Bipedal Walking by Reinforcement Learning

Anonymous author

Abstract

This paper proposes using Truncated Quantile Critics (TQC) with D2RL architecture and mixed-strategy buffer comprising Emphasising Recent Experiences (ERE) and Prioritised Experience Replay (PER). It found promising hyperparameters through a Bayesian search, and the resulting agent is able to solve both BipedalWalker and BipedalWalkerHardcore.

1 Methodology

We propose using Truncated Quantile Critics (TQC) [8] with D2RL architecture [11], with an ERE+PER buffer [14, 10]. TQC was found to perform better than Soft Actor-Critic (SAC) [6] and Twin-Delayed Deep Deterministic (TD3) [5] on BipedalWalker; it converged the fastest and provided excellent sample efficiency. A thorough parameter sweep was conducted with Weights & Biases [3] using a Bayesian search with a Gaussian Process Regression surrogate model. Table 1 shows the five best runs from the sweep on balance for convergence speed and video quality on BipedalWalker.

Batch Size	Num Critics	Num Quantiles	Tau	Initial Steps	Buffer Size	Learning Rate	Discount (γ)	Episodes to 300
256	2	20	0.02	10000	$0.1*10^6$	0.000377	0.99	**100**
64	5	15	0.01	1000	$0.2*10^6$	0.000383	0.99	**100**
64	10	20	0.02	100	$1*10^6$	0.000343	0.99	**100**
64	5	30	0.02	10000	$1*10^6$	0.000464	0.99	**120**
64	5	30	0.005	10000	$0.1*10^6$	0.000289	0.99	**140**

Table 1: These hyperparameters were found to impact performance the most in the normal environment. Final hyperparameters were taken to be the most common as shown here and the mean for learning rate. Format inspired by [12].

The hardcore environment parameters were informed by this but heuristically adjusted to account for the extra difficulty. For example, buffer size on normal was 0.1M transitions, but we set it to 1M in hardcore because there are a lot more useful memories to store.

1.1 Truncated Quantile Critics with D2RL

TQC has three key elements: (1) distributional critics as in QR-DQN [4], (2) ensembling of critics and (3) truncating the right tail of this mixture of distributions. These allow TQC to effectively mitigate overestimation bias [13], which is prevalent in off-policy learning. This is when the Q-function overestimates the action values, as shown by Jensen's inequality:

$$\mathbb{E}[\max\{Q(a) + U(a)\}] \geq \max \mathbb{E}[Q(a) + U(a)] = \max Q(a) \tag{1}$$

for any action-dependent random noise $U(a)$ such that $\forall a\ \mathbb{E}_U[U(a)] = 0$. This bias results in the policy exploiting the critic's erroneous estimates. Temporal difference learning is particularly susceptible as these errors propagate backwards through episodic time and accumulate, potentially causing a positive feedback loop. As such, alleviating overestimation is crucial for good performance.

In TQC, the process is as follows (Fig. 1): we first feed the next state s' and corresponding action a' to N approximators, resulting in N sets of M predicted atoms. These $N * M$

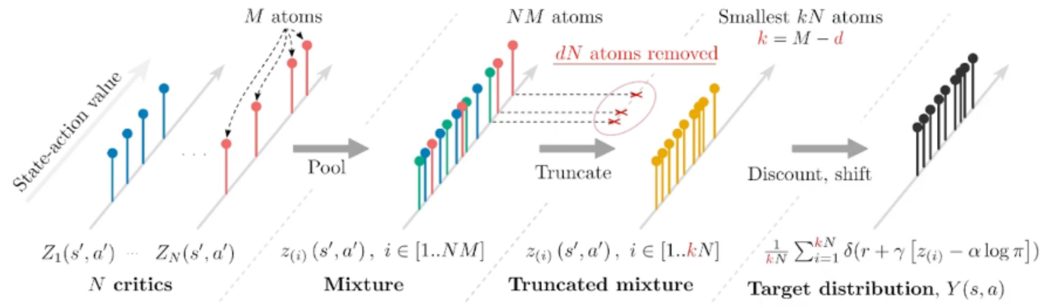

State-action value
M atoms
$Z_1(s',a')$ ··· $Z_N(s',a')$
N critics
Pool
NM atoms
$z_{(i)}(s',a')$, $i \in [1..NM]$
Mixture
dN atoms removed
Truncate
$z_{(i)}(s',a')$, $i \in [1..kN]$
Truncated mixture
Smallest kN atoms
$k = M - d$
Discount, shift
$\frac{1}{kN}\sum_{i=1}^{kN}\delta(r+\gamma[z_{(i)} - \alpha\log\pi])$
Target distribution, $Y(s,a)$

Figure 1: The Truncated Quantile Critics (TQC) process. Image adapted from [8].

atoms are then pooled together into a mixture, which is truncated by removing $d * N$ atoms from the right tail (d for every critic). The Q-value is estimated as the expectation of the location of the remaining atoms. The truncation accounts for overestimation due to high return variance: a higher variance implies lower Q-value estimate post-truncation. This provides fine-grained control over the balance between under- and over-estimation, as well as decoupling overestimation mitigation from the critic ensemble, providing further performance boost. Crucially, pooling must occur before truncation, as if each critic is truncated individually then a single critic may vastly overestimate and not be sufficiently cut down to reduce bias.

The temporal difference target distribution can be calculated:

$$Y(s,a) := \frac{1}{kN}\sum_{i=1}^{kN}\delta\Big(r(s,a) + \gamma[z_{(i)}(s',a') - \alpha\log\pi_\phi(a'|s')]\Big). \tag{2}$$

where r is the reward, γ is the discount and α is the entropy coefficient. We apply quantile regression [4] loss between the target and each critic, similarly to QR-DQN [4]. This has the effect of critics learning to predict the locations of quantiles in the target distribution, which equivalently minimises the 1-Wasserstein distance between target and critic [8]

1.1.1 D2RL

We employ a D2RL architecture [11] for the actor and critics as this has been shown to work effectively with BipedalWalker [12, 9]. D2RL uses dense connections and concatenates the state or state-action pair to the output of each hidden layer. This overcomes the performance degradation that can occur with deeper nets as the connection between input and output is damaged. Simultaneously, the improved feature representation due to network depth boosts performance.

1.2 Prioritising Recent Experiences

1.2.1 ERE

The Emphasising Recent Experiences (ERE) replay buffer [14] prioritises sampling recent experiences as opposed to uniform sampling. This is motivated by the fact that early on in training, actions are suboptimal and quickly become obsolete which means that recent memories are more likely to contain high-quality samples. ERE (1) samples more recent data more frequently and (2) arranges updates to avoid older data overwriting fresher data. Mathematically, standard ERE samples uniformly from the most recent c_k data points, where:

$$c_k = \max\{N \cdot \eta^{k\frac{1000}{K}}, c_{\min}\} \tag{3}$$

where $\eta \in (0,1]$ is a hyperparameter determining the emphasis to place on recent data and $k * 1000/K$ is a measure of progress through K mini-batch updates such that the first mini-batch update has uniform sampling and the last has η^{1000}. The c_{min} parameter prevents sampling from a very small amount of data (which would likely cause overfitting) by providing a minimum number of samples required. η is also annealed over the entire training process towards 1, which represents uniform sampling, as shown:

$$\eta_t = \eta_0 + (1 - \eta_0) \cdot \frac{t}{T} \tag{4}$$

where η_0 is the initial eta, t is the number of timesteps so far and T is the total number of timesteps. Selecting T can be hard since episodes do not always meet the maximum number of timesteps. Continually decrementing T to account for some episodes finishing early would cause eta to anneal much faster over successive short episodes than over longer ones. As such, we heuristically fixed T at the start of training to the maximum number of timesteps allowed in training, rather than treating it as a tuned hyperparameter.

1.2.2 PER

The Prioritized Experience Replay (PER) [10] buffer ideally prioritises sampling memories that an agent could best learn from in its current state. This is motivated by the fact that typical uniform sampling can miss crucial but rare memories in favour of one of the many poor, unhelpful ones. It approximates the usefulness of a state by its TD error δ - essentially a measure of how 'surprising' a given state is [1]. However, a greedy strategy that always selects the transition with the highest TD error fails due to lack of observation diversity. As such, PER implements proportional prioritisation by interpolating between pure greedy prioritisation and uniform random sampling. Formally, a transition i is selected with probability:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{5}$$

where α is a hyperparameter to control the level of prioritisation used. Note that $\alpha = 0$ is uniform sampling. The priority of a transition is $p_i = |\delta_i| + \epsilon$ where ϵ is a small constant that prevents any priorities being zero. The priorities are efficiently stored in and sampled from a 'sum-tree' data structure where every node is the sum of its children, and the leaf nodes are priorities. This means the root note is the sum of all priorities, p_{total}. To sample a minibatch of size k, divide $[0, p_{total}]$ into k equally-sized ranges and sample uniformly from each range. The transitions that correspond to these are sampled from the tree. This enables $\mathcal{O}(\log N)$ updates and sampling.

Estimating a value via stochastic updates assumes that it is in the same distribution as the value's expectation. Since prioritised sampling creates a bias by disrupting this, importance sampling (IS) weights are used during gradient descent:

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\beta \tag{6}$$

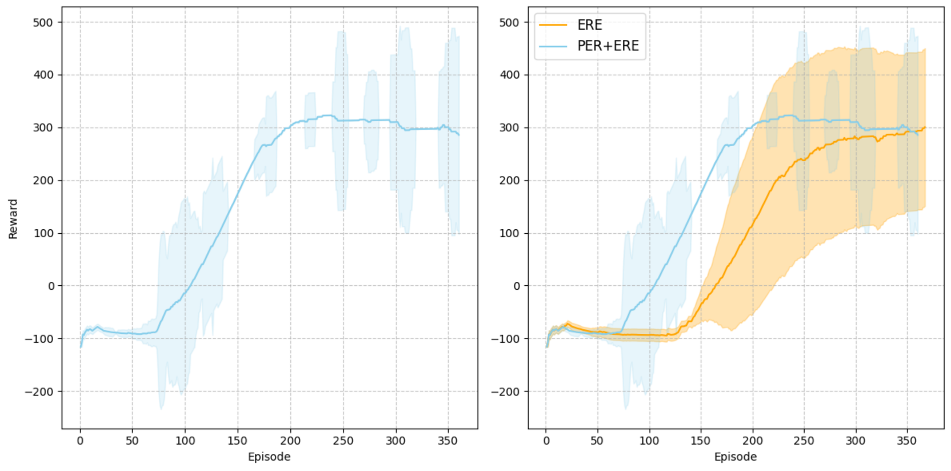
where $w_i * \delta_i$ is used in the Q-update instead of just δ_i. We also define a schedule to linearly anneal β towards 1 which increases the IS correction over training. This enables us to encourage more prioritisation early on in training, which is important in BipedalWalker when a lot of mistakes are made.

1.3 Algorithm

The algorithm proceeds as follows: initialise the policy, critic and critic target as well as the ERE+PER replay buffer D as the empty set. The entropy target is set to $-\dim(\mathcal{A})$ where $\mathcal{A}$ is the action space, which is a familiar heuristic [6]. For each episode, sample from the environment according to a policy. If the agent falls over and receives a reward of -100, this is adjusted to -10 to improve training stability. Otherwise, the reward is squared to further encourage forward motion. Append this experience to D. After completing initial random steps to improve exploration, we take a gradient step on the policy every episode where we sample from D and update the entropy temperature coefficient α, policy and two critics as in Kuznetsov et al. [8]. If using PER, the IS weights are applied as previously discussed to the quantile loss for the critics.

2 Convergence Results

The agent successfully solved the normal and hardcore environments in about 74 and 850 steps respectively. Below we present the average reward over training on the normal environment (left) and an ablation study demonstrating the performance enhancement from PER (right). We plot the mean over the last 100 episodes (or all episodes if less than 100 have happened) and shade the standard deviation on either side. Note that one plot is truncated due to exiting the environment early after reaching an average score above 300 over 100 episodes.



The oscillating variance past episode 150 (many episodes after the agent solved the environment) is likely due to PER's prioritisation of high TD-error transitions: surprising transitions may not be good to learn from after completing the environment. This is expected and built into the priority annealing, so these oscillations eventually disappear. However, this same mechanism allows it to converge much faster as shown in the ablation study.

3 Limitations and Future Work

Although it provides good performance, the distributional critic ensemble of TQC is computationally expensive and can be a bottleneck in training. To alleviate this, several critics may be able to be trained in parallel on different environments and hardware, though this provides extra complexity. The ERE buffer performed well, and the PER was less sample efficient for BipedalWalker as well as adding further complexity. However, it did help to reduce convergence times by taking advantage of older memories with high TD error which ERE may overlook due to recency bias. Tuning PER's hyperparameters specifically in conjunction with ERE might yield better results.

This approach is model-free. Sample efficiency could be drastically improved with a 'Dreamer' world model [7], and intrinsic rewards may be useful to encourage the agent to explore where the model is uncertain [2].

References

[1] David Andre, Nir Friedman, and Ronald Parr. “Generalized prioritized sweeping”. In: *Advances in neural information processing systems* 10 (1997).

[2] Arthur Aubret, Laetitia Matignon, and Salima Hassas. “A survey on intrinsic motivation in reinforcement learning”. In: *arXiv preprint arXiv:1908.06976* (2019).

[3] Lukas Biewald. *Experiment Tracking with Weights and Biases.* 2025. URL: `https://www.wandb.ai/`.

[4] Will Dabney et al. “Distributional reinforcement learning with quantile regression”. In: *Proceedings of the AAAI conference on artificial intelligence.* Vol. 32. 1. 2018.

[5] Scott Fujimoto, Herke Hoof, and David Meger. “Addressing function approximation error in actor-critic methods”. In: *International Conference on Machine Learning.* PMLR. 2018, pp. 1587–1596.

[6] 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.

[7] Danijar Hafner et al. “Mastering diverse domains through world models”. In: *arXiv preprint arXiv:2301.04104* (2023).

[8] Arsenii Kuznetsov et al. “Controlling overestimation bias with truncated mixture of continuous distributional quantile critics”. In: *International conference on machine learning.* PMLR. 2020, pp. 5556–5566.

[9] Arijus Lengvenis. *Dreamwalker: Sample Efficient Soft Actor-Critic Agent with Dreamer Model.* `https://github.com/AirjusLengvenis/bipedal-walker-dreamer`. Accessed: 4 May 2025.

[10] Tom Schaul et al. “Prioritized experience replay”. In: *arXiv preprint arXiv:1511.05952* (2015).

[11] Samarth Sinha et al. “D2rl: Deep dense architectures in reinforcement learning”. In: *arXiv preprint arXiv:2010.09163* (2020).

[12] Lauren Stumpf. *TQC-D2RL: Truncated Quantile Critics for Practical Deep Reinforcement Learning.* `https://github.com/Lauren-Stumpf/TQC-D2RL`. Accessed: 4 May 2025.

[13] Sebastian Thrun and Anton Schwartz. “Issues in using function approximation for reinforcement learning”. In: *Proceedings of the 1993 connectionist models summer school.* Psychology Press. 2014, pp. 255–263.

[14] Che Wang and Keith Ross. “Boosting soft actor-critic: Emphasizing recent experience without forgetting the past”. In: *arXiv preprint arXiv:1906.04009* (2019).