A PROOFS

A.1 THEOREM 1

This proof (of Theorem 1) uses techniques from the proof of Theorem 1 in the paper (Pan et al., 2020), adapting them to the setting considered in this paper. An informal overview is as follows. The maximum of the two Q values is a particular case in the computation of the operator. Here, we mainly show this process also can guarantee convergence. We start with a proposition from the paper (Pan et al., 2020) that shows the relation between the soft-max and the log-sum-exp function.

Proposition 1

$$lse_\beta(\mathbf{X}) - sm_\beta(\mathbf{X}) = \frac{1}{\beta}H(\mathbf{X}) = \frac{1}{\beta}\sum_{i=1}^{n} -p_i \log(p_i) \leq \frac{\log(n)}{\beta}, \tag{10}$$

where $p_i = \frac{e^{\beta x_i}}{\sum_{j=1}^{n} e^{\beta x_j}}$ *denotes the weights of the softmax distribution,* $lse_\beta(\mathbf{X})$ *denotes the log-sum-exp function* $lse_\beta(\mathbf{X}) = \frac{1}{\beta}\log(\sum_{i=1}^{n} e^{\beta x_i})$*, and* $sm_\beta(\mathbf{X})$ *denotes the softmax function* $sm_\beta(\mathbf{X}) = \frac{\sum_{i=1}^{n} e^{\beta x_i} x_i}{\sum_{j=1}^{n} e^{\beta x_j}}$*.* $H(\mathbf{X})$ *is the entropy of the distribution. It is easy to check that the maximum entropy is achieved when* $p_i = \frac{1}{n}$*, where the entropy equals to* $\log(n)$*.*

Proof.

$$\begin{aligned}
&\frac{1}{\beta}\sum_{i=1}^{n} -p_i \log(p_i) \\
=&\frac{1}{\beta}\sum_{i=1}^{n}\left(-\frac{e^{\beta x_i}}{\sum_{j=1}^{n} e^{\beta x_j}} \log\left(\frac{e^{\beta x_i}}{\sum_{j=1}^{n} e^{\beta x_j}}\right)\right) \\
=&\frac{1}{\beta}\sum_{i=1}^{n}\left(-\frac{e^{\beta x_i}}{\sum_{j=1}^{n} e^{\beta x_j}}\left(\beta x_i - \log\left(\sum_{j=1}^{n} e^{\beta x_j}\right)\right)\right) \\
=&-\sum_{i=1}^{n}\frac{e^{\beta x_i} x_i}{\sum_{j=1}^{n} e^{\beta x_j}} + \frac{1}{\beta}\log\left(\sum_{j=1}^{n} e^{\beta x_j}\right)\frac{\sum_{i=1}^{n} e^{\beta x_i}}{\sum_{j=1}^{n} e^{\beta x_j}} \\
=&-\mathrm{sm}_\beta(\mathbf{X}) + \mathrm{lse}_\beta(\mathbf{X})
\end{aligned}$$

□

Theorem 1 (Convergence of value iteration with the DDQS operator) *For any dynamic double Q softmax operator* gdq_{β_t}*, if* β_t *approaches* ∞*, the value function after* t *iterations* v_t *converges to the optimal value function* $V*$*.*

Proof.

$$||(\mathcal{T}_{\beta_t} V_i) - (\mathcal{T}_{\beta_t} V_j)||_\infty \tag{11}$$

$$= \max_s |\mathrm{ddqs}_{\beta_t}(Q_i^{max}(s,\cdot)) - \mathrm{ddqs}_{\beta_t}(Q_j(s,\cdot))| \tag{12}$$

$$\begin{aligned}
\leq &\max_s |\mathrm{lse}_{\beta_t}(Q_i^{(}s,\cdot)) - \frac{1}{\beta}H(Q_i(s,\cdot)) \\
&- \mathrm{lse}_{\beta_t} Q_j(s,\cdot)) + \frac{1}{\beta}H(Q_j(s,\cdot))|
\end{aligned} \tag{13}$$

$$\leq \underbrace{\max_s |\mathrm{lse}_{\beta_t}(Q_i(s,\cdot)) - \mathrm{lse}_{\beta_t}((Q_j(s,\cdot))|}_{(A)} + \underbrace{\frac{\log(|A|)}{\beta_t}}_{(B)}, \tag{14}$$

For the term (A), the log-sum-exp operator has been proved as a non-expanding operator in the paper (Fox et al., 2015). In mathematics, the log-sum-exp function is almost equal to the max function. That is why this makes sense. Here, we prove that it's also a non-expanding operator in our greedy Q setting. Define a norm on Q value as $\|Q_i - Q_j\| \triangleq \max_{\mathbf{s},\mathbf{a}} |Q_i(\mathbf{s},\mathbf{a}) - Q_j(\mathbf{s},\mathbf{a})|$. Suppose $\epsilon = \|Q_i - Q_j\|$. Please note that in our setting,$Q_i(\mathbf{s},\mathbf{a}) = \min Q_i^1(\mathbf{s},\mathbf{a}), Q_i^2(\mathbf{s},\mathbf{a})$, $Q_j(\mathbf{s},\mathbf{a})$ is similar defined. Then

$$\begin{aligned}\log \int \exp(Q_i(\mathbf{s}',\mathbf{a}'))\, d\mathbf{a}' &\leq \log \int \exp(Q_j(\mathbf{s}',\mathbf{a}') + \epsilon)\, d\mathbf{a}' \\ &= \log \left(\exp(\epsilon) \int \exp Q_j(\mathbf{s}',\mathbf{a}')\, d\mathbf{a}' \right) \\ &= \epsilon + \log \int \exp Q_j(\mathbf{a}',\mathbf{a}')\, d\mathbf{a}'. \end{aligned} \tag{15}$$

Similarly, $\log \int \exp Q_i(\mathbf{s}',\mathbf{a}')\, d\mathbf{a}' \geq -\epsilon + \log \int \exp Q_j(\mathbf{s}',\mathbf{a}')\, d\mathbf{a}'$. Therefore $\|\mathcal{T}Q_i - \mathcal{T}Q_j\| \leq \gamma\epsilon = \gamma\|Q_i - Q_j\|$.

Consider for Q^1 and Q^2, we expand the composed Q function, we derive

$$\|\mathcal{T}Q_i - \mathcal{T}Q_j\| \leq \gamma\|Q_i - Q_j^1\|, \|\mathcal{T}Q_i - \mathcal{T}Q_j\| \leq \gamma\|Q_i - Q_j^2\|$$

when $Q_i = Q_i^1$, it means Q_i^1 is the min value, for the term (A),we have

$$\begin{aligned}&\max_s |\mathrm{lse}_{\beta_t}(Q_i(s,\cdot)) - \mathrm{lse}_{\beta_t}((Q_j(s,\cdot))| \\ &\leq \max_{s,a} \frac{1}{\beta_t} |\beta_t Q_i^1 - \beta_t Q_j^1| \\ &\leq \gamma \max_{s,a} \sum_{s'} p(s'|s,a)|V_i(s') - V_j(s')| \\ &\leq \gamma ||V_i - V_j||_\infty \end{aligned}$$

So $\mathcal{T}$ is a contraction. We can get the same result when $Q_i = Q_i^2$. And due to the min operator, Q^1 and Q^2 are updated iteratively. Consequently, the two Q functions converge to the optimal value, satisfying the modified Bellman equation. Thus, the optimal policy is unique.

For the term (B), the details can be found in (Pan et al., 2020). A direct understanding is that as β increases, this term will eventually become zero. ☐

B ALGORITHM

Compared to SAC and OAC algorithms, the BACC algorithm 1 primarily makes modifications in the exploration component. In practical applications, instead of calculating the state-value function, BACC directly updates the Q-functions. In our algorithm, several key steps are followed: 1)Dynamic Increase of β_t (line 3): This is done to ensure the convergence of the Q-value, as described in section 4.1; 2)Action Sampling from Exploration Policy (line 5): Actions are sampled from the exploration policy to interact with the environment, as detailed in section 4.2; 3)Storing Transitions in Memory Buffer: The resulting transitions are stored in a memory buffer; 4)Updating the Q-function (line 12) and Actor (line 13): BACC samples transitions from the memory buffer to update both the Q-function and the actor, as explained in section 4.3.

C HYPER PARAMETERS

Table 1 provides an overview of the common BACC parameters utilized in the comparative evaluation presented in Figure 3. However, for the results in Figure 5(a-b), which were evaluated on the RoboSchool benchmark, we made slight adjustments to the hyperparameters. In the HumanoidFlagrunHarder-v1 environment, we set β_t to 10, and s_r to 3. In the HumanoidFlagrun-v1 environment, we used β_t=1 and s_r=0.1.

Algorithm 1 Bold Actor and Conservative Critic (BACC).

```
Require: $\theta_1, \theta_2, \phi$    ▷ Initial parameters $\theta_1, \theta_2$ of the Q function and $\phi$ of the target policy $\pi_T$.
$\breve{\theta}_1 \leftarrow \theta_1, \breve{\theta}_2 \leftarrow \theta_2, \mathcal{D} \leftarrow \emptyset$    ▷ Initialize target network weights and replay buffer
for each iteration do
    increase $\beta_t$ according to the iteration number
    for each environment step do
        $\mathbf{a}_t \sim \pi_E(\mathbf{a}_t|\mathbf{s}_t, \beta_t)$    ▷ Sample action from exploration policy as in (4.2).
        $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$    ▷ Sample state from the environment
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, R(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$    ▷ Store the transition in the replay buffer
    end for
    for each training step do
        sample batch transition $(\mathbf{s}_t, \mathbf{a}_t, R(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})$ from the buffer
        compute target $\hat{Q}^i(\mathbf{s}_t, \mathbf{a}_t)$ for $i \in 1, 2$
        update $\theta_i$ with $\hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in 1, 2$    ▷ Q parameter update
        update $\phi$ with $\hat{\nabla}_{\phi} J_\pi(\phi)$    ▷ Policy parameter update
        $\breve{\theta}_1 \leftarrow \tau\theta_1 + (1-\tau)\breve{\theta}_1, \breve{\theta}_2 \leftarrow \tau\theta_2 + (1-\tau)\breve{\theta}_2$    ▷ Update target networks
    end for
end for
Ensure: $\theta_1, \theta_2, \phi$    ▷ Optimized parameters
```

Parameter	Value
Shared	
optimizer	Adam
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers (all networks)	2
number of hidden units per layer	256
number of samples per minibatch	256
nonlinearity	ReLU
SAC	
target smoothing coefficient (τ)	0.005
target update interval	1
gradient steps	1
OAC	
beta UB	4.66
delta	23.53
BACC	
dynamic weight (β_t)	epoch number*1
sample range (s_r)	7
sample size (s_n)	32

Table 1: BACC Hyper-parameters

D ENVIRONMENT PROPERTIES

The properties of each environment are summarized in Table 2.

Environment	State dim	Action dim	Episode Length
Humanoid-v2	376	17	1000
Ant-v2	111	8	1000
HalfCheetah-v2	17	6	1000
Walker2d-v2	17	6	1000
Hopper-v2	11	3	1000
Swimmer-v2	8	2	1000

Table 2: The details of Mujoco Environments used in this paper.

E MORE RESULTS

E.1 RELATION BETWEEN BETA AND SAMPLE SIZE

We sample n integers uniformly from [1,1000] and give the numerical result of the $\text{lse}_\beta(\mathbf{X})$, $\text{sm}_\beta(\mathbf{X})$ and $\frac{1}{\beta}H(\mathbf{X})$. According to the results shown in Table 3,Table 4,Table 5, we can see that a large beta will lead $\text{sm}_\beta(\mathbf{X})$ give a consistent result with the max operator. And, $\text{sm}_\beta(\mathbf{X})$ is better than $\text{lse}_\beta(\mathbf{X})$, it approximate the maximum from the lower bound. With a large beta, the maximum value can be aprroximated regardless of the sample size.

n, $beta$=0.01	$\text{lse}_\beta(\mathbf{X})$	$\text{sm}_\beta(\mathbf{X})$	$\frac{1}{\beta}H(\mathbf{X})$	maximum
10	992.46	932.61	59.85	976
100	1252.17	900.62	351.50	995
1000	1461.84	899.60	561.094	999
10000	1692.55	900.98	779.09	999
100000	1920.21	899.25	896.54	999
1000000	2150.74	899.56	nan	999

Table 3: The results when $beta$=0.01.

n, $beta$=1	$\text{lse}_\beta(\mathbf{X})$	$\text{sm}_\beta(\mathbf{X})$	$\frac{1}{\beta}H(\mathbf{X})$	maximum
10	834.69	834.00	0.69	834
100	970.02	969.92	0.09	970
1000	1000.29	998.70	1.58	999
10000	1001.69	998.31	3.38	999
100000	1004.10	998.41	5.68	999
1000000	1006.36	998.41	7.85	999

Table 4: The results when $beta$=1.

E.2 REWARD DIFFERENCE BETWEEN EXPLORATION AND EVALUATION

As shown in Figure 6, Because both algorithms are related to exploration, the evaluation return is higher than the exploration return. However, something goes wrong in OAC exploration. As we use transition sampled from the replay buffer to train the policy, it does not seem to have much impact on policy learning. Instead, it shows that our exploration strategy is better than the OAC method.

E.3 REWARD COMPARISON BETWEEN BACC AND OAC

As shown in Figure 7, our method performs better in policy evaluation. Another thing to note is that our method is inspired by OAC, and we found some problems with the exploration strategy of OAC. Therefore, we need to prove that our exploration strategy is better, as shown in Figure 8, this figure shows that our exploration strategy is better.

n, $beta$=100	$\text{lse}_\beta(\mathbf{X})$	$\text{sm}_\beta(\mathbf{X})$	$\frac{1}{\beta}H(\mathbf{X})$	maximum
10	947.0	947.0	0.0	947
100	996.0	996.0	0.0	996
1000	997.0	997.0	0.0	997
10000	999.02	999.00	0.02	999
100000	999.05	999.0	0.05	999
1000000	999.07	999.0	0.07	999

Table 5: The results when $beta$=100.

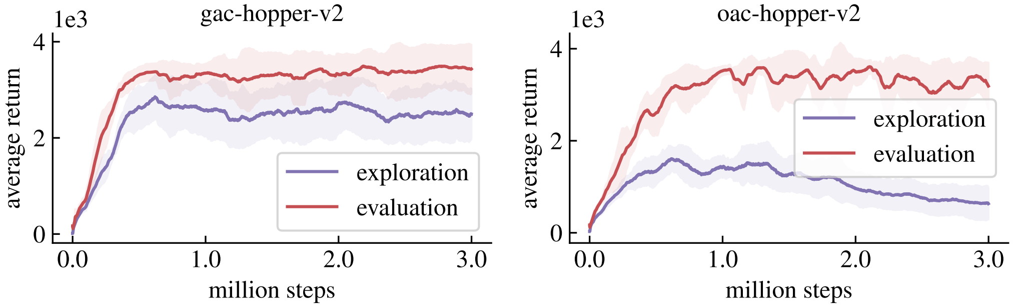


Figure 6: Exploration and evaluation difference of BACC and OAC

E.4 More Hyper-Parameter Experiments

Learning the Q value. The parameter β_t influences the learning of the Q function. In practice, we dynamically increase the β_t by setting it as the multiplication of β and the epoch number of timestep t. In the early stage of training, the Q function cannot provide little information, so a small β can be used to encourage the exploration. The different β results are shown in Figure 9(a). As we can see, a smaller β can produce a slightly better result. Our method is not so sensitive to this parameter in our setting. Nevertheless, a smaller β indeed takes a better result.

Better exploration. As shown in Figure 9(b), if we use a smaller value of s_r, that is, we sample action around the current policy, we can see the final result compared to the two larger values is terrible, which shows that **aimless exploration** does lead to poor results, it is difficult to get good results if s_r is too small. Policy learning depends on how well the initial policy is. When s_r=7, we get better results indicating that the **policy divergence** problem can be solved by explicit sampling point out-of-distribution. Thus, we can do more OOD sampling with the Q function to explore action space. Additionally, according to the results of s_r=7 and s_r=9, we know that s_r should not be as big as possible. Bigger s_r does not mean better results because the policy gradient comprises the $\nabla_\phi Q$ and $\nabla_\phi \pi$. Suppose we optimize the policy with many low-probability actions, which may prefer by the Q function. In that case, the policy gradient may be too small to promote the parameter update.

Exploration policy. When constructing an exploration policy, a key question is how many actions need to be evaluated to obtain a usable exploration policy. If constructing the policy needs many actions to be evaluated, our method becomes computationally burdensome and resource-intensive. As shown in Figure 9(c), when s_n=10, the final result is the best; when s_n=1000, the best result is achieved at about one million steps, but the final result does not gain an advantage. According to the results in the figure, it can be seen that BACC can be effective by evaluating only a limited number of actions, and the improvement of the final performance does not lie in the complete evaluation of actions in the action space. Instead, the exploration strategy is the main factor for performance improvement.

Time cost. Our experimental setup consists of a computer with Ubuntu 18 operating system, equipped with a 9900K CPU and an RTX 2060 GPU. Without using the exploration strategy pro-

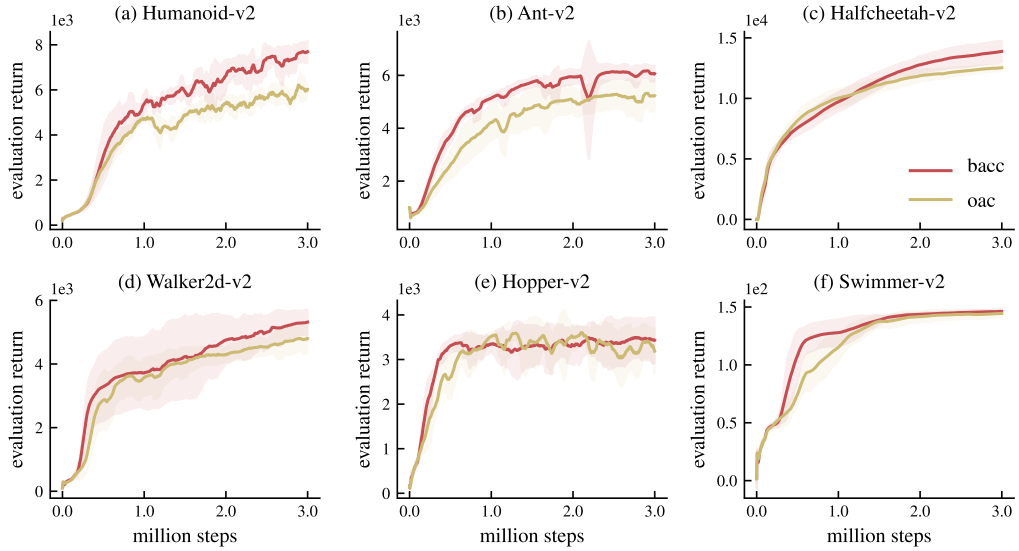


Figure 7: Average evaluation episode return of BACC and OAC

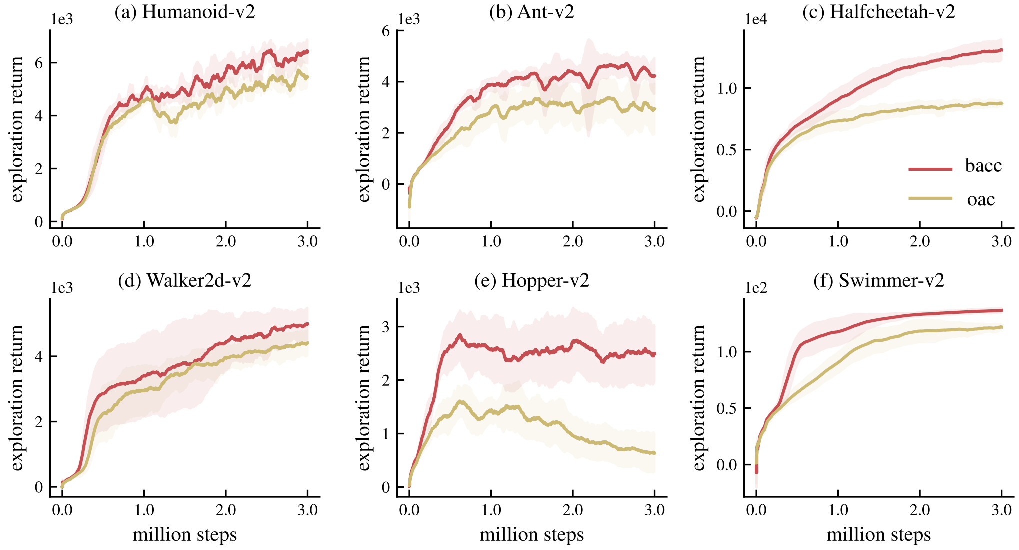
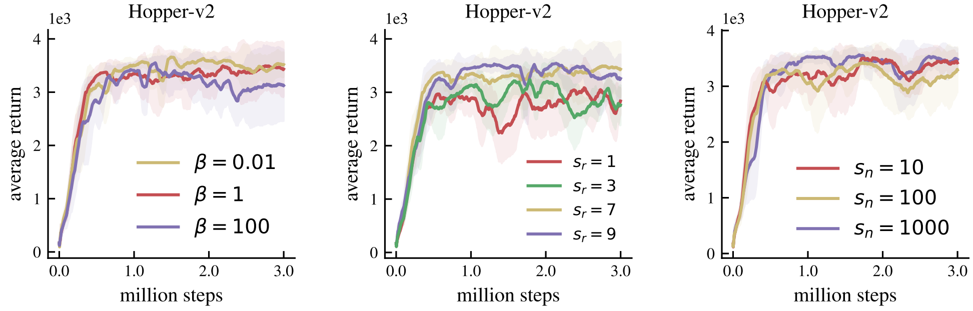


Figure 8: Average exploration episode return of BACC and OAC

posed in this paper, it takes 6.619 seconds to complete one epoch on average. In most of our experiments, we evaluate 32 actions. With this setting, running one epoch takes an average of 7.29 seconds. If we evaluate 64 actions, running one epoch takes an average of 7.09 seconds. The GPU may be more efficient when computing data with a batch size 64. From this, the additional time added due to exploration is insignificant. One epoch requires 1000 interactions with the environment. When averaged per exploration step, the time consumed is almost negligible.

We show all the results of different hyper-parameters on these six environments. As shown in Figure 10, Figure 11 and Figure 12, combine the numerical results, we can better choose the value for hyper-parameters.

E.5 VISUALIZATION FOR THE Q VALUE



(a) Hyper-parameter β (b) Sample range (c) Sample size

Figure 9: Three hyper-parameters related to the exploration policy. (a): In practice, the value of β_t is obtained by multiplying β with the epoch number of timestep t. (b): The parameter s_r determines the sample range, where a large value indicates that sampled actions could deviate further from the distribution of the current policy. (c): We uniformly sample s_n actions within the sample range to construct our exploration policy.

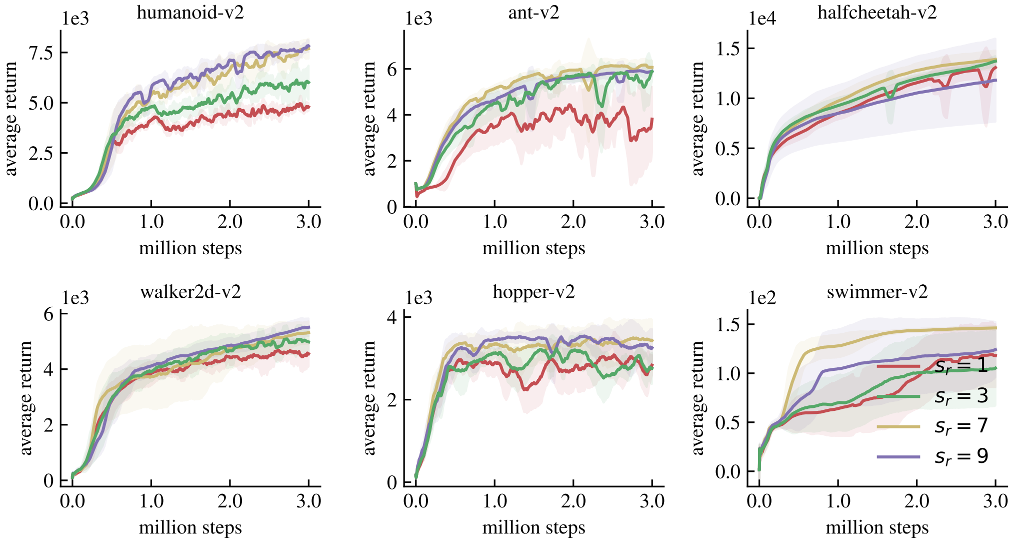


Figure 10: The results of different sample range on the six environments.

Visualization of the Q function. Our approach is based on the maximum entropy framework and assumes that the Q function is in an energy-based form. We design experiments to validate this assumption. The action space in the Simmer environment is two-dimensional, making it an ideal validation environment. We select an intermediate state of the Q network during the training process, sample 400*400 points across the entire action space, and calculate the corresponding Q values. The results we obtained are shown in Figure 13. We plot the 3d surface of the Q function, and a 2d plane for rotor2=-1.

To observe the surface of the Q network, we plot different stage Q value, which is evaluated based on a random start state and sampled actions in the Swimmer-v2 environments. As shown in Figure 14,in the initial stage, the surface is not flat, which is influenced by the input (state and action); if the state and action is zero vector, this surface should be flat, all zero. This phenomenon shows that neural networks imply prior knowledge about choosing actions. Policy initialization is closely related to policy learning. As training progresses, the final optimal action dramatically differs from the initial policy.

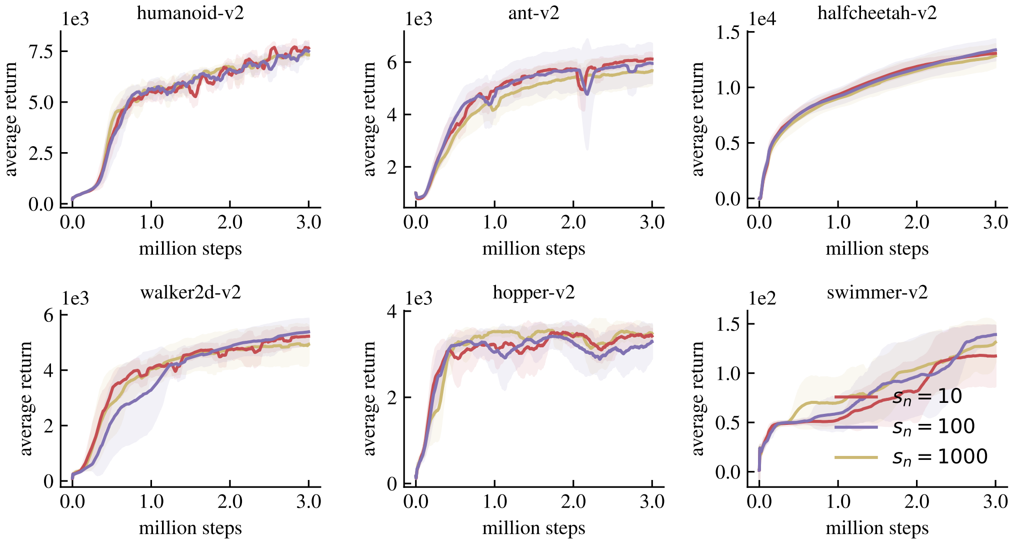


Figure 11: The results of different sample size on the six environments.

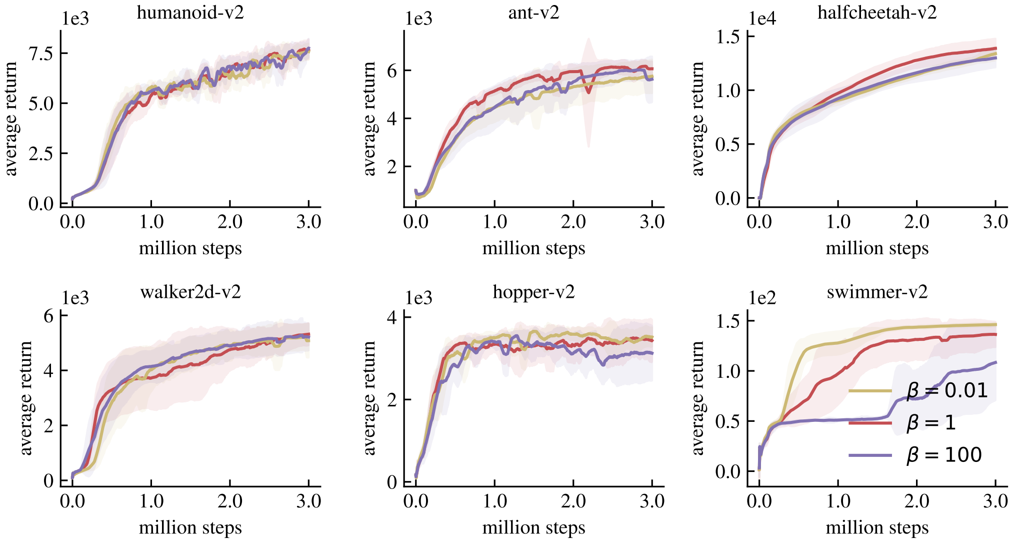
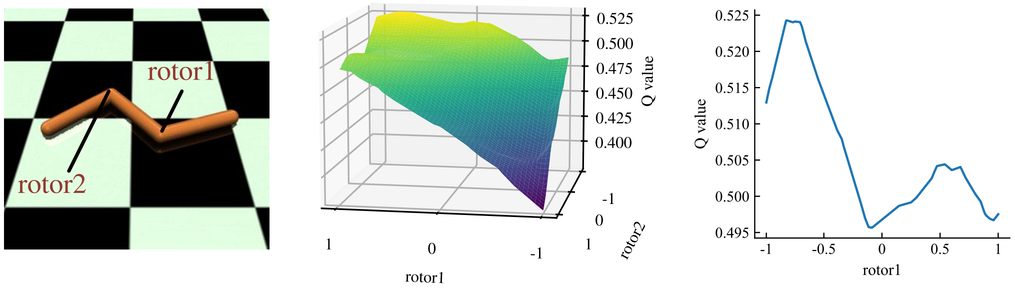


Figure 12: The results of different beta on the six environments.

F LIMITATIONS AND BROADER IMPACTS

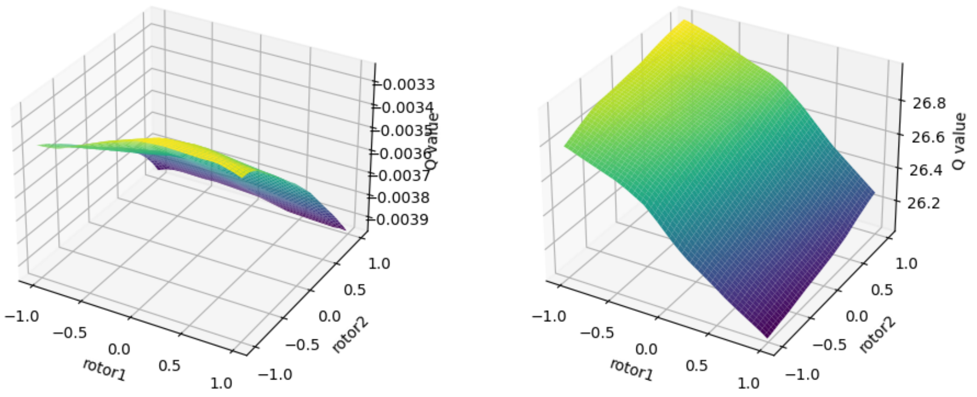
Limitations. In low-dimensional action spaces, our method shows little improvement. It is particularly noticeable that in the swimmer-v2 environment, state-of-the-art results can reach an episode reward of 350. Furthermore, exploration costs should also increase as the action space's dimensionality increases. The conclusions we drew earlier may have limitations. However, it is challenging to develop environments with higher-dimensional action spaces, and we still need to fully validate our conclusions in such environments.

Broader impacts. We do not anticipate any negative consequences from using our method in practice.

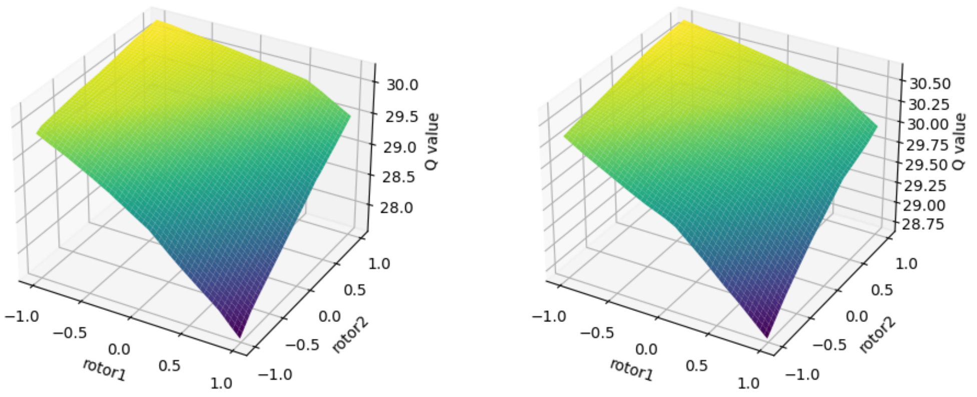


(a) Swimmer (b) 3D visualization (c) 2D plane

Figure 13: Visualization of the Q function. (a): The swimmer has two rotors, and its moving is controlled by adjusting the torque applied to the two rotors. (b): The Q values of the two-dimension actions are plotted in 3D Space. (c): We plot a particular case for rotor2=-1 to show that the Q function has an energy-based form in early-stage training.

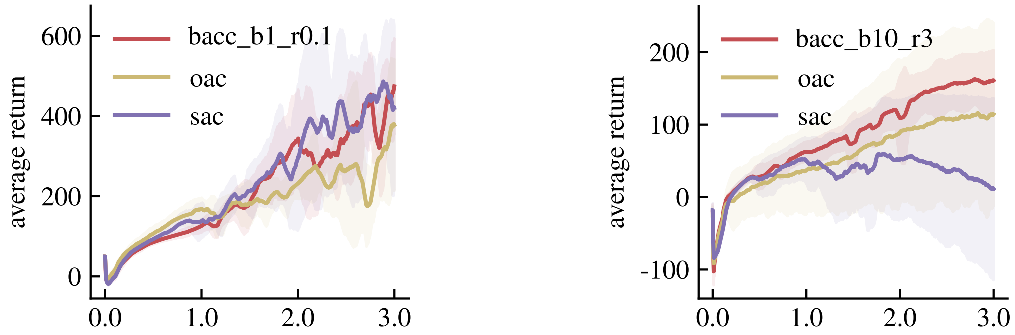


(a) Epoch 0 (b) Epoch 500



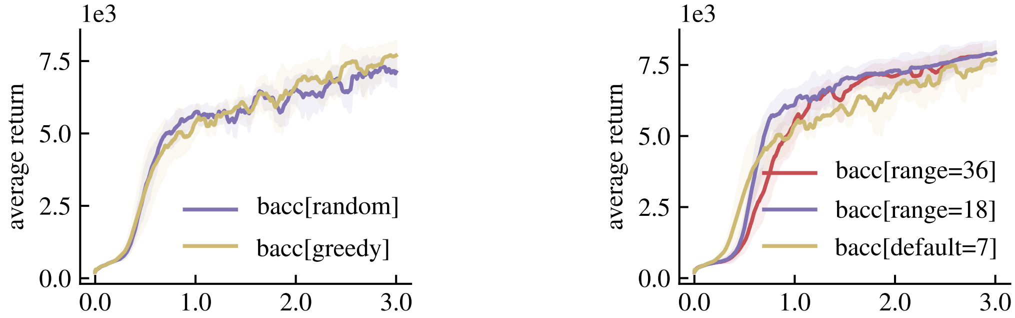
(c) Epoch 1500 (d) Epoch 2000

Figure 14: 3D surfaces of different epoch Q function

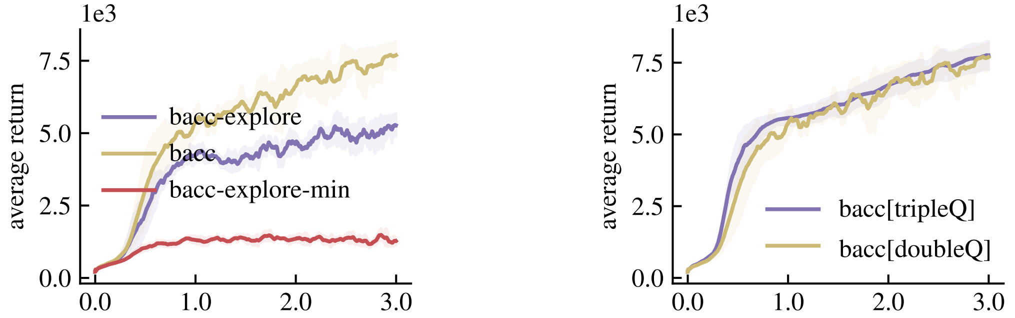


(a) HumanoidFlagrun-v1 (b) HumanoidFlagrunHarder-v1

Figure 15: Additional evaluation in the Roboschool simulation. We use different hyper-parameters. “bacc_b1_r0.1” means that $\beta = 1$ and $s_r = 0.1$.



(a) random vs greedy (b) large range effect



(c) stable learning and greedy exploration (d) more Q network

Figure 16: More results in the Humanoid-v2 environment. (a) “random” means randomly taking $Q_1(s,a)$ or $Q2(s,a)$ when constructing π_E. (b) “range$\bar{3}$6” means s_r=36. (c) “bacc-explore” means expore without greedy Q, just with policy. Furthermore, “bacc-explore-min” means no use of the min value as the target value. (d) “tripleQ” means the value function ensemble framework comprises triple Q networks.