Learning to Walk Randomly by applying techniques to reduce overestimation

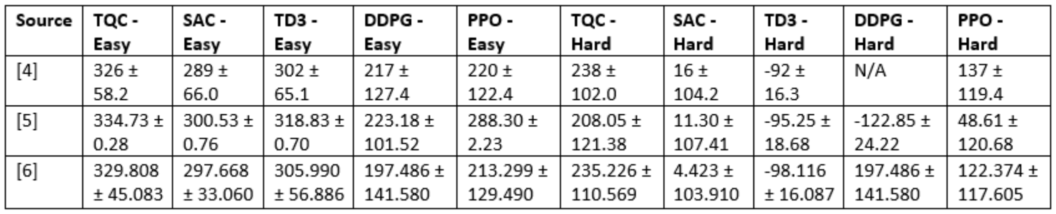
Anonymous author

Abstract

This paper proposes training a bipedal robot to learn to walk using Truncated Quantile Critic [8] as our baseline model. This was decided based on research comparison of other state-of-the-art algorithms that have attempted to solve the same environment. We proceeded to enhance this algorithm by integrating Adaptively Calibrated Critic [1] on top of Ada-TQC [9] to overcome overestimation bias. We found that the end result overtook our baseline algorithm and solved both Bipedal Walker environments.

1 Methodology

When exploring different algorithms for the Bipedal Walker, we referred to three sources [6] , [5] and [10] [3] which demonstrates the findings for the following algorithms shown in a summarised table in Table 1.



Source	TQC - Easy	SAC - Easy	TD3 - Easy	DDPG - Easy	PPO - Easy	TQC - Hard	SAC - Hard	TD3 - Hard	DDPG - Hard	PPO - Hard
[4]	326 ± 58.2	289 ± 66.0	302 ± 65.1	217 ± 127.4	220 ± 122.4	238 ± 102.0	16 ± 104.2	-92 ± 16.3	N/A	137 ± 119.4
[5]	334.73 ± 0.28	300.53 ± 0.76	318.83 ± 0.70	223.18 ± 101.52	288.30 ± 2.23	208.05 ± 121.38	11.30 ± 107.41	-95.25 ± 18.68	-122.85 ± 24.22	48.61 ± 120.68
[6]	329.808 ± 45.083	297.668 ± 33.060	305.990 ± 56.886	197.486 ± 141.580	213.299 ± 129.490	235.226 ± 110.569	4.423 ± 103.910	-98.116 ± 16.087	197.486 ± 141.580	122.374 ± 117.605

Table 1: Comparing the average return ± standard deviation for different algorithms

From these three different sources we concluded that TQC was by far the best algorithm to explore. Therefore, we will be studying [8] to find out more about the basis for this algorithm.

One of the biggest problems in Reinforcement learning is the overestimation bias which can occur when an agent estimates its own performance higher than it truly is. In off-policy learning, we aim to accurately approximate the Q-function, however it tends to be overestimated due to Jensen's inequality:

$$\max_a Q\left(s_{t+1}, a\right)=\max_a \mathbb{E}\left[Q\left(s_{t+1}, a\right)+\epsilon_a\right] \leq \mathbb{E}\left[\max_a\left\{Q\left(s_{t+1}, a\right)+\epsilon_a\right\}\right]$$

This error will be propagated through temporal difference learning and will harm the decisions the agent makes, hence the results as well.

1.1 TQC

Rather than using the Q function, distributional reinforcement learning approximates the random variable: $Z_\theta(s, a)$. [2]

TQC uses multiple approximations $\hat{Z}_{\psi_n}(s,a)$ of the policy conditioned return distribution $Z^\pi(s,a)$ each predicting the distribution from the QR-DQN method : $Z_{\theta_n}\left(s_t, a_t\right)=\frac{1}{M} \sum_{m=1}^M \delta\left(\theta_n^m\left(s_t, a_t\right)\right)$. supported on atoms $(\theta_1, \cdots, \theta_N)$. [1] These are trained with the temporal difference target distribution Y(s,a) and build the set: Z(s,a):=mn(s,a)n[1..N],m[1..M] with atoms of distribution: Z1(s,a),. . . ,ZN(s,a). Let $k \in \{1, \dots, M\}$, then the kN smallest of these targets define atoms: $y_m(s_t, a_t) = r_t + \gamma(\theta^m(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1} \mid s_{t+1}))$,via the Huber quantile loss, of the target distribution $Y(s_t, a_t) = \frac{1}{kN}\sum_{i=1}^{kN} \delta(y_i(s_t, a_t))$. When minimising the 1-Wasserstein distance between the atoms and $Y(s_t, a_t)$, we need to learn the locations for each quantile fraction: $\tau_m = \frac{2m-1}{m}, m \in \{1, \dots, M\}$ We then train to minimize the quantile Huber loss :

$$J(\phi) = \underset{\substack{s \sim \mathcal{D} \\ a \sim \pi}}{\mathbb{E}}[L(s_t, a_t; \theta_n)]$$

where:

$$L(s_t, a_t; \theta_n) = \frac{1}{kNM}\sum_{m,i=1}^{M,kN} \rho^H_{\tau_m}(y_i(s_t, a_t) - \theta^m_n(s_t, a_t)),$$

where $\rho^H_\tau(u) = |\tau - \nVdash(u < 0)|\mathcal{L}^1_H(u)$ and $\mathcal{L}^1_H(u)$ is the Huber loss. [2]

As in SAC, the policy is trained and we maximize the entropy penalized estimate of the Q-value by minimizing the loss [4] :

$$J(\phi) = \underset{\substack{s \sim \mathcal{D} \\ a \sim \pi}}{\mathbb{E}}\left[\alpha \log \pi_\phi(a \mid s) - \frac{1}{NM}\sum_{m,n=1}^{M,N} \theta^m_n(s, a)\right].$$

1.2 Enhancements

In [9] it is said that TQC is affected by η^{TQC} (number of truncated atoms) which has varying optimal values for each environment. [8]

[1] explains that though, TQC achieves impressive results, the parameter d, respresented as: $d = M - k$ (number of dropped targets) must be individually set for each environment.

Both [1] and [9] explain how these hyper-parameters were able to control overestimation bias, and we will be looking into these papers for our enhancements of TQC. For improving TQC we will be looking into two different models: Adaptively Calibrated Critics (ACC) and Adaptive TQC (Ada-TQC). Both methods are used in reducing overestimation bias in off-policy DRL, however, they take two different approaches to achieve this as described below:

1.2.1 ADA-TQC

The reduction of overestimation bias typically involves our controlling hyperparameter: , which changes in each environment which in turn reduces the sample efficiency. Therefore, Ada-TQC aims to change automatically without any outer hyperparameter tuning.

Ada-TQC relies on a data-driven estimation of the aggregated bias, which firstly requires a source of on-policy trajectories. For this we use a replay buffer D_{fresh} containing trajectories from several recent policies to obtain near on-policy trajectories. Secondly we require an unbiased state-action value function and this will use k-step discounted sum of future rewards:

$$\tilde{Q}(\rho) = \sum_{i=t}^{\min(t+k-1, T-1)} \gamma^{i-t} r_i + \underbrace{\gamma^k \hat{Q}_\psi(s_{t+k}, a_{t+k})}_{\text{if } t+k<T},$$

where $\rho = (s_t, a_t, r_t, \dots, r_{\min(t+k-1,T-1)}, s_{\min(t+k,T)})$ is a rollout starting at a state s_t and an action a_t. The rollout ρ contains at most k future steps. T is a length of the the episode representing done signal. These are combined to estimate the aggregated bias: where $\mathcal{B}$ is a

$$\widetilde{\text{aggbias}}(\hat{Q}_\psi, \mathcal{B}) = \frac{1}{|\mathcal{B}|}\sum_{\rho \in \mathcal{B}} \left(\hat{Q}_\psi(s_t, a_t) - \tilde{Q}(\rho)\right)$$

mini-batch of rollouts sampled from $\mathcal{D}_{\text{fresh}}$ and s_t, a_t are the starting state and action from rollout ρ.

We use D_{fresh}^{valid} instead of D_{fresh} when environments have a time limit, as future rewards may not be available for some state-action pairs. We also remove states with not enough future rewards from D_{fresh}^{valid} to obtain D_{fresh} , which requires at least k future rewards available. After "done" the rewards are available and will equal to zero.

To adjust during training, we state the bias at each optimization step t+1 relies on the hyper-parameter η_{t+1} through the parameters $_{t+1}$ of $\hat{Q}_{t+1}$. An increase in η_t reduces the overestimation at step t+1, while a decrease in η_t reduces the underestimation at step t+1. The bias control updates, for continuous hyper-parameters can be approximated as:

$$\eta_{t+1} = \eta_t + \lambda \operatorname{sign}\left(\nabla_\eta \operatorname{aggbias}\left(\hat{Q}_{\psi_{t+1}(\eta)}, \pi\right)\Big|_{\eta=\eta_t} \right),$$

where sign$(\cdot)$ is normalization which selects step-size λ independently of the absolute value of returns.

When comparing the methods, [2] uses $N = 2$ as the number of critics, the number of the dropped quantiles η^{TQC}were adjusted, and use $\eta^{TQC} \in \{0, 2, 4, 6, 8, 10\}$ as the grid for TQC.

The paper found that adaptive η converged towards best performing constant η-s for the majority of the environments.

1.2.2 ACC

Adaptively Calibrated Critics (ACC) uses a parameter that is updated by comparing the current Q-value estimates with the most recent on-policy returns, resulting in bias-calibrated TD targets. ACC incorporates information from unbiased sample returns into the TD targets while mitigating the high variance of the samples. With ACC, the parameter d can be automatically adjusted during training, eliminating the need for costly hyper-parameter tuning.

In preparation for our application of ACC on TQC we will set the following parameters: The maximum number of targets to drop per network is denoted by $d_{\max}$, and the current number of targets dropped is $d = d_{\max}$ - β, where d is continuous. Q_β is the estimate with dN targets dropped from the set of all targets, where $Q_{\beta_{\max}}$ is when no targets are dropped, and $Q_{\beta_{\min}}$ is dropped targets per net. β is a continuous value ranging from 0 to $d_{\max}$. The total number of dropped targets is rounded to the nearest integer when computing the temporal difference target because the number of dropped targets must be a discrete value in $\{0, \ldots, NM\}$.

ACC has Q-value estimators for each state-action pair (s, a), we have $\left\{\hat{Q}_\beta(s,a)\right\}_{\beta \in [\beta_{\min}, \beta_{\max}] \subset \mathbb{R}}$, for $Q(s, a)$ - the true Q-value of the policy π where: $\hat{Q}_{\beta_{\min}}(s,a) \leq Q(s,a) \leq \hat{Q}_{\beta_{\max}}(s,a)$and Q_β a continuous monotone increasing function in β.

Unbiased estimators of $Q(s, a)$ are obtained through Monte Carlo estimation. $\hat{Q}_{\beta^*}(s,a)$ is the estimator with $\beta^*(s, a)$ being the value of β that minimizes the difference:

$$\beta^*(s,a) = \underset{\beta \in [\beta_{\min}, \beta_{\max}]}{\arg\min} \left| \hat{Q}_\beta(s,a) - \frac{1}{N} \sum_{i=1}^{N} R_i(s,a) \right|$$

ACC can adaptively adjust the value of β^* using on-policy rollouts to reduce bias in the TD targets, eliminating the alternative need for extensive tuning as demonstrated with:

$$\beta_{\text{new}} = \beta_{\text{old}} + \alpha \sum_{t=1}^{T_\beta} \left[R(s_t, a_t) - \hat{Q}(s_t, a_t) \right],$$

where α is a step size parameter and $(s_t, a_t)_{t=1}^{T_\beta}$ are the $T_\beta \in \mathbb{N}$ most recent state-action pairs and the adjustment of β depends on the difference. β_{new} is then normalized by the moving average of the absolute difference between returns and estimated Q-values.

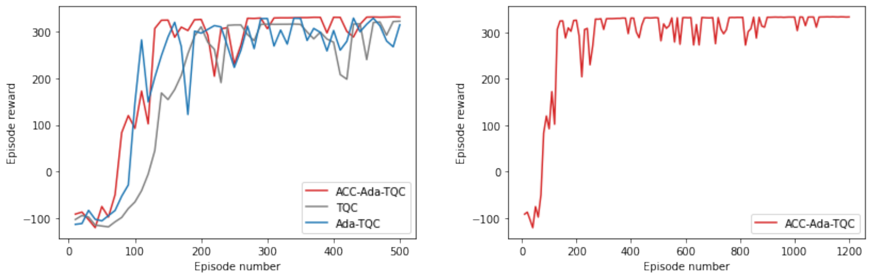
1.2.3 HYPERPARAMETERS

For our hyperparameters, we use the values found in all three papers: We keep our timesteps and $num_{episodes}$ equal to the assignment template given From [1] and [9] , their experimental results show that their hyperparameters are the same as TQC in all environments.

Hybrid When thinking about incorporating both these techniques, we wondered if using two algorithms for the same purpose would work or give us any benefit. However, from [1] we learnt that ACC could be applied to any algorithm that applies Q-value estimators therefore, we initialised that perhaps implementing Ada-TQC and then integrating ACC into the code by updating beta, states and including dropped quantiles in our trainer and also including the number of critics in our train function.

2 Convergence Results

The graph below shows our BipedalWalker-v3 results for 3 different models: TQC, Ada-TQC and our proposed ACC-Ada-TQC. Overall, our findings show the benefits of ACC-



Ada-TQC, where it shows better results than the other two models. Though, Ada-TQC is initally increasing greatly, our ACC-Ada-TQC is more consistent and eventually solves the easy environment first. However, for our Bipedal-Walker-Hardcore, though it has been solved we had a lack of time resources to produce good results hence, only producing 750 episodes. However, we can see its promising results, from graph, video and log evidence and we can predict a high convergence from this. Overall we found our highest score for our easy environment to be 334 within 1000 episodes and our hard environment to be 312 within 750 episodes.

3 Limitations and Future Work

We found that running on a CPU for training our enhancements, was extremely slow, yet despite using a GPU, it took at least 30 hours to reach 1000 episodes for the easy model due to the complexity of the algorithm. The issues with NCC prevented the ability to train our hardcore model better. In TQC, ensembling and distribution networks result in additional computational resources for maintenance and calculations [7] . Furthermore, despite ACC's and Ada's minimal computational costs in comparison to TQC, they both inevitably increase the complexity of the computations. ACC adds costs from its use of multiple critics and ensembling [1] , whilst Ada-tqc also has its calculations from the adaptive sampling step [9] . As a result, in future work we would aim to train it on better GPU resources, and with more time available. Another limitation we had was the reliability of our hyperparameters. Whilst we have several sources to back up our choices, these state-of-the-art models do not have much research for the Bipedal Walker or the new combined model we proposed. As a result, if we did have more time resources, we would use the Optuna library to find the optimal parameters for our model.

References

[1] Dorka N et al. "Adaptively Calibrated Critic Estimates for Deep Reinforcement Learning". In: (2022), pp. 624–631.

[2] *Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics.* `https://deepai.org/publication/controlling-overestimation-bias-with-truncated-mixture-of-continuous-distributional-quantile-critics`. Accessed: 2022-07-01.

[3] Sean Gillen, Asutay Ozmen, and Katie Byl. "Direct random search for fine tuning of deep reinforcement learning policies". In: *arXiv preprint arXiv:2109.05604* (2021).

[4] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[5] *Hugging Face Bipedal Walker Hardcore-v3 models.* `https://huggingface.co/models?other=BipedalWalkerHardcore-v3`. Accessed: 2022-07-01.

[6] *Hugging Face Bipedal Walker-v3 models.* `https://huggingface.co/models?other=BipedalWalker-v3`. Accessed: 2022-07-01.

[7] *Hyperparameters in Deep RL.* `https://towardsdatascience.com/hyperparameters-in-deep-rl-f8a9cf264cd6`. Accessed: 2022-07-01.

[8] A et al Kuznetsov. "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1587–1596.

[9] Arsenii Kuznetsov et al. "Automating Control of Overestimation Bias for Reinforcement Learning". In: *arXiv preprint arXiv:2110.13523* (2021).

[10] *Parameter Sharing Paper.* `https://github.com/jkterry1/parameter-sharing-paper/blob/9282494755d31857614235d5f57555bd7c3bf6d2/logs/benchmark/benchmark.md`. Accessed: 2022-07-01.