On the Estimation Bias in Double Q-Learning

Zhizhou Ren[1†], **Guangxiang Zhu**[2], **Hao Hu**[2], **Beining Han**[2], **Jianglun Chen**[2], **Chongjie Zhang**[2]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign
[2]Institute for Interdisciplinary Information Sciences, Tsinghua University
zhizhour@illinois.edu, guangxiangzhu@outlook.com
{hu-h19, hbn18, chen-jl18}@mails.tsinghua.edu.cn
chongjie@tsinghua.edu.cn

Abstract

Double Q-learning is a classical method for reducing overestimation bias, which is caused by taking maximum estimated values in the Bellman operation. Its variants in the deep Q-learning paradigm have shown great promise in producing reliable value prediction and improving learning performance. However, as shown by prior work, double Q-learning is not fully unbiased and suffers from underestimation bias. In this paper, we show that such underestimation bias may lead to multiple non-optimal fixed points under an approximate Bellman operator. To address the concerns of converging to non-optimal stationary solutions, we propose a simple but effective approach as a partial fix for the underestimation bias in double Q-learning. This approach leverages an approximate dynamic programming to bound the target value. We extensively evaluate our proposed method in the Atari benchmark tasks and demonstrate its significant improvement over baseline algorithms.

1 Introduction

Value-based reinforcement learning with neural networks as function approximators has become a widely-used paradigm and shown great promise in solving complicated decision-making problems in various real-world applications, including robotics control (Lillicrap et al., 2016), molecular structure design (Zhou et al., 2019), and recommendation systems (Chen et al., 2018). Towards understanding the foundation of these successes, investigating algorithmic properties of deep-learning-based value function approximation has attracted a growth of attention in recent years (Van Hasselt et al., 2018; Fu et al., 2019; Achiam et al., 2019; Dong et al., 2020). One of the phenomena of interest is that Q-learning (Watkins, 1989) is known to suffer from overestimation issues, since it takes a maximum operator over a set of estimated action-values. Comparing with underestimated values, overestimation errors are more likely to be propagated through greedy action selections, which leads to an overestimation bias in value prediction (Thrun and Schwartz, 1993). This overoptimistic behavior of decision making has also been investigated in the literature of management science (Smith and Winkler, 2006) and economics (Thaler, 1988).

In deep Q-learning algorithms, one major source of value estimation errors comes from the optmization procedure. Although a deep neural network may have a sufficient expressiveness power to represent an accurate value function, the back-end optimization is hard to solve. As a result of computational considerations, stochastic gradient descent is almost the default choice for training deep Q-networks. As pointed out by Riedmiller (2005) and Van Hasselt et al. (2018), a mini-batch gradient update may have unpredictable effects on state-action pairs outside the training batch. The high variance of gradient estimation by such stochastic methods would lead to an unavoidable approximation error in value prediction, which cannot be eliminated by simply increasing sample size and

[†]Work done while Zhizhou was an undergraduate at Tsinghua University.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

network capacity. Through the maximum operator in the Q-learning paradigm, such approximation error would propagate and accumulate to form an overestimation bias. In practice, even if most benchmark environments are nearly deterministic (Brockman et al., 2016), a dramatic overestimation can be observed (Van Hasselt et al., 2016).

Double Q-learning (Van Hasselt, 2010) is a classical method to reduce the risk of overestimation, which is a specific variant of the double estimator (Stone, 1974) in the Q-learning paradigm. Instead of taking the greedy maximum values, it uses a second value function to construct an independent action-value evaluation as a cross validation. With proper assumptions, double Q-learning was proved to slightly underestimate rather than overestimate the maximum expected values (Van Hasselt, 2010). This technique has become a default implementation for stabilizing deep Q-learning algorithms (Hessel et al., 2018). In continuous control domains, a famous variant named clipped double Q-learning (Fujimoto et al., 2018) also shows great success in reducing the accumulation of errors in actor-critic methods (Haarnoja et al., 2018; Kalashnikov et al., 2018).

To understand algorithmic properties of double Q-learning and its variants, most prior work focus on the characterization of one-step estimation bias, i.e., the expected deviation from target values in a single step of Bellman operation (Lan et al., 2020; Chen et al., 2021). In this paper, we present a different perspective on how these one-step errors accumulate in stationary solutions. We first review a widely-used analytical model introduced by Thrun and Schwartz (1993) and reveal a fact that, due to the perturbation of approximation error, both double Q-learning and clipped double Q-learning have multiple approximate fixed points in this model. This result raises a concern that double Q-learning may easily get stuck in some local stationary regions and become inefficient in searching for the optimal policy. Motivated by this finding, we propose a novel value estimator, named *doubly bounded estimator*, that utilizes an abstracted dynamic programming as a lower bound estimation to rule out the potential non-optimal fixed points. The proposed method is easy to be combined with other existing techniques such as clipped double Q-learning. We extensively evaluate our approach on a variety of Atari benchmark tasks, and demonstrate significant improvement over baseline algorithms in terms of sample efficiency and convergence performance.

2 Background

Markov Decision Process (MDP; Bellman, 1957) is a classical framework to formalize an agent-environment interaction system which can be defined as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$. We use $\mathcal{S}$ and $\mathcal{A}$ to denote the state and action space, respectively. $P(s'|s,a)$ and $R(s,a)$ denote the transition and reward functions, which are initially unknown to the agent. γ is the discount factor. The goal of reinforcement learning is to construct a policy $\pi : \mathcal{S} \to \mathcal{A}$ maximizing cumulative rewards

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \;\middle|\; s_0 = s, s_{t+1} \sim P(\cdot|s_t, \pi(s_t))\right].$$

Another quantity of interest can be defined through the Bellman equation $Q^\pi(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[V^\pi(s')\right]$. The optimal value function Q^* corresponds to the unique solution of the Bellman optimality equation, $Q^*(s,a) = R(s,a) + \gamma\, \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\max_{a' \in \mathcal{A}} Q^*(s',a')\right]$. Q-learning algorithms are based on the Bellman optimality operator $\mathcal{T}$ stated as follows:

$$(\mathcal{T}Q)(s,a) = R(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)}\left[\max_{a' \in \mathcal{A}} Q(s',a')\right]. \tag{1}$$

By iterating this operator, value iteration is proved to converge to the optimal value function Q^*. To extend Q-learning methods to real-world applications, function approximation is indispensable to deal with a high-dimensional state space. Deep Q-learning (Mnih et al., 2015) considers a sample-based objective function and deploys an iterative optimization framework

$$\theta_{t+1} = \mathop{\arg\min}_{\theta \in \Theta} \mathop{\mathbb{E}}_{(s,a,r,s') \sim \mathcal{D}}\left[\left(r + \gamma \max_{a' \in \mathcal{A}} Q_{\theta_t}(s',a') - Q_\theta(s,a)\right)^2\right], \tag{2}$$

in which Θ denotes the parameter space of the value network, and $\theta_0 \in \Theta$ is initialized by some predetermined method. (s,a,r,s') is sampled from a data distribution $\mathcal{D}$ which is changing during exploration. With infinite samples and a sufficiently rich function class, the update rule stated in

Eq. (2) is asymptotically equivalent to applying the Bellman optimality operator $\mathcal{T}$, but the underlying optimization is usually inefficient in practice. In deep Q-learning, Eq. (2) is optimized by mini-batch gradient descent and thus its value estimation suffers from unavoidable approximation errors.

3 On the Effects of Underestimation Bias in Double Q-Learning

In this section, we will first revisit a common analytical model used by previous work for studying estimation bias (Thrun and Schwartz, 1993; Lan et al., 2020), in which double Q-learning is known to have underestimation bias. Based on this analytical model, we show that its underestimation bias could make double Q-learning have multiple fixed-point solutions under an approximate Bellman optimality operator. This result suggests that double Q-learning may have extra non-optimal stationary solutions under the effects of the approximation error.

3.1 Modeling Approximation Error in Q-Learning

In Q-learning with function approximation, the ground truth Bellman optimality operator $\mathcal{T}$ is approximated by a regression problem through Bellman error minimization (see Eq. (1) and Eq. (2)), which may suffer from unavoidable approximation errors. Following Thrun and Schwartz (1993) and Lan et al. (2020), we formalize underlying approximation errors as a set of random noises $e^{(t)}(s,a)$ on the regression outcomes:

$$Q^{(t+1)}(s,a) = (\mathcal{T}Q^{(t)})(s,a) + e^{(t)}(s,a). \tag{3}$$

In this model, double Q-learning (Van Hasselt, 2010) can be modeled by two estimator instances $\{Q_i^{(t)}\}_{i\in\{1,2\}}$ with separated noise terms $\{e_i^{(t)}\}_{i\in\{1,2\}}$. For simplification, we introduce a policy function $\pi^{(t)}(s) = \arg\max_a Q_1^{(t)}(s,a)$ to override the state value function as follows:

$$\begin{aligned} V^{(t)}(s) &= Q_2^{(t)}\left(s,\ \pi^{(t)}(s) = \arg\max_{a\in\mathcal{A}} Q_1^{(t)}(s,a)\right), \\ \forall i\in\{1,2\},\quad Q_i^{(t+1)}(s,a) &= \underbrace{R(s,a) + \gamma\mathbb{E}_{s'}\left[V^{(t)}(s')\right]}_{\text{target value}} + \underbrace{e_i^{(t)}(s,a)}_{\text{approximation error}}. \end{aligned} \tag{4}$$

A minor difference of Eq. (4) from the definition of double Q-learning given by Van Hasselt (2010) is the usage of a unified target value $V^{(t)}(s')$ for both two estimators. This simplification does not affect the derived implications, and is also implemented by advanced variants of double Q-learning (Fujimoto et al., 2018; Lan et al., 2020).

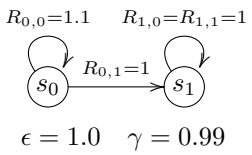
To establish a unified framework for analysis, we use a stochastic operator $\widetilde{\mathcal{T}}$ to denote the Q-iteration procedure $Q^{(t+1)} \leftarrow \widetilde{\mathcal{T}}Q^{(t)}$, e.g., the updating rules stated as Eq. (3) and Eq. (4). We call such an operator $\widetilde{\mathcal{T}}$ as a *stochastic Bellman operator*, since it approximates the ground truth Bellman optimality operator $\mathcal{T}$ and carries some noises due to approximation errors. Note that, as shown in Eq. (4), the target value can be constructed only using the state-value function $V^{(t)}$. We can define the stationary point of state-values $V^{(t)}$ as the fixed point of a stochastic Bellman operator $\widetilde{\mathcal{T}}$.

Definition 1 (Approximate Fixed Points)**.** *Let $\widetilde{\mathcal{T}}$ denote a stochastic Bellman operator, such as what are stated in Eq.* (3) *and Eq.* (4)*. A state-value function V is regarded as an approximate fixed point under a stochastic Bellman operator $\widetilde{\mathcal{T}}$ if it satisfies $\mathbb{E}[\widetilde{\mathcal{T}}V] = V$, where $\widetilde{\mathcal{T}}V$ denotes the output state-value function while applying the Bellman operator $\widetilde{\mathcal{T}}$ on V.*

Remark. In prior work (Thrun and Schwartz, 1993), value estimation bias is defined by expected one-step deviation with respect to the ground truth Bellman operator, i.e., $\mathbb{E}[(\widetilde{\mathcal{T}}V^{(t)})(s)] - (\mathcal{T}V^{(t)})(s)$. The approximate fixed points stated in Definition 1 characterizes the accumulation of estimation biases in stationary solutions.

In Appendix A.2, we will prove the existence of such fixed points as the following statement.

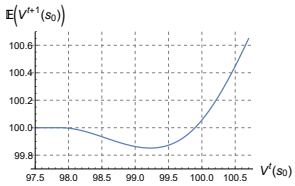
Proposition 1. *Assume the probability density functions of the noise terms $\{e(s,a)\}$ are continuous. The stochastic Bellman operators defined by Eq.* (3) *and Eq.* (4) *must have approximate fixed points in arbitrary MDPs.*



(a) A simple construction

$V(s_0)$	$V(s_1)$	$\tilde{\pi}(a_0\|s_0)$
100.162	100.0	62.2%
101.159	100.0	92.9%
110.0	100.0	100.0%

(b) Numerical solutions of fixed points



(c) Visualizing non-monotonicity

Figure 1: (a) A simple infinite-horizon MDP where double Q-learning stated as (4) has multiple approximate fixed points. $R_{i,j}$ is a shorthand of $R(s_i, a_j)$. (b) The numerical solutions of the fixed points produced by double Q-learning in the MDP presented above. $\tilde{\pi}$ denotes the expected policy generated by the corresponding fixed point under the perturbation of noise $e(s, a)$. A formal description of $\tilde{\pi}$ refers to Definition 2 in Appendix A.3. (c) The relation between the input state-value $V^{(t)}(s_0)$ and the expected output state-value $\mathbb{E}[V^{(t+1)}(s_0)]$ generated by double Q-learning in the constructed MDP, in which we assume $V^{(t)}(s_1) = 100$.

3.2 Existence of Multiple Approximate Fixed Points in Double Q-Learning Algorithms

Given the definition of the approximate fixed point, a natural question is whether such kind of fixed points are unique or not. Recall that the optimal value function Q^* is the unique solution of the Bellman optimality equation, which is the foundation of Q-learning algorithms. However, in this section, we will show that, under the effects of the approximation error, the approximate fixed points of double Q-learning may not be unique.

An Illustrative Example. Figure 1a presents a simple MDP in which double Q-learning stated as Eq. (4) has multiple approximate fixed points. For simplicity, this MDP is set to be fully deterministic and contains only two states. All actions on state s_1 lead to a self-loop and produce a unit reward signal. On state s_0, the result of executing action a_0 is a self-loop with a slightly larger reward signal than choosing action a_1 which leads to state s_1. The only challenge for decision making in this MDP is to distinguish the outcomes of executing action a_0 and a_1 on state s_0. To make the example more accessible, we assume the approximation errors $\{e^{(t)}(s, a)\}_{t,s,a}$ are a set of independent random noises sampled from a uniform distribution *Uniform*$(-\epsilon, \epsilon)$. This simplification is also adopted by Thrun and Schwartz (1993) and Lan et al. (2020) in case studies. Here, we select the magnitude of noise as $\epsilon = 1.0$ and the discount factor as $\gamma = 0.99$ to balance the scale of involved amounts.

Considering to solve the equation $\mathbb{E}[\widetilde{\mathcal{T}}V] = V$ according to the definition of the approximate fixed point (see Definition 1), the numerical solutions of such fixed points are presented in Table 1b. There are three different fixed point solutions. The first thing to notice is that the optimal fixed point V^* is retained in this MDP (see the last row of Table 1b), since the noise magnitude $\epsilon = 1.0$ is much smaller than the optimality gap $Q^*(s_0, a_0) - Q^*(s_0, a_1) = 10$. The other two fixed points are non-optimal and very close to $Q(s_0, a_0) \approx Q(s_0, a_1) = 100$. Intuitively, under the perturbation of approximation error, the agent cannot identify the correct maximum-value action for policy improvement in these situations, which is the cause of such non-optimal fixed points. To formalize the implications, we would present a sufficient condition for the existence of multiple extra fixed points.

Mathematical Condition. Note that the definition of a stochastic Bellman operator can be decoupled to two parts: (1) Computing target values $\mathcal{T}Q^{(t)}$ according to the given MDP; (2) Perform an imprecise regression and some specific computations to obtain $Q^{(t+1)}$. The first part is defined by the MDP, and the second part is the algorithmic procedure. From this perspective, we can define the input of a learning algorithm as a set of ground truth target values $\{(\mathcal{T}Q^{(t)})(s, a)\}_{s,a}$. Based on this notation, a sufficient condition for the existence of multiple fixed points is stated as follows.

Proposition 2. *Let $f_s\left(\{(\mathcal{T}Q)(s, a)\}_{a\in\mathcal{A}}\right) = \mathbb{E}[(\widetilde{\mathcal{T}}V)(s)]$ denote the expected output value of a learning algorithm on state s. Assume $f_s(\cdot)$ is differentiable. If the algorithmic procedure $f_s(\cdot)$ satisfies Eq.* (5)*, there exists an MDP such that it has multiple approximate fixed points.*

$$\exists s,\ \exists i,\ \exists X \in \mathbb{R}^{|\mathcal{A}|}, \quad \frac{\partial}{\partial x_i} f_s(X) > 1, \tag{5}$$

where $X = \{x_i\}_{i=1}^{|\mathcal{A}|}$ denotes the input of the function f_s.

The proof of Proposition 2 is deferred to Appendix A.4. This proposition suggests that, in order to determine whether a Q-learning algorithm may have multiple fixed points, we need to check whether its expected output values could change dramatically with a slight alter of inputs. Considering the constructed MDP as an example, Figure 1c visualizes the relation between the input state-value $V^{(t)}(s_0)$ and the expected output state-value $\mathbb{E}[V^{(t+1)}(s_0)]$ while assuming $V^{(t)}(s_1) = 100$ has converged to its stationary point. The minima point of the output value is located at the situation where $V^{(t)}(s_0)$ is slightly smaller than $V^{(t)}(s_1)$, since the expected policy derived by $\widetilde{\mathcal{T}}V^{(t)}$ will have a remarkable probability to choose sub-optimal actions. This local minima suffers from the most dramatic underestimation among the whole curve, and the underestimation will eventually vanish as the value of $V^{(t)}(s_0)$ increases. During this process, a large magnitude of the first-order derivative could be found to meet the condition stated in Eq. (5).

Remark. In Appendix A.5, we show that clipped double Q-learning, a popular variant of double Q-learning, has multiple fixed points in an MDP slightly modified from Figure 1a. Besides, the condition presented in Proposition 2 does not hold in standard Q-learning that uses a single maximum operator (see Proposition 6 in Appendix). It remains an open question whether standard Q-learning with overestimation bias has multiple fixed points.

3.3 Diagnosing Non-Optimal Fixed Points

In this section, we first characterize the properties of the extra non-optimal fixed points of double Q-learning in the analytical model. And then, we discuss its connections to the literature of stochastic optimization, which motivates our proposed algorithm in section 4.

Underestimated Solutions. The first notable thing is that, the non-optimal fixed points of double Q-learning would not overestimate the true maximum values. More specifically, every fixed-point solution could be characterized as the ground truth value of some stochastic policy as the follows:

Proposition 3 (Fixed-Point Characterization)**.** *Assume the noise terms e_1 and e_2 are independently generated in the double estimator stated in Eq.* (4)*. Every approximate fixed point V is equal to the ground truth value function $V^{\tilde{\pi}}$ with respect to a stochastic policy $\tilde{\pi}$.*

The proof of Proposition 3 is deferred to Appendix A.3. In addition, the corresponding stochastic policy $\tilde{\pi}$ can be interpreted as

$$\tilde{\pi}(a|s) = \mathbb{P}\left[a = \arg\max_{a' \in \mathcal{A}} \left(\underbrace{R(s,a') + \gamma \mathbb{E}_{s'}\left[V(s')\right]}_{(\mathcal{T}Q)(s,a')} + e(s,a') \right)\right],$$

which is the expected policy generated by the corresponding fixed point along with the random noise $e(s, a')$. This stochastic policy, named as *induced policy*, can provide a snapshot to infer how the agent behaves and evolves around these approximate fixed points. To deliver intuitions, we provide an analogical explanation in the context of optimization as the following arguments.

Analogy with Saddle Points. Taking the third column of Table 1b as an example, due to the existence of the approximation error, the induced policy $\tilde{\pi}$ suffers from a remarkable uncertainty in determining the best action on state s_0. Around such non-optimal fixed points, the greedy action selection may be disrupted by approximation error and deviate from the correct direction for policy improvement. These approximate fixed points are not necessary to be strongly stationary solutions but may seriously hurt the learning efficiency. If we imagine each iteration of target updating as a step of "*gradient update*" for Bellman error minimization, the non-optimal fixed points would refer to the concept of *saddle points* in the context of optimization. As stochastic gradient may be trapped in saddle points, Bellman operation with approximation error may get stuck around non-optimal approximate fixed points. Please refer to section 5.1 for a visualization of a concrete example.

Escaping from Saddle Points. In the literature of non-convex optimization, the most famous approach to escaping saddle points is *perturbed gradient descent* (Ge et al., 2015; Jin et al., 2017). Recall that, although gradient directions are ambiguous around saddle points, they are not strongly convergent solutions. Some specific perturbation mechanisms with certain properties could help to make the optimizer to escape non-optimal saddle points. Although these methods cannot be directly applied to double Q-learning since the Bellman operation is not an exact gradient descent, it motivates us to construct a specific perturbation as guidance. In section 4, we would introduce a perturbed target updating mechanism that uses an external value estimation to rule out non-optimal fixed points.

4 Doubly Bounded Q-Learning through Abstracted Dynamic Programming

As discussed in the last section, the underestimation bias of double Q-learning may lead to multiple non-optimal fixed points in the analytical model. A major source of such underestimation is the inherent approximation error caused by the imprecise optimization. Motivated by the literature of escaping saddle points, we introduce a novel method, named *Doubly Bounded Q-learning*, which integrates two different value estimators to reduce the negative effects of underestimation.

4.1 Algorithmic Framework

As discussed in section 3.3, the geometry property of non-optimal approximate fixed points of double Q-learning is similar to that of saddle points in the context of non-convex optimization. The theory of escaping saddle points suggests that, a well-shaped perturbation mechanism could help to remove non-optimal saddle points from the landscape of optimization (Ge et al., 2015; Jin et al., 2017). To realize this brief idea in the specific context of iterative Bellman error minimization, we propose to integrate a second value estimator using different learning paradigm as an external auxiliary signal to rule out non-optimal approximate fixed points of double Q-learning. To give an overview, we first revisit two value estimation paradigms as follows:

1. **Bootstrapping Estimator:** As the default implementation of most temporal-difference learning algorithms, the target value y^{Boots} of a transition sample (s_t, a_t, r_t, s_{t+1}) is computed through bootstrapping the latest value function back-up $V_{\theta_{\text{target}}}$ parameterized by θ_{target} on the successor state s_{t+1} as follows:
$$y^{\text{Boots}}_{\theta_{\text{target}}}(s_t, a_t) = r_t + \gamma V_{\theta_{\text{target}}}(s_{t+1}),$$
where the computations of $V_{\theta_{\text{target}}}$ differ in different algorithms (e.g., different variants of double Q-learning).
2. **Dynamic Programming Estimator:** Another approach to estimating state-action values is applying dynamic programming in an abstracted MDP (Li et al., 2006) constructed from the collected dataset. By utilizing a state aggregation function $\phi(s)$, we could discretize a complex environment to a manageable tabular MDP. The reward and transition functions of the abstracted MDP are estimated through the collected samples in the dataset. An alternative target value y^{DP} is computed as:
$$y^{\text{DP}}(s_t, a_t) = r_t + \gamma V^*_{\text{DP}}(\phi(s_{t+1})), \tag{6}$$
where V^*_{DP} corresponds to the optimal value function of the abstracted MDP.

The advantages and bottlenecks of these two types of value estimators lie in different aspects of error controlling. The generalizability of function approximators is the major strength of the *bootstrapping estimator*, but on the other hand, the hardness of the back-end optimization would cause considerable approximation error and lead to the issues discussed in section 3. The tabular representation of the *dynamic programming estimator* would not suffer from systematic approximation error during optimization, but its performance relies on the accuracy of state aggregation and the sampling error in transition estimation.

Doubly Bounded Estimator. To establish a trade-off between the considerations in the above two value estimators, we propose to construct an integrated estimator, named *doubly bounded estimator*, which takes the maximum values over two different basis estimation methods:
$$y^{\text{DB}}_{\theta_{\text{target}}}(s_t, a_t) = \max\left\{y^{\text{Boots}}_{\theta_{\text{target}}}(s_t, a_t),\ y^{\text{DP}}(s_t, a_t)\right\}. \tag{7}$$
The targets values $y^{\text{DB}}_{\theta_{\text{target}}}$ would be used in training the parameterized value function Q_θ by minimizing
$$L(\theta; \theta_{\text{target}}) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim D}\left(Q_\theta(s_t, a_t) - y^{\text{DB}}_{\theta_{\text{target}}}(s_t, a_t)\right)^2,$$
where D denotes the experience buffer. Note that, this estimator maintains two value functions using different data structures. Q_θ is the major value function which is used to generate the behavior policy for both exploration and evaluation. V_{DP} is an auxiliary value function computed by the abstracted dynamic programming, which is stored in a discrete table. The only functionality of V_{DP} is computing the auxiliary target value y^{DP} used in Eq. (7) during training.

Remark. The name "*doubly bounded*" refers to the following intuitive motivation: Assume both basis estimators, y^{Boots} and y^{DP}, are implemented by their conservative variants and do not tend to overestimate values. The doubly bounded target value $y^{\text{DB}}(s_t, a_t)$ would become a good estimation if either of basis estimator provides an accurate value prediction on the given state-action pair (s_t, a_t). The outcomes of abstracted dynamic programming could help the bootstrapping estimator to escape the non-optimal fixed points of double Q-learning. The function approximator used by the bootstrapping estimator could extend the generalizability of discretization-based state aggregation. The learning procedure could make progress if either of estimators can identify the correct direction for policy improvement.

Practical Implementation. To make sure the dynamic programming estimator does not overestimate the true values, we implement a tabular version of batch-constrained Q-learning (BCQ; Fujimoto et al., 2019) to obtain a conservative estimation. The abstracted MDP is constructed by a simple state aggregation based on low-resolution discretization, i.e., we only aggregate states that cannot be distinguished by visual information. We follow the suggestions given by Fujimoto et al. (2019) and Liu et al. (2020) to prune the unseen state-action pairs in the abstracted MDP. The reward and transition functions of remaining state-action pairs are estimated through the average of collected samples. A detailed description is deferred to Appendix B.5.

4.2 Underlying Bias-Variance Trade-Off

In general, there is no existing approach can completely eliminate the estimation bias in Q-learning algorithm. Our proposed method also focuses on the underlying bias-variance trade-off.

Provable Benefits on Variance Reduction. The algorithmic structure of the proposed *doubly bounded estimator* could be formalized as a stochastic Bellman operator $\widetilde{\mathcal{T}}^{\text{DB}}$:

$$(\widetilde{\mathcal{T}}^{\text{DB}}V)(s) = \max\left\{(\widetilde{\mathcal{T}}^{\text{Boots}}V)(s),\ V^{\text{DP}}(s)\right\}, \tag{8}$$

where $\widetilde{\mathcal{T}}^{\text{Boots}}$ is the stochastic Bellman operator corresponding to the back-end bootstrapping estimator (e.g., Eq. (4)). V^{DP} is an arbitrary deterministic value estimator such as using abstracted dynamic programming. The benefits on variance reduction can be characterized as the following proposition.

Proposition 4. *Given an arbitrary stochastic operator $\widetilde{\mathcal{T}}^{Boots}$ and a deterministic estimator V^{DP},*

$$\forall V,\ \forall s \in \mathcal{S}, \quad Var[(\widetilde{\mathcal{T}}^{DB}V)(s)] \leq Var[(\widetilde{\mathcal{T}}^{Boots}V)(s)],$$

where $(\widetilde{\mathcal{T}}^{DB}V)(s)$ is defined as Eq. (8)*.*

The proof of Proposition 4 is deferred to Appendix A.6. The intuition behind this statement is that, with a deterministic lower bound cut-off, the variance of the outcome target values would be reduced, which may contribute to improve the stability of training.

Trade-Off between Different Biases. In general, the proposed *doubly bounded estimator* does not have a rigorous guarantee for bias reduction, since the behavior of abstracted dynamic programming depends on the properties of the tested environments and the accuracy of state aggregation. In the most unfavorable case, if the dynamic programming component carries a large magnitude of error, the lower bounded objective would propagate high-value errors to increase the risk of overestimation. To address these concerns, we propose to implement a conservative approximate dynamic programming as discussed in the previous section. The asymptotic behavior of batch-constrained Q-learning does not tend to overestimate extrapolated values (Liu et al., 2020). The major risk of the dynamic programming module is induced by the state aggregation, which refers to a classical problem (Li et al., 2006). The experimental analysis in section 5.2 demonstrate that, the error carried by abstracted dynamic programming is acceptable, and our proposed method definitely works well in most benchmark tasks.

5 Experiments

In this section, we conduct experiments to demonstrate the effectiveness of our proposed method. We first perform our method on the two-state MDP example discussed in previous sections to visualize its algorithmic effects. And then, we set up a performance comparison on Atari benchmark with several baseline algorithms. A detailed description of experiment settings is deferred to Appendix B.

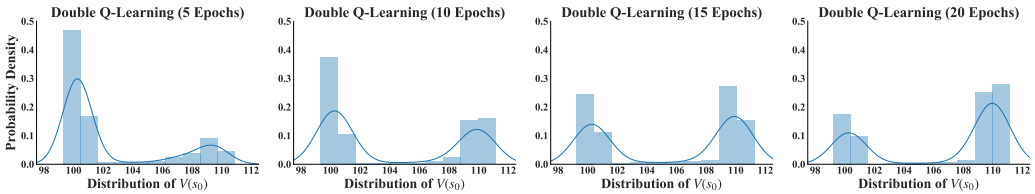


Figure 2: Visualization of the probability density of $V(s_0)$ learned by double Q-learning. In this section, all plotted distributions are estimated by 10^3 runs with random seeds. We utilize `seaborn.distplot` package to plot the kernel density estimate curves and histogram bins.

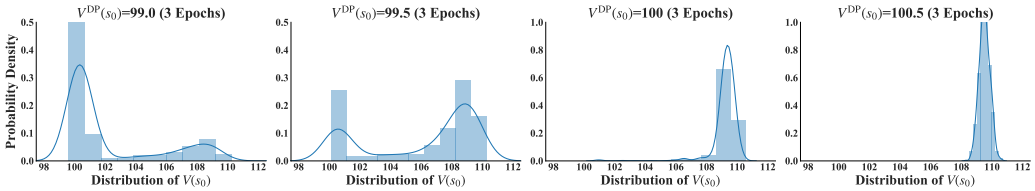


Figure 3: Visualization of the probability density of $V(s_0)$ learned by doubly bounded Q-learning with an imprecise dynamic programming planner. $V^{\mathrm{DP}}(s_0)$ denotes the value estimation given by dynamic programming, which serves as a lower bound in our method.

5.1 Tabular Experiments on a Two-State MDP

To investigate whether the empirical behavior of our proposed method matches it design purpose, we compare the behaviors of double Q-learning and our method on the two-state MDP example presented in Figure 1a. In this MDP, the non-optimal fixed points would get stuck in $V(s_0) \approx 100$ and the optimal solution has $V^*(s_0) = 110$. In this tabular experiment, we implement double Q-learning and our algorithm with table-based Q-value functions. The Q-values are updated by iteratively applying Bellman operators as what are presented in Eq. (4) and Eq. (8). To approximate practical scenarios, we simulate the approximation error by entity-wise Gaussian noises $\mathcal{N}(0, 0.5)$. Since the stochastic process induced by such noises suffers from high variance, we perform soft update $Q^{(t+1)} = (1-\alpha)Q^{(t)} + \alpha(\widetilde{\mathcal{T}}Q^{(t)})$ to make the visualization clear, in which α refers to learning rate in practice. We consider $\alpha = 10^{-2}$ for all experiments presented in this section. In this setting, we denote one epoch as $\frac{1}{\alpha(1-\gamma)} = 10^4$ iterations. A detailed description for the tabular implementation is deferred to Appendix B.2.

We investigate the performance of our method with an imprecise dynamic programming module, in which we only apply lower bound for state s_0 with values $V^{\mathrm{DP}}(s_0) \in \{99.0, 99.5, 100.0, 100.5\}$. The experiment results presented in Figure 2 and Figure 3 support our claims in previous sections:

1. The properties of non-optimal fixed points are similar to that of saddle points. These extra fixed points are relatively stationary region but not truly static. A series of lucky noises can help the agent to escape from non-optimal stationary regions, but this procedure may take lots of iterations. As shown in Figure 2, double Q-learning may get stuck in non-optimal solutions and it can escape these non-optimal regions by a really slow rate. After 20 epochs (i.e., $2 \cdot 10^5$ iterations), there are nearly a third of runs cannot find the optimal solution.
2. The design of our doubly bounded estimator is similar to a perturbation on value learning. As shown in Figure 3, when the estimation given by dynamic programming is slightly higher than the non-optimal fixed points, such as $V_{\mathrm{DP}}(s_0) = 100.5$, it is sufficient to help the agent escape from non-optimal stationary solutions. A tricky observation is that $V_{\mathrm{DP}}(s_0) = 100$ also seems to work. It is because cutting-off a zero-mean noise would lead to a slight overestimation, which makes the actual estimated value of $V^{\mathrm{DP}}(s_0)$ to be a larger value.

5.2 Performance Comparison on Atari Benchmark

To demonstrate the superiority of our proposed method, *Doubly Bounded Q-Learning through Abstracted Dynamic Programming* (DB-ADP), we compare with six variants of deep Q-networks as baselines, including DQN (Mnih et al., 2015), double DQN (DDQN; Van Hasselt et al., 2016),

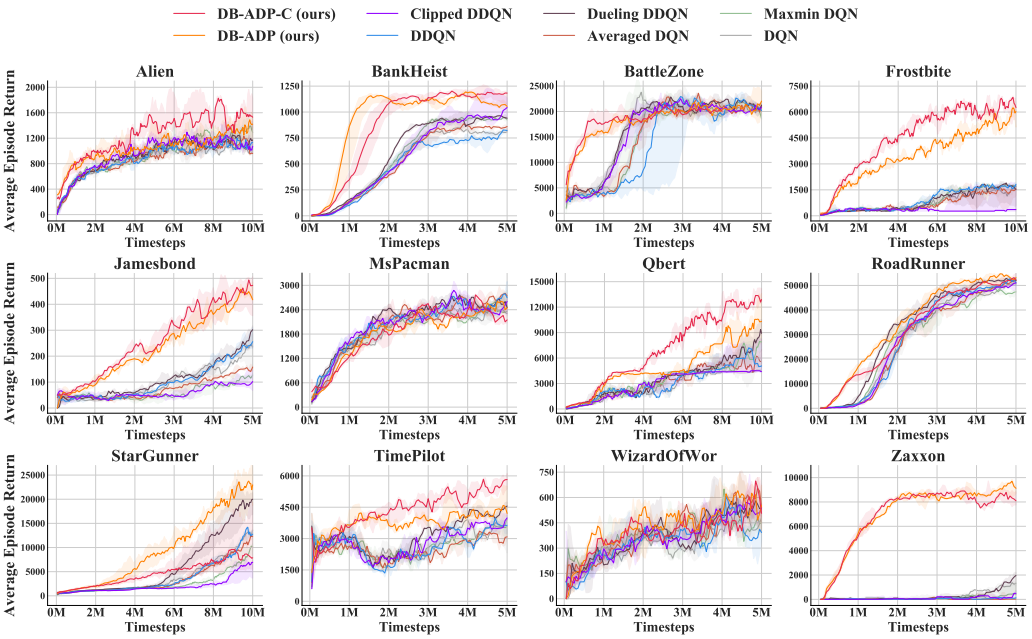


Figure 4: Learning curves on a suite of Atari benchmark tasks. DB-ADP and DB-ADP-C refer to our proposed approach built upon double Q-learning and clipped double Q-learning, respectively.

dueling DDQN (Wang et al., 2016), averaged DQN (Anschel et al., 2017), maxmin DQN (Lan et al., 2020), and clipped double DQN adapted from Fujimoto et al. (2018). Our proposed doubly bounded target estimation y^{DB} is built upon two types of bootstrapping estimators that have clear incentive of underestimation, i.e., double Q-learning and clipped double Q-learning. We denote these two variants as DB-ADP-C and DB-ADP according to our proposed method with or without using clipped double Q-learning.

As shown in Figure 4, the proposed doubly bounded estimator has great promise in bootstrapping the performance of double Q-learning algorithms. The improvement can be observed both in terms of sample efficiency and final performance. Another notable observation is that, although clipped double Q-learning can hardly improve the performance upon Double DQN, it can significantly improve the performance through our proposed approach in most environments (i.e., DB-ADP-C vs. DB-ADP in Figure 4). This improvement should be credit to the conservative property of clipped double Q-learning (Fujimoto et al., 2019) that may reduce the propagation of the errors carried by abstracted dynamic programming.

5.3 Variance Reduction on Target Values

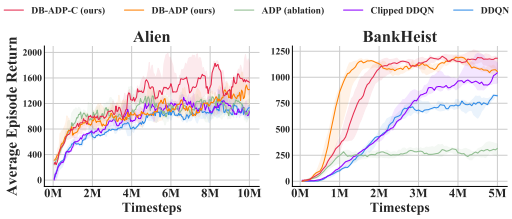
To support the theoretical claims in Proposition 4, we conduct an experiment to demonstrate the ability of doubly bounded estimator on variance reduction. We evaluate the standard deviation of the target values with respect to training networks using different sequences of training batches. Table 5a presents the evaluation results on our proposed methods and baseline algorithms. The †-version corresponds to an ablation study, where we train the network using our proposed approach but evaluate the target values computed by bootstrapping estimators, i.e., using the target value formula of double DQN or clipped double DQN. As shown in Table 5a, the standard deviation of target values is significantly reduced by our approaches, which matches our theoretical analysis in Proposition 4. It demonstrates a strength of our approach in improving training stability. A detailed description of the evaluation metric is deferred to Appendix B.

5.4 An Ablation Study on the Dynamic Programming Module

To support the claim that the dynamic programming estimator is an auxiliary module to improving the strength of double Q-learning, we conduct an ablation study to investigate the individual performance of dynamic programming. Formally, we exclude Bellman error minimization from the training

TASK NAME	DB-ADP-C	DB-ADP-C†	CDDQN
ALIEN	**0.006**	0.008	0.010
BANKHEIST	**0.009**	0.010	0.010
QBERT	**0.008**	0.010	0.011
TASK NAME	DB-ADP	DB-ADP†	DDQN
ALIEN	0.008	0.009	0.012
BANKHEIST	**0.009**	0.011	0.013
QBERT	0.009	0.011	0.012

(a) Variance reduction on target values



(b) Ablation study on dynamic programming

Figure 5: (a) Evaluating the standard deviation of target values w.r.t. different training batches. The presented amounts are normalized by the value scale of corresponding runs. "†" refers to ablation studies. (b) An ablation study on the individual performance of the dynamic programming module.

procedure and directly optimize the following objective to distill the results of dynamic programming into a generalizable parametric agent:

$$L^{\mathrm{ADP}}(\theta) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim D}\left[\left(Q_\theta(s_t,a_t) - y^{\mathrm{DP}}(s_t,a_t)\right)^2\right],$$

where $y^{\mathrm{DP}}(s_t, a_t)$ denotes the target value directly by dynamic programming. As shown in Figure 5b, without integrating with the bootstrapping estimator, the abstracted dynamic programming itself cannot outperform deep Q-learning algorithms. It remarks that, in our proposed framework, two basis estimators are supplementary to each other.

6 Related Work

Correcting the estimation bias of Q-learning is a long-lasting problem which induces a series of approaches (Lee et al., 2013; D'Eramo et al., 2016; Chen, 2020; Zhang and Huang, 2020), especially following the methodology of double Q-learning (Zhang et al., 2017; Zhu and Rigotti, 2021). The most representative algorithm, clipped double Q-learning (Fujimoto et al., 2018), has become the default implementation of most advanced actor-critic algorithms (Haarnoja et al., 2018). Based on clipped double Q-learning, several methods have been investigated to reduce the its underestimation and achieve promising performance (Ciosek et al., 2019; Li and Hou, 2019). Other recent advances usually focus on using ensemble methods to further reduce the error magnitude (Lan et al., 2020; Kuznetsov et al., 2020; Chen et al., 2021). Statistical analysis of double Q-learning is also an active area (Weng et al., 2020; Xiong et al., 2020) that deserves future studies.

Besides the variants of double Q-learning, using the softmax operator in Bellman operations is another effective approach to reduce the effects of approximation error (Fox et al., 2016; Asadi and Littman, 2017; Song et al., 2019; Kim et al., 2019; Pan et al., 2019). The characteristic of our approach is the usage of an approximate dynamic programming. Our analysis would provide a theoretical support for memory-based approaches, such as episodic control (Blundell et al., 2016; Pritzel et al., 2017; Lin et al., 2018; Zhu et al., 2020; Hu et al., 2021), which are usually designed for near-deterministic environments. Instead of using an explicit planner, Fujita et al. (2020) adopts the trajectory return as a lower bound for value estimation. This simple technique also shows promise in improving the efficiency of continuous control with clipped double Q-learning.

7 Conclusion

In this paper, we reveal an interesting fact that, under the effects of approximation error, double Q-learning may have multiple non-optimal fixed points. The main cause of such non-optimal fixed points is the underestimation bias of double Q-learning. Regarding this issue, we provide some analysis to characterize what kind of Bellman operators may suffer from the same problem, and how the agent may behave around these fixed points. To address the potential risk of converging to non-optimal solutions, we propose doubly bounded Q-learning to reduce the underestimation in double Q-learning. The main idea of this approach is to leverage an abstracted dynamic programming as a second value estimator to rule out non-optimal fixed points. The experiments show that the proposed method has shown great promise in improving both sample efficiency and convergence performance, which achieves a significant improvement over baselines algorithms.

Acknowledgments and Disclosure of Funding

The authors would like to thank Kefan Dong for insightful discussions. This work is supported in part by Science and Technology Innovation 2030 – "New Generation Artificial Intelligence" Major Project (No. 2018AAA0100904), a grant from the Institute of Guo Qiang, Tsinghua University, and a grant from Turing AI Institute of Nanjing.

References

Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*, pages 176–185. PMLR, 2017.

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252, 2017.

Richard Bellman. Dynamic programming. *Princeton University Press*, 89:92, 1957.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Luitzen Egbertus Jan Brouwer. Über abbildung von mannigfaltigkeiten. *Mathematische annalen*, 71 (1):97–115, 1911.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G Bellemare. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

Gang Chen. Decorrelated double q-learning. *arXiv preprint arXiv:2006.06956*, 2020.

Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1187–1196, 2018.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, pages 1785–1796, 2019.

Kefan Dong, Yuping Luo, Tianhe Yu, Chelsea Finn, and Tengyu Ma. On the expressivity of neural networks for deep reinforcement learning. In *International Conference on Machine Learning*, pages 2627–2637. PMLR, 2020.

Carlo D'Eramo, Marcello Restelli, and Alessandro Nuara. Estimating maximum expected value through gaussian approximation. In *International Conference on Machine Learning*, pages 1032–1040. PMLR, 2016.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 202–211, 2016.

Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning*, pages 2021–2030, 2019.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

Yasuhiro Fujita, Kota Uenishi, Avinash Ummadisingu, Prabhat Nagarajan, Shimpei Masuda, and Mario Ynocente Castro. Distributed reinforcement learning of targeted grasping with active vision for mobile manipulators. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9712–9719. IEEE, 2020.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on learning theory*, pages 797–842. PMLR, 2015.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Hao Hu, Jianing Ye, Zhizhou Ren, Guangxiang Zhu, and Chongjie Zhang. Generalizable episodic memory for deep reinforcement learning. *arXiv preprint arXiv:2103.06469*, 2021.

Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1181–1189. Citeseer, 2015.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pages 1724–1732. PMLR, 2017.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*. PMLR, 2018.

Richard M Karp and Michael O Rabin. Efficient randomized pattern-matching algorithms. *IBM journal of research and development*, 31(2):249–260, 1987.

Seungchan Kim, Kavosh Asadi, Michael Littman, and George Konidaris. Deepmellow: removing the need for a target network in deep q-learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2733–2739, 2019.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, 2020.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *International Conference on Learning Representations*, 2020.

Donghun Lee, Boris Defourny, and Warren B Powell. Bias-corrected q-learning to control max-operator bias in q-learning. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 93–99. IEEE, 2013.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4:5, 2006.

Zhunan Li and Xinwen Hou. Mixing update q-value for deep reinforcement learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2019.

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. In *International Conference on Learning Representations*, 2016.

Zichuan Lin, Tianqi Zhao, Guangwen Yang, and Lintao Zhang. Episodic memory deep q-networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2433–2439, 2018.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. In *Advances in Neural Information Processing Systems*, 2020.

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

Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13(1):103–130, 1993.

Ling Pan, Qingpeng Cai, Qi Meng, Wei Chen, Longbo Huang, and Tie-Yan Liu. Reinforcement learning with dynamic boltzmann softmax updates. *arXiv preprint arXiv:1903.05926*, 2019.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2827–2836, 2017.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

James E Smith and Robert L Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

Zhao Song, Ron Parr, and Lawrence Carin. Revisiting the softmax bellman operator: New benefits and new perspective. In *International Conference on Machine Learning*, pages 5916–5925, 2019.

Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

Richard H Thaler. The winner's curse. *The Journal of Economic Perspectives*, 2(1):191–202, 1988.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

Hado Van Hasselt. Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2094–2100, 2016.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016.

Chris Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Wentao Weng, Harsh Gupta, Niao He, Lei Ying, and R Srikant. The mean-squared error of double q-learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Huaqing Xiong, Lin Zhao, Yingbin Liang, and Wei Zhang. Finite-time analysis for double q-learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Huihui Zhang and Wu Huang. Deep reinforcement learning with adaptive combined critics. 2020.

Zongzhang Zhang, Zhiyuan Pan, and Mykel J Kochenderfer. Weighted double q-learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3455–3461, 2017.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.

Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic reinforcement learning with associative memory. In *International Conference on Learning Representations*, 2020.

Rong Zhu and Mattia Rigotti. Self-correcting q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11185–11192, 2021.

A Omitted Statements and Proofs

A.1 The Relation between Estimation Bias and Approximate Fixed Points

An intuitive characterization of such fixed point solutions is considering one-step estimation bias with respect to the maximum expected value, which is defined as

$$\mathcal{E}(\widetilde{\mathcal{T}}, V, s) = \mathbb{E}[(\widetilde{\mathcal{T}}V)(s)] - (\mathcal{T}V)(s), \tag{9}$$

where $(\mathcal{T}V)(s)$ corresponds to the precise state value after applying the ground truth Bellman operation. The amount of estimation bias $\mathcal{E}$ characterizes the deviation from the standard Bellman operator $\mathcal{T}$, which can be regarded as imaginary rewards in fixed point solutions.

Every approximate fixed point solution under a stochastic Bellman operator can be characterized as the optimal value function in a modified MDP where only the reward function is changed.

Proposition 5. *Let $\widetilde{V}$ denote an approximation fixed point under a stochastic Bellman operator $\widetilde{\mathcal{T}}$. Define a modified MDP $\widetilde{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, P, R + \widetilde{R}, \gamma \rangle$ based on $\mathcal{M}$, where the extra reward term is defined as*

$$\widetilde{R}(s,a) = \mathcal{E}(\widetilde{\mathcal{T}}, \widetilde{V}, s) = \mathbb{E}[(\widetilde{\mathcal{T}}\widetilde{V})(s)] - (\mathcal{T}\widetilde{V})(s),$$

where $\mathcal{E}$ is the one-step estimation bias defined in Eq. (9)*. Then $\widetilde{V}$ is the optimal state-value function of the modified MDP $\widetilde{\mathcal{M}}$.*

Proof. Define a value function $\widetilde{Q}$ based on $\widetilde{V}$, $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\widetilde{Q}(s,a) = R(s,a) + \widetilde{R}(s,a) + \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)}[\widetilde{V}(s')].$$

We can verify $\widetilde{Q}$ is consistent with $\widetilde{V}$, $\forall s \in \mathcal{S}$,

$$\begin{aligned}
\widetilde{V}(s) &= \mathbb{E}[(\widetilde{\mathcal{T}}\widetilde{V})(s)] \\
&= \mathbb{E}[(\widetilde{\mathcal{T}}\widetilde{V})(s)] - (\mathcal{T}\widetilde{V})(s) + \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)}[\widetilde{V}(s')] \right) \\
&= \max_{a \in \mathcal{A}} \left(R(s,a) + \widetilde{R}(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)}[\widetilde{V}(s')] \right) \\
&= \max_{a \in \mathcal{A}} \widetilde{Q}(s,a).
\end{aligned}$$

Let $\mathcal{T}_{\widetilde{\mathcal{M}}}$ denote the Bellman operator of $\widetilde{\mathcal{M}}$. We can verify $\widetilde{Q}$ satisfies Bellman optimality equation to prove the given statement, $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned}
(\mathcal{T}_{\widetilde{\mathcal{M}}}\widetilde{Q})(s,a) &= R(s,a) + \widetilde{R}(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[\max_{a' \in \mathcal{A}} \widetilde{Q}(s',a') \right] \\
&= R(s,a) + \widetilde{R}(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)}[\widetilde{V}(s')] \\
&= \widetilde{Q}(s,a).
\end{aligned}$$

Thus we can see $\widetilde{V}$ is the solution of Bellman optimality equation in $\widetilde{\mathcal{M}}$. □

A.2 The Existence of Approximate Fixed Points

The key technique for proving the existence of approximate fixed points is Brouwer's fixed point theorem.

Lemma 1. *Let $B = [-L, -L]^d$ denote a d-dimensional bounding box. For any continuous function $f : B \to B$, there exists a fixed point x such that $f(x) = x \in B$.*

Proof. It refers to a special case of Brouwer's fixed point theorem (Brouwer, 1911). □

Lemma 2. *Let $\widetilde{\mathcal{T}}$ denote the stochastic Bellman operator defined by Eq.* (3)*. There exists a real range L, $\forall V \in [L, -L]^{|\mathcal{S}|}$, $\mathbb{E}[\widetilde{\mathcal{T}}V] \in [L, -L]^{|\mathcal{S}|}$.*

Proof. Let R_{max} denote the range of the reward function for MDP $\mathcal{M}$. Let R_e denote the range of the noisy term. Formally,

$$R_{\text{max}} = \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} |R(s,a)|,$$

$$R_e = \max_{s\in\mathcal{S}} \mathbb{E}\left[\max_{a\in\mathcal{A}} |e(s,a)|\right].$$

Note that the L_∞-norm of state value functions satisfies $\forall V \in \mathbb{R}^{|\mathcal{S}|}$,

$$\|\mathbb{E}[\widetilde{\mathcal{T}}V]\|_\infty \leq R_{\text{max}} + R_e + \gamma\|V\|_\infty.$$

We can construct the range $L = (R_{\text{max}} + R_e)/(1-\gamma)$ to prove the given statement. □

Lemma 3. *Let $\widetilde{\mathcal{T}}$ denote the stochastic Bellman operator defined by Eq.* (4)*. There exists a real range L, $\forall V \in [L, -L]^{|\mathcal{S}|}$, $\mathbb{E}[\widetilde{\mathcal{T}}V] \in [L, -L]^{|\mathcal{S}|}$.*

Proof. Let R_{max} denote the range of the reward function for MDP $\mathcal{M}$. Formally,

$$R_{\text{max}} = \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} |R(s,a)|.$$

Note that the L_∞-norm of state value functions satisfies $\forall V \in \mathbb{R}^{|\mathcal{S}|}$,

$$\|\mathbb{E}[\widetilde{\mathcal{T}}V]\|_\infty \leq R_{\text{max}} + \gamma\|V\|_\infty.$$

We can construct the range $L = R_{\text{max}}/(1-\gamma)$ to prove the given statement. □

Proposition 1. *Assume the probability density functions of the noise terms $\{e(s,a)\}$ are continuous. The stochastic Bellman operators defined by Eq.* (3) *and Eq.* (4) *must have approximate fixed points in arbitrary MDPs.*

Proof. Let $f(V) = \mathbb{E}[\widetilde{\mathcal{T}}V]$ denote the expected return of a stochastic Bellman operation. This function is continuous because all involved formulas only contain elementary functions. The given statement is proved by combining Lemma 1, 2, and 3. □

A.3 The Induced Policy of Double Q-Learning

Definition 2 (Induced Policy)**.** *Given a target state-value function V, its induced policy $\tilde{\pi}$ is defined as a stochastic action selection according to the value estimation produced by a stochastic Bellman operation $\tilde{\pi}(a|s) =$*

$$\mathbb{P}\left[a = \arg\max_{a'\in\mathcal{A}}\left(\underbrace{R(s,a') + \gamma\mathbb{E}_{s'}\left[V(s')\right]}_{(\mathcal{T}Q)(s,a')} + e_1(s,a')\right)\right],$$

where $\{e_1(s,a)\}_{s,a}$ are drawing from the same noise distribution as what is used by double Q-learning stated in Eq. (4)*.*

Proposition 3 (Fixed-Point Characterization)**.** *Assume the noise terms e_1 and e_2 are independently generated in the double estimator stated in Eq.* (4)*. Every approximate fixed point V is equal to the ground truth value function $V^{\tilde{\pi}}$ with respect to a stochastic policy $\tilde{\pi}$.*

Proof. Let V denote an approximate fixed point under the stochastic Bellman operator $\widetilde{\mathcal{T}}$ defined by Eq. (4). By plugging the definition of the induced policy into the stochastic operator of double Q-learning, we can get

$$\begin{aligned} V(s) &= \mathbb{E}[\widetilde{\mathcal{T}}V(s)] \\ &= \mathbb{E}\left[(\widetilde{\mathcal{T}}Q_2)\left(s,\ \arg\max_{a\in\mathcal{A}}(\widetilde{\mathcal{T}}Q_1)(s,a)\right)\right] \end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}\left[(\widetilde{\mathcal{T}}Q_2)\left(s,\ \arg\max_{a\in\mathcal{A}}\left((\mathcal{T}Q_1)(s,a)+e_1(s,a)\right)\right)\right] \\
&= \mathop{\mathbb{E}}_{a\sim\tilde{\pi}(\cdot|s)}\left[(\widetilde{\mathcal{T}}Q_2)(s,a)\right] \\
&= \mathop{\mathbb{E}}_{a\sim\tilde{\pi}(\cdot|s)}\left[R(s,a)+\gamma\mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)}V(s')+e_2(s,a)\right] \\
&= \mathop{\mathbb{E}}_{a\sim\tilde{\pi}(\cdot|s)}\left[R(s,a)+\gamma\mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)}V(s')\right],
\end{aligned}
$$

which matches the Bellman expectation equation. □

As shown by this proposition, the estimated value of a non-optimal fixed point is corresponding to the value of a stochastic policy, which revisits the incentive of double Q-learning to underestimate true maximum values.

A.4 A Sufficient Condition for Multiple Fixed Points

Proposition 2. *Let $f_s\left(\{(\mathcal{T}Q)(s,a)\}_{a\in\mathcal{A}}\right) = \mathbb{E}[(\widetilde{\mathcal{T}}V)(s)]$ denote the expected output value of a learning algorithm on state s. Assume $f_s(\cdot)$ is differentiable. If the algorithmic procedure $f_s(\cdot)$ satisfies Eq.* (5)*, there exists an MDP such that it has multiple approximate fixed points.*

$$
\exists s,\ \exists i,\ \exists X\in\mathbb{R}^{|\mathcal{A}|},\quad \frac{\partial}{\partial x_i}f_s(X)>1, \tag{5}
$$

where $X=\{x_i\}_{i=1}^{|\mathcal{A}|}$ denotes the input of the function f_s.

Proof. Suppose f_s is a function satisfying the given condition.

Let $x_i=\bar{x}$ and X denote the corresponding point satisfying Eq. (5).

Let $g(x)$ denote the value of f_s while only changing the input value of x_i to x. Note that, according to Eq. (5), we have $g'(\bar{x})>1$.

Since f_s is differentiable, we can find a small region $\bar{x}_L<\bar{x}<\bar{x}_R$ around $\bar{x}$ such that $\forall x\in[\bar{x}_L,\bar{x}_R],\ g'(x)>1$. And then, we have $g(\bar{x}_R)-g(\bar{x}_L)>\bar{x}_R-\bar{x}_L$.

Consider to construct an MDP with only one state (see Figure 6a as an example). We can use the action corresponding to x_i to construct a self-loop transition with reward r. All other actions lead to a termination signal and an immediate reward, where the immediate rewards correspond to other components of X. By setting the discount factor as $\gamma=\frac{\bar{x}_R-\bar{x}_L}{g(\bar{x}_R)-g(\bar{x}_L)}<1$ and the reward as $r=\bar{x}_L-\gamma g(\bar{x}_L)=\bar{x}_R-\gamma g(\bar{x}_R)$, we can find both $\bar{x}_L$ and $\bar{x}_R$ are solutions of the equation $x=r+\gamma g(x)$, in which $g(\bar{x}_L)$ and $g(\bar{x}_R)$ correspond to two fixed points of the constructed MDP. □

Proposition 6. *Vanilla Q-learning does not satisfy the condition stated in Eq.* (5) *in any MDPs.*

Proof. In vanilla Q-learning, the expected state-value after one iteration of updates is

$$
\begin{aligned}
\mathbb{E}[V^{(t+1)}(s)] &= \mathbb{E}\left[\max_{a\in\mathcal{A}}Q^{(t+1)}(s,a)\right] \\
&= \mathbb{E}\left[\max_{a\in\mathcal{A}}\left((\mathcal{T}Q^{(t)})(s,a)+e^{(t)}(s,a)\right)\right] \\
&= \int_{w\in\mathbb{R}^{|\mathcal{A}|}}\mathbb{P}[e^{(t)}=w]\left(\max_{a\in\mathcal{A}}\left((\mathcal{T}Q^{(t)})(s,a)+w(s,a)\right)\right)\mathrm{d}w.
\end{aligned}
$$

Denote

$$
f(\mathcal{T}Q^{(t)},w)=\max_{a\in\mathcal{A}}\left((\mathcal{T}Q^{(t)})(s,a)+w(s,a)\right).
$$

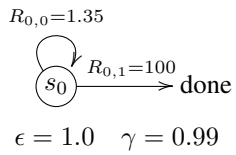
Note that the value of $f(\mathcal{T}Q^{(t)},w)$ is 1-Lipschitz w.r.t. each entry of $\mathcal{T}Q^{(t)}$. Thus we have $\mathbb{E}[V^{(t+1)}(s)]$ is also 1-Lipschitz w.r.t. each entry of $\mathcal{T}Q^{(t)}$. The condition stated in Eq. (5) cannot hold in any MDPs. □

A.5 A Bad Case for Clipped Double Q-Learning

The stochastic Bellman operator corresponding to clipped double Q-learning is stated as follows.

$$\begin{aligned} \forall i \in \{1,2\}, \quad Q_i^{(t+1)}(s,a) &= R(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a)} \left[V^{(t)}(s') \right] + e_i^{(t)}(s,a), \\ V^{(t)}(s) &= \min_{i \in \{1,2\}} Q_i^{(t)} \left(s,\ \arg\max\nolimits_{a \in \mathcal{A}} Q_1^{(t)}(s,a) \right). \end{aligned} \tag{10}$$

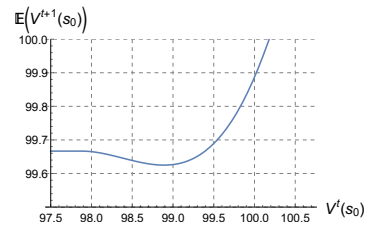
An MDP where clipped double Q-learning has multiple fixed points is illustrated as Figure 6.



(a) A simple construction

$V(s_0)$	$\tilde{\pi}(a_0\|s_0)$
100.491	83.1%
101.833	100.0%
101.919	100.0%

(b) Numerical solutions of fixed points



(c) Visualizing non-monotonicity

Figure 6: (a) A simple MDP where clipped double Q-learning stated as Eq. (10) has multiple approximate fixed points. $R_{i,j}$ is a shorthand of $R(s_i, a_j)$. (b) The numerical solutions of the fixed points produced by clipped double Q-learning in the MDP presented above. (c) The relation between the input state-value $V^{(t)}(s_0)$ and the expected output state-value $\mathbb{E}[V^{(t+1)}(s_0)]$ generated by clipped double Q-learning in the constructed MDP.

A.6 Provable Benefits on Variance Reduction

Lemma 4. *Let x denote a random variable. y denotes a constant satisfying $y \leq \mathbb{E}[x]$. Then, Var$[\max\{x, y\}] \leq$ Var$[x]$.*

Proof. Let $\mu = \mathbb{E}[x]$ denote the mean of random variable x. Consider

$$\begin{aligned} \text{Var}[x] &= \int_{-\infty}^{\infty} \mathbb{P}[x = t](t-\mu)^2 \mathrm{d}t \\ &= \int_{-\infty}^{\mu} \mathbb{P}[x = t](t-\mu)^2 \mathrm{d}t + \int_{\mu}^{\infty} \mathbb{P}[x = t](t-\mu)^2 \mathrm{d}t \\ &\geq \int_{-\infty}^{\mu} \mathbb{P}[x = t](\mu - \max\{t, y\})^2 \mathrm{d}t + \int_{\mu}^{\infty} \mathbb{P}[x = t](t-\mu)^2 \mathrm{d}t \\ &= \int_{-\infty}^{\infty} \mathbb{P}[x = t](\mu - \max\{t, y\})^2 \mathrm{d}t \\ &\geq \int_{-\infty}^{\infty} \mathbb{P}[x = t](\mathbb{E}[\max\{x, y\}] - \max\{t, y\})^2 \mathrm{d}t \qquad (11) \\ &= \text{Var}[\max\{x, y\}], \end{aligned}$$

where Eq. (11) holds since the true average point $\mathbb{E}[\max\{x, y\}]$ leads to the minimization of the variance formula. □

Proposition 4. *Given an arbitrary stochastic operator $\widetilde{\mathcal{T}}^{Boots}$ and a deterministic estimator V^{DP},*

$$\forall V,\ \forall s \in \mathcal{S}, \quad \textit{Var}[(\widetilde{\mathcal{T}}^{DB} V)(s)] \leq \textit{Var}[(\widetilde{\mathcal{T}}^{Boots} V)(s)],$$

where $(\widetilde{\mathcal{T}}^{DB} V)(s)$ is defined as Eq. (8)*.*

Proof. When $V^{\text{DP}}(s)$ is larger than all possible output values of $(\widetilde{\mathcal{T}}^{\text{Boots}} V)(s)$, the given statement directly holds, since Var $\left[\max\left\{(\widetilde{\mathcal{T}}^{\text{Boots}} V)(s),\ V^{\text{DP}}(s)\right\}\right]$ would be equal to zero.

Otherwise, when $\mathbb{E}[V^{\text{Boots}}(s)]$ is smaller than $V^{\text{DP}}(s)$, we can first apply a lower bound cut-off by value $y = \mathbb{E}[V^{\text{Boots}}(s)] < V^{\text{DP}}(s)$, which gives $\text{Var}[\max\{y, V^{\text{Boots}}(s)\}] \leq \text{Var}[V^{\text{Boots}}(s)]$ and $\mathbb{E}[\max\{y, V^{\text{Boots}}(s)\}] > \mathbb{E}[V^{\text{Boots}}(s)]$. By repeating this procedure several times, we can finally get $V^{\text{DP}}(s) \leq \mathbb{E}[\max\{y, V^{\text{Boots}}(s)\}]$ and close the proof. □

B Experiment Settings and Implementation Details

B.1 Evaluation Settings

Probability Density of $V(s_0)$ on Two-State MDP. All plotted distributions are estimated by 10^3 runs with random seeds, i.e., suffering from different noises in approximate Bellman operations. When applying doubly bounded Q-learning, we only set lower bound for $V(s_0)$ and do nothing for $V(s_1)$. The initial Q-values are set to 100.0 as default. The main purpose of setting this initial value is avoiding the trivial case where doubly bounded Q-learning learns $V(s_0)$ faster than $V(s_1)$ so that does not get trapped in non-optimal fixed points at all.

Cumulative Rewards on Atari Games. All curves presented in this paper are plotted from the median performance of 5 runs with random initialization. To make the comparison more clear, the curves are smoothed by averaging 10 most recent evaluation points. The shaded region indicates 60% population around median. The evaluation is processed in every 50000 timesteps. Every evaluation point is averaged from 5 trajectories. Following Castro et al. (2018), the evaluated policy is combined with a 0.1% random execution.

Standard Deviation of Target Values. The evaluation of target value standard deviations contains the following steps:

1. Every entry of the table presents the median performance of 5 runs with random network initialization.
2. For each run, we first perform 10^6 regular training steps to collect an experience buffer and obtain a basis value function.
3. We perform a target update operation, i.e., we use the basis value function to construct frozen target values. And then we train the current values for 8000 batches as the default training configuration to make sure the current value nearly fit the target.
4. We sample a batch of transitions from the replay buffer as the testing set. We focus on the standard deviations of value predictions on this testing set.
5. And then we collect 8000 checkpoints. These checkpoints are collected by training 8000 randomly sampled batches successively, i.e., we collect one checkpoint after perform each batch updating.
6. For each transition in the testing set, we compute the standard deviation over all checkpoints. We average the standard deviation evaluation of each single transition as the evaluation of the given algorithm.

B.2 Tabular Implementation of Doubly Bounded Q-Learning

Algorithm 1 Tabular Simulation of Doubly Bounded Q-Learning

Initialize $V^{(0)}$
for $t = 1, 2, \cdots, T-1$ **do**
  for $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    $Q^{(t)}(s, a) \leftarrow \mathbb{E}[r + \gamma V^{(t)}(s')]$
  $Q_1 \leftarrow Q^{(t)} + \text{noise}$
  $Q_2 \leftarrow Q^{(t)} + \text{noise}$
  for $s \in \mathcal{S}$ **do**
    $V^{(t+1)}(s) \leftarrow (1-\alpha)V^{(t)}(s) + \alpha \max\{Q_2(s, \arg\max_a Q_1(s, a)), V_{\text{DP}}(s)\}$
return $V^{(T)}$

B.3 DQN-Based Implementation of Double Bounded Q-Learning

Algorithm 2 DQN-Based Implementation of Doubly Bounded Q-Learning

Initialize θ, θ^{target}
$\mathcal{D} \leftarrow \emptyset$
for $t = 1, 2, \cdots, T-1$ **do**
  Collect one step of transition (s_t, a_t, r_t, s_{t+1}) using the policy given by Q_θ
  Store (s_t, a_t, r_t, s_{t+1}) in to replay buffer $\mathcal{D}$
  Update the DP value on state s_t, ▷ Prioritized sweeping (Moore and Atkeson, 1993)

$$V^{DP}(s_t) \leftarrow \max_{a:(s_t,a)\in\mathcal{D}} \frac{1}{|\mathcal{D}(s_t,a)|} \sum_{(\hat{r},\hat{s}')\in\mathcal{D}(s_t,a)} \left(\hat{r} + \gamma V^{DP}(\hat{s}')\right)$$

  Perform one step of gradient update for θ, ▷ Estimated by mini-batch

$$\theta \leftarrow \theta - \alpha \nabla \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}} \left[\left(r + \gamma \max \left\{ \max_{a'\in\mathcal{A}} Q_{\theta^{\text{target}}}(s',a'), V^{\text{DP}}(s') \right\} - Q_\theta(s,a) \right)^2 \right],$$

  where α denotes the learning rate
  if t mod $H = 0$ **then** ▷ Update target values
    $\theta^{\text{target}} \leftarrow \theta$
    Update the DP values over the whole replay buffer $\mathcal{D}$
return θ

B.4 Implementation Details and Hyper-Parameters

Our experiment environments are based on the standard Atari benchmark tasks supported by OpenAI Gym (Brockman et al., 2016). All baselines and our approaches are implemented using the same set of hyper-parameters suggested by Castro et al. (2018). More specifically, all algorithm investigated in this paper use the same set of training configurations.

- Number of `noop` actions while starting a new episode: 30;
- Number of stacked frames in observations: 4;
- Scale of rewards: clipping to $[-1, 1]$;
- Buffer size: 10^6;
- Batch size: 32;
- Start training: after collecting 20000 transitions;
- Training frequency: 4 timesteps;
- Target updating frequency: 8000 timesteps;
- ϵ decaying: from 1.0 to 0.01 in the first 250000 timesteps;
- Optimizer: Adam with $\varepsilon = 1.5 \cdot 10^{-4}$;
- Learning rate: $0.625 \cdot 10^{-4}$.

For ensemble-based methods, Averaged DQN and Maxmin DQN, we adopt 2 ensemble instances to ensure all architectures presented in this paper use comparable number of trainable parameters.

All networks are trained using a single GPU and a single CPU core.

- GPU: GeForce GTX 1080 Ti
- CPU: Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz

In each run of experiment, 10M steps of training can be completed within 36 hours.

B.5 Abstracted Dynamic Programming for Atari Games

State Aggregation. We consider a simple discretization to construct the state abstraction function $\phi(\cdot)$ used in Eq. (6). We first follow the standard Atari pre-processing proposed by Mnih et al. (2015) to rescale each RGB frame to an 84×84 luminance map, and the observation is constructed as a stack of 4 recent luminance maps. We round the each pixel to 256 possible integer intensities and use a standard static hashing, Rabin-Karp Rolling Hashing (Karp and Rabin, 1987), to set up the table for storing V_{DP}. In the hash function, we use two large prime numbers ($\approx 10^9$) and select their primary roots as the rolling basis. From this perspective, each image would be randomly projected to an integer within a range ($\approx 10^{18}$).

Conservative Action Pruning. To obtain a conservative value estimation, we follow the suggestions given by Fujimoto et al. (2019) and Liu et al. (2020) to prune the unseen state-action pairs in the abstracted MDP. Formally, in the dynamic programming module, we only allow the agent to perform state-action pairs that have been collected at least once in the experience buffer. The reward and transition functions of remaining state-action pairs are estimated through the average of collected samples.

Computation Acceleration. Note that the size of the abstracted MDP is growing as the exploration. Regarding computational considerations, we adopt the idea of *prioritized sweeping* (Moore and Atkeson, 1993) to accelerate the computation of tabular dynamic programming. In addition to periodically applying the complete Bellman operator, we perform extra updates on the most recent visited states, which would reduce the total number of operations to obtain an acceptable estimation. Formally, our dynamic programming module contains two branches of updates:

1. After collecting each complete trajectory, we perform a series of updates along the collected trajectory. In the context of *prioritized sweeping*, we assign the highest priorities to the most recent visited states.
2. At each iteration of target network switching, we perform one iteration of value iteration to update the whole graph.

Connecting to Parameterized Estimator. Finally, the results of abstracted dynamic programming would be delivered to the deep Q-learning as Eq. (7). Note that the constructed doubly bounded target value $y^{\text{DB}}_{\theta_{\text{target}}}$ is only used to update the parameterized value function Q_θ and would not affect the computation in the abstracted MDP.

C Visualization of Non-Optimal Approximate Fixed Points

As shown in Figure 4, the sample efficiency of our methods dramatically outperform double DQN in an Atari game called Zaxxon. We visualize a screen snapshot of a scenario that the double DQN agent gets stuck in for a long time (see Figure 7), which may refer to a practical example of non-optimal fixed points.

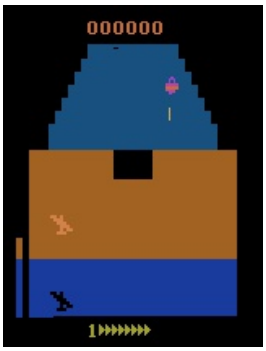


Figure 7: Visualization of a non-optimal approximate fixed point in Zaxxon.

In this scenario, the agent (i.e., the aircraft at the left-bottom corner) needs to pass through a gate (i.e., the black cavity in the center of the screen). Otherwise it will hit the wall and crash. The double DQN usually gets stuck in a policy that cannot pass through this gate, but our method can find this solution very fast. We provide a possible interpretation here. In this case, the agent may find the path to pass the gate by random exploration, but when it passes the gate, it will be easier to be attacked by enemies. Due to the lack of data, the value estimation may suffer from a large amount of error or variance in these states. It would make the agent be confused on the optimal action. By contrast, the dynamic programming planner used by double bounded Q-learning is non-parametric that can correctly reinforce the best experience.

D Additional Experiments on Random MDPs

In addition to the two-state toy MDP, we evaluate our method on a benchmark task of random MDPs.

Experiment Setting. Following Jiang et al. (2015), we generate 1000 random MDPs with 10 states and 5 actions from a distribution. For each state-action pair (s, a), the transition function is determined by randomly choosing 5 non-zero entries, filling these 5 entries with values uniformly drawn from $[0, 1]$, and finally normalizing it to $P(\cdot|s, a)$. i.e.,

$$P(s'|s,a) = \frac{\widehat{P}(s'|s,a)}{\sum_{s''\in\mathcal{S}} \widehat{P}s''|s,a)} \quad \text{where} \quad \widehat{P}(s'|s,a) \sim \text{Uniform}(0,1).$$

The reward values $R(s, a)$ are independently and uniformly drawn from $[0, 1]$. We consider the discount factor $\gamma = 0.99$ for all MDPs. Regarding the simulation of approximation error, we consider the same setting as the experiments in section 5.1. We consider Gaussian noise $\mathcal{N}(0, 0.5)$ and use soft updates with $\alpha = 10^{-2}$.

Approximate Dynamic Programming. Our proposed method uses a dynamic programming module to construct a second value estimator as a lower bound estimation. In this experiment, we consider the dynamic programming is done in an approximate MDP model, where the transition function of each state-action pair (s, a) is estimated by K samples. We aim to investigate the dependency of our performance on the quality of MDP model approximation.

Experiment Results. We evaluate two metrics on 1000 random generated MDPs: (1) the value estimation error $(V - V^*)$, and (2) the performance of the learned policy $(V^\pi - V^*)$ where $\pi = \arg\max Q$. All amounts are evaluated after 50000 iterations. The experiment results are shown as follows:

Evaluation Metric	Estimation Error $(V - V^*)$	Policy Performance $(V^\pi - V^*)$
double Q-learning	-16.36	-0.68
Q-learning	33.15	-0.73
ours ($K = 10$)	0.53	-0.62
ours ($K = 20$)	0.54	-0.62
ours ($K = 30$)	0.37	-0.61

Table 1: Evaluation on random MDPs.

We conclude the experiment results in Table 1 by two points:

1. As shown in the above table, Q-learning would significantly overestimate the values, and double Q-learning would significantly underestimate the values. Comparing to baseline algorithms, the value estimation of our proposed method is much more reliable. Note that our method only cuts off low-value noises, which may lead to a trend of overestimation. This overestimation would not propagate and accumulate during learning, since the first estimator y^{Boots} has incentive to underestimate values. The accumulation of overestimation errors cannot exceed the bound of y^{DP} too much. As shown in experiments, the overestimation error would be manageable.
2. The experiments show that, although the quality of value estimation of Q-learning and double Q-learning may suffer from significant errors, they can actually produce polices with acceptable performance. This is because the transition graphs of random MDPs are strongly connected, which induce a dense set of near-optimal polices. When the tasks have branch structures, the quality of value estimation would have a strong impact on the decision making in practice.

E Additional Experiments for Variance Reduction on Target Values

Table 2: Evaluating the standard deviation of target values w.r.t. different sequences of training batches. The presented amounts are normalized by the value scale of corresponding runs. "†" refers to ablation studies where we train the network using our proposed approach but evaluate the target values computed by bootstrapping estimators.

TASK NAME	DB-ADP-C	DB-ADP-C†	CDDQN	DB-ADP	DB-ADP†	DDQN
ALIEN	**0.006**	0.008	0.010	0.008	0.009	0.012
BANKHEIST	**0.009**	0.010	0.010	**0.009**	0.011	0.013
BATTLEZONE	**0.012**	**0.012**	0.031	0.014	0.014	0.036
FROSTBITE	**0.006**	0.007	0.012	0.007	0.007	0.015
JAMESBOND	0.010	0.010	**0.009**	0.010	0.011	0.010
MSPACMAN	**0.007**	0.009	0.012	**0.007**	0.009	0.013
QBERT	**0.008**	0.010	0.011	0.009	0.011	0.012
ROADRUNNER	**0.009**	0.010	0.010	**0.009**	0.011	0.012
STARGUNNER	**0.009**	0.010	**0.009**	**0.009**	0.010	0.010
TIMEPILOT	0.012	0.013	0.012	**0.011**	0.013	**0.011**
WIZARDOFWOR	**0.013**	0.017	0.029	0.017	0.018	0.034
ZAXXON	**0.010**	0.012	**0.010**	0.011	0.011	**0.010**

As shown in Table 2, our proposed method can achieve the lowest variance on target value estimation in most environments.

F Additional Experiments for Baseline Comparisons and Ablation Studies

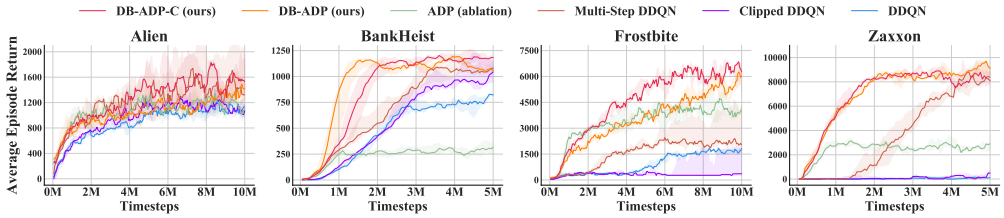


Figure 8: Learning curves on a suite of Atari benchmark tasks for comparing two additional baselines.

We compare our proposed method with two additional baselines:

- **ADP (ablation):** We conduct an ablation study that removes Bellman error minimization from the training and directly optimize the following objective:

$$L^{\text{ADP}}(\theta) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim D} \left(Q_\theta(s_t,a_t) - y^{\text{DP}}(s_t,a_t)\right)^2,$$

  where $y^{\text{DP}}(s_t,a_t)$ denotes the target value directly generated by dynamic programming. As shown in Figure 8, without integrating with Bellman operator, the abstracted dynamic programming itself cannot find a good policy. It remarks that, in our proposed framework, two basis estimators are supplementary to each other.

- **Multi-Step DDQN:** We also compare our method to a classical technique named multi-step bootstrapping that modifies the objective function as follows:

$$L^{\text{Multi-Step}}(\theta;\theta_{\text{target}},K) = \mathop{\mathbb{E}}_{(s_t,a_t)\sim D} \left(Q_\theta(s_t,a_t) - \left(\sum_{u=0}^{K-1} r_{t+u} + \gamma^K V_{\theta_{\text{target}}}(s_{t+K})\right)\right)^2,$$

  where we select $K = 3$ as suggested by (Castro et al., 2018). As shown in Figure 8, our proposed approach also outperforms this baseline.