A. Experimental Setting

We would like to caution about the use of MuJoCo 2.0 with versions of Gym at least up to `v0.15.4` (the last released at the moment). For these versions Gym incorrectly nullifies state components corresponding to contact forces, which, in turn makes results incomparable to previous works.

In our work we use MuJoCo 1.5 and `v3` versions of environments. Versions of all other packages we used are listed in the Conda environment file, distributed with the source code[6].

B. Hyperparameters

Critic networks are fully-connected, with the last layer output size equal to the number of atoms M.

Table 5. Hyperparameters values.

Hyperparameter	TQC	SAC
Optimizer	Adam	
Learning rate	$3 \cdot 10^{-4}$	
Discount γ	0.99	
Replay buffer size	$1 \cdot 10^6$	
Number of critics N	5	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 $\mathcal{H}_T$	$-\dim \mathcal{A}$	
Nonlinearity	ReLU	
Target smoothing coefficient β	0.005	
Target update interval	1	
Gradient steps per iteration	1	
Environment steps per iteration	1	
Number of atoms M	25	—
Huber loss parameter κ	1	—

Table 6. Environment dependent hyperparameters for TQC.

Environment	Number of dropped atoms, d	Number of environment steps
Hopper	5	$3 \cdot 10^6$
HalfCheetah	0	$5 \cdot 10^6$
Walker2d	2	$5 \cdot 10^6$
Ant	2	$5 \cdot 10^6$
Humanoid	2	$10 \cdot 10^6$

C. Toy Experiment Setting

The task is an infinite horizon MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, p_0)$ with a single state $\mathcal{S} = \{s_0\}$ and one-dimensional continuous action space $\mathcal{A} = [-1, 1]$. Since there is only one state, the state transition function $\mathcal{P}$ and initial state distribution p_0 are delta functions. On each step agent receives stochastic reward $r(a) \sim f(a) + \mathcal{N}(0, \sigma)$, where $\sigma = 0.25$. The mean reward function is the cosine with slowly increasing amplitude (Figure 9):

$$f(a) = \left[A_0 + \frac{A_1 - A_0}{2}(a+1)\right] \cos \nu a, \quad \text{where } A_0 = 0.3; \quad A_1 = 0.9; \quad \nu = 5 \tag{19}$$

[6] https://github.com/bayesgroup/tqc

The reward function gives rise to three locally optimal actions: near the left end, $a \approx -0.94$, in the right half, $a^* \approx 0.31$ (global) and at the right end $a = 1$. The optimal policy in this environment always selects $a^* = \arg\max_a f(a)$. The discount factor is $\gamma = 0.99$.

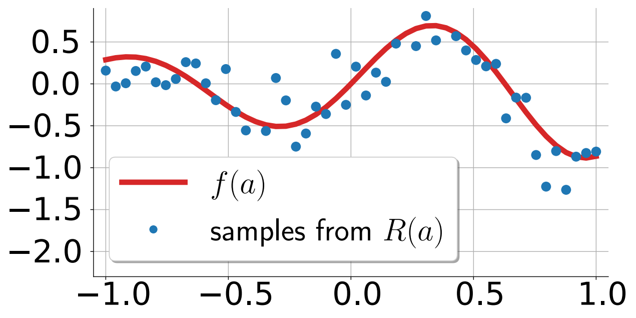

Figure 9. Reward function. x-axis represents one dimensional action space, y-axis - corresponding stochastic rewards and their expectation.

We evaluate three bias correction techniques (Table 1). We train Q-networks (or Z-networks, depending on the method) with two hidden layers of size 50 on the buffer of size 50 for 3000 iterations. Each iteration includes Temporal Difference (TD) target computation on all elements of the buffer and a gradient step minimizing the loss function specific to a particular method. We populate the buffer only once, at the start, with 50 pairs (a, r): we sample a reward for each of the actions from the uniform grid.

For TD target computation, instead of the target network, we plugged in the approximation and stopped gradient, as usual. To prevent interference of policy optimization into conclusions about Q-function approximation quality, we use implicit greedy deterministic policy induced by value networks: the argmax of the approximation, listed in Table 1. To find the maximum, we evaluated the approximation over the dense uniform grid over dense uniform grid of actions of size 2000 .

For each method we vary the parameter responsible for overestimation control. For AVG and MIN we vary the number of networks N from $[3, 5, 10, 20, 50]$ and $[2, 3, 4, 6, 8, 10]$ correspondingly. For TQC — the number of dropped atoms per network $d = M - k$ from $[0, 1, 2, 3, 4, 5, 6, 7, 10, 13, 16]$ out of 25.

For each variation we report the robust average (10% of each tail is truncated) over 100 seeds of $\mathbb{E}_{a\sim\mathcal{U}(-1,1)}\left[\Delta(a)\right]$ and $\mathbb{V}\text{ar}_{a\sim\mathcal{U}(-1,1)}\left[\Delta(a)\right]$. The $\Delta(a) := \widehat{Q}^\pi(a) - Q^\pi(a)$ is a signed discrepancy between the approximate and the true Q-value. For example, for TQC $\widehat{Q}^\pi(a) = \mathbb{E}\widehat{Z}^\pi(a) = \frac{1}{kN}\sum_{i=1}^{kN} z_{(i)}(a)$. Expectation and variance are estimated over dense uniform grid of actions of size 2000.

We present the results in Figure 4 with bubbles of diameter, inversely proportional to the averaged over the seeds absolute distance between the optimal a^* and the $\arg\max$ of the policy objective.

D. Extended Ablation

D.1. Number of critics

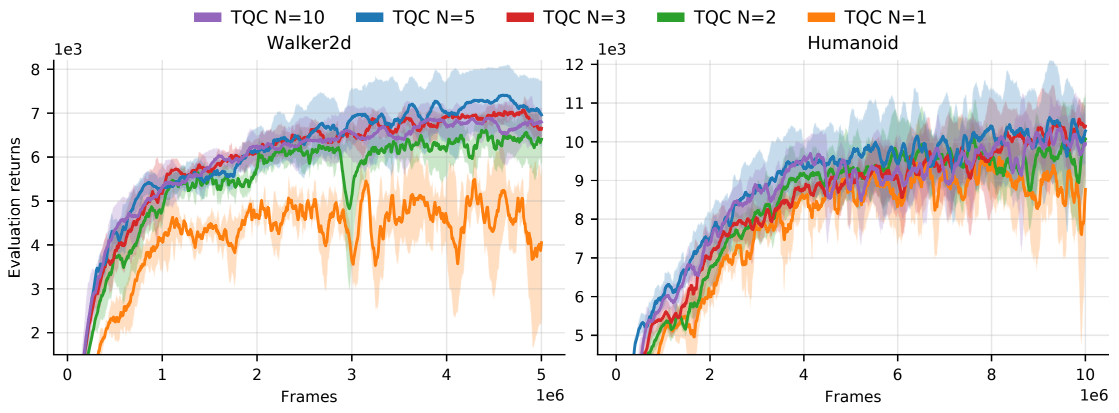

Figure 10. Varying the number of critic networks N for TQC with $M = 25$ atoms per critic and $d = 2$ of dropped atoms per critic. Smoothed with a window of 100, $\pm$ std is plotted.

D.2. Total number of atoms M

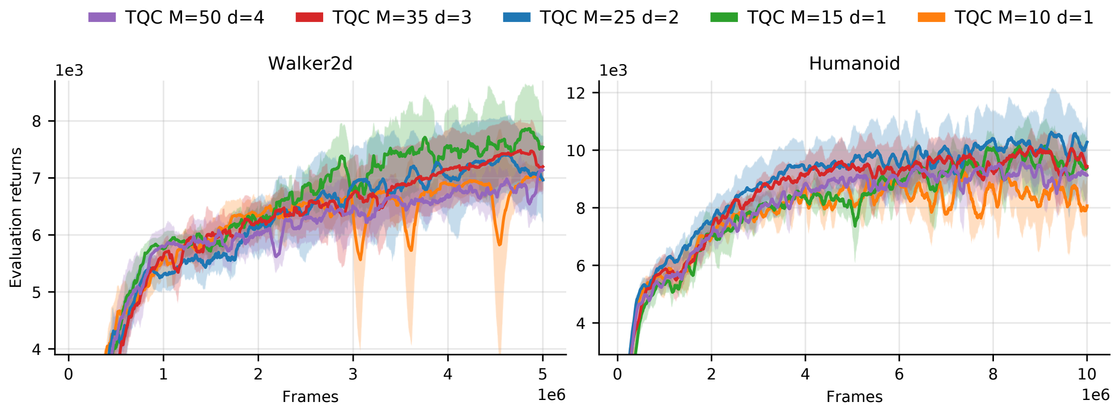

Figure 11. Varying the number of atoms per critic M for TQC with $N = 5$ critics and corresponding d to keep approximately the same proportion of dropped atoms. Smoothed with a window of 100, $\pm$ std is plotted.

D.3. Varying the number of dropped atoms for Hopper and HalfCheetah

As always, for TQC we remove dN atoms with largest locations from the mixture composed of individual critics predictions. d controls the intesity of overestimation control, while N defines the size of the ensemble.

Figure 12 shows the dependence of TQC on d for Hopper and HalfCheetah. The results indicate that the performance increases with an increase in d for Hopper and with a decrease in d for Halfcheetah. This provides additional evidence for the per-environment optimal level of overestimation also reported by Lan et al. (2020).

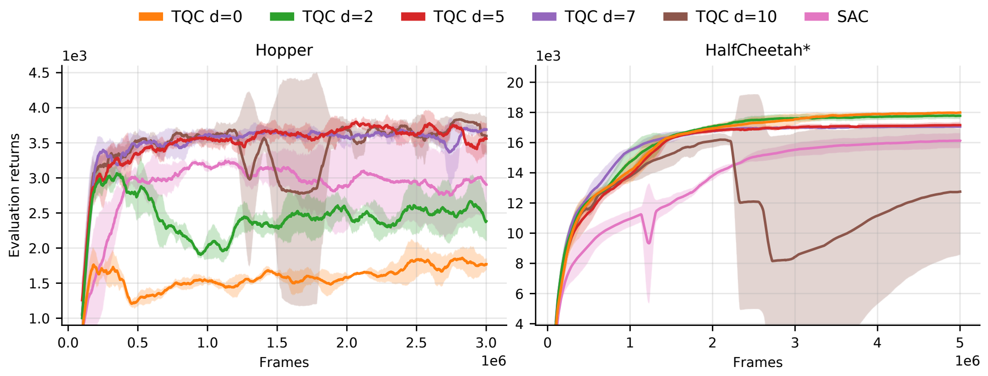

Figure 12. Varying the number of dropped atoms per critic d. $N = 5$ networks, $M = 25$ atoms. Smoothed with a window of 100, $\pm$ std is plotted. The stared HalfCheetah means extended number of initial exploration steps, as is suggested by https://github.com/rail-berkeley/softlearning/issues/75. The additional exploration prevent premature convergence and better conveys the dependence on d.

E. Application of TQC components to TD3

Could distributional critics, ensembling and truncation improve performance of other algorithms? Figure 13 quantifies performance improvements brought by various modifications of TQC on top of the TD3 (Fujimoto et al., 2018b). For TD3 the largest part of the improvement originates from the distributional critic architecture.

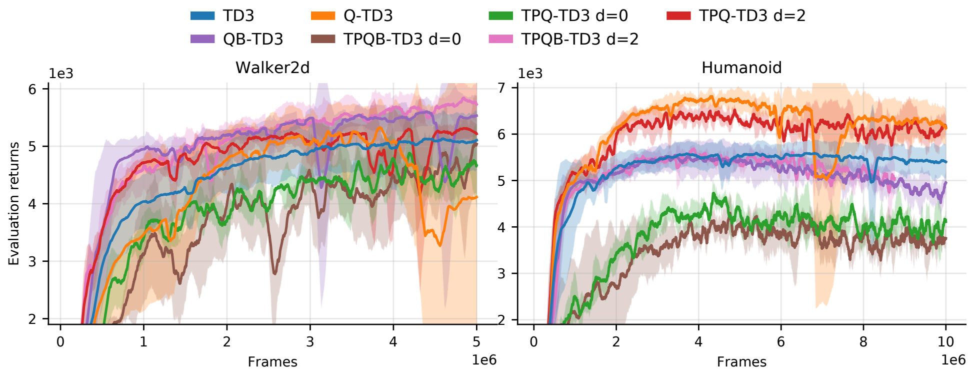

Figure 13. Applying **Q**uantile, **B**ig, **T**runcated and **P**ooled modifications to TD3 algorithm with $d \in \{0, 2\}$ (where applicable). The naming convention is consistent with the ablation section 5. $N = 5$ networks, $M = 25$ atoms. Smoothed with a window of 100, $\pm$ std is plotted.

F. Various Types of Overestimation Correction with Small and Big Networks

F.1. Clipped Double Q-learning

To ensure that it is not possible to match the performance of TQC with careful tuning of previous methods, we varied the number of critic networks used in the Clipped Double Q-learning estimate (Fujimoto et al., 2018b) for SAC (Haarnoja et al., 2018b). The larger the number of networks under the $\min$, the more the underestimation (Lan et al., 2020).

We have found that for MuJoCo benchmarks it is not possible to improve performance upon the published results by controlling the overestimation in such a coarse way for both the regular network size (Figure 14), and for the increased network size (Figure 15).

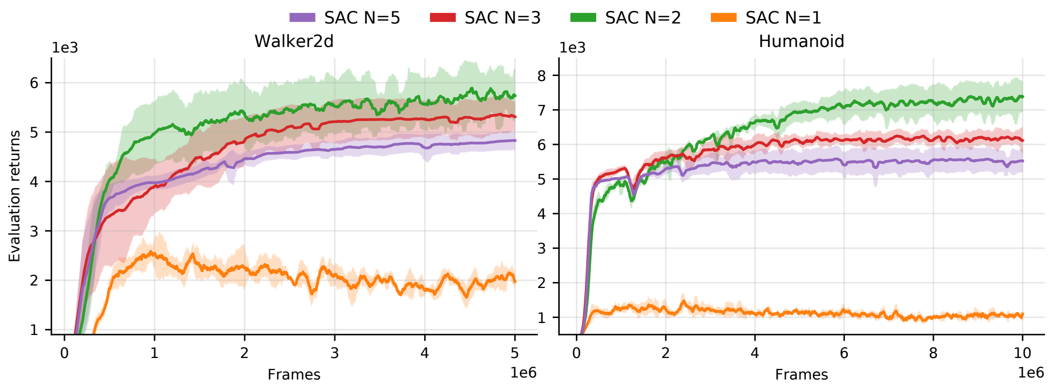

Figure 14. Varying the number of critic networks N under the $\min$ operation of the Clipped Double Q-learning estimate for SAC. Each critic networks is 2 layers deep with 256 neurons in each layer (the same network structure as in SAC (Haarnoja et al., 2018b)). Smoothed with a window of 100, ± std is plotted.

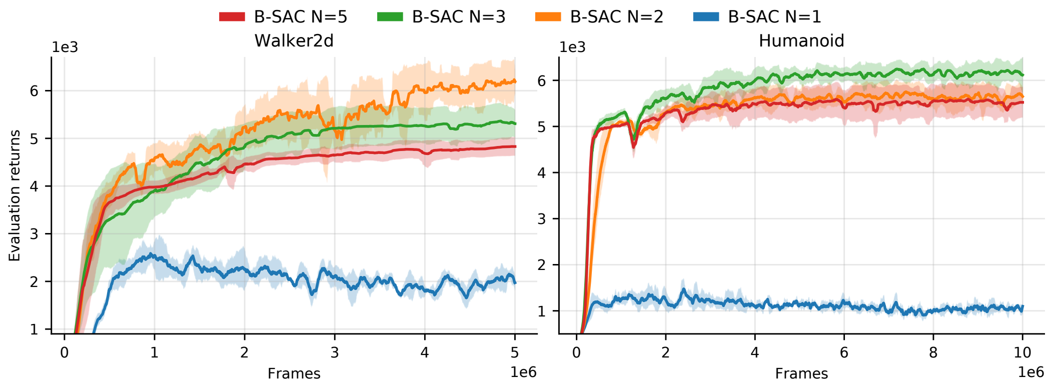

Figure 15. Varying the number of critic networks N under the $\min$ operation of the Clipped Double Q-learning estimate for SAC. Each critic networks is 3 layers deep with 512 neurons in each layer. Smoothed with a window of 100, ± std is plotted.

F.2. k-SAC and kQ-SAC

Additionally, we benchmarked a modification of the Clipped Double Q-learning. The modification, introduced by Fujimoto et al. (2018a), which we here call *k-SAC*, uses a convex combination of two Q-values to compute the target:

$$y(s,a) = r + \gamma \left[k \min\left(Q_1(s',a'), Q_2(s',a')\right) + (1-k)\max\left(Q_1(s',a'), Q_2(s',a')\right) - \alpha \log \pi(a'|s')\right] \tag{20}$$

Hyperparameter $k \in [0,1]$ interpolates between the usual Clipped Estimate ($k=1$) and its upper bound: $\max$ of two Q-functions ($k=0$). By benchmarking k-SAC we can test if the overestimation compensation of the usual estimate ($k=1$) is excessive.

Figures 16 and 17 shows the performance of k-SAC and its Quantile modification (see Section 5 for details). For both architectures performance increases with an increase in k, culminating at $k = 1$ for almost all settings. This suggest, that for SAC and QSAC the Clipped Estimate may be very close to optimal.

Interestingly, QSAC benefits much more than SAC from the increased network size .

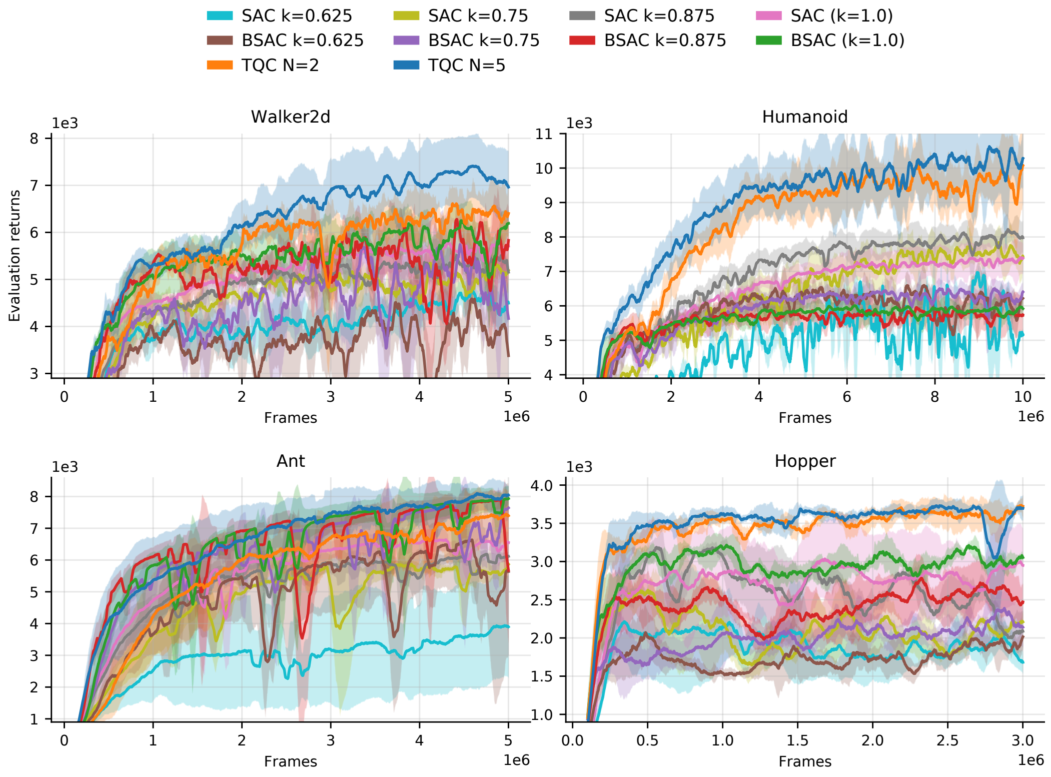

Figure 16. Varying the k in the convex combination of $\min$ and $\max$ over two critic networks. Big SAC, BSAC, has 3 layers of 512 neurons each. Smoothed with a window of 100, $\pm$ std is plotted.

F.3. TQC with small and large networks

In Appendx F.1 we have compared SAC with small and big networks. In Figure 18 we report the performance of TQC with small and big networks. The performance degradation due to a reduction in network size is almost invisible on Humanoid, while being large and noticable on Walker for $N = 5$ and $N = 2$ respectively.

G. Measuring Overestimation in MuJoCo

Usually, overestimation is measured as the difference between the estimated value $\widehat{Q}^\pi$ and its true counterpart. In complex environments, such as MuJoCo, we do not know the true Q-values. Thus, we have to rely on Monte Carlo estimation and compare the $\widehat{Q}^\pi$ with Q^π_{MC}. Usually (Fujimoto et al., 2018a; Van Hasselt et al., 2015) the reported metric is the difference between the predicted value and estimated true value, averaged over the state and action distributions.

Results. Figure 19 shows distributions of $\Delta(s,a) = \widehat{Q}^\pi(s,a) - Q^\pi_{\text{MC}}(s,a)$ at the end of the training for TQC, SAC, k-SAC (see Appendix F.2 for details on k-SAC). Each violin in the figure corresponds to 400 points: 100 state-action samples per each of the 4 trained agents. To estimate Q^π_{MC} we made 20 rollouts. By varying hyperparameters responsible for the overestimation control, we can match average overestimation level of TQC and SAC. However, as we see in Figure 19, the level of overestimation does not fully explain the difference in performance.

While interpreting the average Δ it is important to keep in mind the following.

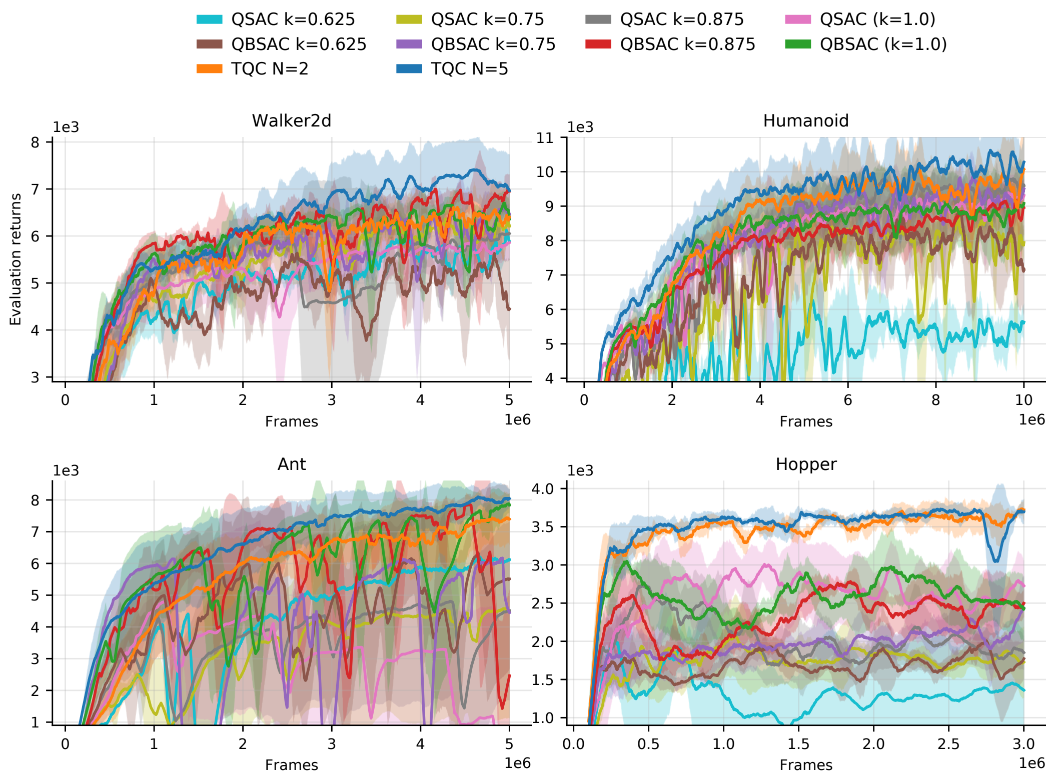

Figure 17. Varying the k in the convex combination of $\min$ and $\max$ over two critic networks. Big QSAC, QBSAC, has 3 layers of 512 neurons each. For the details on the learning process of QSAC and QBSAC see Section 5. Smoothed with a window of 100, $\pm$ std is plotted.

1. **Incompleteness of measurements**. We may miss crucial overestimations because of two factors: space and time. First, the state-action product space is multidimensional, so the Monte Carlo estimation is likely to miss some of overestimations. Second, overestimations change in time during the optimization, and there is a chance to miss some wild overestimation occurring at the moment different from the snapshot dump.

2. **Issues of averaging**. Averaging of many harmless underestimations ($\Delta < 0$) may compensate a harmful overestimation ($\Delta > 0$) leading to zero average difference. Thus, the small average Δ does not mean there are no *harmful* overestimations.

3. **Existence of harmless overestimations**. Some overestimations are harmless: e.g. a constant shift in all state-action values does not change optimization at all. Also, the policy may be unaffected by some erroneous estimates due to insufficiently flexible approximation family or due to the noise in gradients.

4. **Existence of useful overestimations**. Overestimation is not 100% evil. Previous works (Lan et al., 2020) showed an agent can benefit from overestimations: High return stochasticity is believed to promote overestimations, which, in turn, promote exploration. If the return stochasticity correlates positively with average returns, overestimations help an agent to find those regions faster.

How to account for these peculiarities and what can serve a reliable measure of overestimation remain an open question.

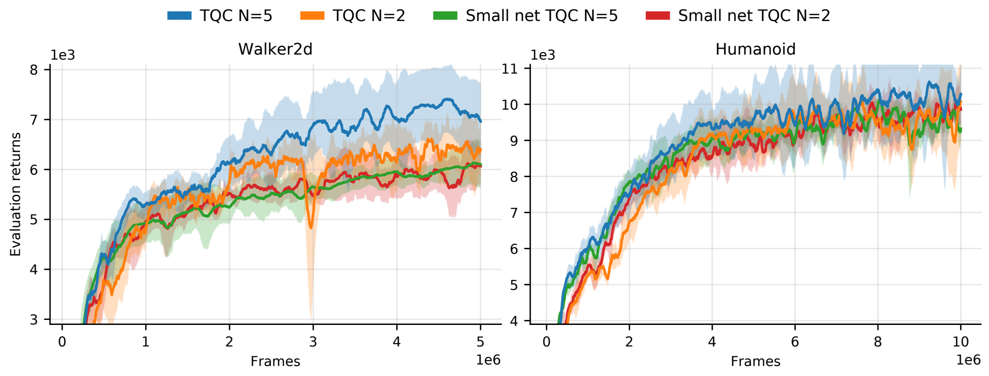

Figure 18. TQC small and big networks, with ensemble size $N \in \{2, 5\}$. As before, the small network consists of 2 layers of 256 neurons each. The big network consists of 3 layers of 512 neurons each. Smoothed with a window of 100, $\pm$ std is plotted.

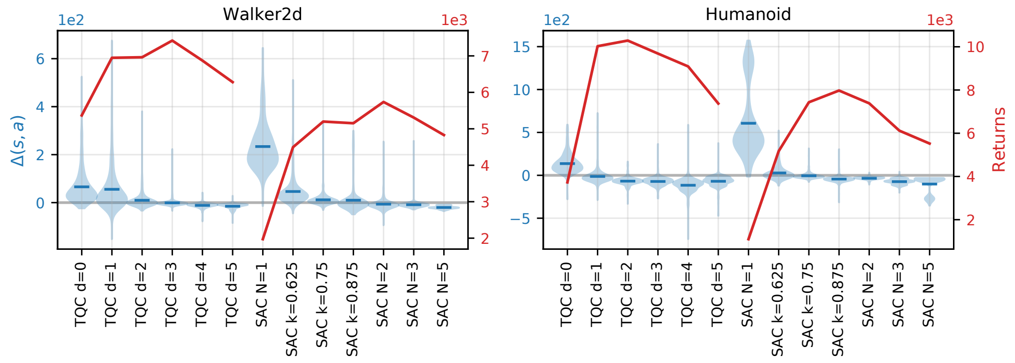

Figure 19. The dependence of the performance (red) and the distribution of value estimation error (blue) for TQC, SAC and k-SAC varying hyperparameters responsible for overestimation control. A blue bar depicts the mean of the distribution.

H. Removed Atoms Stats

TQC drops atoms with largest locations after the pooling of atoms from multiple Z-networks. Experimentally, this procedure drops more atoms for some Z-networks than for the others. To quantify this disbalance, we compute the ratio of dropped atoms to the total M atoms for each of Z-networks. These proportions, once sorted and averaged over the replay, are approximately constant throughout the learning: $65/35\%$ for $N = 2$ and $35/25/18/13/9\%$ for $N = 5$ (Figure 20).

Interestingly, without sorting the averaging over the replay gives almost perfectly equal proportions (Figure 21). These results suggest, that a critic overestimates in some regions of the state action space more, than any other critic. In other words, in practice the systematic overestimation of a single critic (w.r.t. other critics predictions) on the whole state action space does not occur.

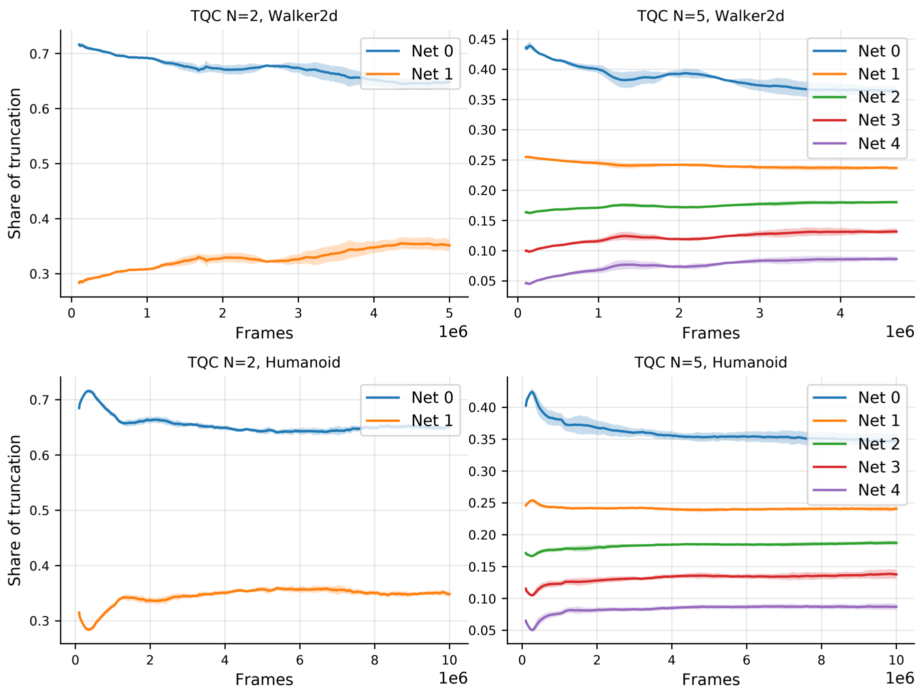

Figure 20. Proportions of atoms dropped per critic, *sorted* and averaged over the minibatches drawn from the experience replay for TQC with $N = 2$ and $N = 5$ critics with $M = 25$ and $d = 2$ dropped atoms per critic. For example, the upper right plot, should be read as "on average the largest proportion of dropped atoms per critic is 35%, i.e. out of $N \cdot d = 10$ atoms dropped approximately 4 were predicted by a single critic. Smoothed with a window of 100, $\pm$ std is plotted.

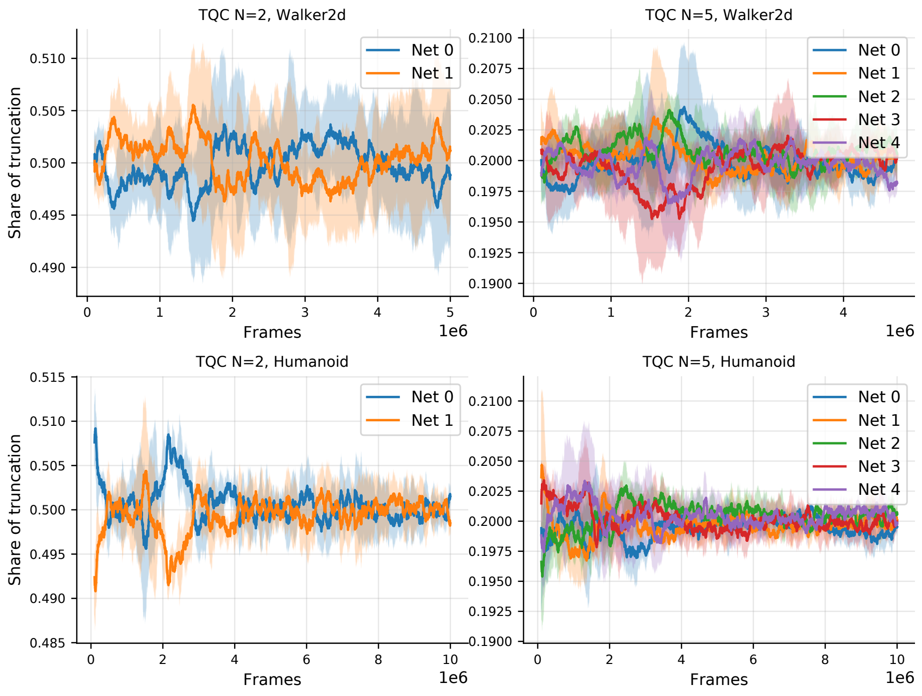

Figure 21. Proportions of atoms dropped per critic, averaged over the minibatches drawn from the experience replay for TQC with $N = 2$ and $N = 5$ critics with $M = 25$ and $d = 2$ dropped atoms per critic. Same plot as 20, but without sorting. The figure illustrates that there is no a single critic consistently overestimating more than the others over the whole state action space. Smoothed with a window of 100, $\pm$ std is plotted.