# CROSS$Q$: BATCH NORMALIZATION IN DEEP REINFORCEMENT LEARNING FOR GREATER SAMPLE EFFICIENCY AND SIMPLICITY

**Aditya Bhatt** [*] [1,4] **Daniel Palenicek** [*] [1,2] **Boris Belousov** [1,4] **Max Argus** [3]
**Artemij Amiranashvili** [3,6] **Thomas Brox** [3] **Jan Peters** [1,2,4,5]
[*]Equal contribution [1]Intelligent Autonomous Systems, TU Darmstadt [2]Hessian.AI [3]University of Freiburg
[4]German Research Center for AI (DFKI) [5]Centre for Cognitive Science, TU Darmstadt [6]Amazon

## ABSTRACT

Sample efficiency is a crucial problem in deep reinforcement learning. Recent algorithms, such as REDQ and DroQ, found a way to improve the sample efficiency by increasing the update-to-data (UTD) ratio to 20 gradient update steps on the critic per environment sample. However, this comes at the expense of a greatly increased computational cost. To reduce this computational burden, we introduce Cross$Q$: a lightweight algorithm that makes careful use of Batch Normalization and removes target networks to surpass the state-of-the-art in sample efficiency while maintaining a low UTD ratio of 1. Notably, Cross$Q$ does not rely on advanced bias-reduction schemes used in current methods. Cross$Q$'s contributions are thus threefold: (1) state-of-the-art sample efficiency, (2) substantial reduction in computational cost compared to REDQ and DroQ, and (3) ease of implementation, requiring just a few lines of code on top of SAC.

## 1 INTRODUCTION

Sample efficiency is a crucial concern when applying Deep Reinforcement Learning (Deep RL) methods on real physical systems. One of the first successful applications of Deep RL to a challenging problem of quadruped locomotion was achieved using Soft Actor-Critic (SAC) (Haarnoja et al., 2018a), allowing a robot dog to learn to walk within 2h of experience (Haarnoja et al., 2018b). Subsequently, it was noted that the critic in SAC may be underfitted, as only a single gradient update step on the network parameters is performed for each environment step. Therefore, Randomized Ensembled Double Q-Learning (REDQ) (Chen et al., 2021) was proposed that increased this number of gradient steps, termed Update-To-Data (UTD) ratio. In addition, Dropout Q-functions (DroQ) (Hiraoka et al., 2021) improved the computational efficiency of REDQ while maintaining the same sample efficiency by replacing its ensemble of critics with dropout. This enabled learning quadruped locomotion in a mere 20min (Smith et al., 2022). Thus, REDQ and DroQ represent the state-of-the-art in terms of sample efficiency in DeepRL for continuous control.

Importantly, both REDQ and DroQ showed that naively increasing the UTD ratio of SAC does not perform well due to the critic networks' Q-value estimation bias. Therefore, ensembling techniques were introduced for bias reduction (explicit ensemble in REDQ and implicit ensemble via dropout in DroQ), which allowed increasing the UTD to 20 critic updates per environment step.

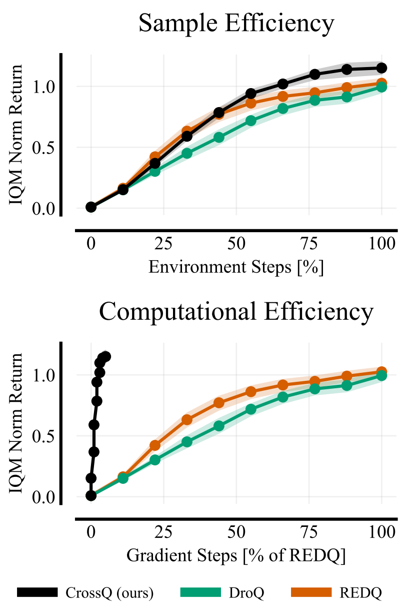

Figure 1: **Cross$Q$ training performance aggregated over environments.** Cross$Q$ is more sample efficient (top) while being significantly more computationally efficient (bottom) in terms of the gradient steps, thanks to a low UTD $= 1$. Following Agarwal et al. (2021), we normalize performance by the maximum of REDQ in each environment.

Higher UTD ratios improve sample efficiency by paying the price of increased computational cost, which manifests in higher wallclock time and energy consumption. It is, therefore, desirable to seek alternative methods that achieve the same or better sample efficiency at a lower computational cost, e.g., by using lower UTDs while keeping the performance of REDQ and DroQ.

It turns out that even UTD $= 1$ can perform surprisingly well if other algorithmic components are adjusted appropriately. In this paper, we introduce **Cross$\boldsymbol{Q}$**, a lightweight algorithm that achieves superior performance by 'crossing out' parts of SAC. First, it *removes target networks*, an ingredient widely believed to slow down training in exchange for stability (Mnih et al., 2015; Lillicrap et al., 2016; Kim et al., 2019; Fan et al., 2020). Second, we find that use of *Batch Normalization* (BatchNorm) (Ioffe & Szegedy, 2015), when applied in a particular manner, is efficient at stabilizing training and significantly improving sample efficiency. This contradicts other observations that it hurts the learning performance in Deep RL, e.g., (Hiraoka et al., 2021). Third, Cross$Q$ uses *wider critic layers*, motivated by prior research on the ease of optimization of wider networks (Ota et al., 2021). In addition to the first two improvements, wider networks lead to achieving even higher returns.

**Contributions.** (1) We present the Cross$Q$ algorithm, which marks the new state-of-the-art for model-free off-policy RL in both sample efficiency and computational complexity; (2) By removing target networks, we are able to show the first successful application of BatchNorm in off-policy Deep RL; (3) We provide empirical investigations and hypotheses for Cross$Q$'s success. Cross$Q$'s changes mainly pertain to the deep network architecture of SAC; therefore, our study is chiefly empirical: through a series of ablations, we isolate and study the contributions of each part. We find that Cross$Q$ matches or surpasses the state-of-the-art algorithms in sample efficiency while being up to $4\times$ faster in terms of wallclock time, without requiring critic ensembles, target networks, or high UTD ratios. We will provide the source code of Cross$Q$ and our experiments after publication and during the rebuttal.

## 2 Background

We briefly introduce the necessary background for the remainder of the paper.

### 2.1 Off-policy Reinforcement Learning and Soft Actor-Critic

We consider a discrete-time Markov Decision Process (MDP) (Puterman, 2014), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho, \gamma \rangle$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability $\boldsymbol{s}_{t+1} \sim \mathcal{P}(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$, reward function $r_t = \mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, initial state distribution $\boldsymbol{s}_0 \sim \rho$ and discount factor $\gamma \in [0, 1)$. RL describes the problem of an agent learning an optimal policy $\pi$ for a given MDP. At each time step $t$, the agent receives a state $s_t$ and interacts with the environment according to its policy $\pi$. We focus on the Maximum Entropy RL setting (Ziebart et al., 2008), where the agent's objective is to find the optimal policy $\pi^*$, which maximizes the expected cumulative reward while keeping the entropy $\mathcal{H}$ high; $\arg\max_{\pi^*} \mathbb{E}_{\boldsymbol{s}_0 \sim \rho} \left[\sum_{t=0}^{\infty} \gamma^t (r_t - \alpha \mathcal{H}(\pi(\,\cdot\,|\boldsymbol{s}_t)))\right]$. The action-value function is defined by $Q(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\pi, \mathcal{P}} \left[\sum_{t=0}^{\infty} \gamma^t (r_t - \alpha \log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}\right]$ and describes the expected reward when taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$.

Soft Actor-Critic (SAC) (Haarnoja et al., 2018a) is a popular algorithm that solves the MaxEnt RL problem. SAC parametrizes the Q function and policy as neural networks and trains two independent versions of the Q function, using the minimum of their estimates to compute the regression targets for Temporal Difference (TD) learning. This *clipped double-Q* trick, originally proposed by Fujimoto et al. (2018), helps in reducing the potentially destabilizing overestimation bias inherent in approximate Q learning (Hasselt, 2010).

### 2.2 High update-to-data ratios, REDQ, and DroQ

Despite its popularity among practitioners and as a foundation for other more complex algorithms, SAC leaves much room for improvement in terms of sample efficiency. Notably, SAC performs exactly one gradient-based optimization step per environment interaction. SAC's UTD $= 1$ setting is analogous to simply training for fewer epochs in supervised learning. Therefore, in recent years, gains in sample efficiency within RL have been achieved through increasing the update-to-data ratio (UTD) (Janner et al., 2019; Chen et al., 2021; Hiraoka et al., 2021; Nikishin et al., 2022).

```python
def critic_loss(Q_params, policy_params, obs, acts, rews, next_obs):
    next_acts, next_logpi = policy.apply(policy_params, obs)

    # Concatenated forward pass
    all_q, new_Q_params = Q.apply(Q_params,
        jnp.concatenate([obs, next_obs]),
        jnp.concatenate([acts, next_acts]),
    )
    # Split all_q predictions and stop gradient on next_q
    q, next_q = jnp.split(all_q, 2)
    next_q = jnp.min(next_q, axis=0)    # min over double Q function
    next_q = jax.lax.stop_gradient(next_q - alpha * next_logpi)
    return jnp.mean((q - (rews + gamma * next_q))**2), new_Q_params
```

Figure 2: **Cross$Q$ critic loss in JAX.** The Cross$Q$ critic loss is easy to implement on top of an existing SAC implementation. One just adds Batch Normalization into the critic network and removes the target network. Since now there exists only a single double Q function network, one can simply concatenate observations and next observations, as well as actions and next actions along the batch dimension, perform a joint forward pass, and split up the batches afterward. This removes the computational need for two individual forward passes through the same network.

Different algorithms, however, substantially vary in their approaches to achieving high UTD ratios. Janner et al. (2019) uses a model to generate synthetic data, which allows for more overall gradient steps. Nikishin et al. (2022) adopt a simpler approach: they increase the number of gradient steps, while periodically resetting the policy and critic networks to fight premature convergence to local minima. We now briefly outline the two high-UTD methods to which we compare Cross$Q$.

**REDQ.** Chen et al. (2021) find that merely raising SAC's UTD ratio hurts performance. They attribute this to the accumulation of the learned Q functions' estimation bias over multiple update steps — despite the clipped double-Q trick — which destabilizes learning. To remedy this bias more strongly, they increase the number of Q networks from two to an ensemble of 10. Their method, called REDQ, permits stable training at high UTD ratios up to 20.

**DroQ.** Hiraoka et al. (2021) note that REDQ's ensemble size, along with its high UTD ratio, makes training computationally expensive. They instead propose using a smaller ensemble of Q functions equipped with Dropout (Srivastava et al., 2014), along with Layer Normalization (Ba et al., 2016) to stabilize training in response to the noise introduced by Dropout. Called DroQ, their method is computationally cheaper than REDQ, yet still expensive due to its UTD ratio of 20.

## 3 The Cross$Q$ Algorithm

In this paper, we challenge this current trend of high UTD ratios and set a new bar in sample efficiency and computational efficiency simultaneously, at only UTD $= 1$. Cross$Q$ is our new state-of-the-art off-policy actor-critic algorithm. Based on SAC, it uses purely network-architectural engineering insights from deep learning to accelerate training at UTD $= 1$. As a result, it ~~crosses out~~ much of the algorithmic design complexity that was added over the years and which led to the current state-of-the-art methods. In doing so, we present a much simpler yet more efficient algorithm. In the following paragraphs, we introduce the three design choices that constitute the Cross$Q$ algorithm.

### 3.1 Design Choice 1: Removing Target Networks

Mnih et al. (2015) originally introduced target networks to stabilize the training of value-based off-policy RL methods, and today, most algorithms require them (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018a). SAC updates the critics' target networks with Polyak Averaging

$$\boldsymbol{\theta}^{\circ} \leftarrow (1-\tau)\boldsymbol{\theta}^{\circ} + \tau\boldsymbol{\theta}, \tag{1}$$

where $\boldsymbol{\theta}^{\circ}$ are the target network parameters, and $\boldsymbol{\theta}$ are those of the trained critic. $\tau$ is the *target network momentum*; with a high $\tau = 1$ (equivalent to cutting out the target network), SAC training

SAC:

$$Q_{\boldsymbol{\theta}}(\boldsymbol{S}_t, \boldsymbol{A}_t) = \boldsymbol{q}_t$$
$$Q_{\boldsymbol{\theta}^\circ}(\boldsymbol{S}_{t+1}, \boldsymbol{A}_{t+1}) = \boldsymbol{q}^\circ_{t+1}$$
$$\mathcal{L}_{\boldsymbol{\theta}} = (\boldsymbol{q}_t - \boldsymbol{r}_t - \gamma\, \boldsymbol{q}^\circ_{t+1})^2$$

Cross$Q$ (Ours):

$$Q_{\boldsymbol{\theta}}\left(\begin{bmatrix}\boldsymbol{S}_t \\ \boldsymbol{S}_{t+1}\end{bmatrix}, \begin{bmatrix}\boldsymbol{A}_t \\ \boldsymbol{A}_{t+1}\end{bmatrix}\right) = \begin{bmatrix}\boldsymbol{q}_t \\ \boldsymbol{q}_{t+1}\end{bmatrix}$$
$$\mathcal{L}_{\boldsymbol{\theta}} = (\boldsymbol{q}_t - \boldsymbol{r}_t - \gamma\, \boldsymbol{q}_{t+1})^2$$

Figure 3: SAC without BatchNorm in the critic $Q_{\boldsymbol{\theta}}$ (left) requires target $Q$ values $\boldsymbol{q}^\circ_{t+1}$ to stabilize learning. Cross$Q$ with BatchNorm in the critic $Q_{\boldsymbol{\theta}}$ (right) removes the need for target networks and allows for a joint forward pass of both current and future values. Batches are sampled from the replay buffer $\mathcal{B}$: $\boldsymbol{S}_t, \boldsymbol{A}_t, \boldsymbol{r}_t, \boldsymbol{S}_{t+1} \sim \mathcal{B}$ and $\boldsymbol{A}_{t+1} \sim \pi(\boldsymbol{S}_{t+1})$ from the current policy.

can diverge, leading to explosive growth in $\boldsymbol{\theta}$ and the $Q$ predictions. Target networks stabilize training by explicitly delaying value function updates, arguably slowing down online learning (Plappert et al., 2018; Kim et al., 2019; Morales, 2020).

Recently, Yang et al. (2021) found that critics with Random Fourier Features can be trained without target networks, suggesting that the choice of layer activations affects the stability of training. Our experiments in Section 4.4 uncover an even simpler possibility: using bounded activation functions or feature normalizers is sufficient to prevent critic divergence in the absence of target networks, whereas the common choice of $\mathrm{ReLU}$ without normalization diverges. While others have used normalizers in Deep RL before, we are the first to identify that they make target networks redundant. Our next design choice exploits this insight to obtain an even greater boost.

## 3.2 Design Choice 2: Using Batch Normalization

Up to now, BatchNorm has not seen wide adoption in value-based off-policy RL methods, despite its success and widespread use in supervised learning (He et al., 2016; Santurkar et al., 2018). Some methods use BatchNorm in decoupled feature extractors for Deep RL networks (Ota et al., 2020; 2021), but not in the critic networks. Hiraoka et al. (2021) report that using BatchNorm in critics causes training to fail, and instead opt to use LayerNorm in DroQ.

**We find using BatchNorm *carefully*, when *additionally* removing target networks, performs surprisingly well, trains stably, and is, in fact, algorithmically simpler than current methods.**

We first explain why BatchNorm needs to be used *carefully*. Within the critic loss $[Q_{\boldsymbol{\theta}}(\boldsymbol{S}, \boldsymbol{A}) - (\boldsymbol{r} + \gamma Q_{\boldsymbol{\theta}^\circ}(\boldsymbol{S}', \boldsymbol{A}'))]^2$, predictions are made for two differently distributed batches of state-action pairs; $(\boldsymbol{S}, \boldsymbol{A})$ and $(\boldsymbol{S}', \boldsymbol{A}')$, where $\boldsymbol{A}' \sim \pi_\phi(\boldsymbol{S}')$ is sampled from the *current policy iterate*, while $\boldsymbol{A}$ originates from the replay buffer following historical behavior policies.

As the target network parameters are updated by Polyak Averaging from the live network (Equation 1), this also applies to the BatchNorm parameters. It is trivial to see that the BatchNorm running statistics of the live network, which were estimated from batches of $(\boldsymbol{s}, \boldsymbol{a})$ pairs, will not have *seen* samples $(\boldsymbol{s}', \pi(\boldsymbol{s}'))$ and will further not match their statistics. In other words, the state-action inputs evaluated by the target network will be out-of-distribution given its mismatched BatchNorm running statistics. It is well known that the prediction quality of BatchNorm-equipped networks degrades in the face of such test-time distribution shifts (Pham et al., 2022; Lim et al., 2023).

Removing the target network *elegantly* avoids this problem entirely. With the target network out of the picture, we simply concatenate both batches and feed them through the $Q$ network in a single forward pass, as illustrated in Figure 3 and shown in code in Figure 2. This simple trick ensures that BatchNorm's normalization moments arise from the union of both batches, corresponding to a $50 : 50$ mixture of their respective distributions. Such normalization layers *do not* perceive the $(\boldsymbol{s}', \pi_\phi(\boldsymbol{s}'))$ batch as being out-of-distribution. This small change to SAC allows the safe use of BatchNorm, and greatly accelerates training.

We are not the only ones to identify this way of using BatchNorm to tackle the distribution mismatch problem; indeed, other works in supervised learning, e.g., Test-Time Adaptation (Lim et al., 2023), EvalNorm (Singh & Shrivastava, 2019), and *Four Things Everyone Should Know to Improve Batch Normalization* (Summers & Dinneen, 2020) also use mixed moments to bridge this gap.

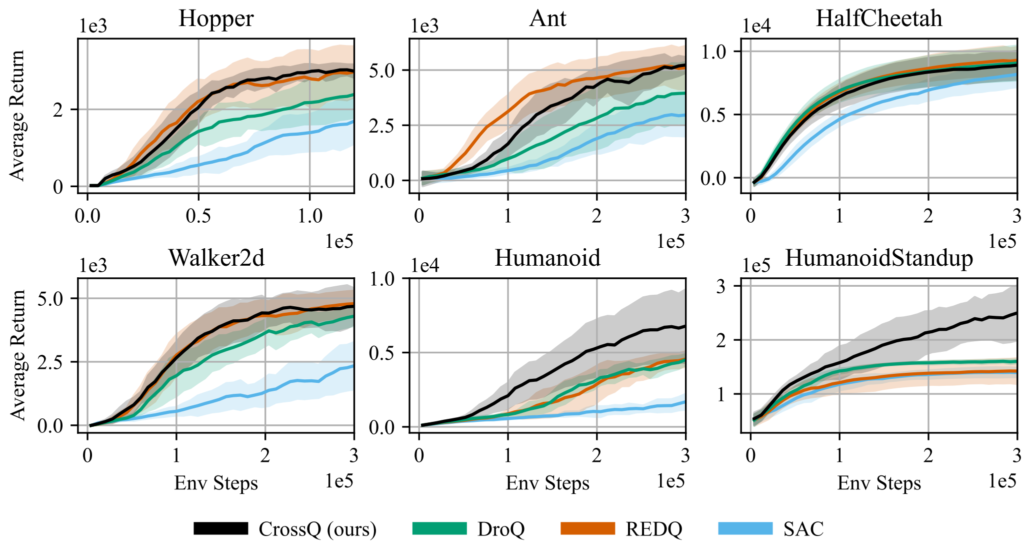

Figure 4: **Sample efficiency of Cross$Q$.** Compared to the REDQ and DroQ baselines (UTD $= 20$) Cross$Q$ (UTD $= 1$) performs either comparably, better, or — for the more challenging Humanoid tasks — substantially better. We plot the mean and std achieved average episode reward against the required environment steps across 10 random seeds.

### 3.3 Design Choice 3: Wider Critic Networks

Following Ota et al. (2021), we find that wider critic network layers in Cross$Q$ lead to even faster learning. As we show in our ablations in Section 4.4, most performance gains originate from the first two design choices; however, wider critic networks further boost the performance, making Cross$Q$ state-of-the-art in terms of sample efficiency.

We want to stress again that **Cross$Q$**, a UTD $= 1$ method, ***does not use bias-reducing ensembles, high UTD ratios or target networks***. Despite this, it achieves state-of-the-art sample efficiency at a fraction of the compute cost of REDQ and DroQ (see Figures 4 and 5).

## 4 Experiments and Analysis

We conduct experiments to provide empirical evidence for Cross$Q$'s performance, and investigate:

1. Sample efficiency of Cross$Q$ compared to REDQ and DroQ;
2. Computational efficiency in terms of wallclock time and performed gradient step;
3. Effects of the proposed design choices on the performance via Q function bias evaluations;

and conduct further ablation studies for the above design choices. We evaluate across a wide range of continuous-control `MuJoCo` (Todorov et al., 2012) environments, with 10 random seeds each. Following Janner et al. (2019); Chen et al. (2021) and Hiraoka et al. (2021), we evaluate on the same four `Hopper`, `Walker2d`, `Ant`, and `Humanoid` tasks, as well as two additional tasks: `HalfCheetah` and the more challenging `HumanoidStandup` from Gymnasium (Towers et al., 2023). We adapted the JAX version of stable-baselines (Raffin et al., 2021) for our experiments.

### 4.1 Sample Efficiency of Cross$Q$

Figure 4 compares our proposed Cross$Q$ algorithm with REDQ, DroQ, and SAC in terms of their sample efficiency, i.e., average episode return at a given number of environment interactions. We perform periodic evaluations during training to obtain the episodic reward. From these, we report the mean and standard deviations over 10 random seeds. All subsequent experiments in this paper follow the same protocol.

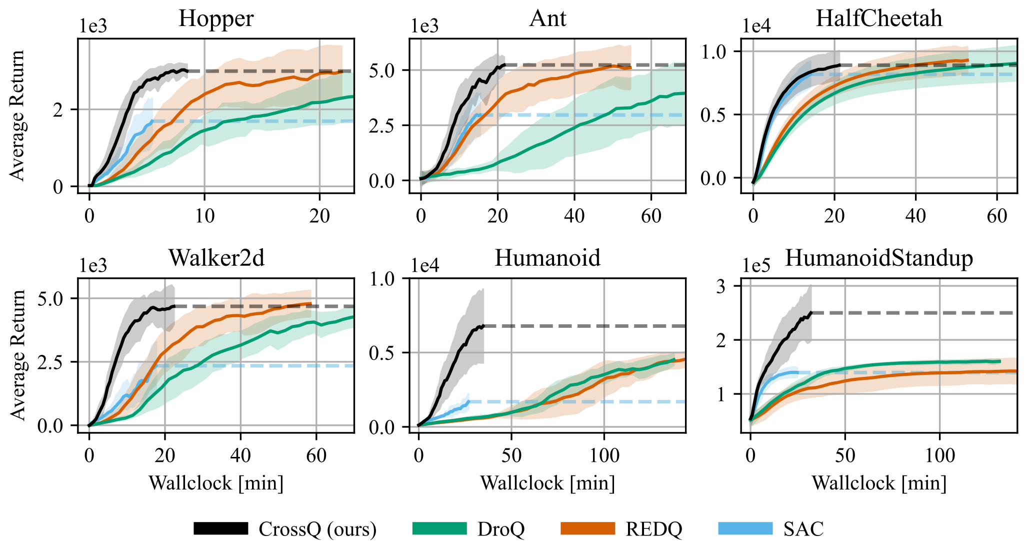

Figure 5: **Computational efficiency.** Cross$Q$ trains an order of magnitude faster: taking only $3 \times 10^5$ gradient steps to match or surpass the performance of the baselines at $6 \times 10^6$ steps, substantially saving on wallclock time. The dashed horizontal lines for Cross$Q$ and SAC are visual aids to better compare their final performance after training for the same amount of environment steps as REDQ and DroQ but with significantly less wallclock time, measurement described in Appendix A.2.

This experiment shows that Cross$Q$ matches or outperforms the best baseline in all the presented environments except on Ant, where REDQ performs better in the early training stage, but Cross$Q$ eventually matches it. On `Hopper`, `Walker`, and `HalfCheetah`, the learning curves of Cross$Q$ and REDQ overlap, and there is no significant difference. On the harder Humanoid and Humanoid-Standup tasks, Cross$Q$ substantially surpasses all other baselines.

## 4.2 Computational Efficiency of Cross$Q$

Figure 5 compares the computational efficiency of Cross$Q$ with the baselines. This metric is where Cross$Q$ makes the biggest leap forward. Cross$Q$ requires $20\times$ less gradient steps than REDQ and DroQ, which results in roughly $4\times$ faster wallclock speeds (Table 1). Especially on the more challenging `Humanoid` and `HumanoidStandup` tasks the speedup is the most pronounced. In our view, this is a noteworthy feature. On the one hand, it opens up the possibility of training agents in a truly online and data-efficient manner, such as in real-time robot learning. On the other hand, in the case of large computing budgets, it can allow the training of even larger models for longer than what is currently possible. The reason for the increased computational efficiency of Cross$Q$ lies in the use of a low UTD $= 1$.

Table 1: **Wallclock times.** Evaluated for Cross$Q$ and baselines across environments in minutes and recorded on an `RTX 3090`, the details of the measurement procedure are described in Appendix 4.2. Comparing Cross$Q$ with Cross$Q$ (Small) and SAC, it is apparent that using wider critic networks does come with a performance penalty. However, compared to REDQ and DroQ, one clearly sees the substantial improvement in Wallclock time of Cross$Q$ over those baselines.

| | Wallclock Time [minutes] | | | | |
|---|---|---|---|---|---|
| | SAC | Cross$Q$ (small) | **Cross$Q$ (ours)** | REDQ | DroQ |
| `Hopper-v4` | 5.43 | 5.30 | 8.46 | 21.89 | 23.82 |
| `Ant-v4` | 15.39 | 16.82 | 21.65 | 54.51 | 69.74 |
| `HalfCheetah-v4` | 14.05 | 15.64 | 21.12 | 52.70 | 65.72 |
| `Walker2d-v4` | 17.62 | 14.98 | 22.32 | 58.37 | 70.85 |
| `Humanoid-v4` | 26.76 | 26.00 | 34.66 | 138.27 | 137.55 |
| `HumanoidStandup-v4` | 24.68 | 24.41 | 31.85 | 141.39 | 131.51 |

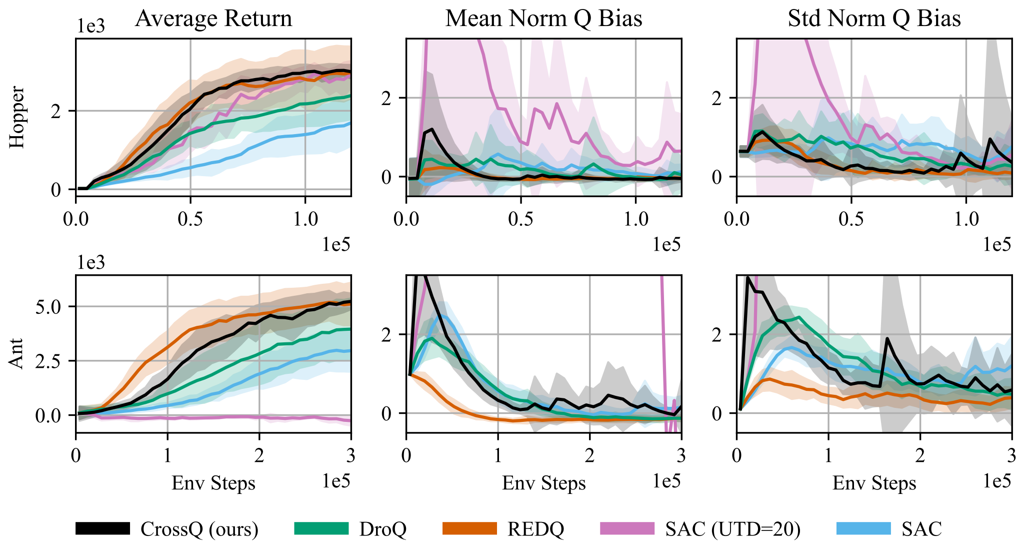

Figure 6: ***Q* estimation bias.** Following the analysis of Chen et al. (2021), we plot the mean and standard deviation of the normalized Q function bias. REDQ generally has the least bias. Cross$Q$ performs better than DroQ and SAC while having comparatively less and more bias in `Hopper` and `Walker`, respectively. The full set of environments is shown in Fig. 10.

## 4.3 Evaluating $Q$ Function Estimation Bias

All methods we consider in this paper are based on SAC and, thus, include the clipped double-Q trick to reduce Q function overestimation bias (Fujimoto et al., 2018). Chen et al. (2021) and Hiraoka et al. (2021) stress the importance of keeping this bias even lower to achieve their high performances and intentionally design REDQ and DroQ to additionally reduce bias with explicit and implicit ensembling. In contrast, Cross$Q$ outperforms both baselines without any ensembling. Could Cross$Q$'s high performance be attributed to implicitly reducing the bias as a side effect of our design choices? Using the same evaluation protocol as Chen et al. (2021), we compare the normalized Q prediction biases in Figure 4.3. Due to space constraints, here we show `Hopper` and `Ant` and place the rest of the environments in Figure 10 in the Appendix.

We find that REDQ and DroQ indeed have lower bias than SAC and significantly lower bias than SAC with UTD $= 20$. The results for Cross$Q$ are mixed: while its bias trend exhibits a lower mean and variance than SAC, in some environments, its bias is higher than DroQ, and in others, it is lower or comparable. REDQ achieves comparable or worse returns than CrossQ while maintaining the least bias. As Cross$Q$ performs better *despite* — perhaps paradoxically — having generally higher Q estimation bias, we conclude that the relationship between performance and estimation bias is complex, and one does not seem to have clear implications on the other.

## 4.4 Ablations

We conduct ablation studies to better understand the impact of different design choices in Cross$Q$.

### 4.4.1 Disentangling the Effects of Target Networks and BatchNorm

Cross$Q$ changes SAC in three ways; of these, two explicitly aim to accelerate optimization: the removal of target networks, and the introduction of BatchNorm. Unfortunately, SAC without target networks diverges, so to study the contribution of the first change, we need a way to compare SAC — divergence-free — *with and without target networks*. Fortunately, such a way exists: according to our supplementary experiments in Appendix A.3, simply using bounded activation functions in the critic appears to prevent divergence. This is a purely empirical observation, and an in-depth study regarding the influence of activations and normalizers on the stability of Deep RL is beyond the scope of this paper. In this specific ablation, we use `tanh` activations instead of `relu`, solely as a tool to make the intended comparison possible.

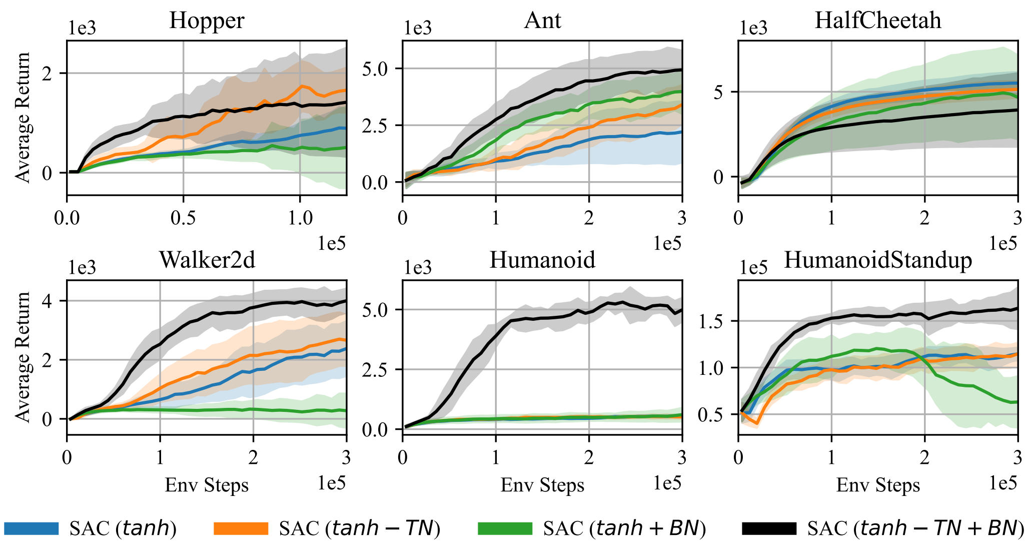

Figure 7: **The effects of target networks and BatchNorm on sample efficiency.** All SAC variants in this experiment use critics with $\tanh$ activations, since we find they allow divergence-free training without target networks, making this comparison possible. Removing target networks ($-\text{TN}$) provides only small improvements over the SAC baseline with target nets. Using BatchNorm together with target nets ($+\text{TN} + \text{BN}$) exhibits instabilities. Using BatchNorm after removing target nets ($-\text{TN} + \text{BN}$) — the configuration most similar to Cross$Q$ — shows the best aggregate performance. All curves are averaged over 10 random seeds.

Figure 7 shows the results of our experiment. The performance of SAC without target networks supports the common intuition that target networks indeed slow down learning to a small extent. We find that the combination of BatchNorm and Target Networks performs inconsistently, failing to learn anything in half of the environments. Lastly, the configuration of BatchNorm without target networks — and the closest to Cross$Q$ — achieves the best aggregate performance, with the boost being significantly bigger than that from removing target networks alone. In summary, even though removing target networks may slightly improve in performance in some environments, it is the combination of removing target networks and adding BatchNorm that accelerates learning the most.

### 4.4.2 Ablating the Different Design Choices and Hyperparameters

In this subsection, we ablate the different design choices of Cross$Q$ in order to show their combined effectiveness and individual importance.

**Hyperparameters.** Cross$Q$ uses the best hyperparameters obtained from a series of grid searches. Of these, only three are different from SAC's default values. First, we increase the *policy delay* — the number of times we update the critic per actor update — from $1$ to $3$. Second, we reduce the $\beta_1$ momentum for the Adam optimizer (Kingma & Ba, 2015) from $0.9$ to $0.5$[1] Third, we increase the critic network width from $256$ to $2048$. Figure 8 shows that resetting any one of those hyperparameters back to their SAC default values reduces performance slightly, but still allows strong performance across most environments. One important observation is that the Cross$Q$ (small) with layer width of $256$ trains almost as fast as the original SAC, however, still exhibits comparably strong performance in four out of the six environments. This suggests that practitioners may be able to make use of a larger compute budget, i.e., train efficiently across a range of different network sizes, by scaling up layer widths according to the available hardware resources.

[1]Reducing Adam's momentum is a common choice in Generative Adversarial Network training, specifically to help the generator-discriminator optimization dynamics to converge (Radford et al., 2016; Salimans et al., 2016; Gemp & McWilliams, 2019), a framework formally similar to actor-critic training (Finn et al., 2016).

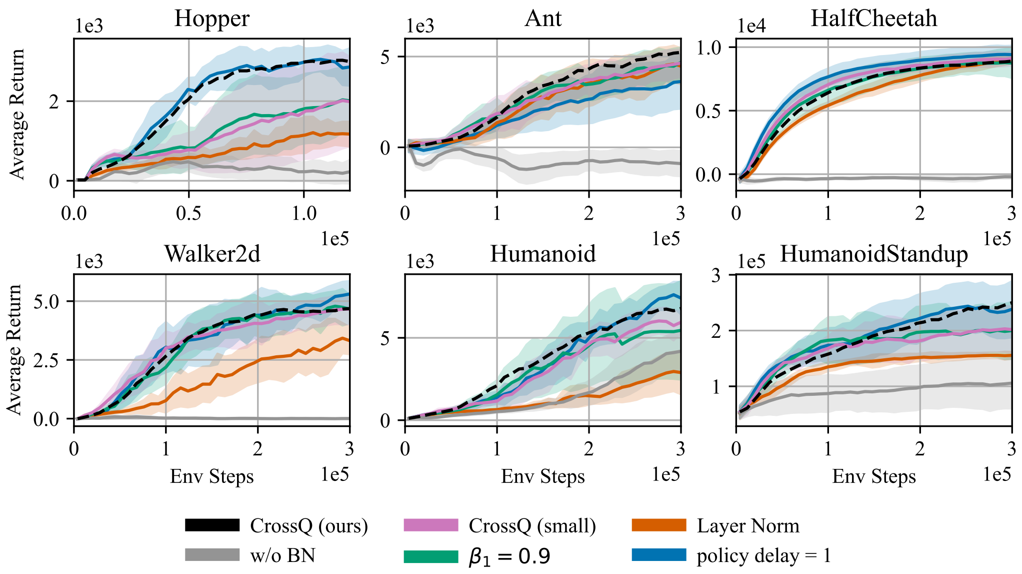

Figure 8: **Cross$Q$ ablation study.** We ablate across different hyperparameter settings and architectural configurations. In reducing the network width to be the same as SAC, Cross$Q$ (small) shows weaker performance, is still competitive with Cross$Q$ in four out of six environments. Using the default Adam momentum $\beta_1 = 0.9$ (green) instead of $0.5$ degrades performance in some environments. Using a policy delay of 1 (blue) instead of 3 has a very small effect, except on Ant. Using LayerNorm (orange) instead of BatchNorm results in slower learning; it also trains stably without target networks. Removing BatchNorm (gray) results in failure of training due to divergence.

**Removing BatchNorm.** As we would expect, removing BatchNorm does not work well (grey curve in Figure 8). This particular configuration amounts to training SAC with `ReLU` activations and without target networks, which is known to be unstable.

**Using Layer Normalization.** Another interesting question is whether other normalization strategies in the critic, such as Layer Normalization (LayerNorm) (Ba et al., 2016), would also give the same results. Figure 8 shows that replacing BatchNorm with LayerNorm degrades the performance.

## 5 Conclusion & Future Work

We introduced Cross$Q$, a new state-of-the-art off-policy RL algorithm, both in terms of sample efficiency and computational efficiency. To the best of our knowledge, Cross$Q$ is the first successful application of BatchNorm in off-policy actor-critic RL. Through benchmarks and ablations, we confirmed that target networks do indeed slow down training and showed a way to remove them without sacrificing training stability. We also showed that BatchNorm has the same accelerating effect on training in Deep RL as it does in supervised deep learning. The combined effect of removing target networks and adding BatchNorm is what makes Cross$Q$ so efficient. We investigated the relationship between the Q estimation bias and the learning performance of Cross$Q$, but did not identify a straightforward dependence. This indicates that the relationship between the Q estimation bias and the agent performance is more complex than previously thought.

In future work, it would be interesting to analyze the Q estimation bias more extensively, similar to (Li et al., 2022). Furthermore, a deeper theoretical analysis of the used BatchNorm approach in the context of RL would be valuable, akin to the works in supervised learning, e.g., (Summers & Dinneen, 2020). Although the wider critic networks do provide an additional performance boost, they increase the computation cost, which could potentially be reduced. Finally, while our work focuses on the standard continuous control benchmarking environments, a logical extension would be applying Cross$Q$ to a real system and using visual observations in addition to the robot state.

## Acknowledgments

We acknowledge the grant “Einrichtung eines Labors des Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) an der Technischen Universität Darmstadt” of the Hessisches Ministerium für Wissenschaft und Kunst. This work is also funded by the Hessian Ministry of Science and the Arts (HMWK) through the project “The Third Wave of Artificial Intelligence - 3AI”, Hessian.AI and by the German Research Foundation (DFG): 417962828.

## References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in neural information processing systems*, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International conference on learning representations*, 2021.

Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for dynamics and control*, 2020.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 2018.

Ian M. Gemp and Brian McWilliams. The unreasonable effectiveness of adam on cycles. In *NeurIPS Workshop on Bridging Game Theory and Deep Learning*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Hado Hasselt. Double q-learning. In *Advances in neural information processing systems*, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on computer vision and pattern recognition*, 2016.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International conference on learning representations*, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 2015.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in neural information processing systems*, 2019.

Seungchan Kim, Kavosh Asadi, Michael L. Littman, and George Dimitri Konidaris. Deepmellow: Removing the need for a target network in deep q-learning. In *International joint conference on artificial intelligence*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International conference on learning representations*, 2015.

Qiyang Li, Aviral Kumar, Ilya Kostrikov, and Sergey Levine. Efficient deep reinforcement learning requires regulating overfitting. In *International conference on learning representations*, 2022.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International conference on machine learning*, 2016.

Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. TTN: A domain-shift aware batch normalization in test-time adaptation. In *International conference on learning representations*, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

M. Morales. *Grokking Deep Reinforcement Learning*. Manning Publications, 2020.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, 2022.

Kei Ota, Tomoaki Oiki, Devesh Jha, Toshisada Mariyama, and Daniel Nikovski. Can increasing input dimensionality improve deep reinforcement learning? In *International conference on machine learning*, 2020.

Kei Ota, Devesh K Jha, and Asako Kanezaki. Training larger networks for deep reinforcement learning. *arXiv preprint arXiv:2102.07920*, 2021.

Quang Pham, Chenghao Liu, and Steven HOI. Continual normalization: Rethinking batch normalization for online continual learning. In *International conference on learning representations*, 2022.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International conference on learning representations*, 2016.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, 2016.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in neural information processing systems*, 2018.

Saurabh Singh and Abhinav Shrivastava. Evalnorm: Estimating batch normalization statistics for evaluation. In *International conference on computer vision*, 2019.

Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

Cecilia Summers and Michael J. Dinneen. Four things everyone should know to improve batch normalization. In *International conference on learning representations*, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International conference on intelligent robots and systems*, 2012.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, 2023.

Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. In *International conference on learning representations*, 2021.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

## A APPENDIX

### A.1 HYPERPARAMETERS

Experiment hyperparameters, used in the main paper. We adapted most hyperparameters that are commonly used in other works (Haarnoja et al., 2018b; Chen et al., 2021; Hiraoka et al., 2021).

Table 2: Learning Hyperparameters

| **Parameter** | SAC | REDQ | DroQ | Cross$Q$ (ours) |
|---|---|---|---|---|
| Discount Factor ($\gamma$) | 0.99 | 0.99 | 0.99 | 0.99 |
| Learning Rate (Actor+Critic) | 0.001 | 0.001 | 0.001 | 0.001 |
| Replay Buffer Size | $10^6$ | $10^6$ | $10^6$ | $10^6$ |
| Batch Size | 256 | 256 | 256 | 256 |
| Activation Function | `ReLU` | `ReLU` | `ReLU` | `ReLU` |
| Layer Normalization | No | No | Yes | No |
| Dropout Rate | N/A | N/A | 0.01 | N/A |
| Batch Normalization | No | No | No | Yes |
| Critic Width | 256 | 256 | 256 | 2048 |
| Target Update Rate ($\tau$) | 0.001 | 0.001 | 0.001 | N/A |
| Adam $\beta_1$ | 0.9 | 0.9 | 0.9 | 0.5 |
| Update-to-Data ratio (UTD) | 1 | 20 | 20 | 1 |
| Policy Delay | 2 | 20 | 20 | 3 |
| Number of Critics | 2 | 10 | 2 | 2 |

### A.2 WALLCLOCK TIME MEASUREMENT

Wallclock times were measured by timing and averaging over four seeds each and represent *pure training times*, without the overhead of synchronous evaluation and logging, until reaching $3 \times 10^5$ environment steps and $1.2 \times 10^5$ for `Hopper-v4`. The times are recorded on an `Nvidia RTX 3090 Turbo` with an `AMD EPYC 7453` CPU.

### A.3 ACTIVATIONS

Figure 7 depicts a small exploratory experiment in which we remove target networks from SAC, and train it with different activation functions and feature normalizers. We do this only to explore whether the boundedness of activations has an influence on training stability. We learn from this experiment that SAC with $\tanh$ activations trains without divergence, allowing us to conduct the study in Section 4.4.1. We also observe that at least two feature normalization schemes (on top of the unbounded relu activations) permit divergence-free optimization.

For vectors $\boldsymbol{x}$, $\text{relu\_over\_max}(\boldsymbol{x})$ denotes a simple normalization scheme using an underlying unbounded activation: $\text{relu}(\boldsymbol{x})/\max(\boldsymbol{x})$, with the maximum computed over the entire feature vector. $\text{layernormed\_relu}$ simply denotes LayerNorm applied *after* the $\text{relu}$ activations. Both of these schemes prevent divergence. Using LayerNorm *before* the $\text{relu}$ activations also prevents divergence, and is already explored in the ablations in Figure 8. None of these normalizers perform as strongly as BatchNorm.

A thorough theoretical or experimental study of how activations and normalizers affect the stability of Deep RL is beyond the scope of this paper. We hope, however, that our observations help inform future research directions for those interested in this topic.

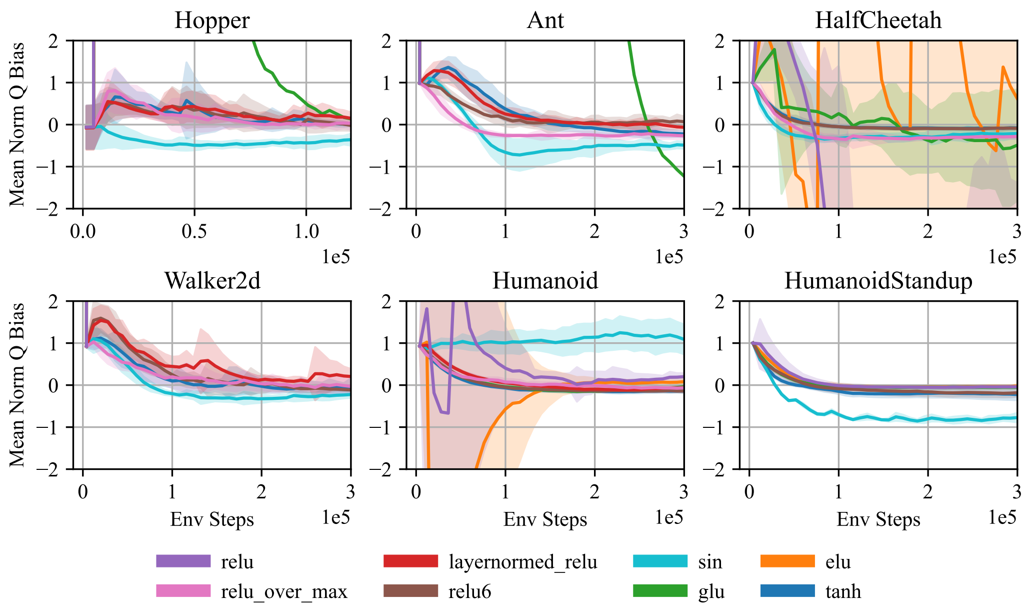

Figure 9: (In)stability of SAC without target networks, observed through the Q estimation bias. In this small-scale experiment, we run SAC with unbounded $(\text{relu}, \text{glu}, \text{elu})$ and bounded $(\tanh, \text{relu6}, \sin)$ activation functions, as well as "indirectly" bounded activations through the use of two custom normalizers other than BatchNorm $(\text{relu\_over\_max}, \text{layernormed\_relu})$. SAC variants with unbounded activations appear highly unstable in most environments, whereas the variants with bounded activations (as well as the normalizers) do not diverge, maintaining relatively low bias.

### A.4 Normalized $Q$-Bias Plots

Figure 10 shows the results of the Q function bias analysis for all environments.

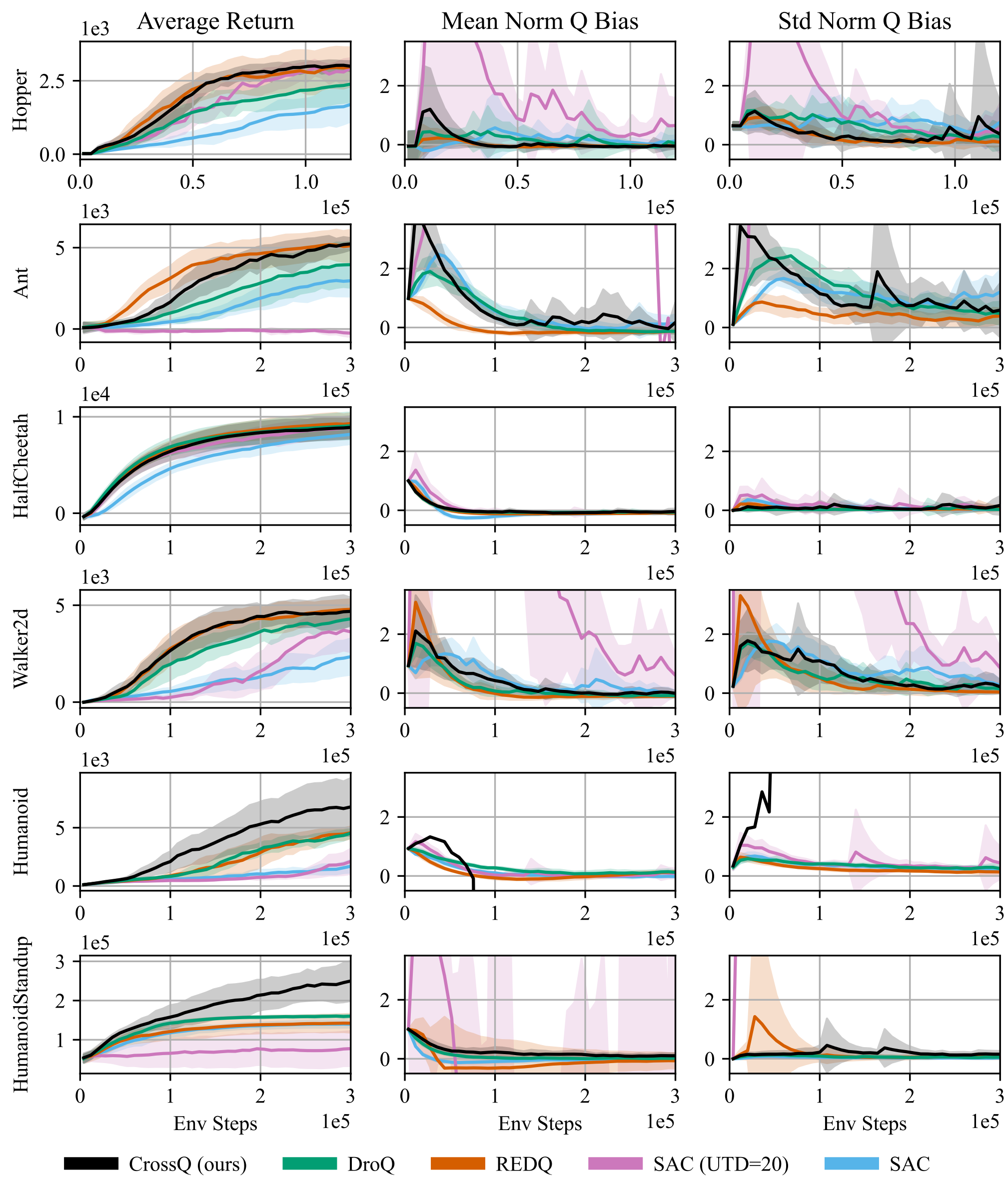

Figure 10: Q estimation bias. Mean and standard deviation of the normalized Q function bias, computed as described by Chen et al. (2021). As in the main paper, we do not find a straightforward connection between normalized Q function bias and learning performance. Cross$Q$ generally shows the same or larger Q estimation bias compared to REDQ but matches or outperforms REDQ in learning speed, especially on the challenging `Humanoid` tasks. We note that on `Humanoid`, the normalized Q biases explode. The reason for this is that two seeds do not learn (flatline) with exploding Q function predictions. Their influence on the Q bias evaluation is disproportionately high. Removing the two seeds from the plotting `Humanoid` shows similar Q bias trends as in the other environments, but in full transparency, we did not remove them here.