policy is already optimal, that is, $V^\pi = V^*$.*

In the finite action space setting repeated application of a PIO $\mathcal{I}$ to some initial deterministic policy yields a sequence that is guaranteed to converge to the optimal policy (Sutton & Barto, 2018). Policy Iteration relies on this property for convergence, and can be described as maintaining a policy π_i and its value q^{π_i} and improving the policy at each step with $\pi_{i+1} = \mathcal{I}_{\arg\max}(\pi_i)$. Here, $\mathcal{I}_{\arg\max}(\pi_i)$ denotes an operator that produces an $\arg\max$ policy with respect to q^{π_i}. We define any policy π as an $\arg\max_a q$ policy if it has support only on the actions that maximize q. We note that every optimal policy is an $\arg\max$ policy on q^*. In Policy Iteration, the value $q^{\pi_{i+1}}$ is often computed with $q^{\pi_{i+1}} = (\mathcal{T}^{\pi_{i+1}})^k q^{\pi_i}$, by choosing a sufficiently large k for the equality to hold. In practice however, computing the exact value q^π is usually expensive and often infeasible and the majority of algorithms rely instead on an approximation of q^π for *approximate* policy improvement. We therefore define an *approximate* PIO (APIO) in Definition 2.

Definition 2 (Approximate Policy Improvement Operator)**.** *An approximate policy improvement operator is a mapping $\mathcal{I} : \Pi \times \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \to \Pi$ such that $V^{\mathcal{I}(\pi,Q^\pi)}(s) \geq V^\pi(s), \forall s \in \mathcal{S}$, and $\big(\mathcal{I}(\pi,q)\cdot q\big)(s) \geq \big(\pi \cdot q\big)(s)$, $\forall \pi \in \Pi$, $\forall q \in R^{|\mathcal{S}||\mathcal{A}|}$, as well as $\exists s \in \mathcal{S}$ such that $V^{\mathcal{I}(\pi,Q^\pi)}(s) > V^\pi$, unless the policy is already optimal, that is, $V^\pi = V^*$.*

For the exact state-action value q^π, APIOs yield strictly improved policies, like PIOs. The Value Iteration algorithm relies on $\mathcal{I}_{\arg\max}$ as an APIO to update a q vector at every step with the Bellman Optimality Operator $q_{i+1} = \mathcal{T}^* q_i = \mathcal{T}^{\mathcal{I}_{\arg\max}(\cdot,q_i)} q_i$. The Optimistic Policy Iteration (OPI) algorithm (Tsitsiklis, 2002) generalizes both algorithms into one framework by allowing a variable number of updates with the Bellman Operator, i.e. $\pi_{i+1} = \mathcal{I}_{\arg\max}(\cdot, q_i)$ and $q_{i+1} = (\mathcal{T}^{\pi_{i+1}})^k q_i$.

2.2 Actor Critic, DDPG and TD3

In this work we consider Actor-Critic (AC) algorithms as RL algorithms that maintain a policy π (actor) and an approximation to the value of the policy $q \approx Q^\pi$ (critic) (Konda & Tsitsiklis, 1999). This framework can be thought of in many respects as an interaction-based variation of DP's OPI. The majority of modern ACs use deep neural networks (DNNs) to parameterize the actor π_θ and the critic q_ϕ and optimize their parameters using stochastic gradient descent or related gradient-based optimizers. Optimization of the critic often relies on Temporal-Difference methods that approximate the Bellman update empirically. To update and improve the policy, a commonly used PIO is the *policy gradient*, described by the *policy gradient theorem* (Sutton et al., 1999):

$$\nabla_\theta J(\pi_\theta) = \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} \tfrac{\partial \pi_\theta(a|s)}{\partial\theta} Q^\pi(s,a). \tag{1}$$

The term d^π denotes the discounted improper distribution defined as $d^\pi(s_0) = \sum_{t=1}^\infty \gamma^{t-1} P(s_t = s, s_0, \pi)$, where $P(s_t = s, s_0, \pi)$ denotes the probability of being in state s after starting in state s_0 and following the policy π for t transitions. One major difference between ACs and OPI is that OPI traditionally uses $\mathcal{I}_{\arg\max}$, or more generally, PIOs that operate on deterministic policies (Williams & Baird III, 1993), while ACs usually build on the policy gradient which is originally framed exclusively for stochastic policies, a design choice many modern approaches still adhere to. A deterministic variation of the stochastic actor π_θ used in the original policy gradient theorem was proposed by Silver et al. (2014) with the Deterministic Policy Gradient (DPG) algorithm. DPG was later extended for neural networks with Deep-DPG (DDPG, Lillicrap et al., 2015) and again to better account for function approximation errors with Twin-Delayed DDPG (TD3, Fujimoto et al., 2018). TD3 introduces three main contributions: (i) To improves learning stability through robustness to approximation errors in the critic, Fujimoto et al. (2018) propose to sample actions close to the predictions of the deterministic policy π_θ when training the value function. Specifically, the actions $\tilde{a} \sim \mathcal{N}(\pi(s'), 1).clip(-\beta, \beta)$ are sampled from a normal distribution with mean $\pi(s')$ and clipped. (ii) To reduce over-estimation bias, Fujimoto et al. (2018) propose to use *Clipped Double Q-learning*, which takes the minimum over two critics for the value bootstrap. Putting together (i) and (ii), the value target used to train both critics is computed as follows: $y(s, a) = r(s, a) + \gamma \min_i q_{\phi_i}(s', \tilde{a})$, $i \in \{1, 2\}$. (iii) Finally, TD3 delays updates to the actor such that the two critics are closer to their convergence value (i.e. updated more often) before updating the actor.

3 Dynamic Programming with Stochastic Policy Improvement Operators

To facilitate theoretical analysis of an approach that utilizes two separate improvement operators applied to the policy and value respectively, we begin this work by outlining an extended form of OPI that generalizes for stochastic policies and applicable APIOs in Algorithm 1:

Algorithm 1 Generalized Optimistic Policy Iteration (GOPI).

```
1: For starting vectors $q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $\pi \in \Pi$, SAPIO $\mathcal{I}$, $k \geq 1$ and chosen $\epsilon > 0$
2: while $\|\mathcal{T}^* q - q\| > \epsilon$ or $\|\pi \cdot q - \max_a q\| > \epsilon$ do
3:     $\pi \leftarrow \mathcal{I}(\pi, q)$
4:     $q \leftarrow (\mathcal{T}^\pi)^k q$
   return $q, \pi$
```

Algorithm 1 is general in that it reduces to standard OPI when $\mathcal{I} = \mathcal{I}_{\arg\max}$ (or more generally, when Π is the class of deterministic policies). While previous works have investigated convergence properties for OPI with a large range of APIOs and deterministic policies (Williams & Baird III, 1993), similar guarantees for the setting of Algorithm 1 with stochastic policies and appropriate (A)PIOs are not yet established to the best of our knowledge. When the policy π is stochastic, the (A)PIO property that $\exists s \in \mathcal{S}$ such that $V^{\mathcal{I}(\pi)}(s) > V^\pi(s)$ is not sufficient for Algorithm 1 to converge, because unlike deterministic policies, stochastic policies allow for infinitesimally small improvement even when the action space is finite. For this reason (i) we formulate the convergence of π in the **while** condition in line 2 in Algorithm 1 with $\|\pi \cdot q - \max_a q\| \leq \epsilon$ and (ii) we extend the definition of APIOs with the *sufficiency* condition, which is necessary to guarantee convergence of stochastic policies to $\arg\max$ policies:

Definition 3 (Sufficient Approximate Policy Improvement Operator)**.** *Let $q^0, q^1, \ldots$ be a sequence of q vectors such that $\lim_{m\to\infty} q^m = q$. Let $\pi^0, \pi^1, \ldots$ be a sequence of policies where $\pi^{i+1} = \mathcal{I}(\pi^i, q^i)$. An operator $\mathcal{I} : \Pi \times \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \to \Pi$ is a sufficient APIO (SAPIO) if $\mathcal{I}$ satisfies the conditions of an APIO (Definition 2) and the sequence (π^i, q^i) converges such that $\lim_{m\to\infty} \|\pi^m \cdot q^m - \max_a q\| = 0$, for any π^0.*

The sufficiency condition guarantees that the convergence of SAPIOs is sufficiently stable even in the presence of non-constant values q_i, as long as the sequence q_i itself converges. A simple example of an APIO that violates the sufficiency condition is policy gradient with a learning rate sequence $\alpha_i = \alpha/2^i$, which may decay too fast to converge to an optimal policy, or more generally any APIO that converges to local optima. A simple example of an operator that satisfies all conditions is $\mathcal{I}_{\arg\max}$. We can now present our first theoretical result, convergence for SAPIO-based GOPI:

Theorem 1 (Convergence for Algorithm 1)**.** *For any finite horizon MDP $\mathcal{M}$, SAPIO $\mathcal{I}$, $k \geq 1$, $q^0 \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and $\pi^0 \in \Pi$, the policy and value iterates produced by Algorithm 1 converge to the optimal policy and value respectively. That is, for every $\epsilon > 0$ there exists an $M_\epsilon \in \mathbb{N}^+$ such that after m iterations of the algorithm, $||q^m - q^*|| < \epsilon$ and $||\pi^m \cdot q^m - \max_a q^*|| < \epsilon$, for all $m \geq M_\epsilon$.*

The proof uses induction and builds on immediate convergence of values of terminal states s_H, convergence of policies at states s_{H-1} and finally on showing that given that q, π converge for all states s_{t+1}, they also converge for all states s_t. The full proof is provided in Appendices A.1 and A.2.

4 Value-Improved Generalized Optimistic Policy Iteration

Algorithm 1 unifies many policy and value based DP approaches into one framework by distilling the improvement into one operation. However, it is possible to describe a different framework that leverages both improvements *separately*. We frame this approach in Algorithm 2.

Algorithm 2 Value-Improved Generalized Optimistic Policy Iteration (VI-GOPI).

```
1: For starting vectors $q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $\pi \in \Pi$, SAPIO $\mathcal{I}_1$, APIO $\mathcal{I}_2$, $k \geq 1$ and chosen $\epsilon > 0$
2: while $\|\mathcal{T}^* q - q\| > \epsilon$ or $\|\pi \cdot q - \max_q q\| > \epsilon$ do
3:     $\pi \leftarrow \mathcal{I}_1(\pi, q)$
4:     $q \leftarrow (\mathcal{T}^{\mathcal{I}_2(\pi,q)})^k q$
   return $q, \pi$
```

Algorithm 2 extends Algorithm 1 with an additional improvement step that only influences the value update (marked in blue), in the spirit of value-based approaches. Note that line 4 in Algorithm 2 computes a second improved policy $\mathcal{I}_2(\pi, q)$, uses it to update q with the Bellman operator and then proceeds to discard it. That may seem like an odd choice: if we already endeavor to update the policy and use it to update the value, why would we discard it? Imagine the main policy maintained by the algorithm π_θ is parametric, as it often is in modern RL. In this case we are limited to improvement operators $\mathcal{I}_1$ that return parametric policies. However, $\mathcal{I}_2$ may not have the same limitation, as the policy produced by $\mathcal{I}_2(\pi, q)$ can be a non-parametric policy (sampling-based, for example). If we choose an operator $\mathcal{I}_2$ that produces policies that are not representable by π_θ, we essentially have to discard $\mathcal{I}_2(\pi, q)$ after every update. We verify that this new scheme converges in our second theoretical result, the convergence for Algorithm 2:

Theorem 2 (Convergence of Value-Improved Optimistic Policy Iteration)**.** *For any finite horizon MDP $\mathcal{M}$, SAPIO $\mathcal{I}_1$, APIO $\mathcal{I}_2$, $k \geq 1$, $q^0 \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $\pi^0 \in \Pi$ and $\epsilon > 0$, the policy and value maintained by Algorithm 2 converge to the optimal policy and value respectively. That is, there exists an $M_\epsilon \in \mathbb{N}^+$ such that after m iterations, $||q^m - q^*|| < \epsilon$ and $||\pi^m \cdot q^m - \max_a q^*|| < \epsilon$, $\forall m \geq M_\epsilon$.*

The proof follows the same structure as the proof for Algorithm 1 and is provided in Appendix A.3.

5 Value-Improved Actor Critic Algorithms

As a base for a practical Value-Improved AC (VI-AC) algorithm in the spirit of VI-GOPI we choose the popular off-policy AC algorithms DDPG and TD3, which learn implicitly deterministic policies but act and in TD3 also propagate the value of stochastic ones, making them a good fit for a framework where the critic and actor policy-classes differ. A simple APIO $\mathcal{I}_2$ that can be used in the value update is the sampling-based $\arg\max$, where we sample n actions $A_n = \{a_1, \ldots, a_n\}$ from $\pi(s)$ (if π is not a stochastic policy, one can sample zero-mean noise and add it to the prediction $\pi(s)$). Then $\mathcal{I}_2(\pi(s), q(s)) = \arg\max_{a_i \in A_n} q(s, a_i)$. This operator is very likely to be sensitive to errors in the q function however, as argued by Fujimoto et al. (2018). A more robust option is the Mean-Top-k operator $\mathcal{I}_{mtk}$. $\mathcal{I}_{mtk}$ samples n actions $A_n = \{a_1, \ldots, a_n\}$ from $\pi(s)$ and sorts the actions according to the q values $q(s, a_i)$. Rather than returning the $\arg\max$, $\mathcal{I}_{mtk}$ returns a uniform policy across the top k sorted actions. The $\mathcal{I}_{mtk}$ operator does not directly incentivize the policy to optimize for actions where the critic is smooth however, which is the motivation for the introduction of Gaussian noise to TD3. Specifically, the smaller k, the more sensitive the operator is to errors in the q function, while larger k/n reduce the significance of the improvement. As a better fit for TD3 we construct $\mathcal{I}_\mathcal{N}$, a clipped Gaussian variation of the sample-max operator in Algorithm 3.

Similarly, $\mathcal{I}_\mathcal{N}$ samples n actions $A_n = \{a_1, \ldots, a_n\}$ from $\pi(s)$. We interpret each action $a_i \in A_n$ as a proposal mean for an improved clipped-Gaussian policy $\pi_i = \mathcal{N}(a_i, 1).clip(-\beta, \beta)$. Additional m actions $\mathcal{A}_{i,m} = \{a_1, \ldots, a_m\}$ are sampled from each proposed policy and the mean q value of these actions is computed: $v_i(s) = \frac{1}{m} \sum_{a_j \in \mathcal{A}_{i,m}} q(s, a_j)$ as to approximate the value of the clipped Gaussian policy π_i. Finally, $\mathcal{I}_\mathcal{N}$ chooses $a_i = a_{\arg\max_i v_i(s)}(s)$ as the mean of the improved policy. That is, the improved policy $\mathcal{I}_\mathcal{N}(\pi, q)(s) = \pi_i(s)$. It is easy to see that in expectation both $\mathcal{I}_{mtk}$ and $\mathcal{I}_\mathcal{N}$ satisfy the requirements of APIO: in expectation over the sampling process, the $\arg\max$ operator will return the policy π_i whose value v_i is the largest, and given access to true value Q^π, true value V^{π_i} is the largest.

Algorithm 3 Clipped Gaussian Policy Improvement $\mathcal{I}_N$

```
    Input: actor $\pi$, a critic $q$, a state $s$ and hyper parameter $\beta$
1: $\{\alpha_1, \dots, \alpha_n\} \sim \mathcal{N}(0,1)$                                  ▷ Sample $n$ samples
2: $\Pi_n(s) \leftarrow \{\mathcal{N}(\pi_\theta(s) + \alpha_1, 1), \dots, \mathcal{N}(\pi_\theta(s) + \alpha_n, 1)\}$    ▷ Instantiate $n$ policies
3: $V_n = \{\}$                                   ▷ Instantiate an empty set of policy-values.
4: for each policy $\pi'_i \in \Pi_n$ do                    ▷ Will be done in parallel in practice
5:     $\mathcal{A}_m \leftarrow \{a_1, \dots, a_m\} \sim \pi'_i(s).clip(-\beta, \beta)$.      ▷ Sample $m$ samples from each policy
6:     $v_i(s) \leftarrow \frac{1}{m} \sum_{a \in \mathcal{A}_m} q(s,a)$          ▷ Approximate the sampling-based value of the policy.
7:     Add $v_i(s)$ to the set $V_n$
8: $\pi'(s) \leftarrow \pi'_{\arg\max_i v_i(s) \in V_n}$                        ▷ Identify the best policy $\pi'(s)$
9: return $\pi'(s)$.
```

As discussed in Section 2.2, Fujimoto et al. (2018) identify over-estimation bias as a significant detrimental effect in DDPG, and this effect can be expected to increase in VI-DDPG by further maximizing the value targets. To mitigate this effect, one can use Fujimoto et al.'s *Clipped Double Q-learning* approach of replacing the value-bootstrap prediction in the value target with the minimum over two critics (referred to as *double-critics* from here on). We employ this for $\mathcal{I}_{mtk}$ and $\mathcal{I}_{\mathcal{N}}$ by replacing every prediction of $q_\phi(s,a)$ in the value update step with $\min_{\phi_i} q_{\phi_i}(s,a), i = \{1,2\}$. We investigate the interaction between over-estimation that may be introduced with VI and the clipped Q-learning over-estimation-reduction mechanism in Section 6. We summarize VI-DDPG and VI-TD3 in Algorithms 4 and 5 respectively. Differences to baseline DDPG/TD3 are marked in blue. To maintain as many of the original design choices and existing differences between DDPG and TD3 to better evaluate the effect of VI, in VI-TD3 actions are resampled from the improved policy (line 10 in Algorithm 5) while in VI-DDPG they are not. For additional details see Appendix C.1.

Algorithm 4 Value-Improved DDPG (VI-DDPG)

```
 1: Initialize policy network $\pi_\theta$, $Q$ networks $q_{\phi_1}, q_{\phi_2}$, APIO $\mathcal{I}$ and replay buffer $\mathcal{B}$
 2: for each episode do
 3:     for each environment interaction $t$ do
 4:         Take action $a_t \sim \mathcal{N}(\pi_\theta(s_t), \sigma^2)$        ▷ Using the exploration-noise hyperparameter $\sigma^2$
 5:         Observe $s_{t+1}, r_t$
 6:         Add the transition $(s_t, a_t, r_t, s_{t+1})$ to the buffer $\mathcal{B}$
 7:         Sample a batch $b$ of transitions of the form $(s_t, a_t, r_t, s_{t+1})$ from $\mathcal{B}$
 8:         Update $\pi_\theta$ using the deterministic policy-gradient $\forall s_t \in b$
 9:         Approximate a deterministic improved policy $\pi'(s_{t+1}) = \mathcal{I}(\pi_\theta, \min_j q_{\phi_j})(s_{t+1})$
10:         Compute the value targets $y(s_t, a_t) \leftarrow r_t + \gamma q(s_{t+1}, \pi'(s_{t+1})), \forall (s_t, a_t, r_t, s_{t+1}) \in b$
11:         Update $q_{\phi_1}, q_{\phi_2}$ with gradient descent and MSE loss using targets $y$
    return $\pi_\theta$.
```

Algorithm 5 Value-Improved TD3 (VI-TD3)

```
 1: Initialize policy network $\pi_\theta$, $Q$ networks $q_{\phi_1}, q_{\phi_2}$, APIO $\mathcal{I}$ and replay buffer $\mathcal{B}$
 2: for each episode do
 3:     for each environment interaction $t$ do
 4:         Take action $a_t \sim \mathcal{N}(\pi_\theta(s_t), \sigma^2)$        ▷ Using the exploration-noise hyperparameter $\sigma^2$
 5:         Observe $s_{t+1}, r_t$
 6:         Add the transition $(s_t, a_t, r_t, s_{t+1})$ to the buffer $\mathcal{B}$
 7:         Sample a batch $b$ of transitions of the form $(s_t, a_t, r_t, s_{t+1})$ from $\mathcal{B}$
 8:         Update $\pi_\theta$ every $n$ value updates using the deterministic policy-gradient $\forall s_t \in b$
 9:         Approximate an improved policy $\pi'(s_{t+1}) = \mathcal{I}(\pi_\theta, \min_j q_{\phi_j})(s_{t+1}), \forall s_{t+1} \in b$
10:         Sample an action from the improved policy $a \sim \pi'(s_{t+1}), \forall s_{t+1} \in b$
11:         Compute the value targets $y(s_t, a_t) \leftarrow r_t + \gamma \min_j q_{\phi_j}(s_{t+1}, a), \forall (s_t, a_t, r_t, s_{t+1}) \in b$
12:         Update $q_{\phi_1}, q_{\phi_2}$ with gradient descent and MSE loss using targets $y$
    return $\pi_\theta$
```

6 Experiments

We evaluate VI-TD3 and VI-DDPG on the Mujoco benchmark in Figure 1. We include popular baselines in the form of PPO (Schulman et al., 2017b) and MPO (Abdolmaleki et al., 2018) for reference. Performance is presented using mean and 2 standard errors over 20 seeds of evaluation episodes taken throughout training. Our implementation builds on the popular code base CleanRL (Huang et al., 2022). See Appendix C for full experimental details. The VI-TD3 agent uses $\mathcal{I}_{\mathcal{N}}$ with $n = m = 16$. Two variations of VI-DDPG are included, both use double-critics as in TD3. One variation uses $\mathcal{I}_{\mathcal{N}}$ with $n = m = 16$ and the other uses $\mathcal{I}_{mtk}$ with $n = 128,\ k = 4$. Reference algorithms are dashed. Agents that appear in multiple figures retain their legend colour from the first figure they appear in. In all environments, VI-TD3 and VI-DDPG with $\mathcal{I}_{mtk}$ match or outperform their respective baselines TD3 or DDPG. VI-DDPG with $\mathcal{I}_{\mathcal{N}}$ significantly outperforms DDPG in all environments except Swimmer and HalfCheetah, where the double-critics mechanism has a strong detrimental effect (see Figure 2 below and Figure 5 in Appendix B).

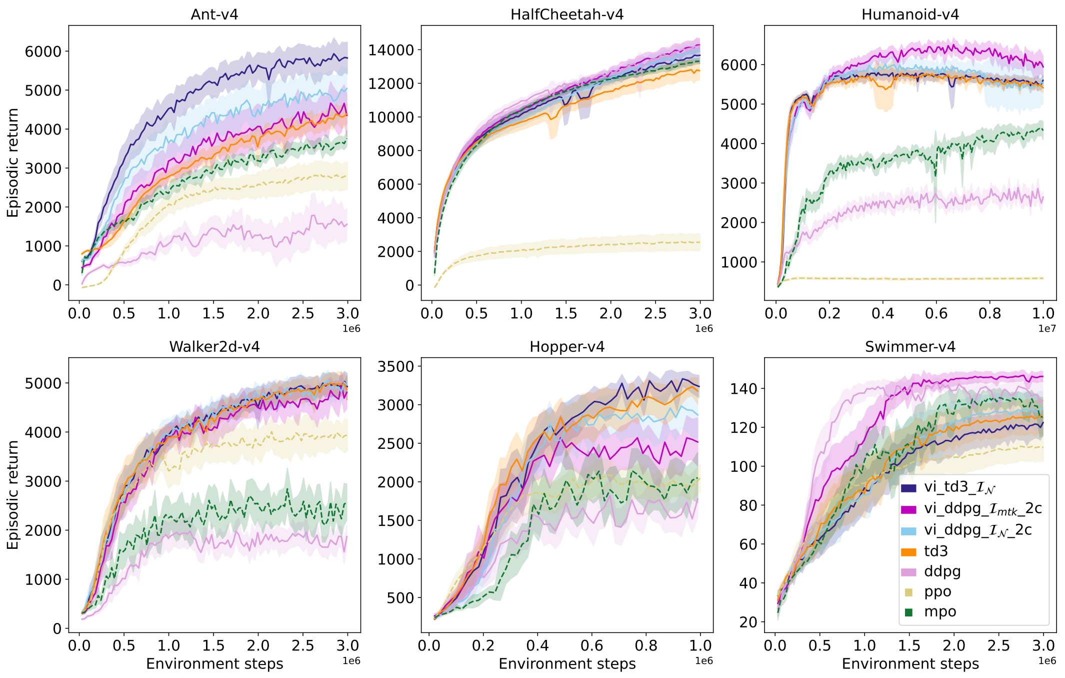

Figure 1: Mean and 2 standard errors in the shaded area across 20 seeds in evaluation for VI-TD3 with $\mathcal{I}_{\mathcal{N}}$, VI-DDPG with $\mathcal{I}_{\mathcal{N}}$ and double-critics (2c), VI-DDPG with $\mathcal{I}_{mtk}$ and 2c, TD3, DDPG, PPO and MPO.

In Figure 2 we investigate the interaction between applying an APIO in the value-update and overestimation bias, through the lens of VI-DDPG. We compare DDPG with and without double-critics to VI-DDPG with $\mathcal{I}_{\mathcal{N}}$, $n = m = 16$, with and without double-critics. Where double-critics are beneficial – Ant and Humanoid – VI-DDPG with double-critics outperforms or matches all other agents in Figure 2, while without it is the least performing agent. Where double-critics are detrimental – HalfCheetah – VI-DDPG with double-critics demonstrates significant improvement over double-critics DDPG, about matching the performance of (VI-)DDPG without double-critics. We include ablations in additional environments in Appendix B where the same conclusion holds.

In Figure 3 we investigate the behaviour of VI-DDPG/TD3 across different values of the hyperparameters n, m and n, k introduced with $\mathcal{I}_{\mathcal{N}}$ and $\mathcal{I}_{mtk}$ respectively. In the top row we investigate the trade-off between greediness and robustness induced in $\mathcal{I}_{mtk}$ by different values of k for a constant $n = 128$ with VI-DDPG, including DDPG baselines with and without double-critics respectively. In the bottom row we use VI-TD3 to investigate the gain from increasing values of the hyperparameters n, m which improve the robustness of the operator $\mathcal{I}_{\mathcal{N}}$. The general trend for $n = m$ is as one might expect, larger values of n and m improve or match on smaller. For $\mathcal{I}_{mtk}$ there is an apparent trade off between $k \gg 1$ which demonstrates improved stability and $k \ll n$ which has a stronger maximization effect. In Swimmer and HalfCheetah where the effect of double-critics is detrimental,

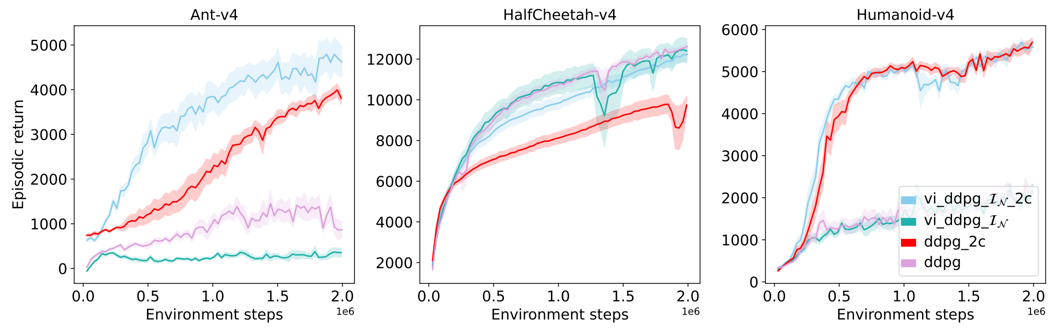

Figure 2: Mean and one standard error in the shaded area across 10 seeds in evaluation for VI-DDPG with $\mathcal{I}_\mathcal{N}$, $n = m = 16$ and DDPG, both with and without double-critics (2c).

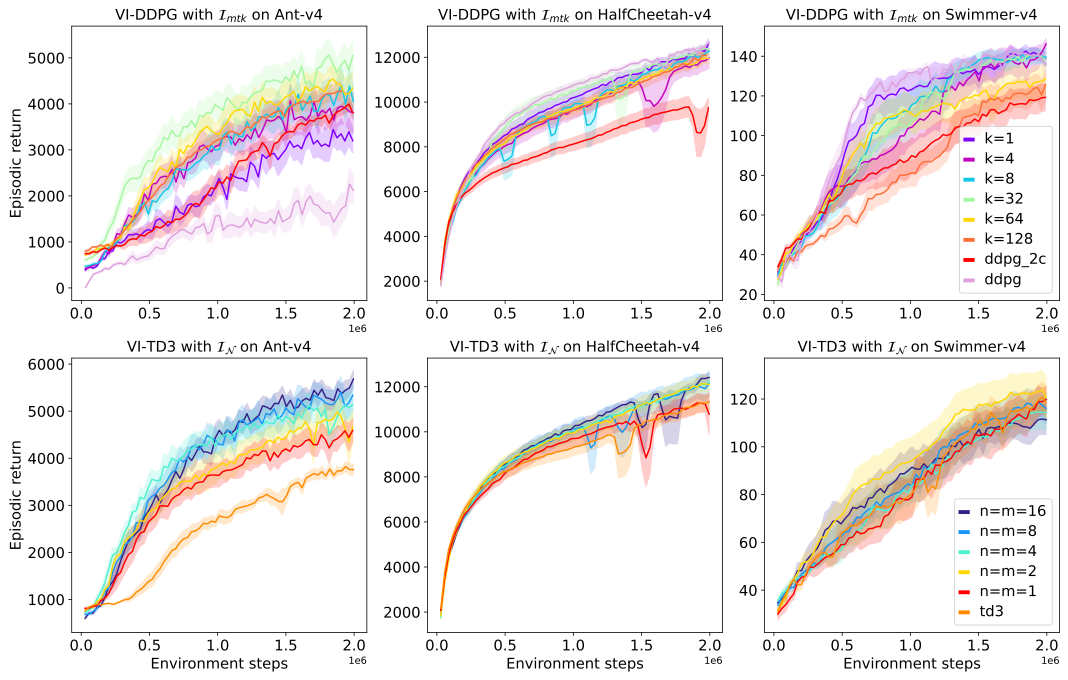

Figure 3: Mean and one standard error in the shaded area across 10 seeds in evaluation. Top row: VI-DDPG with $\mathcal{I}_{mtk}$, $n = 128$ double-critics and different values of k, as well as DDPG with and without double-critics (2c). Bottom row: VI-TD3 with different values of $n = m$ as well as baseline TD3.

values of $k < n$ regain some / all of the performance lost by double-critics and $k = n$ regains the least or not at all. On Ant however small values of k are among the worst performers while $k = n$ is among the three best performers, demonstrating a different trade off between stability and maximization. Overall, $\mathcal{I}_\mathcal{N}$ achieves better performance in some environments, presumably due to being more robust, while $\mathcal{I}_{mtk}$ achieves better performance in other environments, presumably due to a greedier policy improvement.

6.1 Limitations

Apart from the additional algorithmic complexity and hyperparameters introduced to the training process of the AC framework, the introduction of an additional policy improvement operator used in the value update step can have both beneficial and detrimental effects, which are likely to depend on the specific algorithm and operator combination. We list limitations and / or concerns we believe might be relevant, to contrast with the benefits demonstrated in Figure 1: (i) Computing an improved policy in the value update step will incur additional compute cost that depends on the APIO $\mathcal{I}_2$ used.

(ii) Disconnecting the critic from the policy may result in additional sources of instability as a result of the value-update being more off-policy. (iii) As mentioned in Section 5 and demonstrated in Figure 2, updating the critic with respect to an improved policy is likely to result in an increase to over-estimation effects.

Specifically for $\mathcal{I}_{\mathcal{N}}$, the compute cost introduced in number of sequential operations is one additional forward pass for the actor and one for the critic networks, n (parallel) averaging operations across m and a maximization operation across n. The number of parallel operations scales with the product $nm|b|$ where $|b|$ is the batch size. $\mathcal{I}_{mtk}$'s number of parallel operations scales with $n|b|$. In terms of sequential operations, $\mathcal{I}_{mtk}$ is bounded by the cost of one sorting operation across n as well as averaging across k. In Figure 2 the comparison between VI-DDPG with and without double-critics demonstrates the reliance of VI-DDPG on over-estimation-mitigating mechanisms. These mechanisms by themselves can be detrimental in some environments (as demonstrated by agents employing double-critics in HalfCheetah and Swimmer). On the other hand, both $\mathcal{I}_{\mathcal{N}}$ and $\mathcal{I}_{mtk}$ demonstrate capacity to regain performance that was lost due to the double-critics mechanism.

7 Related Work

Dynamic Programming & Convergence Williams & Baird III (1993) investigate convergence among many other theoretical properties of OPI algorithms with deterministic policies and applicable PIOs. Littman & Szepesvári (1996) describe a general DP framework that extends Value Iteration to a large set of PIOs that induce convergence through contraction. Perkins & Precup (2002) investigate convergence of Policy Iteration with linear approximation with a variety of PIOs. Tsitsiklis (2002) prove a variety of convergence properties for OPI algorithms that rely on $\mathcal{I}_{\arg\max}$ specifically. Bertsekas (2011) prove many theoretical properties of approximate OPI for deterministic policies and $\mathcal{I}_{\arg\max}$. Smirnova & Dohmatob (2019) investigate convergence of variations of approximate OPI with PIOs that operate directly on the estimated values, i.e. $\mathcal{I} : \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \to \Pi$. Previous works prove a range of convergence results for AC algorithms, although works in this area generally focus specifically on the policy-gradient improvement operator and function-approximation related challenges (see Xu et al., 2020; Holzleitner et al., 2021; Qiu et al., 2021).

Connecting Value and Policy Based Methods Ghosh et al. (2020) connect policy based and value based RL methods by casting them in operator form and showing that policy-based Reinforce (Williams, 1992) and value-based Q-learning (Sutton & Barto, 2018) can be seen as two sides of the same coin. O'Donoghue et al. (2016); Nachum et al. (2017); Schulman et al. (2017a) connect *soft* policy and value based methods by showing equivalence between Q-learning and policy gradient methods in the setting of entropy-regularized RL, which optimizes for a different objective than standard RL.

Connections to Existing Algorithms As touched on in Section 5, TD3 learns a *deterministic* policy but approximates the value of a *stochastic* policy. This can be thought of as similar to Algorithm 2 where instead of $\mathcal{I}_2$ being an APIO it acts as a projection operator. GreedyAC (Neumann et al., 2023) maintains two actors and one critic. Both actors are improved with respect to the same critic, although the critic only approximates the value of one of the two actors. In a similar vein, in Algorithm 2 the actor is improved with respect to a critic that does not approximate its value. A variation of Reanalyze (RZ, Schrittwieser et al., 2021) is perhaps the most similar to the essence of Algorithm 2. In RZ, the value-bootstrap in the TD-based value-targets can be computed in one of two ways: (i) using the value function $v \approx V^{\pi}$, or (ii) with a prediction from MCTS which (in principle) approximates V^* directly. While Ye et al. (2021) found the second variation to not be worth the compute cost, we note that RZ uses n-step value targets which reduce the contribution of the bootstrapped value by γ^n, and planning with MCTS is very computationally intensive unlike the operators proposed in this work. Generating value targets with (ii) is an example of an approach used in model based RL where value targets are generated from planning (see Moerland et al., 2023). From the perspective that views $\arg\max_a q$ as 1-step-look-ahead planning, VI-AC can be thought of as a model-free agent using two separate 1-step-look-ahead planners $\mathcal{I}_1$ and $\mathcal{I}_2$ to generate value targets and policy improvement respectively.

8 Conclusions

We extend the AC framework with an additional policy improvement operator applied in the value-update step, which we call Value-Improved (VI-)AC. To facilitate theoretical analysis of this approach we describe two Dynamic Programming algorithms: the first extends Optimistic Policy Iteration to general policy improvement operators and stochastic policies (GOPI), and the second extends GOPI by incorporating an additional improvement operator in the value update step, in the spirit of VI-AC. We verify that both approaches converge in the finite horizon setting in the presence of *sufficient* policy improvement operators. The convergence of GOPI strengthens the theoretical motivation behind the choice to base novel RL algorithms on novel policy improvement operators, and provides a set of sufficient conditions for these operators to induce convergence in finite horizon with DP, as a first step in identifying necessary and sufficient conditions under more general settings. We design two practical VI-AC algorithms building upon the popular online off-policy ACs TD3 and DDPG. We evaluate VI-TD3/DDPG in the Mujoco benchmark where they outperform or match the performance of TD3 and DDPG respectively in all environments tested. We believe that this work provides motivation for future AC algorithms to be designed with multiple improvement steps in mind. In addition, our work provides a new venue to benefit from novel sampling-based improvement operators.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

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

Dimitri P Bertsekas. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335, 2011.

D.P. Bertsekas. Approximate dynamic programming. In *Dynamic Programming and Optimal Control*, number v. 2 in Athena Scientific optimization and computation series, chapter 6. Athena Scientific, 3 edition, 2007. ISBN 9781886529304.

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

Dibya Ghosh, Marlos C Machado, and Nicolas Le Roux. An operator view of policy gradient methods. *Advances in Neural Information Processing Systems*, 33:3397–3406, 2020.

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

Markus Holzleitner, Lukas Gruber, José Arjona-Medina, Johannes Brandstetter, and Sepp Hochreiter. Convergence proof for actor-critic methods applied to ppo and rudder. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XLVIII: Special Issue In Memory of Univ. Prof. Dr. Roland Wagner*, pp. 105–130. Springer, 2021.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

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

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Michael L Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In *ICML*, volume 96, pp. 310–318, 1996.

Thomas M Moerland, Joost Broekens, Aske Plaat, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Samuel Neumann, Sungsu Lim, Ajin George Joseph, Yangchen Pan, Adam White, and Martha White. Greedy actor-critic: A new conditional cross-entropy method for policy improvement. In *The Eleventh International Conference on Learning Representations*, 2023.

Brendan O'Donoghue, Remi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. Combining policy gradient and q-learning. In *International Conference on Learning Representations*, 2016.

Theodore Perkins and Doina Precup. A convergent form of approximate policy iteration. In S. Becker, S. Thrun, and K. Obermayer (eds.), *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002.

Shuang Qiu, Zhuoran Yang, Jieping Ye, and Zhaoran Wang. On finite-time convergence of actor-critic algorithm. *IEEE Journal on Selected Areas in Information Theory*, 2(2):652–664, 2021.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591, 2021.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

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

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

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

Elena Smirnova and Elvis Dohmatob. On the convergence of approximate and regularized policy iteration schemes. *arXiv preprint arXiv:1909.09621*, 2019.

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

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

John N Tsitsiklis. On the convergence of optimistic policy iteration. *Journal of Machine Learning Research*, 3(Jul):59–72, 2002.

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

Ronald J Williams and Leemon C Baird III. Analysis of some incremental variants of policy iteration: First steps toward understanding actor-critic learning systems. Technical report, Citeseer, 1993.

Tengyu Xu, Zhe Wang, and Yingbin Liang. Non-asymptotic convergence analysis of two time-scale (natural) actor-critic algorithms. *arXiv preprint arXiv:2005.03557*, 2020.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering Atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

A Proofs

A.1 Proof for Theorem 1 for $k = 1$

In this section we will prove Theorem 1 for $k = 1$ for readability, and in the following Section, Appendix A.2 we extend the proof for $k \geq 1$.

Proof. Theorem 1 We will prove by backwards induction from the terminal states that the sequence $\lim_{m\to\infty}(\pi^m, q^m)$ induced by Algorithm 1 converges for any q^0, π^0, SAPIO $\mathcal{I}$ and $k \geq 1$. That is, for every $\epsilon > 0$ there exists a M_ϵ such that $\|q^m - q^*\| \leq \epsilon$ and $\|\pi^m \cdot q^m - \max_a q^*\| < \epsilon$ for all $m \geq M_\epsilon$, $q^0 \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and $\pi^0 \in \Pi$.

We let s_t denote a state $(\cdot, t) \in \mathcal{S}$, that is, a state in the environment arrived at after t transitions. The states s_H are terminal states, and the indexing begins from s_0. We let q^m, π^m denote the vectors at iteration m of Algorithm 1. We let q^m_t, π^m_t denote the sub-vectors of all entries in q^m, π^m associated with states s_t. In this notation q^1_{H-1} is the q vector for all terminal transitions $(s_{H-1}, \cdot)$ after the one iteration of the algorithm.

Induction Hypothesis: For every $\epsilon > 0$ there exist M^ϵ_{t+1} such that for all $m \geq M^\epsilon_{t+1}$ we have $\|q^m_{t+1} - q^*_{t+1}\| \leq \epsilon$, and $\|\pi^m_{t+1} \cdot q^m_{t+1} - \max_a q^*_{t+1}\| \leq \epsilon$.

Base Case $t = H - 1$: Let $\epsilon > 0$. Since states s_H are terminal, and have therefore value 0, we have $q^m_{H-1} = \mathcal{R}_{H-1} = q^*_{H-1}$ and therefore $\|q^m_{H-1} - q^*_{H-1}\| \leq \epsilon$ trivially holds for all $m \geq 1$.

By the Sufficiency condition of the SAPIO which induces convergence of π^m to an $\arg\max$ policy with respect to q there exists M^ϵ_{H-1} such that:

$$\|\pi^m_{H-1} \cdot q^m_{H-1} - \max_a q^*_{H-1}\| = \|\pi^m_{H-1} \cdot q^*_{H-1} - \max_a q^*_{H-1}\| \leq \epsilon$$

for all $m \geq M^\epsilon_{H-1}$. Thus the Induction Hypothesis holds at the base case.

Case $t < H - 1$: We will show that if the Induction Hypothesis holds for all states $t + 1$, it also holds for states t.

Step 1: Let $\epsilon > 0$. Assume the Induction Hypothesis holds for states $t + 1$. Then there exists M^ϵ_{t+1} such that $\|q^m_{t+1} - q^*_{t+1}\| \leq \epsilon$ and $\|\pi^m_{t+1} \cdot q^m_{t+1} - \max_a q^*_{t+1}\| \leq \epsilon$ for all $m \geq M^\epsilon_{t+1}$.

Let us define the transition matrix $\mathcal{P} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|}$ with $\mathcal{P}_{s,a,s'} = P(s'|s, a)$.

First, for all $m \geq M^\epsilon_{t+1}$ we have:

$$\begin{aligned}
\|q^{m+1}_t - q^*_t\| &= \|\mathcal{R} + \gamma\mathcal{P}(\pi^{m+1}_{t+1} \cdot q^m_{t+1}) - \mathcal{R} - \gamma\mathcal{P}\max_a q^*_{t+1}\| && (2)\\
&= \gamma\|\mathcal{P}(\pi^{m+1}_{t+1} \cdot q^m_{t+1}) - \mathcal{P}\max_a q^*_{t+1}\| && (3)\\
&\leq \|\mathcal{P}\|\|\pi^{m+1}_{t+1} \cdot q^m_{t+1} - \max_a q^*_{t+1}\| && (4)\\
&\leq \|\pi^{m+1}_{t+1} \cdot q^m_{t+1} - \max_a q^*_{t+1}\| && (5)\\
&\leq \epsilon && (6)
\end{aligned}$$

(2) is by substitution based on step 4 in Algorithm 1 for $k = 1$. (4) is by the definition of the operator norm $\|\mathcal{P}\|$. (5) is by the fact that the operator norm in sup-norm of all transition matrices is 1 (Bertsekas, 2007). (6) is slightly more involved, and follows from the Induction Hypothesis and the *approximate improvement* of the SAPIO.

Let us show that (6), i.e. $\|\pi^{m+1}_{t+1} \cdot q^m_{t+1} - \max_a q^*_{t+1}\| \leq \epsilon$ holds. Under the infinity norm holds point-wise for each state $s \in \mathcal{S}$:

$$\begin{aligned}
-\epsilon &\leq [\pi^m_{t+1} \cdot q^m_{t+1}](s) - \max_a q^*_{t+1}(s, a) && (7)\\
&\leq \boldsymbol{[\pi^{m+1}_{t+1} \cdot q^m_{t+1}](s) - \max_a q^*_{t+1}(s, a)} && (8)\\
&\leq \max_{a'} q^m_{t+1}(s, a') - \max_a q^*_{t+1}(s, a) && (9)\\
&\leq \max_{a'} \left(q^m_{t+1}(s, a') - q^*_{t+1}(s, a')\right) && (10)\\
&\leq \epsilon\,. && (11)
\end{aligned}$$

(7) is the induction hypothesis $\|\pi_{t+1}^m \cdot q_{t+1}^m - \max_a q_{t+1}^*\| \leq \epsilon$, which holds under the infinity norm point wise, (8) uses the SAPIO property $[\pi_{t+1}^m \cdot q_{t+1}^m](s) \leq [\pi_{t+1}^{m+1} \cdot q_{t+1}^m](s)$, (9) the inequality $[\pi_{t+1}^{m+1} \cdot q_{t+1}^m](s) \leq \max_{a'} q_{t+1}^m(s, a')$, (10) the inequality $-\max_a q_{t+1}^*(s,a) \leq -q_{t+1}^*(s,a'), \forall a' \in \mathcal{A}$, and (11) the induction hypothesis $\|q_{t+1}^* - q_{t+1}^m\| \leq \epsilon$.

Step 2: Pick $M_t^\epsilon \geq M_{t+1}^\epsilon$ such that for all $m \geq M_t^\epsilon$ we have $\|\pi_t^m \cdot q_t^m - \max_a q_t^*\| \leq \epsilon$ which must exist by SAPIO's sufficiency condition, Step 1 and the Induction Hypothesis. Thus, the Induction Hypothesis holds for all states t if it holds for states $t+1$.

Finally, let $\epsilon > 0$. By backwards induction, for each $t = 0, \ldots, H-1$ there exists M_ϵ^t such that for all $m \geq M_\epsilon^t$ we have $\|q_t^m - q_t^*\| \leq \epsilon$, and $\|\pi_t^m \cdot q_t^m - \max_a q_t^*\| \leq \epsilon$. Therefore, we can pick $N_\epsilon = \max_{t=0,\ldots,H-1} M_\epsilon^t$ such that $\|q_t^m - q_t^*\| \leq \epsilon$, and $\|\pi_t^m \cdot q_t^m - \max_a q_t^*\| \leq \epsilon$ for all $m \geq N_\epsilon$ and $t = 0, \ldots, H-1$, proving that Algorithm 1 converges to an optimal policy and optimal q-values for any $\pi^0 \in \Pi, q^0 \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, $k = 1$ and SAPIO $\mathcal{I}$. □

We proceed to extend the proof for $k \geq 1$ below.

A.2 Extension of the Proof for Theorem 1 to $k \geq 1$

In this section we will extend the proof of Theorem 1 from Appendix A.1 to $k \geq 1$. Much of the proof need not be modified. In order to extend the proof to $k \geq 1$, we only need to show the following: For all $k \geq 1$ and every $\epsilon > 0$ such that the Induction Hypothesis holds, there exists an M_ϵ^t such that $\|q_t^{m+1} - q_t^*\| \leq \epsilon$.

We will first extend the notation: let $q_t^{m,i}$ denote the vector q at states t after m algorithm iterations and $i \geq 1$ Bellman updates, such that $q_t^{m,i} = (\mathcal{T}^{\pi^{m+1}} q^{m,i-1})_t$, $q_t^{m,0} = q_t^m$ and finally $q_t^{m+1} = q_t^{m,k}$.

Second, we will extend the Induction Hypothesis:

Extended Induction Hypothesis: For every $\epsilon > 0$ there exist M_{t+1}^ϵ such that for all $m \geq M_{t+1}^\epsilon$ and $i \geq 0$ we have $\|q_{t+1}^{m,i} - q_{t+1}^*\| \leq \epsilon$, and $\|\pi_{t+1}^m \cdot q_{t+1}^{m,i} - \max_a q_{t+1}^*\| \leq \epsilon$.

The Base Case does not change, so we will proceed to *Step 1* in the Inductive Step. We need to show that there exists an M_t^ϵ such that $\|q_t^{m,i} - q_t^*\| \leq \epsilon$ for all $i \geq 0$ and $m \geq M_t^\epsilon$.

Let $\epsilon > 0$ and $m \geq M_t^\epsilon \geq M_{t+1}^\epsilon$.

First, for any $i \geq 1$:

$$\begin{aligned}
\|q_t^{m,i} - q_t^*\| &= \|\mathcal{R} + \gamma \mathcal{P}(\pi_{t+1}^{m+1} \cdot q_{t+1}^{m,i-1}) - q_t^*\| \\
&\leq \|\mathcal{P}\| \|\pi_{t+1}^{m+1} \cdot q_{t+1}^{m,i-1} - \max_a q_{t+1}^*\| \\
&\leq \epsilon
\end{aligned}$$

The first equality is the application of the Bellman Operator in line 4 in Algorithm 1 the ith time. The rest follows from Proof A.1 and the extended Induction Hypothesis.

Second, we need to show that this holds for $i = 0$ as well:

$$\|q_t^{m,0} - q_t^*\| = \|q_t^{m-1,k} - q_t^*\| \leq \|\pi_{t+1}^m \cdot q_{t+1}^{m-1,k-1} - \max_a q_{t+1}^*\| \leq \epsilon$$

The first equality is by definition, and the the first and second inequalities are by the same argumentation as above.

The rest of the proof need not be modified.

A.3 Proof for Theorem 2

Proof. Theorem 2

Similarly to the proof of Theorem 1 from Appendix A.1 (and A.2) we will prove by backwards induction from the terminal states s_H that the sequence $\lim_{m\to\infty}(\pi^m, q^m)$ induced by Algorithm 2 converges for any q^0, π^0, SAPIO $\mathcal{I}_1$, APIO $\mathcal{I}_2$ and $k \geq 1$. That is, for every $\epsilon > 0$ there exists a M_ϵ such that $\|q^m - q^*\| \leq \epsilon$ and $\|\pi^m \cdot q^m - \max_a q^*\| < \epsilon$ for all $m \geq M_\epsilon$, $q^0 \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and $\pi^0 \in \Pi$.

The proof follows directly from the proof in Appendix A.1. The base case is not modified - the qs converge immediately and the policy convergence is not influenced by the introduction of $\mathcal{I}_2$. The Induction Hypothesis need not be modified. In the inductive step, *Step 1* follows directly from the Induction Hypothesis, and *Step 2* need not be modified for the same reason the base case need not be modified. $\square$

B Additional Results

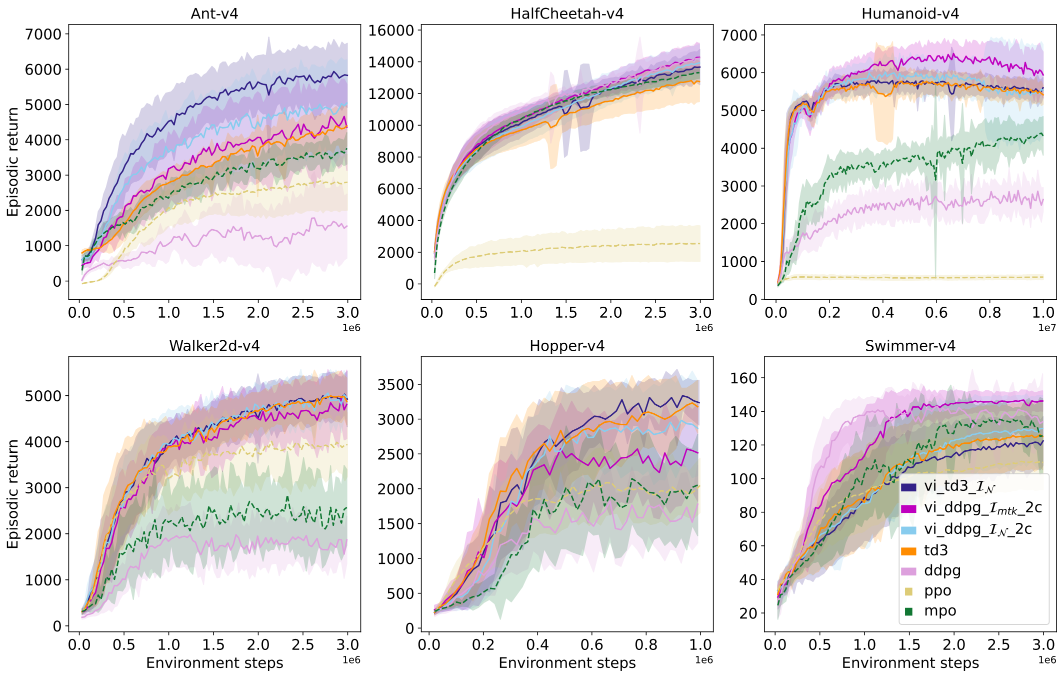

Figure 4: Mean and one standard deviation across 20 seeds in evaluation for VI-TD3 with $\mathcal{I}_\mathcal{N}$, VI-DDPG with $\mathcal{I}_\mathcal{N}$ and double-critics (2c), VI-DDPG with $\mathcal{I}_{mtk}$ and 2c, TD3, DDPG, PPO and MPO. The same agents (and seeds) as in Figure 1, which presented 2 x standard error.

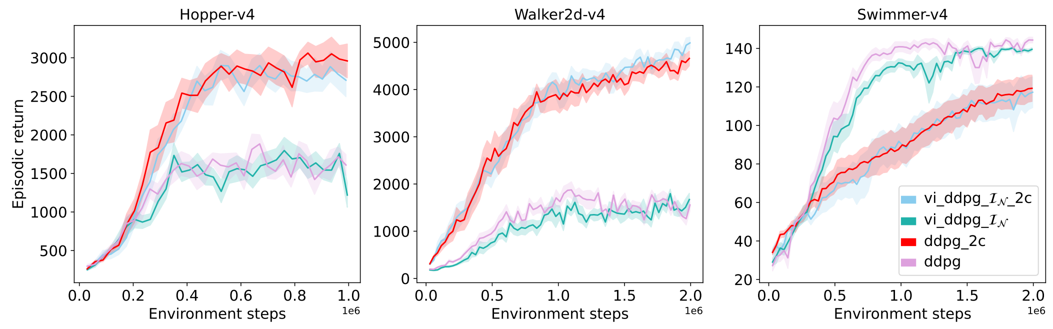

Figure 5: Mean and one standard error across 10 seeds in evaluation for VI-DDPG with $\mathcal{I}_\mathcal{N}$, $n = m = 16$ and baseline DDPG, both with double critics as well as without ($2c$) on additional environments.

C Experimental Details

C.1 Implementation Details

We have designed $\mathcal{I}_{mtk}$ and $\mathcal{I}_{\mathcal{N}}$ explicitly for DDPG and TD3, and our design choices are influenced by the design choices taken in those algorithms. For example, In line 1 in Algorithm 3 ($\mathcal{I}_{\mathcal{N}}$) we've tried both sampling from a clipped Gaussian policy as TD3 would, as well as from a squashed Gaussian in the manner used by SAC Haarnoja et al. (2018) with the same mean 0 and variance 1. In preliminary experiments the results were comparable, and in the final results we've used sampling from the squashed Gaussian. For VI-DDPG and $\mathcal{I}_{mtk}$ the distribution sampled from to generate n actions is the one proposed by TD3 (albeit for a different purpose): $\mathcal{N}(0,1).clip(\beta,\beta)$. We've used the same value of β for (VI-)TD3 and VI-DDPG with $\mathcal{I}_{mtk}$. We note that in principle that is a free hyperparameter that can be tuned separately, as the variance of the Gaussian. In addition, for VI-DDPG with $\mathcal{I}_{mtk}$ in order to preserve the deterministic value targets that are not based in sampling from a policy, the mean of the q predictions (computed with $\min_{\phi_i} q_{\phi_i}(s,a), i=\{1,2\}$ for each action) of the top k actions is used directly as the bootstrap in the value target.

C.2 Evaluation Method

We plot the mean and standard error for *evaluation curves* across multiple seeds. Evaluation curves are computed as follows: after every $n = 5000$ interactions with the environment, $m = 3$ evaluation episodes are ran with the latest network of the agent (actor and critic). The score of the agent is the return averaged across the m episodes. The actions in evaluation are chosen deterministically for TD3, DDPG and MPO with the mean of the policy (the agents use Gaussian policies). The evaluation episodes are not included in the agent's replay buffer or used for training, nor do they count towards the number of interactions. We observed similar or higher returns for all agents in evaluation compared to standard learning curves, with the exception of PPO. Since PPO learns an explicitly stochastic policy and performed much better in training than in deterministic evaluation, and since PPOs action selection in training does not contain any explicit exploration, we present the learning curves (which are equivalent to evaluation curves with sampled actions in this case), for PPO.

C.3 Hyperparameters Tuning

We rely on the popular code base CleanRL (Huang et al., 2022). CleanRL consists of implementations of many popular RL algorithms which are carefully tuned to match or improve upon the performance reported in either the original paper, or popularly used variations of the same algorithm. All baselines used in this paper included Mujoco in their tuning. For these reasons, we have not tuned any of the hyperparameters of any of the baselines ourselves. For VI-TD3/DDPG we used the same hyperparameters used by TD3 and DDPG respectively. The exception to this are the new hyperparameters n, m and n, k introduced by $\mathcal{I}_N$ and $\mathcal{I}_{mtk}$ respectively. For n, m we chose the largest values of n, m that did not excessively increase wall clock time as a result of parallelization bottlenecking in the hardware we have access to. These were $n = m = 16$. In $\mathcal{I}_{mtk}$ we chose a rather small $k = 4$ to favor the effect of the maximization operation over other effects in the results, and an n that enabled us to present ablations with a range of values of k. Figure 2 demonstrates that for different environments different tradeoffs with different values of k are likely to be beneficial.

C.4 Network Architectures

The experiments presented in this paper rely on standard, pretuned architectures for every baseline. As all agents were tuned over a set of environments that included Mujoco, we did not modify that architectures for which the agents were tuned. TD3 and DDPG used the same architecture. MPO had access to a larger network, while PPO to smaller. Architecture details are provided below.

DDPG and TD3:

Critic: 3 layer MLP of width 256 per layer, with ReLU activations on the hidden layers and tanh activation on the output layer.

Actor: 3 layer MLP of width 256 per layer, with ReLU activations on the hidden layers and no activation on the output layer.

PPO:

Critic: 3 layer MLP of width 64 per layer, with tanh activations on the hidden layers and no activation on the output layer.

Actor: Mean prediction: 3 layer MLP of width 64 per layer, with ReLU activations on the hidden layers and no activation on the output layer. Standard deviation prediction: 1 layer MLP of size action space with exponential activation. In addition, PPO used normalized rewards for training, normalized such that their exponential moving average has a fixed variance. As this is the best setting reported for PPO we did not modify it.

MPO:

Critic: 4 layer MLP of width 256 per layer, with Layer Normalization (Ba et al., 2016) followed by tanh activation on the first hidden layer, Exponential Linear Unit (ELU) activations on the following layers, and no activation on the output layer.

Actor: 3 hidden layers MLP of width 256 per layer followed by a width 256 layer each for mean and standard deviation. A Layer Normalization followed by tanh activation applied on the first hidden layer, followed by ELU activations, no activation on the mean layer and a softplus activation on the standard deviation layer.

C.5 Hyperparemeters

TD3		DDPG		MPO		PPO	
exploration noise	0.1	exploration noise	0.1	ϵ	0.1		
τ	0.005	τ	0.005	ϵ_μ	0.01	clip coef	0.2
noise clip	0.5			ϵ_Σ	10^{-6}	vf coef	0.5
				grad norm clip	40	max grad norm	0.5
Learning rate	0.0003	Learning rate	0.0003	Learning rate	0.0003	Learning rate	0.0003
γ	0.99	γ	0.99	γ	0.99	γ	0.99
Buffer size	10^6	Buffer size	10^6	Buffer size	10^6		
Batch size	256	Batch size	256	Batch size	256		
learning start	10^4	learning start	10^4	learning start	10^4		
evaluation frequency	5000	evaluation frequency	5000	evaluation frequency	5000		

C.6 Computation

The experiments were run on the internal Delft AI Cluster (DAIC) computation cluster, using any of the following GPU architectures: NVIDIA Quadro K2200, Tesla P100, GeForce GTX 1080 Ti, GeForce RTX 2080 Ti, Tesla V100S and Nvidia A-40. Each seed was ran on one GPU, and was given access to 6GB of RAM and 2 CPU cores. Total training wall-clock time averages were in the range of 0.5 to 3 hours per 10^6 environment steps, depending on GPU architecture, the baseline algorithm and the hyperparameters n, m, k for VI variations. On A-40 GPUs for example VI-TD3 with $\mathcal{I}_N$ and $n = m = 8$ wall-clock time averages were roughly 1.5 hours per 10^6 environment steps, while $n = m = 16$ roughly 2 hours. Baseline TD3 wall-clock time averages were roughly 1.25 hours per 10^6 environment steps. On the same hardware VI-DDPG with $\mathcal{I}_{mtk}$ and $n = 128$, $k = 4$ wall-clock time averages were roughly 1.25 hours per 10^6 environment steps. Baseline DDPG wall-clock time averages were roughly 0.8 hours per 10^6 environment steps without double critics, and roughly 1.1 with double critics. The total wall clock time over all experiments presented in this paper (main results, baselines and ablations) is estimated at around an average of 9560 wall-clock hours of the compute resources detailed above: 6440 for the main results, 1440 for the over-estimation ablations and 1680 for the hyperparameter ablations. This estimate is under the rough assumption of an average of 2 wall clock hours per 10^6 environment steps for every seed of every agent. Additional experiments that are not included in the paper were run in the process of implementation and testing.