Lily Xie

University of British Columbia

Self-tuning Q-ensembles (STQE): Using Monte-Carlo Estimates for Adaptive Overestimation Bias Reduction

RWTH Aachen University
Institute for Data Science in Mechanical Engineering
Supervisor: Bernd Frauenknecht

UROP – Undergraduate Research Opportunities Program

Aachen
17.07.2023

Table of Contents:

Self-tuning Q-ensembles (STQE): Using Monte-Carlo Estimates for Adaptive Overestimation Bias Reduction

Abstract

Deep Reinforcement Learning (DRL) shows state-of-the-art performance in challenging domains such as gameplay and robotic control (Haarnoja et al., 2019). However, the Q-function estimation method used in DRL intrinsically introduces bias. This phenomenon of overestimation bias is well known but persisting: it has been studied for the past 3 decades, and numerous research has shown the severe impact of overestimation, as it can lead to performance degradation or even divergence of reinforcement learning (RL) approaches. In this work, we address this issue by combining different recent developments in RL: we combine two effective algorithms: Aggressive Q-Learning with Ensembles (AQE) that utilizes Q-ensembles with Adaptively Calibrated Critic Estimates (ACC) that auto-tunes the quantile of chosen critics during the training, such that the resulting algorithm, Self-tuning Q-ensembles (STQE), is able to adjust the number of critics used to calculate the temporal difference (TD) target automatically throughout the training process.

Introduction

The use of Q-ensembles in deep reinforcement learning (DRL) has been successful despite its simplicity. A single Q-function computes the expected sum of future rewards for a given state s and an action a, assuming that the current policy π is followed after this step; whereas a Q-ensemble employs multiple networks, one for each Q-function

output. Many state-of-the-art algorithms, renownedly Randomized Ensembled Double Q-Learning (REDQ) and aggressive Q-Learning with Ensembles (AQE), both make use of this streamlined yet powerful mechanism. In particular, AQE achieves the overall state-of-the-art performance as of today, by averaging over the most pessimistic quantile of critics in the Q-ensemble, leading to a more efficient ensemble usage.

However, the problem of overestimation still persists, as the optimal number of Q-ensembles used when calculating the targets might change during training, while AQE remains robust as it fixes the number of dropped Q-ensembles. In this paper, we propose a novel algorithm, Self-tuning Q-ensembles (STQE), that further enhances the performance of AQE in terms of bias reduction. We approach this goal by combing AQE with Adaptively Calibrated Critic Estimates (ACC) - a new general algorithm that performs bias correction with the aid of the most recent observed on-policy returns, such that the quantile of critics chosen for target calculation can be adjusted automatically and instantaneously throughout the training. The resulting STQE algorithm demonstrates the feasibility of combining ACC with AQE by providing promising first results as a proof of concept, further supported by theoretical results. Moreover, our research further demonstrates the generality of ACC in bias reduction by applying it to AQE.

Background

Reinforcement Learning

The field of machine learning has been rapidly emerging and expanding nowadays, as it allows both software and hardware applications to perform a variety of complex tasks, by making use of data-trained algorithms to produce models that imitate the way humans learn. Within the domain of machine learning lie three major subfields, namely supervised learning, unsupervised learning and reinforcement learning. Reinforcement learning, “a general paradigm with links to trial-and-error methods and behavioural

conditional studies" (Plaat, 2020), has several unique advantages over the other methods: reinforcement learning agents learn by interacting with the environment, rather than studying through large labelled datasets that are expensive to manually label and generate; RL efficiently and automatically generates agent controls that are goal-oriented (Lorenz, 2022), and this nature further allows the agent to learn sequences of actions, which expands beyond the scope of one-step input-output tasks in supervised learning.

The reinforcement learning paradigm involves mainly an agent and an environment (Plaat, A., 2020). The agent-environment interaction is described in a Markov Decision Process, and the basic mechanism is as follows: initially, the environment is at state s; at time t the agent performs an action a, and resultingly the environment progresses towards a new state s_{t+1}, and generates a new reward value r_{t+1}; this process repeats and the goal of the agent is to maximize the total rewards the environment returns, and the agent can improve and/or reinforce its interaction strategy with the environment by repetition. It is worth noting that minimum information is known to the agent prior to performing the actions: neither the effects of actions nor the temporal delay between actions and its effect on the agent's performance is known. Thus, even where the external evaluation is sparse (Thrun and Schwartz, 1993), reinforcement learning algorithms still allow for finding optimal action sequences in temporal decision tasks. This explains why reinforcement learning has been exceptionally successful when applied to solving a range of challenging tasks that resemble real-world problems, such as sequential decision-making and complex control tasks.

For instance, RL algorithms such as Soft Actor-Critic (SAC) have led to great success in complex task controls in recent years (Haarnoja et al., 2019), including the control of quadrupedal locomotion on Minitaur robot and dexterous hand manipulation. However, the rest of the results were mainly reported in simulated environments, while transferring the capabilities of RL to real-world applications widely remains an open research question. A major hurdle in this transfer is the high sample complexity of RL approaches.

This is, as typically the number of update steps on the policy and/or value functions per observed environment transition is very limited to ensure stable learning.

One approach to overcome this high sample complexity is model-based RL. Here, a model of the system dynamics is utilized to speed up learning. State-of-the-art model-based approaches such as Model-based Policy Optimization (MBPO) (Janner et al., 2021) are reported to reduce the sample complexity by about one order of magnitude while achieving comparable asymptotic performance with respect to their model-free counterparts. However, model usage gives rise to issues such as model exploitation, increased algorithm complexity, and an increased number of hyperparameters to tune.

Yet, a novel approach called Randomized Ensembeled Double Q-Learning (REDQ) reports sample complexity comparable to MBPO without using a model (Chen et al., 2021). This is achieved by utilizing an ensemble of action-value functions (Q-functions) leading to reduced approximation error in the action-value estimate. By reducing the approximation error, multiple up-date steps to the action value function per observed environment transition can be conducted without destabilizing learning. This simple yet efficient algorithmic design enables recent breakthroughs in model-free real-world RL (Smith et al., 2021).

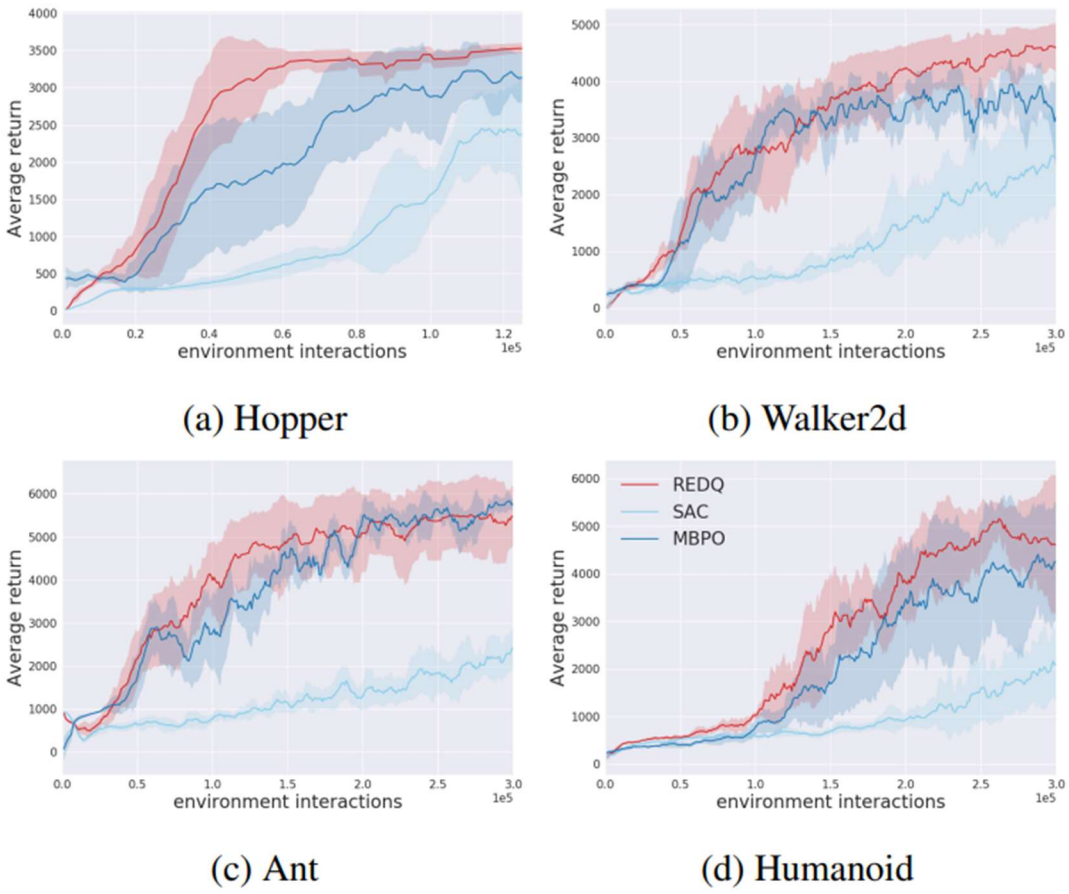


Figure 1: Performance of REDQ compared to MBPO and SAC (Chen et al., 2021)

A drawback of REDQ, however, is its inefficient usage of the ensemble model. Based on insights from distributional RL methods, such as Truncated Quantile Critics (TQC) (Kuznetsov et al., 2020), Chen et al. propose Aggressive Q-learning with Ensembles (AQE) (Wu et al., 2021). Here the in-target minimization of REDQ is replaced by averaging over the most pessimistic quantile of critics, leading to a more efficient ensemble usage. How this quantile is to be chosen is, however, heavily task-dependent and might also change during training. Hence, we investigate the combination of Adaptively Calibrated Critic (ACC) (Dorka et al., 2022) to automatically tune this decision based on Monte Carlo (MC) estimates during learning.

Persisting Q-estimation Bias

To make reinforcement learning algorithms transferable to real-world applications such as robotics, one important property worth concerning is that taking action might be costly. Interactions with the environment could be very time- and cost-intensive; however, the advent of off-policy RL algorithms effectively solves the issue. Off-policy algorithms aim to reuse old experiences, making them more sample efficient than their on-policy counterparts (Dorka et al., 2022), and learning the value of the optimal policy independently of the agent's actions. It treats exploration independently from the learning algorithm (Lillicrap et al., 2019).

One common approach that enables off-policy learning is Watkins' Q-learning (Plaat, 2020): it learns the optimal Q-value function that will maximize the long-term discounted reward over a sequence of timesteps, by updating itself using the Bellman equation.

Usually, Q-value functions must be combined with powerful generalizing function estimators, i.e. artificial neural networks, in order to perform tasks with high dimensional or continuous state and action pairs (Thrun and Schwartz, 1993). However, learning the Q-function off-policy can lead to an overestimation bias (Dorka et al., 2022), and it is especially prominent when using a non-linear function approximator to model the Q-function (Dorka et al., 2022), just like in the case of deep neural networks.

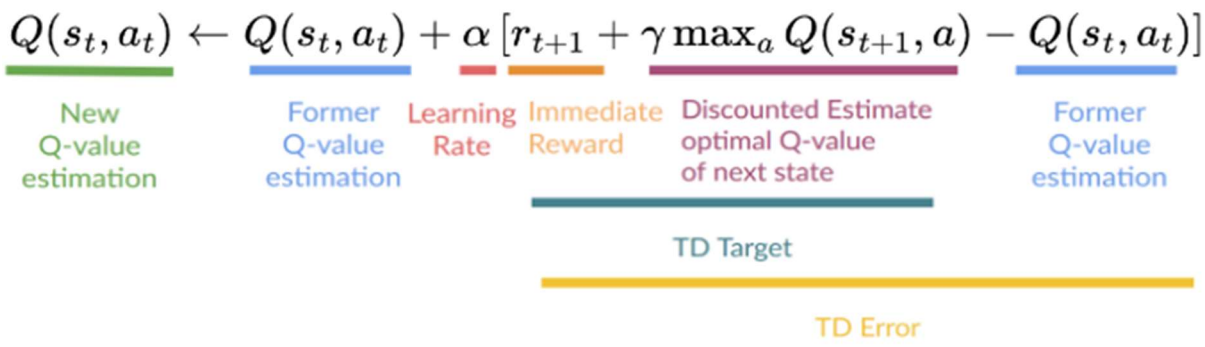


Figure 2: Q-function update formula

Due to the nature of function approximators, they induce some noise or generalization error in the output predictions. Consider the maximum expected future reward calculation in the Q-updating function above: the generation error causes the Q-values to vary, some are too small and some are too large; yet the max operator always selects the largest value (assuming there is more than one action to choose from). If several Q-values are too large simultaneously, the max operator would easily pick up the overall overestimation, resulting in a Q-function particularly sensitive to overestimations. Thus, with the iterative application of the update rule over time, a systematic overestimation effect of values persists due to function approximation when used in recursive value estimation schemes. It is extremely harmful because overestimation bias due to in-target maximization in Q-learning can remarkably slow down the learning performance (Thrun and Schwartz, 1993).

Several research attempts to conquer the problem. Thrun S. and Schwartz A. (1993), after identifying the overestimation issues in using function approximation, suggest multiple potential strategies for diminishing the catastrophic effects of overestimation: using approximators with unbounded memory, incorporating actual sample rewards, using cost instead of discounting, introducing pseudo-costs to offset overestimation, and incorporating function approximators that are biased toward making low predictions (Thrun and Schwartz, 1993). In the domain of Double Q-learning, Fujimoto S. et al (2018) proposed a method that practices generating low predictions called the Clipped Double Q-learning algorithm: in order to perform the target update, upper-bound the less biased value estimate by the biased estimate, so that it results in taking the minimum between the pair of two estimates. This approach significantly mitigates the issue of overestimation bias. Besides TD3 which first incorporates the Clipped-Double Q-learning approach, SAC also utilizes this approach, and further builds it on top of the maximum entropy reinforcement learning framework (Haarnoja et al., 2019).

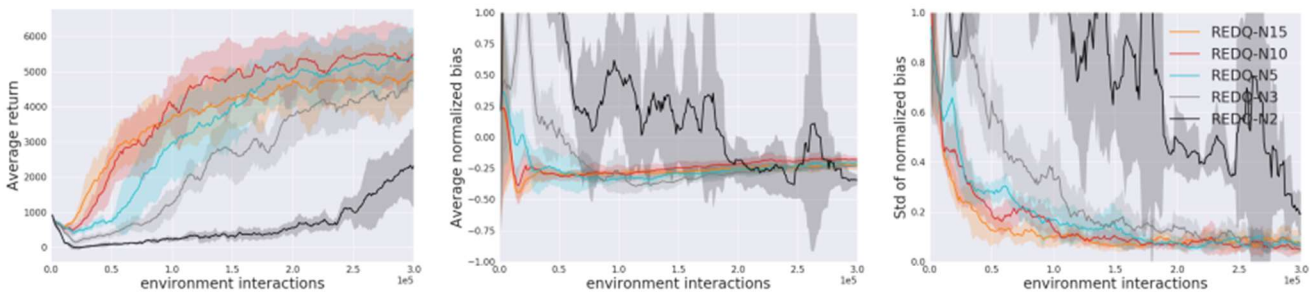
However, due to the coarse nature of simply taking the minimum, as the likelihood of overestimation is reduced by taking the minimum, the chance of underestimation also

increases. Although underestimation bias is far preferable to overestimation, the Clipped-Double Q-learning approach still accumulates errors along the way that it prevents the algorithm from taking a high update-to-data (UTD) ratio: the number of updates taken by the agent divided by the number of actual interactions with the environment. For instance, as the UTD ratio increases in SAC, both the mean and the standard deviation of the Q-estimation bias becomes larger and more unstable.

Related Work

Using Q-Ensembles

Luckily, recent advances in Q-learning have shown that using Q-ensembles can improve the performance of Deep Reinforcement Learning algorithms (Chen et al., 2021). A Q-ensemble employs multiple networks, one for each Q-function output. In REDQ, Q-ensembles are used, and in-target minimization occurs across a random subset of Q functions from the ensemble (Chen et al., 2021). The calculation mechanism is as follows: N denotes the total number of Q-estimations in the Q-ensemble, and M denotes the selected number of Q-functions being used to calculate the temporal-difference (TD) target. The target for the Q-function is obtained by minimizing over a random subset M of the N Q-functions in the ensemble. The default choice for M is M = 2, which makes REDQ a Double Q-learning algorithm. So for instance, let N = 10 and M = 2, then REDQ would randomly select 2 Q-functions from the ensemble, and takes the minimum of the pair. Chen X, et al. (2021) demonstrates that as the ensemble size increases, it results in a more stable and much lower average bias compared to REM (Agarwal et al., 2020) and Maxmin (Lan et al., 2020), a lower standard deviation of bias, and stronger performance. With the use of Q-ensembles, REDQ is able to obtain a high UTD ratio and achieves more than 7 times the sample efficiency of SAC on the challenging MuJoCo environments of Ant and Humanoid. As demonstrated in the case of REDQ, the introduction of Q-ensembles with Clipped-Double Q-learning successfully solves the issue of low UTD ratio, and significantly improves sample efficiency.



(a) Performance, Ant (b) Average bias, Ant (c) std of bias, Ant

Figure 3: REDQ ablation results for Ant. The top row shows the effect of the ensemble size N.

A drawback of REDQ, however, is its inefficient usage of the Q-ensemble. Generally, the subset number M is fixed at 2 for REDQ, which is less than half of the total number of Q-functions in the ensemble, implying that most computed Q-functions are only used once in the training. Moreover, employing one neural network for each of the Q-function outputs in the ensemble can be expensive in terms of computation and memory. Is it possible to maximize the power of the Q-ensemble? A later study named AQE, Aggressive Q-Learning with Ensembles, is able to make extensive use of the Q-ensemble. Rather than minimizing over a subset of the Q-ensemble, AQE still constructs a Q-ensemble to estimate the distribution of the return random variable, but then drops several of the topmost “atoms” from the estimated distribution, and takes the average of the rest to calculate the targets. In this way, AQE naturally controls overestimation bias by only keeping the pessimistic quantile of Q-functions, as well as the standard deviation of the bias. Moving from REDQ to AQE is a major improvement as AQE successfully utilizes most of the ensemble models when calculating the target, therefore maximizing the computational resources, and further enhancing the sampling efficiency (Wu et al., 2021). Moreover, through extensive testing, AQE is shown to surpass the performance of SAC, REDQ and TQC at all stages of training

In spite of AQE’s state-of-the-art performance, a downside of AQE is that the number of topmost Q-functions to be dropped from the target calculation has to be specified as a

hyperparameter before the start of training, whereas this parameter is heavily task-dependent bias-controlling hyperparameter, and might change during the training. Tuning the hyperparameter in a new environment, especially for real-world applications, can be very expensive and sometimes impossible. Therefore, we propose combining AQE with Adaptively Calibrated Critic (ACC) that performs bias correction and auto-tuning of the parameter during that training, which will be further discussed below.

Adaptively Calibrated Critic (ACC)

ACC learns a Q-value function with bias-calibrated TD targets, whereas the bias correction in the targets is determined via β - a bias-controlling parameter that is adjusted by comparing the biased TD estimates with low variance obtained from the Q-ensembles with the unbiased but high-variance Monte-Carlo estimates on the most recent observed data. the overestimation bias is then obtained. After each comparison, β adapts the size of the temporal difference (TD) targets, such that the bias can be corrected in the subsequent updates. By only performing one update step rather than using the minimization, it ensures that β is changing relatively slowly and targets are more stable. As the β parameter changes slower than the rollout returns, our method benefits from stable and low-variance temporal difference targets. Using this technique, ACC allows to adaptively tune the β parameter online, which adjusts the magnitude and direction of bias correction during training in the environment. Therefore, it obviates the need for extensive manual tuning of the hyperparameter in a new environment, promising robust performance across a wide range of tasks.

Self-tuning Q-ensembles (STQE)

Methodology

We propose Self-tuning Q-ensembles (STQE), a novel model-free off-policy algorithm that reduces both over- and under-estimation bias with the flexible usage of the Q-

ensembles. STQE inherits the basic properties of AQE: it has a single policy network and multiple Q-function networks in the Q-ensemble, and targets that average the pessimistic critics by dropping the highest K values among the Q-functions. In other words, the lowest N-K Q-estimates are kept and their average becomes the target. Under such calculation, the estimation bias is highly dependent on the value of K: the bias term could change from overestimation to underestimation by increasing K; and vice versa, the bias term could change from negative to positive by decreasing K. Therefore, tuning of the hyperparameter K becomes especially important for bias control. However, the parameter K has to be set for each environment individually, which is undesirable because it is harder to tune in practice, and it could be very expensive and sometimes impossible in real-world applications. Moreover, fixing the K value makes the algorithm robust, as the optimal number of Q-function varies across environments and might change during the training. To aid in the tuning process, our approach implements Adaptively Calibrated Critic (ACC) to learn a bias-controlling hyperparameter of the AQE algorithm, ACC is a general off-policy algorithm that reduces the bias of value estimates in a principled fashion with the help of the most recent unbiased on-policy rollouts. The K value of kept Q-functions for target calculation can be adjusted automatically during the training, allowing for more fine-grained control of overestimation as well as underestimation bias by choosing how many targets are dropped.

Implementation

The resulting algorithm is implemented and tested building upon the MBRL-Lib Library (Pineda et al., 2021). This library provides an implementation of SAC and is compatible with standard test environments such as OpenAI Gym or MuJoCo.

We start from an existing AQE implementation built upon the MBRL-Lib Library, and initialize values for the hyperparameters, which do not differ much from the original AQE algorithm. In the configuration file, we add two additional arguments, namely the initial

adjusted number of dropped Q-functions and the maximum adjusted number of dropped Q-functions. The initial number is reasonably set to the middle number N/2, or N/2+1 in the case of N being odd, where N is the total number of Q-functions in the Q-ensemble. The maximum dropped number is set to be N-1, and in such extreme cases, the return temporal difference target would be the lowest Q-value estimate among all N Q-functions in the ensemble. Other additional hyperparameters are left unchanged, or directly motivated by the environment.

We modify the update parameter function in AQE so that it adapts the update β function from ACC. The first 5000 environment interactions are generated with a random policy, which allows the agent to conduct initial random exploration without training. Results are compared only after the training steps started to be fair. The updates of β then start.

```
if args.use_acc and epoch % 1 == 0 and t > args.start_steps:
    estimate_return_list, true_return_list = get_true_estimate_value(agent, logger, test_env,
                                                                      max_ep_len)
```

Figure 4: Implementing the ACC mechanism in the code base.

We obtain a list of temporal difference targets as well as a list of Monte Carlo unbiased on-policy rollouts. Each item in the estimate return list is calculated by averaging the N-K Q-functions in the ensemble, and the unbiased estimator is computed by averaging the samples of the discounted return, through Monte Carlo estimation. The Monte Carlo series is the most recent observed high variance return and it serves as the base line in comparison.

```
ep_ret_true = ep_ret_true + r_true * (agent.gamma ** ep_len_true)
```

Figure 5: The Monte Carlo estimation

After obtaining these data crucial for bias correction, the algorithm would call the update β function.

```
agent.update_beta(estimate_return_list, true_return_list)
```

Figure 6: The agent calls the function to update β

The update β function then utilizes the Q-query as well as the Monte Carlo series. The difference between the on-policy returns and temporal difference targets is calculated to update the β parameter, and to adjust the optimal number of Q-functions to drop. When the Q-estimates are larger than the actual observed returns, β is decreased for overestimation; and similarly, increased for overestimation. The number of dropped Q-functions, Q_β, is continuous and increases with β monotonically. Therefore, increasing β increases Q_β and vice versa. We use the general rule of rounding: that is, only when the adjusted Q drop number goes above or below .5 would the algorithm then change the actual dropped number of Q-functions.

```
def update_beta(self, estimate_return, true_return):
    mean_Q_last_eps = np.mean(estimate_return)
    mean_return_last_eps = np.mean(true_return)

    if self.first_training:
        self.diff_mvavg = abs(mean_return_last_eps - mean_Q_last_eps)
        self.first_training = False
    else:
        self.diff_mvavg = (1 - 0.05) * self.diff_mvavg \
                          + 0.05 * abs(mean_return_last_eps - mean_Q_last_eps)

    diff_qret = ((mean_return_last_eps - mean_Q_last_eps) / (self.diff_mvavg + 1e-8))
    self.diff_qret = diff_qret
    print("diff_qret:", diff_qret)
    aux_loss = self.adjusted_Q_drop * diff_qret
    self.Q_drop_optimizer.zero_grad()
    aux_loss.backward()
    self.Q_drop_optimizer.step()
    self.adjusted_Q_drop.data = self.adjusted_Q_drop.clamp(min=0., max=self.adjusted_Q_drop_max)
    print(self.adjusted_Q_drop)
```

Figure 7: Updating the β parameter

During training, the policy is evaluated every 2,000 environment steps by averaging the episode returns of 10 rollouts with the current policy. After each update of β, the oldest episodes in the stored batch are removed until no more than 5,000 state-action pairs are left, so on average β is updated with a batch size slightly > 5,000. The additional computational resources caused by ACC is very minimal compared to AQE: there is only one update to β and at least 1,000 training steps are in between each of these updates, which is very efficient compared to one training step of the actor-critic.

Results and Discussion

We perform extensive and comprehensive experiments using the popular benchmark Pendulum-v0 from the OpenAI Gymnasium test suite. The maximum reward this environment could possibly return is 0. First, we provide experimental results for SAC, REDQ, and AQE. All experiments conducted are trained using our thoroughly tested implementation built upon the MBRL-Lib Library of each individual algorithm (Pineda et al., 2021). This library provides an implementation of SAC and is compatible with standard test environments such as OpenAI Gym or MuJoCo.

In the comparison experiment, we use a total number of Q-functions N = 10 across all experiments. In AQE, the dropped number of Q-functions is set to be 3, therefore we set the initial dropped number of Q-functions to also be 3 in ACC for the first 5000 steps. Notice that this parameter is only useful for the initial random exploration, and once training starts, the dropped number will become the median number 5, and start adjusting from there.

During training, the policy is evaluated every 2,000 environment steps for SAC, REDQ, AQE and STQE (labelled as ACC below), and each episode rollout return with the current policy is recorded in the logger. The data points on the plots below are obtained by averaging the episode returns of 10 rollouts with the current policy.

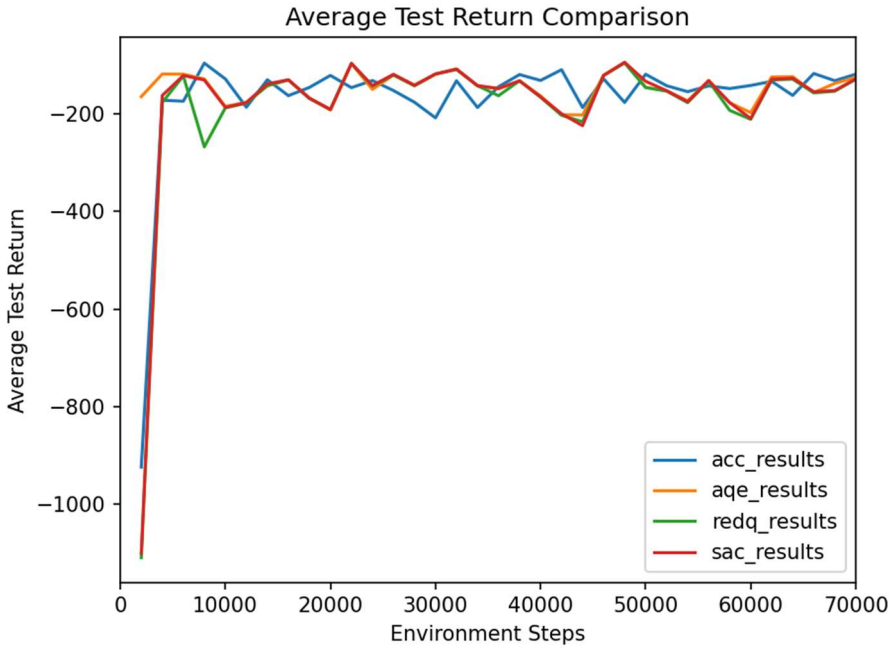


Figure 8: Comparison of average test return per 2,000 steps across SAC, REDQ, AQE and ACC.

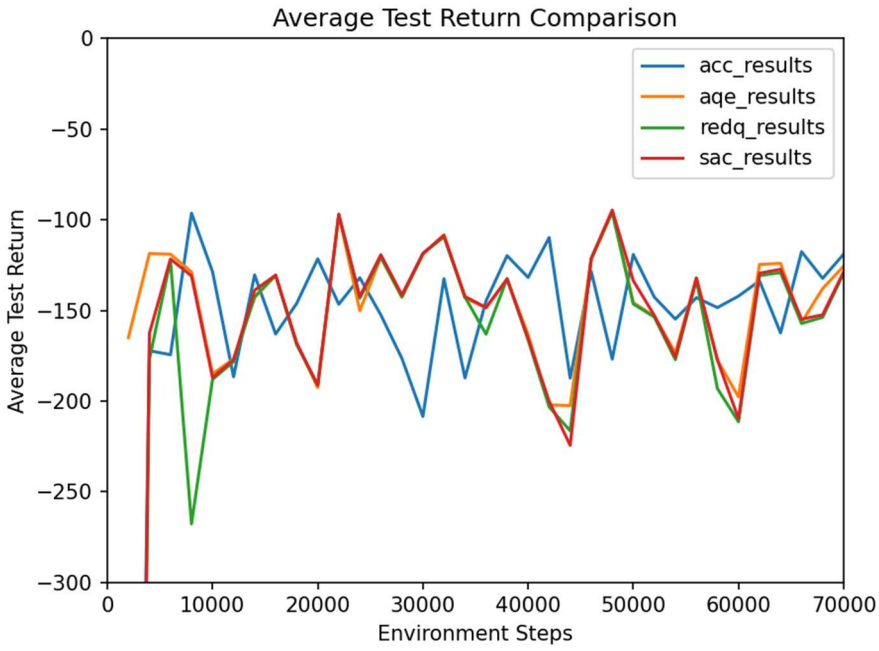


Figure 9: Comparison of average test return, zoomed in with a lower x-bound of -300.

To see a comparison of performance across different stages of training, we constructed the table below demonstrating the average test returns evaluated per 10,000 steps.

Table 1: Comparison of Average Evaluation Returns of SAC, REDQ, AQE and STQE by Stages of Time Steps

Step / Average Evaluation Return	SAC	REDQ	AQE	STQE
0-10000	-340.463989	-372.834221	-143.260604	-299.041745
10000-20000	-161.146171	-162.229554	-162.155741	-149.517515
20000-30000	-123.692911	-124.368817	-126.005075	-163.167204
30000-40000	-139.469741	-142.745871	-139.251447	-143.188397
40000-50000	-154.828377	-156.711426	-153.269955	-144.306953
50000-60000	-169.512431	-173.689995	-166.713645	-146.166639
60000-70000	-138.527682	-140.055365	-133.752042	-132.978786

Figure 10: Table of mean returns across different stages of training. The best results among each stage group are highlighted in yellow.

Based on the table above displaying comparison by stages, it can be inferred that although the best average evaluation return appears to be heterogeneous for earlier

stages, STQE outperforms the other 3 algorithms consistently starting from 50000 steps, showing initial promising results of performance improvement.

We then demonstrate the feasibility of implementing the ACC mechanism on AQE, by visualizing the adjusted dropped number of Q-functions corresponding to normalized bias against environment steps.

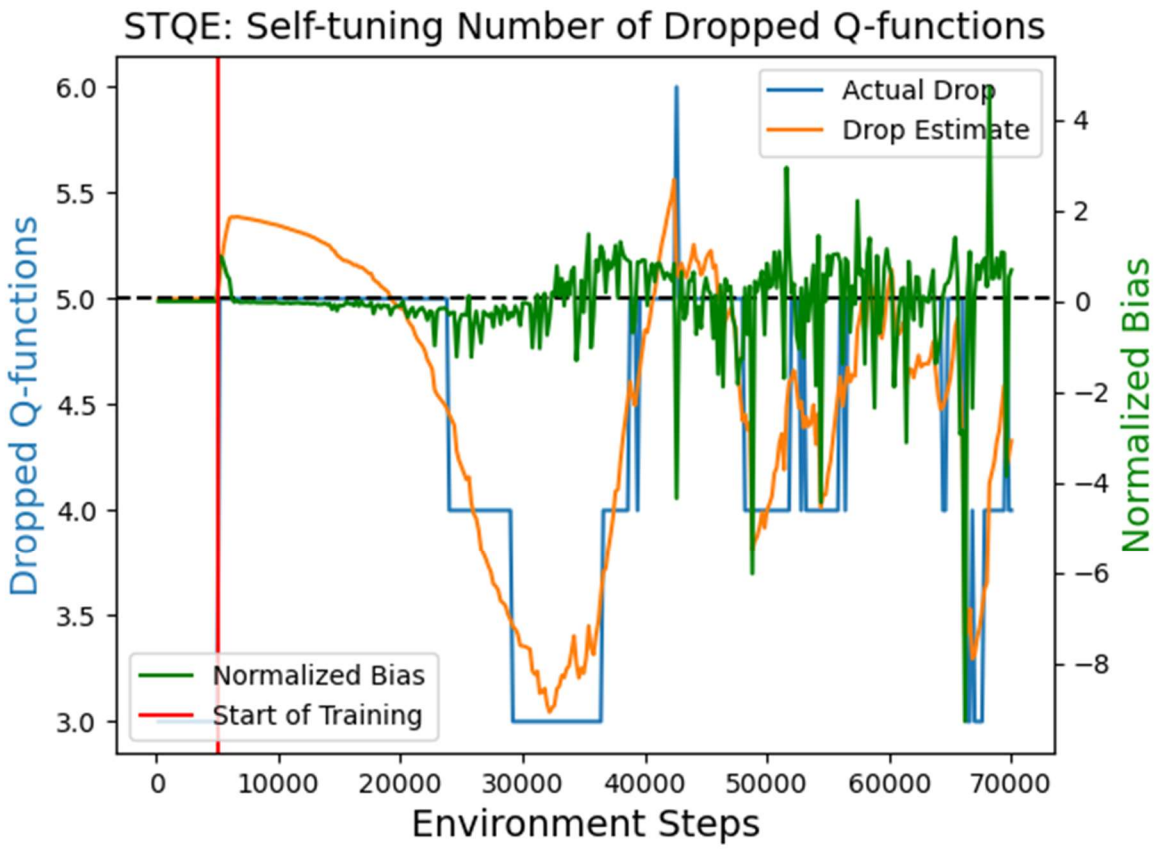


Figure 11: Adjusted dropped number estimates (orange), actual dropped Q-functions (blue) and normalized bias (green) against environment steps, tested on the OpenAI Gymnasium Pendulum-v0 environment.

Figure 11 demonstrates that the dropped number of Q-functions represented by the orange/blue line changes simultaneously with the bias (in green): as overestimation occurs (green line moves above the x-axis), the dropped number of Q-functions also

increases correspondingly to counteract the effect, and vice versa, showing a functioning adaptation of ACC on AQE. This provides valid proof that the self-tuning bias reduction mechanism has been successfully implemented in the Q-ensembles method.

Conclusion

This study successfully implements ACC on top of AQE, realizes the auto-tuning for temporal difference target calculation in the Q-value estimates, and demonstrates preliminary promising results for high performance as well as effective bias reduction.

Due to computational limitations, we were only able to perform comparison experiments on one single OpenAI Gymnasium environment, namely Pendulum-v0, and the number of training steps remains to be 70000 environment steps for all experiments. Past research has shown that definite performance evaluation requires testing in more environments (Wu et al., 2021). Thus, further experiments with more challenging environments and significantly greater trials are strongly recommended. Furthermore, in our research, the evaluation of performance for each individual algorithm is conducted based on only 3 experiments per algorithm, whereas for further statistical analysis of performance mean and standard deviation, far more repetition is needed.

Meanwhile, more parameters are worth discovering as well that are not yet added to our logger printout. We suggest further investigation into actor loss and critic loss, as they are also highly relevant to performance evaluation. Fine-tuning of the hyperparameters, such as finding the optimal total number of Q-value estimates, and how often should the update happen (as AQE has UTD greater than 1) also needs to be explored.

A potential improvement in the algorithm would be incorporating the multi-head architectures as discussed by Wu et al. (2021). It can be expensive in terms of computational resources and memory to employ multiple networks, one for each Q-function output. If possible, network ensemble could be combined with multi-head

architectures to generate multiple Q-function outputs to reduce the computation and memory requirements, and improve algorithm efficiency.

A unique advantage of Self-Tuning Q-Ensembles would be its strong adaptability of hyperparameter tuning across a wide range of different environments, Previous studies show that large differences between the environments (Dorka et al., 2022), and MuJoCo environments surely behave differently compared to the Pendulum environment tested in this study. Therefore, testing on a wide range of environments, particularly environments that are challenging or with discrete action spaces, is strongly encouraged to reveal the effectiveness of Self-Tuning Q-Ensembles. Moreover, the implementation of ACC on TQC has already achieved strong performance on robotics tasks, and we are optimistic to see the performance of STQE in learning on more complicated robotic environments, as well as real robots where tuning hyperparameters is costly.

Acknowledgement

Throughout the 10 weeks in the UROP program, I have gained abundant new experience and insights about conducting research, and I am honoured to have such an opportunity to learn more about reinforcement learning, and to apply my knowledge obtained in school to pioneering real-world applications. I would like to thank my supervisor Bernd Frauenknecht for his valuable and supportive guidance that encouraged me to overcome all the challenges throughout my research internship. I would also like to appreciate the entire DSME team for their welcoming atmosphere and kindness; thanks to Aditya Pradhan for his hard work and implementation that sets the foundation of the research, and Devdutt Subhasish for his kind technical support.

References

Agarwal, R., Schuurmans, D., & Norouzi, M. (2020). An optimistic perspective on offline reinforcement learning. In International Conference on Machine Learning (ICML).

Chen, X., Wang, C., Zhou, Z., & Ross, K. (2021). Randomized Ensembled Double Q-Learning: Learning Fast Without a Model.

Dorka, N., Welschehold, T., Bodecker, J., & Burgard, W. (2022). Adaptively Calibrated Critic Estimates for Deep Reinforcement Learning. IEEE Robotics and Automation Letters.

Fujimoto, S. (2018). Addressing Function Approximation Error in Actor-Critic Methods (p. 10).

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., & Levine, S. (2019). Soft Actor-Critic Algorithms and Applications.

Janner, M., Fu, J., Zhang, M., & Levine, S. (2021). When to Trust Your Model: Model-Based Policy Optimization.

Kuznetsov, A., Shvechikov, P., Grishin, A., & Vetrov, D. (2020). Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics.

Lan, Q., Pan, Y., Fyshe, A., & White, M. (2020). Maxmin q-learning: Controlling the estimation bias of q-learning.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2019). Continuous control with deep reinforcement learning.

Lorenz, U. (2022). Basic Concepts of Reinforcement Learning. In: Reinforcement Learning From Scratch. Springer, Cham. https://doi.org/10.1007/978-3-031-09030-1_2

Plaat, A. (2020). Reinforcement Learning. In: Learning to Play. Springer, Cham. https://doi.org/10.1007/978-3-030-59238-7_3

Pineda, L., Amos, B., Zhang, A., Lambert, N. O., & Calandra, R. (2021). MBRL-Lib: A Modular Library for Model-based Reinforcement Learning.

Smith, L., Kew, J. C., Peng, X. B., Ha, S., Tan, J., & Levine, S. (2021). Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World.

Thrun, S., & Schwartz, A. (1993). Issues in Using Function Approximation for Reinforcement Learning (p. 9).

Wu, Y., Chen, X., Wang, C., Zhang, Y., Zhou, Z., & Ross, K. W. (2021). Aggressive Q-Learning with Ensembles: Achieving Both High Sample Efficiency and High Asymptotic Performance. International Conference on Information Processing Systems.