Automating Control of Overestimation Bias for Continuous Reinforcement Learning

Arsenii Kuznetsov*  Alexander Grishin*  Artem Tsypin
Arsenii Ashukha  Dmitry Vetrov
Samsung AI Center Moscow

Abstract

Bias correction techniques are used by most of the high-performing methods for off-policy reinforcement learning. However, these techniques rely on a pre-defined bias correction policy that is either not flexible enough or requires environment-specific tuning of hyperparameters. In this work, we present a simple data-driven approach for guiding bias correction. We demonstrate its effectiveness on the Truncated Quantile Critics — a state-of-the-art continuous control algorithm. The proposed technique can adjust the bias correction across environments automatically. As a result, it eliminates the need for an extensive hyperparameter search, significantly reducing the actual number of interactions and computation.

1 Introduction

The overestimation bias recently received a lot of attention in reinforcement learning research. It is associated with the inferior performance of reinforcement learning agents, thus inspiring a wide range of algorithms based on bias correction approaches. However, these approaches do not directly optimize the bias, so the over- and underestimation bias is still present (Pan et al., 2020; He and Hou, 2020b; Saglam et al., 2021). In this paper, we explore a data-driven bias correction technique that allows to control the bias only when needed.

The accurate estimation of the state-action value function (Q-function) is the cornerstone of reinforcement learning. Overestimation bias is one of the sources of errors in value function estimates (Thrun and Schwartz, 1993; Hasselt, 2010). The overestimation appears because of the sample-based maximization of inaccurate value estimates, which leads to error accumulation due to the nature of temporal-difference (TD) learning (Fujimoto et al., 2018). As a consequence of the bias, sub-optimal actions may receive high-value estimates. This effect slows down training and leads to the inferior results.

The desire for correction of the overestimation bias has inspired a number of high-performing reinforcement learning algorithms (Fujimoto et al., 2018; Lan et al., 2020; Kuznetsov et al., 2020). However, they use a fixed policy — defined prior to training — for bias correction that either does not entirely eliminate the bias (He and Hou, 2020a) or requires fine-grained tuning of the intensity of the correction (Kuznetsov et al., 2020).

We show that bias can be reduced directly in a data-driven way

We apply the proposed adaptation technique to the soft modification of Truncated Quantile Critics (TQC) algorithm (Kuznetsov et al., 2020). Automatic adaptation allows to avoid extensive environment-specific tuning of hyperparameters which results in nearly ×2 improvement of the actual number of required environment interactions (details in Section 5.2). That results in performance that is state-of-the-art in continuous control at N samples to the best of our knowledge. Theoretically, we prove that under mild conditions—on the randomness of returns—the policy evaluation of AdaTQC tends to decrease the degree of overestimation control, that is consistent with the empirical results.

Our contributions can be summarized as follows: i) we show that bias can be reduced automatically in a data-driven way; ii) We demonstrate state-of-the-art performance and sample efficiency simultaneously; iii) We prove under mild assumptions and demonstrate empirically convergence to the true value function.
2 Background

Reinforcement Learning We consider a conventional reinforcement learning problem where an agent interacts with the environment to maximize its cumulative reward. The environment is an infinite horizon Markov decision process (MDP) which is defined by the tuple $\langle S, A, P, R, \gamma \rangle$ where: $S, A$ are continuous state and action spaces, $P : S \times A \rightarrow \mathcal{P}(S)$ is state transition density, $R : S \times A \rightarrow \mathcal{P}(R)$ is stochastic reward function, $\gamma \in [0, 1)$ is discounting factor, and $\mathcal{P}(\cdot)$ denotes a set of probability distributions over a respective set.

The agent behaves according to the policy $\pi$ where $\phi$ are the parameters of the policy. A trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ is obtained by sequentially selecting actions from the policy $\pi$, executing them in the environment and transitioning to the next state according to $P$. The trajectory distribution of a policy $\pi$ is then given by $p^\pi(\tau) = \prod_{t \geq 0} \pi(a_t | s_t)P(s_{t+1} | s_t, a_t)$. To signify that trajectories are sampled from $p^\pi(\tau)$ we are going to write $\tau \sim \pi$. The objective of Reinforcement learning is then to find a policy that maximizes the discounted trajectories $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$.

Actor-critic methods The actor-critic algorithms solve the problem of cumulative reward maximization. An actor policy $\pi_\phi$ is trained to maximize the expected reward while using a critic as an estimate of a value function (e.g., $Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \sum_{t \geq 0} \gamma^t r_t$) to score the actions. The critic is trained with temporal difference learning, minimizing $n$-step error. The actor is trained via (deterministic) policy gradient theorem (Sutton and Barto, 2018; Silver et al., 2014).

Soft actor-critic The reward can be augmented with policy entropy at each timestamp to encourage better exploration (Haarnoja et al., 2018a) yielding maximum entropy reinforcement learning problem: $C^\text{soft}(\phi) = \mathbb{E}_{\tau \sim \pi_\phi} \sum_{t \geq 0} \gamma^t [r_t - \alpha \log \pi_\phi(a_t | s_t)] \rightarrow \max_{\phi}$.

Distributional RL Distributional reinforcement learning (Bellemare et al., 2017; Dabney et al., 2018b) considers distribution $Z^\pi(s, a)$ over returns as opposed to the expectation of that distribution $Q^\pi(s, a) = \mathbb{E}[Z^\pi(s, a)]$. $Z^\pi(s, a)$ is usually approximated with a parametric model $\tilde{Z}_\psi(s, a)$ that is trained to match TD target distribution. Where $\psi$ are trained by minimizing distributional TD error.

3 Automating Bias Control

We define the bias($\cdot$) of an approximation $\hat{Q}$ of state-action value function $Q^\pi$ for a policy $\pi$ in state $s$ as:

$$\text{bias}(\hat{Q}, \pi, s) := \mathbb{E}_{\tau \sim \pi} (\hat{Q}(s, a) - Q^\pi(s, a)).$$

A lot of bias correction techniques explicitly control the bias with a single scalar parameter which we will call $\eta$ (Kuznetsov et al., 2020; Lan et al., 2020; Fujimoto et al., 2019; Zhang et al., 2017; Anschel et al., 2017). For example, Lan et al. (2020) uses the minimum over varying number of value function to construct a value approximator, Kuznetsov et al. (2020) truncates the right tail of the distributional value function approximations by a varying number of quantiles, Fujimoto et al. (2019) uses varying weight of minimum in convex combination of minimum and maximum of value approximators.

For these methods:

1. $\hat{Q}_\psi$ does not directly depend on control hyperparameter $\eta$, but $\eta$ affects value function approximation $\hat{Q}_\psi$ only through the optimization of its parameters $\psi$ since $\eta$ participates in TD target.

2. The connection between the control hyperparameter $\eta$ and the bias is monotonic—increase of $\eta$ leads to a decrease of bias.

In the following section, we show that control hyperparameter $\eta$ (not specifying to which exact method it corresponds to) satisfying above properties can be used to control the bias in a data-driven way.

3.1 Automating Bias Control Algorithm

We denote $\eta(\pi, \hat{Q})$ at optimization step $t$ as $\eta_t$. Increase of $\eta_t$ at step $t$ reduces the overestimation at step $t + 1$ and decrease of $\eta_t$ relaxes the intensity of correction at step $t + 1$. In other words, bias at step $t + 1$ depends on $\eta_t$ through the parameters $\psi_{t+1}$ of $\hat{Q}_{t+1}$:

$$\text{bias}(\hat{Q}_{t+1}, \pi_{t+1}, s) = \text{bias}(\hat{Q}_{\psi_{t+1}(\eta_t)}, \pi_{t+1}, s).$$

To automate bias control we introduce bias control updates:

- **if** $\text{agg bias}(\hat{Q}_t, \pi_t) > 0$ **overestimation**
  - $\eta_{t+1} = \eta_t + \lambda$ **increase $\eta$**
  - $\eta_{t+1} = \eta_t - \lambda$ **decrease $\eta$**
- **less overestimation at step $t + 1$**
- **less underestimation at step $t + 1$**

where $\lambda$ is step size, $\text{agg bias}(\hat{Q}, \pi) = \mathbb{E}_{p_{agg}}(\text{bias}(\hat{Q}, \pi, s))$ is the bias aggregated over some distribution over states $p_{agg}(s)$, we discuss the possible choices in Section 3.2.

The bias control updates can be understood as the
approximation of the following meta-gradient step

$$\eta_{t+1} = \eta_t + \lambda \text{sign}(\nabla_{\eta} \text{agg bias}(\hat{Q}_{\psi_{t+1}}(\eta), \pi) |_{\eta=\eta_t})$$, \hspace{1cm} (2)$$

where $\text{sign}(\cdot)$ serves as the normalization, that allows to select step-size $\alpha$ independently of the absolute value of returns. The approximation is possible due to the prerequisite on the monotonic influence of $\eta$ on bias that effectively supplies information on the sign of gradient.

The procedure provides a way for data-driven control of the overestimation bias. In contrast to the pre-defined policy, the correction can adapt to the current level of overestimation during the training.

3.2 Bias estimation

Practical implementation of the algorithm requires us to establish an aggregation state distribution that results in an appropriate estimate of the aggregated bias.

**Aggregation distribution.** The purpose of the aggregation distribution is to guide the optimization of $\eta$ by providing the importance of unbiasing value estimates in specific regions for ongoing optimization. The perfect solution would be to sample states proportionally to potential policy performance gain caused by a more accurate value function, which is infeasible. Thus, we propose two feasible options:

- **State uniform:** samples proportionally to state visitation distribution $s \sim d_\pi(s)$. Focuses on states which are visited more often.

- **Trajectory uniform:** samples states uniformly $s \sim \text{Uniform}((s')_{s \in \tau})$ from uniformly sampled trajectories $\tau \sim d_\pi(\tau)$. Focuses on states from short failed trajectories.

The illustration is available in Figure 1.

**Estimation of aggregated bias.** Estimation requires two components: a source of on-policy trajectories and an estimate of the state-action value function. As a source of near on-policy trajectories, we use a small replay buffer $D_{\text{fresh}}$.

The discounted sum of future rewards starting from $s,a$ in sampled trajectory is the unbiased estimate of value function in the corresponding state-action pair. However future rewards are not available for some state-action pairs. Specifically, they are not available for the states near the end of incomplete trajectories—that were interrupted by time limit—so are not the discounted sum of future rewards. Value function for all the other state-action pairs can be easily estimated as future trajectory return $R$. Putting away states without enough number (500 in our case, depends on the value of $\gamma$) of future rewards available from $D_{\text{fresh}}$ we obtain $D_{\text{valid}}^{\text{fresh}}$.

As the result, we sample trajectories from replay buffer $D_{\text{valid}}^{\text{fresh}}$, and then sample with distribution $p_{\text{agg}}(s)$ from these trajectories and average the resulting biases:

$$\text{agg bias}(\hat{Q}, \pi) \approx \mathbb{E}_{\{\tau\}} \mathbb{E}_{s,a}(\hat{Q}(s,a) - z)$$

$$\{\tau\} \sim D_{\text{valid}}^{\text{fresh}} \quad s,a,z \sim p_{\text{agg}}(s,a,z | \{\tau\})$$, \hspace{1cm} (3)

where $\{\tau\}$ denotes a set of trajectories, and $z$ denotes a sample from the return distribution of state-action pair $s,a$ computed from the corresponding trajectory.

4 Adaptive Truncated Quantile Critics

In this section we introduce AdaTQC—soft TQC with bias control updates on top of it. Soft TQC is a modification of TQC that uses $\eta$—the continuous version of the number of dropped quantiles $d$ used in TQC. Continuous $\eta$ is more suitable for fine-grained optimization of overestimation control than its discreet analog $d$. From now on $\eta$ corresponds to this specific bias control hyperparameter instead of the abstract one.
Temporal-difference loss In soft TQC we use $N$ approximations $Z^n$ of distributional value function $Z$. Each approximation $Z^n = Z^n(\cdot; \psi_n)$ is parametrized by the corresponding part of $\psi$. Given a state-action pair each approximation $Z^n(s, a)$ outputs a vector of size $M$, that consists of positions of atoms $Z_m(s, a), m = 1 \ldots M$ approximating quantiles of the return distribution $Z^n(s, a)$. See Figure 2 for illustration.

To construct TD target we use approximations $Z^n(s, a), n = 1 \ldots N$ as follows:

$$y_i(s, a) := r(s, a) + \gamma Z_j(s', a'), \quad i = 1 \ldots MN. \quad (4)$$

We move each of the approximations $Z^n(s, a)$ closer to empirical distribution represented by the targets $y_i(s, a)$. We do that via the conventional quantile regression loss $L(s, a; \eta, \psi_n)$ as follows:

$$\Delta^n_{i, m} = y_i(s, a) - \hat{Z}_m(s, a | \psi_n) \quad (5)$$

$$L(s, a; \eta, \psi_n) = \frac{1}{M} \sum_{m=1}^{M} \left( \sum_{i=1}^{MN} w_i \cdot \rho_{\tau_m} (\Delta^n_{i, m}) \right), \quad (6)$$

where $n$ indexes approximations, $m$ indexes atoms of approximation, $i$ indexes joint targets $y_i$, and $\rho_{\tau_m}(\cdot)$ is a Huber quantile regression loss (Dabney et al., 2018b). The joint loss is $L^D(\eta, \psi_n) = \mathbb{E}_{s, a, \eta} L(s, a; \eta, \psi_n)$.

Quantile loss for each difference $\Delta^n_{i, m}$ is multiplied by the truncation weight

$$w_i(\eta) = \begin{cases} 1, & i < u \\ i - u, & u \leq i < \lfloor u \rfloor \\ 0, & i \geq \lfloor u \rfloor \end{cases}, \quad (7)$$

where $u = NM(1 - \eta)$ denotes the soft number of samples from targets that are used in TD loss.

These weights help to avoid the propagation of the over-estimated values into the learned value function. The truncation weights are controlled by a single parameter $\eta$ that denotes the soft—thus suitable for fine-grained continuous optimization—number of truncated quantities.

Automating choice of $\eta$ We apply the bias control updates to soft TQC algorithm and tune the truncation parameter $\eta$ in a data-driven way as discussed in Section 3. We aggregate bias with trajectory uniform distribution if not stated otherwise.

Policy The policy $\pi(a | s; \phi)$ is trained by distillation of the maximum Q-values:

$$L^\pi(\phi) = -\mathbb{E}_{s \sim D, a \sim \pi(a | s; \phi)} \hat{Q}(s, a) \to \min_{\phi}, \quad (8)$$

where $\hat{Q}(s, a) = \frac{1}{NM} \sum_{m=1}^{M} \sum_{n=1}^{N} \hat{Z}_m(s, a; \psi_n)$, and states are sampled from the replay buffer $D$.

Soft TQC with Automating Bias Control We apply the bias control updates (Section 3) to soft TQC algorithm (Section 4). We also add a weighed policy entropy bonus $\alpha \cdot \log \pi_\phi(\cdot | \cdot)$ to targets $y_i(s, a)$ and policy loss $L_\pi(\psi)$. Here $\alpha$ is dynamically adjusted by target-entropy loss:

$$L^H(\alpha) = \mathbb{E}_{D, \pi_\phi} [\alpha \cdot (\log \pi_\phi(a | s) - H_T)], \quad (9)$$

where $H_T = -\dim(\text{action space})$ is the target entropy. The final method is demonstrated in the algorithm 1.

4.1 On unbiasedness of policy evaluation

Definitions

- Truncation operator $S^u : \mathcal{X} \to \mathcal{X}$

$$F_{S^u}(x) = \begin{cases} F_X(x), & x \leq F_X^{-1}(1 - \eta) \\ 1, & \text{otherwise} \end{cases} \quad (10)$$

- Distributional Bellman operator $\mathcal{T}^\pi$

$$[\mathcal{T}^\pi Z](s, a) \overset{\Delta}{=} R(s, a) + \gamma Z(s', a'), \quad (11)$$

where $s' \sim p_{\text{env}}(\cdot | s, a), \quad a' \sim \pi(\cdot | s')$.

- Truncated distributional Bellman operator $\mathcal{T}^{\pi, \eta}$

$$\mathcal{T}^{\pi, \eta} = \mathcal{T}^\pi \circ S^u \quad (12)$$

- Random field $Z$ is sufficiently stochastic for $\eta$

$$\exists \alpha(\eta) > 0 : \quad \mathbb{E}_{s', a' \sim p_0} \left[ z S^u Z(s', a') - \mathbb{E}_z Z(s', a') \right] \leq -\alpha(\eta), \quad (13)$$
Algorithm 1: Soft TQC with automating bias control. For a mini-batch \( B \), \( \nabla^B \) denotes a stochastic gradient estimation over the mini-batch, \( \text{agg}_{B}^{t} \) bias denotes an estimate of a bias over the mini-batch.

Require: \( \pi_{\phi} \): policy
Require: \( Z_{\psi_n}, Z_{\eta}, n = 1 \ldots N \): critics

1. Initialize experience replays \( D = \emptyset \), \( D_{\text{fresh}} = \emptyset \)
2. Initialize \( \alpha = 1, \beta = 0.005, \eta = \frac{1}{MTN} \)
3. for each iteration do
   4.   for each environment step do
      5.     collect transition \( (s_t, a_t, r_t, s_{t+1}) \) with \( \pi_{\phi} \)
      6.     \( D \leftarrow D \cup \{(s_t, a_t, r_t, s_{t+1})\} \)
      7.   \( D_{\text{fresh}} \leftarrow D_{\text{fresh}} \cup \{(s_t, a_t, r_t, s_{t+1})\} \)
   8. for each gradient step do
      9.     \( B \leftarrow \) sample a batch from the replay \( D \)
     10.    \( B_{\text{fresh}} \leftarrow \) sample a batch from the replay \( D_{\text{fresh}} \)
     11.   \( \alpha \leftarrow \alpha - \lambda_\alpha \nabla^B L^H(\alpha) \)
     12.   \( \phi \leftarrow \phi - \lambda_\phi \nabla^B L^\pi(\phi) \)
     13.   \( \eta \leftarrow \eta + \lambda \text{sign}(\nabla_{\eta} \text{agg bias}(\hat{Q}_{\psi_{t+1}(\eta), \pi})) \)
     14.   \( \psi_n \leftarrow \psi_n - \lambda_\psi \nabla_{\psi_n} L^{TD}(\eta, \psi_n), n = 1 \ldots N \)
     15.   \( \psi_n \leftarrow \beta \psi_n + (1 - \beta) \psi_n, \quad n = 1 \ldots N \)
   16. end for
   17. end for
18. return policy \( \pi_{\phi} \)

where \( p_\theta(s', a') = \mathbb{E}_{s \sim p_{\text{env}}(s)} \mathbb{E}_{a \sim \pi_\theta(s)} p_{\text{env}}(s', a) \pi(a' | s') \).

Simply put, there are enough state-action pairs, where truncation of return distribution causes non-negligible decrease of expectation.

**Theorem 1.** If we evaluate policy by sequentially applying truncated distributional Bellman operator \( Z_{t+1} = T^\pi Z_t \) to some initial random process \( Z_0 \) and if starting from some moment sequence \( Z_t \) will be sufficiently stochastic, then:

\[ \exists \delta > 0, t_2: \forall t', t_2 \quad \mathbb{E}_{s, a \sim p_{\text{opt}}}[E_{Z_t}(s, a) - Q^\pi(s, a)] < -\delta. \quad (14) \]

More formal description and the proof is provided in Appendix.

Simply put, Theorem 1 states that evaluating a fixed policy with a simplified version of soft TQC with a constant \( \eta \) eventually leads to a negative aggregated bias. Effectively, such evaluation augmented with a slow updates of \( \eta \) by automated bias control leads to \( \eta \rightarrow 0, T^{\pi, \alpha, \beta} \rightarrow T^{\pi}, \mathbb{E} Z_t \rightarrow \mathbb{E} Z^\pi \).

**Sufficient stochasticity** in the context of theorem depends on the environment and the policy. We argue that for stochastic policies with approximately fixed entropy and wide class of environments the correspond-

![Figure 3: Relative performance decrease of TQC with different values of hyperparameter \( d \) w.r.t. the performance of the best one for every environment. The best \( d \) for every environment is denoted by green. The optimal value of \( d \) varies dramatically across environments, while there is no good universal choice of \( d \).](image-url)

We empirically evaluate the statement of the Theorem 1 and refer to the Section 5.3 for the results.

## 5 Experiments

We test AdaTQC on MuJoCo (Todorov et al., 2012) and PyBullet (Coumans and Bai, 2021) suites. We implement all models on PyTorch (Paszke et al., 2019). We remove HalfCheetah environment from the Experiments section due to high variance of the performance for most of the methods but report them in Appendix.

### 5.1 Comparison of Performance

**Does TQC need HP selection?** In Figure 3 we show deterioration of the mean score of non-adaptive TQC when a hyper-parameter (HP) \( d \) diverges from the optimal one. With non-optimal \( d \), we can observe as much as a 39% drop of the score (not considering universal bad \( d = 0 \)), so TQC is sensitive to HP choice. Moreover, there is no single HP that is consistently good for all environments. In other words, there is no row without red cells depicting performance decrease.

**Setup** We compare the performance of AdaTQC with: TQC (Kuznetsov et al., 2020), SAC (Haarnoja et al., 2018a), TD3 (Fujimoto et al., 2018), DSAC (Ma et al., 2020). On PyBullet, did not tune any hyperparameters of AdaTQC—we directly use the same set of HPs as for any other environment—while non-adaptive TQC
Table 1: The comparison of the proposed AdaTQC with concurrent continuous control algorithms, all values are scaled by 10$^3$, and also we report the standard deviations of evaluation returns computed over 10 runs of the algorithms. △-s denote the relative difference in the performance to AdaTQC. ▼ means that the corresponding method preforms worse than AdaTQC, and ▲ means that a method preforms better than AdaTQC. AdaTQC closely matches the performance of vanilla TQC while avoiding environment-specific hyperparameter tuning.

Table 2: Average improvement in sample efficiency of AdaTQC in comparison to TQC. Value $k$ means that TQC on average needs at least $k$ full runs with different hyperparameters ($d \in \{0, 1, \ldots, 5\}$) to achieve the same level of performance as AdaTQC with one run. For example, ×3 means that on average TQC needs to try 3 values of hyperparameter $d$, and maximize over them to achieve the same performance as AdaTQC does with just a single run. All the runs consist of the same amount of interactions.

5.2 Sample Efficiency

Results are shown in Table 1. While TQC has shown a state of the art level of results, it requires manual tuning of hyperparameter for each environment. In this section, we quantify the benefits from the absence of hyperparameter tuning. Specifically, we answer the following question:

How many times one needs to run TQC with different hyperparameters to achieve the level of performance of AdaTQC?

We show the results in Table 2. On average, AdaTQC requires ×2 fewer interactions for training. AdaTQC adapts prior knowledge (see Section 3) about $\eta$—overestimation control parameter—that allows to reduce a costly hyperparameter selection to an automatic tuning procedure. The details on sample efficiency computation are presented in Appendix.

5.3 Ablation Study

Number of estimators We show that the typical value of $\eta$ decreases as the number of estimators increases, plots in Appendix. It indicates that overestimation is less severe when the number of networks increases which is consistent with the results from Lan et al. (2020). This experiment serves as a double-check that $\eta$ optimization behaves as expected.

Sampling schemes We compare the i) state uniform and ii) trajectory uniform sampling schemes for estimation of the aggregated bias during training (Section 3.2). The results are presented in Figure 6. The
trajectory-uniform sampling in general leads to better final rewards. It increases the sampling probability of states from the short-failed trajectories, where we believe the more accurate state-action value estimations are needed.

**Bias over the course of training** In Figure 4 we show the value of aggregated bias with trajectory uniform sampling for models throughout the training. Each point represents an average of 4 training seeds with aggregated bias estimated on 1000 additional on-policy trajectories. From the plots we can see that AdaTQC optimizes aggregated bias on par with the tuned TQC and prevents significant deviations from 0.

**Fixed policy** We empirically evaluate the claim of the theorem that for fixed policy $\eta$ converges to zero. In Fig. 5 we show the dynamics of $\eta$ if we freeze only policy at some point in training.

**6 Related work**

Accurate estimation of the value function is one of the core problems in reinforcement learning. The use of approximation-based algorithms induces a consistent and uneven overestimation (Thrun and Schwartz, 1993) which negatively affects their performance (Thrun and Schwartz, 1993; Szita and Lőrincz, 2008; Strehl et al., 2009; Van Hasselt et al., 2016; Lan et al., 2020). This motivated the development of value estimation bias correction techniques.

Some of the techniques for overestimation bias correction are focused on the development of new estimates of the temporal-difference target. Double DQN (Van Hasselt et al., 2016) proposed to disentangle the selection of actions from the estimation of returns by using different Q-functions. Averaged Q-learning (Anschel et al., 2017) takes the average of previously learned state-action values and is shown to reduce the overestimation bias. Maximin Q-learning (Lan et al., 2020) used an ensemble of Q-functions to balance between over- and underestimation by varying the number of critic networks in the ensemble.

In continuous control domain, Twin Delayed DDPG (Fujimoto et al., 2018) introduced clipped double Q-learning. The technique consists in taking a minimum of two independent Q-functions. Although successfully eliminating overestimation bias clipped double Q-learning causes severe underestimation bias (Pan et al., 2020; He and Hou, 2020a) which deteriorates the performance of the algorithm due to the pessimistic underexploration (Ciosek et al., 2019). Triplet-average DDPG (Wu et al., 2020) and Weighted Delayed DDPG (He and Hou, 2020a) used a weighted sum of the minimum of Q-functions and the average of an...
ensemble of Q-functions to mitigate the effect of underestimation.

Softmax Double DDPG (Pan et al., 2020) and Double Actors and Regularized critics (Lyu et al., 2021) leveraged double actors setup to reduce value estimation bias. Pan et al. (2020) also applied Boltzman softmax operator to the Q-function to further improve value estimation. Distributional Soft Actor Critic (Ma et al., 2020) proposed to use a truncated ensemble of Q-functions to mitigate the effect of un-bias. Another method GPL-SAC that is closely related to AdaTQC has been suggested in a concurrent work (Cetin and Celiktutan, 2021). It uses the epistemic uncertainty to control the overestimation bias in a data-driven way. The main difference is that we use the true value of the returns to estimate the bias, whereas GPL-SAC uses a Q-function. The other difference is that unlike AdaTQC, GPL-SAC uses a large update-to-data ratio UTD ≫ 1.

7 Conclusion and future work

In this work, we introduce the approach to adapt a significant class of overestimation control techniques in a data-driven way. We empirically show that the proposed approach on top of TQC is able to automatically reduce bias without hyperparameter tuning while preserving the performance.

We see state-dependent overestimation control parameters η(s) as a promising direction for the future research. It could allow to reduce bias in each state individually and the proposed approach is ready “as is” for such modification.
References

Anschel, O., Baram, N., and Shimkin, N. (2017). Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In ICML.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In International Conference on Machine Learning, pages 449–458. PMLR.

Cetin, E. and Celiktutan, O. (2021). Learning pessimism for robust and efficient off-policy reinforcement learning.

Ciosek, K., Vuong, Q. H., Loftin, R. T., and Hofmann, K. (2019). Better exploration with optimistic actor-critic. In NeurIPS.

Coumans, E. and Bai, Y. (2016–2021). Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018a). Implicit quantile networks for distributional reinforcement learning. In International conference on machine learning, pages 1096–1105. PMLR.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2018b). Distributional reinforcement learning with quantile regression. In Thirty-Second AAAI Conference on Artificial Intelligence.

Duan, J., Guan, Y., Li, S. E., Ren, Y., Sun, Q., and Cheng, B. (2021). Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. IEEE Transactions on Neural Networks and Learning Systems.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In International Conference on Machine Learning, pages 1587–1596. PMLR.

Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In International Conference on Machine Learning, pages 2052–2062. PMLR.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018a). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning, pages 1861–1870. PMLR.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018b). Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

Hasselt, H. (2010). Double q-learning. Advances in neural information processing systems, 23:2613–2621.

He, Q. and Hou, X. (2020a). Reducing estimation bias via weighted delayed deep deterministic policy gradient. arXiv preprint arXiv:2006.12622.

He, Q. and Hou, X. (2020b). Wd3: Taming the estimation bias in deep reinforcement learning. In 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), pages 391–398. IEEE.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Kuznetsov, A., Shvetsikhov, P., Grishin, A., and Vetrov, D. (2020). Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In International Conference on Machine Learning, pages 5556–5566. PMLR.

Lan, Q., Pan, Y., Fyshe, A., and White, M. (2020). Maxmin q-learning: Controlling the estimation bias of q-learning. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Lyu, J., Ma, X., Yan, J., and Li, X. (2021). Efficient continuous control with double actors and regularized critics. arXiv preprint arXiv:2106.03050.

Ma, X., Xia, L., Zhou, Z., Yang, J., and Zhao, Q. (2020). Dsac: Distributional soft actor critic for risk-sensitive reinforcement learning. arXiv preprint arXiv:2004.14547.

Moskovitz, T. H., Parker-Holder, J., Pacchiano, A., Arbel, M., and Jordan, M. I. (2021). Tactical optimism and pessimism for deep reinforcement learning.

Pan, L., Cai, Q., and Huang, L. (2020). Softmax deep double deterministic policy gradients. arXiv preprint arXiv:2010.09177.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E., and Garnett, R., editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc.

Saglam, B., Duran, E., Ciccek, D. C., Mutlu, F. B., and Kozat, S. S. (2021). Parameter-free deterministic reduction of the estimation bias in continuous control. arXiv preprint arXiv:2109.11788.

Silver, D., Lever, G., Heess, N. M. O., Degris, T., Wierstra, D., and Riedmiller, M. A. (2014). Deterministic policy gradient algorithms. In ICML.
Strehl, A. L., Li, L., and Littman, M. L. (2009). Reinforcement learning in finite mdps: Pac analysis. *Journal of Machine Learning Research*, 10(11).

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Szita, I. and Lörincz, A. (2008). The many faces of optimism: a unifying approach. In *Proceedings of the 25th international conference on Machine learning*, pages 1048–1055.

Thrun, S. and Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*, pages 255–263. Hillsdale, NJ.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In 2012 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Wu, D., Dong, X., Shen, J., and Hoi, S. C. (2020). Reducing estimation bias via triplet-average deep deterministic policy gradient. *IEEE transactions on neural networks and learning systems*, 31(11):4933–4945.

Zhang, Z., Pan, Z., and Kochenderfer, M. J. (2017). Weighted double q-learning. In *IJCAI*, pages 3455–3461.
Appendices

A Bias evaluation

In this section, we describe the process of bias estimation in more detail and explain our choice of aggregation distributions. Let us remind that in Section 3.2 we introduced two aggregation distributions: *state uniform* and *trajectory uniform*, that work as follows:

- **State uniform**: states are sampled from state visitation distribution $p(s) = d_\pi(s)$ induced by a policy $\pi$;
- **Trajectory uniform**: states are sampled uniformly from uniformly sampled trajectories:

$$\tau \sim p^\pi(\tau) \quad s|\tau \sim \text{Uniform}\{s \in \tau\}.$$  

For a state-action pair $(s_t, a_t)$ at a step $t$ from a trajectory $\tau$ the bias of an approximation $\hat{Q}$ is estimated as

$$\text{bias}(s_t, a_t, \hat{Q}) = \hat{Q}(s_t, a_t) - \left[r_t + \sum_{t'=t+1}^T \gamma^{t'-t}(r_{t'} + \alpha H_{\text{target}})\right],$$  

where $(s_t, a_t)$ comes from either *state uniform* or *trajectory uniform* distribution. In theory, in place of $H_{\text{target}}$ should be the entropy of the policy in states $s_{t'+1}$, but that would require to evaluate policy neural network multiple times for each $(s_t, a_t)$ pair. As our method is based on Soft Actor Critic with automatic $\alpha$ tuning, average policy entropy is optimized to be equal to hyperparameter $H_{\text{target}}$, that we use instead.

In practice, we use only trajectories where the true return can be estimated accurately. We safely use trajectories that end with *done*. But if a trajectory $\tau$ ends with a *time limit*, we use only $\text{len}(\tau) - 500$ first state-action pairs, as for state-action pairs in the end of the trajectory there is not enough data to estimate return, because the environment has been reset after *time limit*. Consequently, $T$ in (15) is 500, or until *done* if the trajectory is shorter.

The truncation can potentially lead to a biased estimate of the true aggregated bias, although it seems not to harm the correction in practice (Figure 4). In the figure, bias is estimated according to the definition with extra on-policy trajectories, which are played in the environment without the time limit. To estimate one point we take the model from the corresponding seed and average over 1000 on-policy trajectories. The shaded area corresponds to the bias evaluation for different training seeds.
B Main experiments graphs and table

| Simulator | Environment   | $N = 5$ | $N = 2$ |
|-----------|---------------|---------|---------|
|           | AdaTQC (ours) | TQC$^*$ | △       | AdaTQC (ours) | TQC$^*$ | △       | DSAC | SAC | TD3 |
| PyBullet  | Hopper        | 2.61 ± 0.38 | 2.74 ± 0.12 | ▲ 9% | 2.67 ± 0.22 | 2.66 ± 0.17 | ▼ 0% | 2.59 ± 0.18 | 2.44 ± 0.31 | 2.16 ± 0.32 |
|           | Ant           | 3.76 ± 0.34 | 3.74 ± 0.44 | ▼ 0% | 3.89 ± 0.29 | 3.53 ± 0.46 | ▼ 9% | 3.49 ± 0.24 | 3.64 ± 0.34 | 3.43 ± 0.43 |
|           | Walker2D      | 2.68 ± 0.86 | 3.10 ± 0.12 | ▲ 16% | 2.41 ± 1.00 | 2.80 ± 0.58 | ▲ 16% | 2.89 ± 0.11 | 2.37 ± 0.16 | 2.35 ± 0.18 |
|           | Humanoid      | 3.68 ± 0.34 | 4.27 ± 0.49 | ▲ 16% | 3.08 ± 0.21 | 2.78 ± 0.30 | ▼ 10% | 2.18 ± 0.29 | 1.29 ± 0.24 | 1.38 ± 0.56 |
|           | HalfCheetah   | 3.39 ± 0.32 | 3.32 ± 0.63 | ▼ 2% | 3.54 ± 0.35 | 3.47 ± 0.52 | ▼ 2% | 2.92 ± 0.19 | 3.04 ± 0.17 | 2.79 ± 0.26 |
| MuJoCo    | Hopper        | 3.81 ± 0.24 | 3.78 ± 0.19 | ▼ 1% | 3.87 ± 0.23 | 3.71 ± 0.18 | ▼ 4% | 2.37 ± 0.48 | 2.86 ± 0.58 | 3.31 ± 0.55 |
|           | Ant           | 8.05 ± 0.42 | 8.17 ± 0.37 | ▼ 1% | 7.55 ± 0.79 | 7.48 ± 0.69 | ▼ 1% | 6.53 ± 1.11 | 6.16 ± 0.93 | 5.68 ± 1.04 |
|           | Walker2D      | 7.00 ± 0.49 | 6.87 ± 0.77 | ▼ 2% | 6.40 ± 0.43 | 6.51 ± 0.70 | ▼ 2% | 6.44 ± 0.53 | 5.76 ± 0.46 | 5.11 ± 0.52 |
|           | Humanoid      | 9.27 ± 1.08 | 9.17 ± 1.35 | ▼ 1% | 9.48 ± 1.09 | 9.67 ± 1.07 | ▼ 2% | 8.49 ± 0.79 | 7.76 ± 0.46 | 5.40 ± 0.36 |

Table 3: The comparison of the proposed AdaTQC with concurrent continuous control algorithms, all values are scaled by $10^3$. We also report the standard deviations of evaluation returns computed over 10 runs of the algorithms. * at TQC means that the method uses the best hyperparameter $d$. △-s denote the relative difference in the performance of TQC compared to AdaTQC . ▼ means that the corresponding method performs worse than AdaTQC, and ▲ means that a method performs better than AdaTQC.

Figure 7: Optimization curves for AdaTQC, TQC with two different numbers of critics and other methods. Each method is averaged over 10 runs, performance evaluated every 1000 steps, curves are smoothed with a window of 100, ±std of evaluation returns is shaded.
C Theoretical results

By \( F_X(x) \) in this section we define distribution function of random variable \( X \):

\[
F_X(x) = P(X \leq x)
\]  \hspace{1cm} (16)

**Definition.** Let's define truncation operator \( S^\eta(\cdot) \), \( 0 \leq \eta < 1 \) for one-dimensional random variable via distribution function of the result:

\[
F_{S^\eta(Z)}(z) = \begin{cases} 
F_Z(z) \frac{1}{1-\eta}, & z \leq F_Z^{-1}(1-\eta) \\
1, & z > F_Z^{-1}(1-\eta)
\end{cases}
\]  \hspace{1cm} (17)

**Definition.** We introduce non-strict partial order on space of one-dimensional random variables:

\[
Z_1 \preceq Z_2 \text{ if } F_{Z_1}(z) \geq F_{Z_2}(z) \quad \forall z
\]  \hspace{1cm} (18)

**Lemma 1.** \( Z_1 \preceq Z_2 \Rightarrow E Z_1 \leq E Z_2 \)

\[
Z_1 \preceq Z_2 \Rightarrow F_{Z_1}(z) \geq F_{Z_2}(z) \quad \forall z \Rightarrow F_{Z_1}^{-1}(w) \leq F_{Z_2}^{-1}(w) \quad \forall w
\]  \hspace{1cm} (19)

**Proof.** We can rewrite the expectations in Lebesgue form:

\[
E Z_1 = \int_0^1 F_{Z_1}^{-1}(w) \, dw \leq \int_0^1 F_{Z_2}^{-1}(w) \, dw = E Z_2
\]  \hspace{1cm} (20)

**Lemma 2.** \( \forall Z_1, Z_2, \eta : 0 \leq \eta < 1, Z_1 \preceq Z_2 \Rightarrow S^\eta Z_1 \preceq S^\eta Z_2 \)

**Proof.** Let us show that by definition of \( S^\eta \) and order \( \preceq \) the lemma holds true and \( F_{S^\eta(Z_1)}(z) \geq F_{S^\eta(Z_2)}(z) \quad \forall z : \)

\[
F_{Z_1}^{-1}(1-\eta) \leq z \leq F_{Z_2}^{-1}(1-\eta) \Rightarrow \frac{F_{Z_1}(z)}{1-\eta} \geq \frac{F_{Z_2}(z)}{1-\eta}
\]

\[
F_{Z_1}^{-1}(1-\eta) < z \leq F_{Z_2}^{-1}(1-\eta) \Rightarrow 1 \geq \frac{F_{Z_1}(z)}{1-\eta}
\]

\[
z > F_{Z_2}^{-1}(1-\eta) \Rightarrow 1 = 1
\]  \hspace{1cm} (21)

It follows that \( F_{S^\eta(Z_1)}(z) \geq F_{S^\eta(Z_2)}(z) \forall z. \)

**Lemma 3.** \( \forall Z, \eta_1, \eta_2 : 0 \leq \eta_1, \eta_2 < 1, \eta_1 \geq \eta_2 \Rightarrow S^\eta Z \preceq S^{\eta_2} Z \)

**Proof.** Also by definition

\[
F_{Z_1}^{-1}(1-\eta_1) \leq z \leq F_{Z_2}^{-1}(1-\eta_1) \Rightarrow \frac{F_{Z_1}(z)}{1-\eta_1} \geq \frac{F_{Z_2}(z)}{1-\eta_2}
\]

\[
F_{Z_1}^{-1}(1-\eta_1) < z \leq F_{Z_2}^{-1}(1-\eta_2) \Rightarrow 1 \geq \frac{F_{Z_1}(z)}{1-\eta_2}
\]

\[
z > F_{Z_2}^{-1}(1-\eta_2) \Rightarrow 1 = 1
\]  \hspace{1cm} (22)

It follows that \( S^\eta Z \preceq S^{\eta_2} Z. \)

**Lemma 4.** \( Z_1 \preceq Z_2, \eta_1 \geq \eta_2 \Rightarrow S^\eta Z_1 \preceq S^{\eta_2} Z_2 \)

**Proof.** Also by definition

\[
F_{Z_1}^{-1}(1-\eta_1) \leq z \leq F_{Z_2}^{-1}(1-\eta_1) \Rightarrow \frac{F_{Z_1}(z)}{1-\eta_1} \geq \frac{F_{Z_2}(z)}{1-\eta_2}
\]

\[
F_{Z_1}^{-1}(1-\eta_1) < z \leq F_{Z_2}^{-1}(1-\eta_2) \Rightarrow 1 \geq \frac{F_{Z_1}(z)}{1-\eta_2}
\]

\[
z > F_{Z_2}^{-1}(1-\eta_2) \Rightarrow 1 = 1
\]  \hspace{1cm} (23)

It follows that \( S^\eta Z \preceq S^{\eta_2} Z. \)
Proof. Applying previous two lemmas we get the result:

\[ S^{\eta_1}Z_1 \leq S^{\eta_2}Z_1 \leq S^{\eta_2}Z_2 \quad (23) \]

\[ \blacksquare \]

**Lemma 5.** \( \forall Z_1, Z_2, X : \ X \perp Z_1, Z_1 \leq Z_2 \Rightarrow Z_1 + X \leq Z_2 + X \)

**Proof.**

\[ Y_1 = Z_1 + X, \ Y_2 = Z_2 + X \]

\[ F_{Y_1}(y) = P(Y_1 \leq y) = \int_X p(x)P(Z_1 \leq y - x) \, dx = \int_X p(x)F_{Z_1}(y - x) \, dx = F_{Y_2}(y) \quad (24) \]

\[ \blacksquare \]

Now let’s consider random fields over state-action domain instead of scalar random variables. We overload notation and from here we also designate them by \( Z(s, a) \), \( s \in \mathcal{S}, a \in \mathcal{A} \).

**Definition.** Our partial order can be extended on random fields:

\[ Z_1(\cdot, \cdot) \preceq Z_2(\cdot, \cdot) \text{ if } Z_1(s, a) \preceq Z_2(s, a) \forall s \in \mathcal{S}, a \in \mathcal{A}. \quad (25) \]

**Lemma 6.** \( \forall Z_1, Z_2 : \ Z_1 \preceq Z_2, \forall p(s, a) \Rightarrow \mathbb{E}_{s,a \sim p}Z_1(s, a) \preceq \mathbb{E}_{s,a \sim p}Z_2(s, a) \)

**Proof.**

\[ Y_i = \mathbb{E}_{s,a \sim p}Z_i(s, a) \quad (26) \]

\[ F_{Y_1}(y) = P(Y_1 \leq y) = \int_{\mathcal{S},\mathcal{A}} p(s,a)P(Z_1 \leq y|s, a) \, dsda \geq \int_{\mathcal{S},\mathcal{A}} p(s,a)P(Z_2 \leq y|s, a) \, dsda = F_{Y_2}(y) \quad (27) \]

\[ \blacksquare \]

**Lemma 7.** \( \forall \gamma \in [0,1), \forall Z_1, Z_2 : \ Z_1 \preceq Z_2 \Rightarrow \gamma Z_1 \preceq \gamma Z_2 \)

Can be simply shown via definition.

**Definition.** Let us remind the notation introduced in section 4.1.

Distributional Bellman operator \( T^\pi: \)

\[ [T^\pi Z](s, a) \overset{d}{=} R(s, a) + \gamma Z(s', a') \quad (28) \]

where \( s' \sim p_{\text{env}}(\cdot|s, a), \ a' \sim \pi(\cdot|s'). \)

Truncated version: \( T^{\pi, \eta} = T^\pi \circ S^\eta. \)

**Lemma 8.** \( Z_1 \preceq Z_2 \Rightarrow T^\pi Z_1 \preceq T^\pi Z_2 \)

Follows from the previous three lemmas.

Consider supremum norm of action-value functions: \( ||Q||_\infty = \sup_{s, a} |Q(s, a)| \)

**Lemma 9.** \( ||\mathbb{E}_z T^\pi Z - Q^\pi||_\infty \leq \gamma ||\mathbb{E}_z Z - Q^\pi||_\infty \)
Proof.

\[
\forall s, a \quad |\mathbb{E}_s T^\pi Z(s, a) - Q^\pi(s, a)| =
\]

\[
|\mathbb{E}_{s', r \sim p_{\text{env}}(s, a)} E_{a' \sim \pi(s')} [r + \gamma \mathbb{E}_s Z(s', a')] - \mathbb{E}_{s', r \sim p_{\text{env}}(s, a)} E_{a' \sim \pi(s')} [r + \gamma Q^\pi(s', a')] |
\]

\[
\gamma |\mathbb{E}_{s', r \sim p_{\text{env}}(s, a)} E_{a' \sim \pi(s')} [E_s Z(s', a') - Q^\pi(s', a')] | \leq 
\]

\[
\gamma \sup_{s, a} |E_s Z(s, a) - Q^\pi(s, a)| = \gamma ||E_s Z - Q^\pi||_\infty
\]

Applying supremum to left-hand side we get:

\[
||E_s T^\pi Z - Q^\pi||_\infty = \sup_{s, a} |E_s T^\pi Z(s, a) - Q^\pi(s, a)| \leq \gamma ||E_s Z - Q^\pi||_\infty
\]

Now let's consider policy evaluation process with truncation \(Z_{t+1} = T^{\pi, \eta} Z_t\) for fixed \(\pi, \eta\). Also denote sequence without truncation \(Z^0_{t+1} = T^{\pi, 0} Z_t = T^\pi Z_t^0\). Both these sequences start in the same starting point \(Z_0\).

Lemma 10 shows that sequence \(E_s Z^0_t\) exponentially fast and uniformly on \(s, a\) converges to \(Q^\pi\), therefore

\[
\forall \epsilon > 0 \exists t_1 : \forall t > t_1 : \sup_{s, a} |E_s Z^0_t(s, a) - Q^\pi(s, a)| \leq \epsilon
\]

**Lemma 10.**

\[
\forall s, a, t \quad E_s Z_t(s, a) \leq E_s Z^0_t(s, a)
\]

It follows from lemmas 1, 4, 8 and the fact that both sequences have the same starting point.

Let's denote probability density \(\tilde{p}(s', a')\) based on probability distribution \(p_{\text{agg}}(s)\) considered in section 3.2:

\[
\tilde{p}(s', a') = E_s p_{\text{agg}} E_{a \sim \pi(s)} p_{\text{env}}(s'|s, a) \pi(a'|s')
\]

**Definition.** Let's define \(Z_t\) ”sufficiently stochastic” for fixed \(\eta, \pi\) if \(\exists \alpha(\eta) > 0\)

\[
E_{s', a' \sim \tilde{p}} [E_s S^n Z_t(s', a') - E_s Z_t(s', a')] \leq -\alpha(\eta)
\]

**Intuition**  Sufficient stochasticity is a property of an environment and a policy. It states that there are enough state-action pairs, where truncation leads to a non-negligible decrease in the expected value of a distribution over returns. During training, our stochastic policy is explicitly encouraged to have non-zero entropy. That, together with long trajectories lead to the sufficiently stochastic distribution over returns. In other words, we speculate that sufficient stochasticity is quite a weak requirement, and is often satisfied in practice.

**Theorem 1.** Let's assume, that we evaluate policy by sequentially applying truncated distributional Bellman operator \(Z_{t+1} = T^{\pi, \eta} Z_t\) to some initial random field \(Z_0\) and that starting from some moment sequence \(Z_t\) becomes sufficiently stochastic:

\[
\exists t_2, \alpha(\eta) : \forall t' > t_2 \quad 
E_{s', a' \sim \tilde{p}} [E_s S^n Z_t(s', a') - E_s Z_t(s', a')] \leq -\alpha(\eta)
\]

Then \(\exists t_3, \kappa > 0 : \forall t' > t_3 \quad 
E_{s \sim p_{\text{agg}}} E_{a \sim \pi(s)} [E_s Z_{t'}(s, a) - Q^\pi(s, a)] \leq -\kappa
\]
Proof. Let $t = \alpha(\eta)\gamma/2$ in (34). Then take $t' \geq \max(t_1, t_2)$.

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)}E_{z: t+1}Z_{t+1} (s, a) =$$

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)}E_{z: t}T^{\pi, \eta} Z_{t} (s, a) =$$

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)}E_{r, s', a'} [r + \gamma E_{z: t} Z_{t'} (s', a')] \leq$$

sufficient stochasticity

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)}E_{r, s', a'} [r + \gamma E_{z: t} Z_{t'} (s', a')] - \gamma \alpha(\eta) =$$

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)}E_{z: t} T^{\pi, 0} Z_{t} (s, a) - \gamma \alpha(\eta) \leq$$

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)}E_{z: t} T^{\pi, 0} Z_{t}^0 (s, a) - \gamma \alpha(\eta) =$$

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)}E_{z: t} Z_{t+1}^0 (s, a) - \gamma \alpha(\eta) \leq$$

equation (34)

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)} Q^{\pi} (s, a) - \gamma \alpha(\eta) + \frac{\gamma \alpha(\eta)}{2} =$$

$$E_{s \sim \pi_{agg}}E_{a \sim \pi(s)} Q^{\pi} (s, a) - \frac{\gamma \alpha(\eta)}{2}$$

We get that

$$\frac{\gamma \alpha(\eta)}{2} \leq \gamma \alpha(\eta) + \frac{\gamma \alpha(\eta)}{2}$$

The theorem has been proven with $t_3 = \max(t_1, t_2) + 1$, $\kappa = \frac{\gamma \alpha(\eta)}{2}$. \qed

**Understanding** The theorem, in other words, means that during an evaluation of a fixed policy with truncated distributional Bellman operator and a constant $\eta$, starting from some point, the measured aggregated bias will be negative. Effectively leading to a decrease of $\eta$ at optimization steps.

In practice, we expect that if $\eta$ changes slowly enough it will converge to zero ($\eta \to 0$), leading to an unbiased policy evaluation ($T^{\pi, \eta} \to T^\pi$, $E_z Z_t \to Q^\pi$). We support this hypothesis empirically, on a DNN-based model and real optimization trajectories (Figure 5).

## D Sample efficiency computation

In this section, we describe how we quantified sample efficiency of AdaTQC. Sample inefficiency of conventional TQC comes from the need for an environment-specific search for the number of truncated quantities $d$. In Table 2 we compared the number of samples required by AdaTQC to that of a conventional TQC with a grid search for $d \in D = \{0, 1, 2, 3, 4, 5\}$. In Table 4 AdaTQC was compared to TQC with a smaller grid search set of the best performing $d$-s ($d \in D = \{0, 2, 5\}$).

Column “Data reduction” is calculated as follows:

1. For each environment and method (TQC $d = \cdot$, and AdaTQC), we consider the average of evaluation performance over the last $5 \cdot 10^5$ interactions and over 4 seeds as the final performance.

2. For a fixed number $n \in [1, \ldots, |D|]$ of different values for hyperparameter for TQC we average over all possible combinations (without replacement) the maximum of final performance within the combination ($n$ out of $|D|$) and call it hp-averaged performance.

3. For each environment we find the least $n$ such that the corresponding hp-averaged performance is larger than the final performance of AdaTQC and call it data reduction.
Running heading author breaks the line

| Simulator | Environment | Data reduction |
|-----------|-------------|----------------|
| PyBullet  | Hopper      | × 2            |
|           | Ant         | > × 3          |
|           | Humanoid    | > × 3          |
|           | Walker2d    | × 1            |
|           | HalfCheetah | × 3            |
| MuJoCo    | Hopper      | > × 3          |
|           | Ant         | × 2            |
|           | Walker2d    | × 2            |
|           | Humanoid    | × 3            |

Table 4: Average improvement in sample efficiency of AdaTQC in comparison to TQC. Value $k$ means that TQC on average needs at least $k$ full runs with different hyperparameters ($d \in \{0, 2, 5\}$) to achieve the same level of performance as AdaTQC with one run. For example, $\times 3$ means that on average TQC needs to try 3 values of hyperparameter $d$, and maximize over them to achieve the same performance as AdaTQC does with just a single run. All the runs consist of the same amount of interactions.

In other words, “Data reduction” is the minimum amount of hyperparameter tries $n$, that on average is required by TQC to achieve the same or greater final performance than AdaTQC.

To calculate the average “data reduction”—aggregated across environments—we follow the steps (1-3) from above, but:

a) final performance in the step 1 is now the relative performance of TQC with respect to AdaTQC in the same environment. For example, final performance of 1.1 means that TQC obtains a score which is 10% better than the score of AdaTQC.

b) hp-averaged performance in the second step also averaged across environments. For example, hp-averaged performance of 1.01 for $n = 2$ means that TQC on average obtains a score which is 1% better than the score of AdaTQC when all possible pairs of hyperparameter $d$ are considered.

Since the resulting hp-averaged performance for $n = 2$ is slightly more than 1 (1.01) for a grid search with $d \in \{0, 1, 2, 3, 4, 5\}$ and slightly less than 1 (0.99) for a grid search with $d \in \{0, 2, 5\}$—both sets require to try $\times 2$ hyperparameters on average to match the performance of AdaTQC. Thus we report that on average AdaTQC is $\times 2$ more sample efficient than TQC because of the need of hyperparameter tuning of the later.

E Target computation

Here we describe more formally the computation of TD target in soft TQC shown in the scheme on the page 4. To start with, we sample a tuple $(s, a, s', a', r)$ from the replay buffer. Using $N$ approximators $\{\hat{Z}_n^m\}_{n=1}^N$ we obtain $M$ outputs per network totalling to $NM$ predicted atoms $\{\hat{Z}_m^n\}_{n=1}^N, m=1$. Sorted vector of those values constitutes $Z_{\text{joint}}(s', a')$.

F Experimental setup and hyperparameters

In our experiments, we observed that the use of MuJoCo 2.0 with versions of Gym at least up to v0.15.4 nullifies state components that correspond to contact forces which makes the results incomparable with the previous work. All the experiments thus were done with MuJoCo 1.5 and v3 versions of environments. For PyBullet we used v0 versions of environments.

All of the plotted curves were smoothed with a running average over a window of the size 100.

Critic networks are fully connected, with the last layer output size equal to the number of atoms $M$. In DSAC (Ma et al., 2020) one extra fully connected layer is used in the critic to obtain the embeddings of quantile fractions. Note that unlike TQC and AdaTQC, DSAC uses implicit quantile parameterization and samples quantile fractions...
Hyperparameter | AdaTQC | TQC | DSAC | SAC
--- | --- | --- | --- | ---
Optimizer | Adam | | | |
Learning rate | $3 \cdot 10^{-4}$ | | | |
Discount $\gamma$ | 0.99 | | | |
Replay buffer size | $1 \cdot 10^6$ | | | |
Number of critics $N$ | 2 | | | |
Number of hidden layers in critic networks | 3 | 2 | | |
Size of hidden layers in critic networks | 512 | | 256 | |
Number of hidden layers in policy network | 2 | | | |
Size of hidden layers in policy network | 256 | | | |
Minibatch size | 256 | | | |
Entropy target $H_T$ | $-\dim A$ | | | |
Temperature parameter $\alpha$ | auto | | | |
Nonlinearity | ReLU | | | |
Target smoothing coefficient $\beta$ | 0.005 | | | |
Target update interval | 1 | | | |
Gradient steps per iteration | 1 | | | |
Environment steps per iteration | 1 | | | |
Number of atoms $M$ | 25 | 25 | 32 | — |
Huber loss parameter $\kappa$ | 1 | 1 | 1 | — |
Quantile fractions | fixed | fixed | random | — |

Table 5: Hyperparameters values.

Hyperparameter | AdaTQC
--- | ---
Initial $\eta$ value | $0.04 = 1/M$
$\eta$ learning rate | $3 \cdot 10^{-6}$
Bias estimation interval | 5
Number of trajectories in ‘fresh’ replay buffer | 200
Number of states sampled per trajectory | 20

Table 6: AdaTQC-specific hyperparameters values.

randomly (Dabney et al., 2018a). The number of critics in TQC and AdaTQC is assumed to be 2 if not otherwise specified. The results for number of critics $N = 5$ are provided in Table 3.

To make the comparison to TQC and AdaTQC fair, $\alpha$ was adjusted automatically (Haarnoja et al., 2018b) for all methods (SAC, TQC, AdaTQC, DSAC).

AdaTQC-specific hyperparameters are listed in Table 7. “Bias estimation interval” stands for the amount of optimization steps between agg bias($\hat{Q}_\psi, \pi$) re-estimations. We do not recalculate aggregated bias every step to decrease the amount of computations required. $\eta$, on the other hand, is optimized at each step (basically, we make the same step on $\eta$ several times). “Number of trajectories in replay buffer” and “number of states sampled per trajectory” are the hyperparameters of the trajectory uniform aggregation distribution. In figure 6 we use $4000 = 200 \cdot 20$ states to estimate the bias with state uniform aggregation distribution, so the overall amount of states remains the same.

G On the optimization of $\eta$ in practice

Update rule provided in equation 2 is realized in practice via gradient steps with Adam optimizer (Kingma and Ba, 2015) on the following loss:

$$\eta \left( \text{agg bias}(\hat{Q}_\psi, \pi) \right)^- \rightarrow \max_{\eta}$$

(38)

where $(\ldots)^-$ means stop-gradient. Because Adam normalizes the gradients, in practice we get an optimization process where all absolute differences between neighboring values of $\eta$ are almost the same.
To not confuse the reader that the magnitude of bias matters in this optimization process, we have decided to write $\eta$ optimization in form 2.

**H  Exclusion of ’HalfCheetah-v3’ environment**

![Graph](image)

Figure 8: Comparison of different methods on ’HalfCheetah-v3’ environment. Performance is evaluated every 1000 steps. Curves are smoothed with a window of 100, each curve represents one seed of learning.

The MuJoCo environment ’HalfCheetah-v3’ has at least two clearly suboptimal convergence points for the policy: around evaluation returns 2500 and 8000. For example, the first point accounts for the case where the agent learns to crawl on his back instead of running. These additional convergence points significantly increase the variance of the training results, and multiple samples of training are required to estimate the performance of an algorithm correctly.

Sometimes additional exploratory steps are taken before the training begins. In this case, the environment will require the change in experiment protocol, and we did not want to do that. Either way, we decided to exclude this environment from the main experiments.

**I  Optimal values for d in TQC on PyBullet**

| Environment     | Humanoid | Walker2D | Hopper | HalfCheetah | Ant |
|-----------------|----------|----------|--------|-------------|-----|
| Optimal $d$     | 3        | 5        | 5      | 2           | 3   |

Table 7: Optimal values for the number of truncated atoms $d$ for TQC. The selection is based on 4 runs with each parameter value.

**J  Reduction of overestimation with the increase of the number of networks**

![Graph](image)

Figure 9: The average $\eta$ over the optimization trajectory of AdaTQC with different numbers of Z-functions for different environments. Error-bars indicate $\pm$std over 4 runs.

In figure 9 we can see that while the size of the critic ensemble increases, the average $\eta$ decreases. It shows that the higher number of estimators helps to eliminate the overestimation bias.