A. A Simple Example of Efficiency of Sample-Aware Entropy Maximization

Here, we provide a toy example showing the effectiveness of maximizing the sample-aware entropy defined as the entropy of a mixture distribution $q_{mix}^{\pi,\alpha} = \alpha\pi + (1-\alpha)q$, where q is the sample action distribution of the replay buffer. For this simple toy example, we consider a discrete MDP case in order to show the intuition of sample-aware entropy maximization.

Let us consider a simple 1-step MDP in which s_0 is the unique initial state, there exist N_a actions ($\mathcal{A} = \{A_1, \cdots, A_{N_a}\}$), s_1 is the terminal state, and r is a deterministic reward function. Then, there exist N_a state-action pairs in total. Let us assume that we already have $N_a - 1$ state-action samples in the replay buffer as $\mathbf{R} = \{(s_0, A_1, r(s_0, A_1)), \cdots, (s_0, A_{N_a-1}, r(s_0, A_{N_a-1}))\}$. In order to estimate the Q-function for all state-action pairs, the policy should sample the last action A_{N_a} (Then, we can reuse all samples infinitely to estimate Q). Here, we will compare two exploration methods.

1) First, if we consider the simple entropy maximization, the policy that maximizes its entropy will choose all actions with equal probability $1/N_a$ (uniformly). Then, N_a samples should be taken on average by the policy to visit the action A_{N_a}.

2) Second, consider the sample-aware entropy maximization. Here, the sample action distribution q in the buffer becomes $q(a_0|s_0) = 1/(N_a - 1)$ for $a_0 \in \{A_1, \cdots, A_{N_a-1}\}$ and $q(A_{N_a}|s_0) = 0$, the mixture distribution becomes $q_{mix}^{\pi,\alpha} = \alpha\pi + (1-\alpha)q$, and we set $\alpha = 1/N_a$. Then, the policy that maximizes the sample-aware entropy is given by $\pi(A_{N_a}|s_0) = 1$ because this policy makes $q_{mix}^{\pi,\alpha}$ uniform and the sample-aware entropy is maximized. In this case, we only need one sample to visit the action A_{N_a}. In this way, the proposed sample-aware entropy maximization can enhance sample-efficiency for exploration by using the previous sample distribution and choosing a proper α. With this motivation, we propose the sample-aware entropy regularization for off-policy RL and an α-adaptation method.

B. Proofs

B.1. Proof of Theorem 1

To prove Theorem 1, we first provide two lemmas. For a fixed policy π, Q^π can be estimated by repeating the Bellman backup operator, as stated in Lemma 1 below. Lemma 1 is based on usual policy evaluation but has a new ingredient of the ratio function in the proposed sample-aware entropy case.

Lemma 1 (Diverse Policy Evaluation) *Define a sequence of diverse Q-functions as $Q_{k+1} = \mathcal{T}^\pi Q_k$, $k \geq 0$, where π is a fixed policy and Q_0 is a real-valued initial Q. Assume that the action space is bounded, and $R^{\pi,\alpha}(s_t, a_t) \in (0,1)$ for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. Then, the sequence $\{Q_k\}$ converges to the true diverse state-action value Q^π.*

Proof. Let $r_{\pi,t} := \frac{1}{\beta} r_t + \gamma \mathbb{E}_{s_{t+1}\sim P}[\mathbb{E}_{a_{t+1}\sim\pi}[\alpha \log R^{\pi,\alpha}(s_{t+1}, a_{t+1}) - \alpha \log \alpha\pi(a_{t+1}|s_{t+1})] + (1-\alpha)\mathbb{E}_{a_{t+1}\sim q}[\log R^{\pi,\alpha}(s_{t+1}, a_{t+1}) - \log \alpha\pi(a_{t+1}|s_{t+1})]]$. Then, we can rewrite the modified Bellman equation (11) into the standard Bellman equation form for the true Q^π as follows:

$$\mathcal{T}^\pi Q(s_t, a_t) = r_{\pi,t} + \gamma \mathbb{E}_{s+1\sim P,\ a_{t+1}\sim\pi}[Q(s_{t+1}, a_{t+1})] \tag{B.1}$$

Under the assumption of a bounded action space and $R^{\pi,\alpha} \in (0,1)$, the reward $r_{\pi,t}$ is bounded and the convergence is guaranteed as the usual policy evaluation (Sutton & Barto, 1998; Haarnoja et al., 2018a). □

Lemma 2 is about diverse policy improvement.

Lemma 2 (Diverse Policy Improvement) *Let π_{new} be the updated policy obtained by solving $\pi_{new} = \arg\max_\pi J_{\pi_{old}}(\pi)$, where $J_{\pi_{old}}(\pi)$ is given in (13). Then, $Q^{\pi_{new}}(s_t, a_t) \geq Q^{\pi_{old}}(s_t, a_t)$, $\forall\ (s_t, a_t) \in \mathcal{S} \times \mathcal{A}$.*

Proof. Since $\pi_{new} = \arg\max_\pi J_{\pi_{old}}(\pi)$, we have $J_{\pi_{old}}(\pi_{new}) \geq J_{\pi_{old}}(\pi_{old})$. Expressing $J_{\pi_{old}}(\pi_{new})$ and $J_{\pi_{old}}(\pi_{old})$ by using the definition of $J_{\pi_{old}}(\pi)$ in (13), we have

$$\begin{aligned}
J_{\pi_{old}}(\pi_{new}(\cdot|s_t)) &= \mathbb{E}_{a_t\sim\pi_{new}}[Q^{\pi_{old}}(s_t, a_t) + \alpha \log R^{\pi_{new},\alpha}(s_t, a_t) - \alpha \log \alpha\pi_{new}(a_t|s_t)] \\
&\qquad + (1-\alpha)\mathbb{E}_{a_t\sim q}[\log R^{\pi_{new},\alpha}(s_t, a_t) - \log \alpha\pi_{new}(a_t|s_t)] \\
&\geq J_{\pi_{old}}(\pi_{old}(\cdot|s_t)) \\
&= \mathbb{E}_{a_t\sim\pi_{old}}[Q^{\pi_{old}}(s_t, a_t) + \alpha \log R^{\pi_{old},\alpha}(s_t, a_t) - \alpha \log \alpha\pi_{old}(a_t|s_t)] \\
&\qquad + (1-\alpha)\mathbb{E}_{a_t\sim q}[\log R^{\pi_{old},\alpha}(s_t, a_t) - \log \alpha\pi_{old}(a_t|s_t)] \\
&= V^{\pi_{old}}(s_t)
\end{aligned} \tag{B.2}$$

by the definition of $V^\pi(s_t)$ in (12). Then, based on (B.2), we obtain the following inequality:

$$\begin{aligned}
&Q^{\pi_{old}}(s_t, a_t) = \frac{1}{\beta} r_t + \gamma \mathbb{E}_{s_{t+1}\sim P}[V^{\pi_{old}}(s_{t+1})] \\
&\quad \overset{(a)}{\leq} \frac{1}{\beta} r_t + \gamma \mathbb{E}_{s_{t+1}\sim P}\{\mathbb{E}_{a_{t+1}\sim\pi_{new}}[\underbrace{Q^{\pi_{old}}(s_{t+1}, a_{t+1})}_{=\frac{1}{\beta}r_{t+1}+\gamma\mathbb{E}_{s_{t+2}\sim P}[V^{\pi_{old}}(s_{t+2})]} + \alpha \log R^{\pi_{new},\alpha}(s_{t+1}, a_{t+1}) - \alpha \log \alpha\pi_{new}(a_{t+1}|s_{t+1})] \\
&\qquad\qquad + (1-\alpha)\mathbb{E}_{a_{t+1}\sim q}[\log R^{\pi_{new},\alpha}(s_{t+1}, a_{t+1}) - \log \alpha\pi_{new}(a_{t+1}|s_{t+1})]\} \\
&\quad \vdots \\
&\quad \leq Q^{\pi_{new}}(s_t, a_t), \quad \text{for each } (s_t, a_t) \in \mathcal{S} \times \mathcal{A},
\end{aligned} \tag{B.3}$$

where Inequality (a) is obtained by applying Inequality (B.2) on $V^{\pi_{old}}(s_{t+1})$, and $Q^{\pi_{old}}(s_{t+1}, a_{t+1})$ in the RHS of Inequality (a) is expressed as $\frac{1}{\beta} r_{t+1} + \gamma \mathbb{E}_{s_{t+2}\sim P}[V^{\pi_{old}}(s_{t+2})]$ and Inequality (B.2) is then applied on $V^{\pi_{old}}(s_{t+2})$; this procedure is repeated to obtain Inequality (B.3). By (B.3), we have the claim. This concludes proof. □

Now, we prove Theorem 1 based on the previous lemmas.

Theorem 1 (Diverse Policy Iteration) *By repeating iteration of the diverse policy evaluation and the diverse policy improvement, any initial policy converges to the optimal policy π^* s.t. $Q^{\pi^*}(s_t,a_t) \geq Q^{\pi'}(s_t,a_t)$, $\forall\ \pi' \in \Pi$, $\forall\ (s_t,a_t) \in \mathcal{S}\times\mathcal{A}$. Also, such π^* achieves maximum J, i.e., $J_{\pi^*}(\pi^*) \geq J_\pi(\pi)$ for any $\pi \in \Pi$.*

Proof. Let Π be the space of policy distributions and let $\{\pi_i, i = 0, 1, 2, \cdots \mid \pi_i \in \Pi\}$ be a sequence of policies generated by the following recursion:

$$\pi_{i+1} = \arg\max_{\pi\in\Pi} J_{\pi_i}(\pi) \qquad \text{with an arbitrary initial policy } \pi_0, \tag{B.4}$$

where the objective function $J_{\pi_i}(\pi)$ is defined in (13).

Proof of convergence of the sequence $\{\pi_i, i = 0, 1, 2, \cdots\}$ to a local optimum is for arbitrary state space $\mathcal{S}$. On the other hand, for proof of convergence of $\{\pi_i, i = 0, 1, 2, \cdots\}$ to the global optimum, we assume finite MDP, as typically assumed for convergence proof in usual policy iteration (Sutton & Barto, 1998).

For any state-action pair $(s,a) \in \mathcal{S}\times\mathcal{A}$, each $Q^{\pi_i}(s,a)$ is bounded due to the discount factor γ (see (8)), and the sequence $\{Q^{\pi_i}(s,a), i = 0, 1, 2, \cdots\}$ is monotonically increasing by Lemma 2. Now, consider two terms $J_{\pi_{i+1}}(\pi_{i+1}(\cdot|s))$ and $J_{\pi_i}(\pi_{i+1}(\cdot|s))$, which are expressed by the definition of $J_{\pi_{old}}(\pi)$ in (13) as follows:

$$\begin{aligned} J_{\pi_{i+1}}(\pi_{i+1}(\cdot|s)) = \beta\{&\mathbb{E}_{a\sim\pi_{i+1}}\left[Q^{\pi_{i+1}}(s,a) + \alpha(\log R^{\pi_{i+1},\alpha}(s,a) - \log\alpha\pi_{i+1}(a|s))\right] \\ &+ (1-\alpha)\mathbb{E}_{a\sim q}\left[\log R^{\pi_{i+1},\alpha}(s,a) - \log\alpha\pi_{i+1}(a|s)\right]\} \end{aligned} \tag{B.5}$$

$$\begin{aligned} J_{\pi_i}(\pi_{i+1}(\cdot|s)) = \beta\{&\mathbb{E}_{a\sim\pi_{i+1}}\left[Q^{\pi_i}(s,a) + \alpha(\log R^{\pi_{i+1},\alpha}(s,a) - \log\alpha\pi_{i+1}(a|s))\right] \\ &+ (1-\alpha)\mathbb{E}_{a\sim q}\left[\log R^{\pi_{i+1},\alpha}(s,a) - \log\alpha\pi_{i+1}(a|s)\right]\}. \end{aligned} \tag{B.6}$$

Note in (B.5) and (B.6) that all the terms are the same for $J_{\pi_{i+1}}(\pi_{i+1}(\cdot|s))$ and $J_{\pi_i}(\pi_{i+1}(\cdot|s))$ except $\beta\mathbb{E}_{a\sim\pi_{i+1}}[Q^{\pi_{i+1}}(s,a)]$ in $J_{\pi_{i+1}}(\pi_{i+1}(\cdot|s))$ and $\beta\mathbb{E}_{a\sim\pi_{i+1}}[Q^{\pi_i}(s,a)]$ in $J_{\pi_i}(\pi_{i+1}(\cdot|s))$. Because $\{Q^{\pi_i}(s,a), i = 0, 1, 2, \cdots\}$ is monotonically increasing by Lemma 2, comparing (B.5) and (B.6) yields

$$J_{\pi_{i+1}}(\pi_{i+1}(\cdot|s)) \geq J_{\pi_i}(\pi_{i+1}(\cdot|s)). \tag{B.7}$$

Furthermore, we have for any $s \in \mathcal{S}$,

$$J_{\pi_i}(\pi_{i+1}(\cdot|s)) \geq J_{\pi_i}(\pi_i(\cdot|s)) \tag{B.8}$$

by the definition of π_{i+1} in (B.4). Combining (B.7) and (B.8), we have

$$J_{\pi_{i+1}}(\pi_{i+1}(\cdot|s)) \geq J_{\pi_i}(\pi_{i+1}(\cdot|s)) \geq J_{\pi_i}(\pi_i(\cdot|s)) \tag{B.9}$$

for any state $s \in \mathcal{S}$. Therefore, the sequence $\{J_{\pi_i}(\pi_i(\cdot|s)), i = 0, 1, 2, \cdots\}$ is monotonically increasing for any $s \in \mathcal{S}$. Furthermore, note from (B.5) that $J_{\pi_i}(\pi_i(\cdot|s))$ is bounded for all i, because the Q-function and the entropy of the mixture distribution are bounded. (Note that the RHS of (B.5) except the term $\mathbb{E}_{a\sim\pi_{i+1}}[Q^{\pi_{i+1}}(s,a)]$ is nothing but the entropy of the mixture distribution $\mathcal{H}(q_{mix}^{\pi_{i+1},\alpha})$. Please see (10) for this.) Note that $J_{\pi_i}(\pi_i)$, which is obtained by setting $\pi_{old} = \pi_i$ and $\pi = \pi_i$ in (13), is nothing but $J(\pi_i)$ with the desired original J defined in (3). Hence, by (B.9) and the boundedness of the sequence $\{J_{\pi_i}(\pi_i)\}$, convergence to a local optimum of J by the sequence $\{\pi_i, i = 0, 1, 2, \cdots\}$ is guaranteed by the monotone convergence theorem.

Now, consider convergence to the global optimum. By the monotone convergence theorem, $\{Q^{\pi_i}(s,a), i = 0, 1, 2, \cdots\}$ and $\{J_{\pi_i}(\pi_i(\cdot|s)), i = 0, 1, 2, \cdots\}$ pointwisely converge to their limit functions $Q^* : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ and $J^* : \mathcal{S} \to \mathbb{R}$, respectively. Here, note that $J^*(s) \geq J_{\pi_i}(\pi_i(\cdot|s))$ for any i because the sequence $\{J_{\pi_i}(\pi_i(\cdot|s)), i = 0, 1, 2, \cdots\}$ is monotonically increasing by (B.9). By the definition of pointwise convergence, for any $s \in \mathcal{S}$, for any $\epsilon > 0$, there exists a sufficiently large $N(s) (> 0)$ depending on s such that $J_{\pi_i}(\pi_i(\cdot|s)) \geq J^*(s) - \frac{\epsilon(1-\gamma)}{\gamma}$ for all $i \geq N(s)$. When $\mathcal{S}$ is finite, we set $\bar{N} = \max_s N(s)$. Then, we have

$$J_{\pi_i}(\pi_i(\cdot|s)) \geq J^*(s) - \frac{\epsilon(1-\gamma)}{\gamma}, \quad \forall s \in \mathcal{S},\ \forall i \geq \bar{N} \tag{B.10}$$

Furthermore, we have

$$J_{\pi_i}(\pi_i(\cdot|s)) \geq J_{\pi_i}(\pi'(\cdot|s)) - \frac{\epsilon(1-\gamma)}{\gamma}, \quad \forall s \in \mathcal{S},\ \forall i \geq \bar{N},\ \forall \pi' \in \Pi. \tag{B.11}$$

(B.11) is valid by the following reason. Suppose that (B.11) is not true. Then, there exist some $s' \in \mathcal{S}$ and some $\pi' \in \Pi$ such that

$$J_{\pi_i}(\pi'(\cdot|s')) \overset{(b)}{>} J_{\pi_i}(\pi_i(\cdot|s')) + \frac{\epsilon(1-\gamma)}{\gamma} \overset{(c)}{\geq} J^*(s'), \tag{B.12}$$

where Inequality (b) is obtained by negating (B.11) and Inequality (c) is obtained by (B.10). Moreover, we have

$$J_{\pi_{i+1}}(\pi_{i+1}(\cdot|s')) \overset{(d)}{\geq} J_{\pi_i}(\pi_{i+1}(\cdot|s')) = \max_{\pi} J_{\pi_i}(\pi(\cdot|s')) \overset{(e)}{\geq} J_{\pi_i}(\pi'(\cdot|s')), \tag{B.13}$$

where Inequality (d) is valid due to (B.7) and Inequality (e) is valid by the definition of π_{i+1} given in (B.4). Combining (B.12) and (B.13) yields

$$J_{\pi_{i+1}}(\pi_{i+1}(\cdot|s')) \geq J_{\pi_i}(\pi_{i+1}(\cdot|s')) \geq J_{\pi_i}(\pi'(\cdot|s')) > J_{\pi_i}(\pi_i(\cdot|s')) + \frac{\epsilon(1-\gamma)}{\gamma} \geq J^*(s'). \tag{B.14}$$

However, this contradicts to the fact that $J^*(s')$ is the limit of the monotone-increasing sequence $J_{\pi_i}(\pi_i(\cdot|s'))$. Therefore, (B.11) is valid.

Based on (B.11), we have the following inequality regarding $Q^{\pi_i}(s_t, a_t)$: For any (s_t, a_t), for all $i \geq \bar{N}$,

$$\begin{aligned}
Q^{\pi_i}(s_t, a_t) &= \frac{1}{\beta} r_t + \gamma \mathbb{E}_{s_{t+1}\sim P}[V^{\pi_i}(s_{t+1})] \\
&= \frac{1}{\beta} r_t + \gamma \mathbb{E}_{s_{t+1}\sim P}[J_{\pi_i}(\pi_i(\cdot|s_{t+1}))] \\
&\overset{(f)}{\geq} \frac{1}{\beta} r_t + \gamma \mathbb{E}_{s_{t+1}\sim P}\left[J_{\pi_i}(\pi'(\cdot|s_{t+1})) - \frac{\epsilon(1-\gamma)}{\gamma}\right], \quad \forall \pi' \in \Pi, \\
&\overset{(g)}{=} \frac{1}{\beta} r_t + \gamma \mathbb{E}_{s_{t+1}\sim P}\{\mathbb{E}_{a_{t+1}\sim\pi}[Q^{\pi_i}(s_{t+1}, a_{t+1}) + \alpha \log R^{\pi',\alpha}(s_{t+1}, a_{t+1}) - \alpha \log \alpha\pi'(a_{t+1}|s_{t+1})] \\
&\qquad\qquad + (1-\alpha)\mathbb{E}_{a_{t+1}\sim q}[\log R^{\pi',\alpha}(s_{t+1}, a_{t+1}) - \log \alpha\pi'(a_{t+1}|s_{t+1})]\} - \epsilon(1-\gamma) \\
&\vdots \\
&\overset{(h)}{\geq} Q^{\pi'}(s_t, a_t) - \epsilon, \quad \forall \pi' \in \Pi,
\end{aligned} \tag{B.15}$$

where Inequality (f) is valid due to (B.11); Equality (g) is obtained by explicitly expressing $J_{\pi_i}(\pi')$ using (13); we express $Q^{\pi_i}(s_{t+1}, a_{t+1})$ as $Q^{\pi_i}(s_{t+1}, a_{t+1}) = \frac{1}{\beta} r_{t+1} + \gamma \mathbb{E}_{s_{t+2}\sim P}[V^{\pi_i}(s_{t+2})]$ and repeat the same procedure on $V^{\pi_i}(s_{t+2}) = J_{\pi_i}(\pi_i(\cdot|s_{t+2}))$; and we obtain the last Inequality (h) by repeating this iteration. Here, the resulting constant term is $-\epsilon(1-\gamma) - \epsilon\gamma(1-\gamma) - \epsilon\gamma^2(1-\gamma) - \cdots = -\epsilon$, as shown in the RHS of Inequality (g). Note that the uniformity condition "$\forall s \in \mathcal{S}$" in the Inequality (B.11) is required because we need to express $J_{\pi_i}(\pi_i(\cdot|s_{t+1})),\ J_{\pi_i}(\pi_i(\cdot|s_{t+2})),\ J_{\pi_i}(\pi_i(\cdot|s_{t+3})),\ \cdots$ in terms of $J_{\pi_i}(\pi'(\cdot|s_{t+1})),\ J_{\pi_i}(\pi'(\cdot|s_{t+2})),\ J_{\pi_i}(\pi'(\cdot|s_{t+3})),\ \cdots$, respectively, by using (B.11) in the above recursive procedure and the support of each element of the sequence $s_{t+1},\ s_{t+2},\ s_{t+3}, \cdots$ is $\mathcal{S}$ in general. Since $\epsilon > 0$ is arbitrary in the above, by taking $i \to \infty$ on both sides of (B.15), we have

$$Q^{\pi_\infty}(s, a) \geq Q^{\pi'}(s, a), \quad \forall \pi' \in \Pi, \quad \forall\, (s, a) \in \mathcal{S} \times \mathcal{A} \tag{B.16}$$

since the sequence $\{Q^{\pi_i}(s, a), i = 0, 1, 2, \cdots\}$ is monotonically increasing.

Now, let us compare $J_{\pi'}(\pi'(\cdot|s))$ and $J_{\pi_\infty}(\pi'(\cdot|s))$. These two terms can be expressed in similar forms to (B.5) and (B.6), respectively. Then, only $Q^{\pi_\infty}(s, a)$ and $Q^{\pi'}(s, a)$ are different in the expressed forms. Comparing $J_{\pi'}(\pi'(\cdot|s))$ and $J_{\pi_\infty}(\pi'(\cdot|s))$ as we did for (B.7), we have

$$J_{\pi_\infty}(\pi'(\cdot|s)) \geq J_{\pi'}(\pi'(\cdot|s)) \tag{B.17}$$

due to Inequality (B.16). In addition, we have $J_{\pi_i}(\pi_i(\cdot|s)) \geq J_{\pi_i}(\pi'(\cdot|s)) - \frac{\epsilon(1-\gamma)}{\gamma}$ due to (B.11). Since $\epsilon > 0$ is arbitrary, by taking $i \to \infty$, we have

$$J_{\pi_\infty}(\pi_\infty(\cdot|s)) \geq J_{\pi_\infty}(\pi'(\cdot|s)). \tag{B.18}$$

Finally, combining (B.17) and (B.18) yields

$$J_{\pi_\infty}(\pi_\infty(\cdot|s)) \geq J_{\pi_\infty}(\pi'(\cdot|s)) \geq J_{\pi'}(\pi'(\cdot|s)), \ \ \forall\, \pi' \in \Pi, \quad \forall\, s \in \mathcal{S}. \tag{B.19}$$

Recall that $J_\pi(\pi)$, which is obtained by setting $\pi_{old} = \pi$ and $\pi = \pi$ in (13), is nothing but $J(\pi)$ of the desired original J defined in (3). Therefore, π_∞ is the optimal policy π^* maximizing J, and $\{\pi_i\}$ converges to the optimal policy π^*. This concludes the proof. □

Remark: Note that what we actually need for proof of convergence to the global optimum is the uniform convergence of $J_{\pi_i}(\pi_i(\cdot|s)) \to J^*(s)$ as functions of s to obtain (B.11). The finite state assumption is one sufficient condition for this. In order to guarantee convergence to global optimum in non-finite MDP (e.g. continuous state-space), we need more assumption as considered in (Puterman & Brumelle, 1979; Santos & Rust, 2004). Here, we do not further detail. In this paper, we just consider function approximation for the policy and the value functions to implement the diverse policy iteration in continuous state and action spaces, based on the convergence proof in finite MDP.

B.2. Proof of Theorem 2

Remark: We defined $J_{\pi_{old}}(\pi)$ as (13), which is restated below:

$$\begin{aligned} J_{\pi_{old}}(\pi(\cdot|s_t)) := \beta\{&\mathbb{E}_{a_t\sim\pi}\left[Q^{\pi_{old}}(s_t,a_t) + \alpha(\log R^{\pi,\alpha}(s_t,a_t) - \log \alpha\pi(a_t|s_t))\right] \\ &+ (1-\alpha)\mathbb{E}_{a_t\sim q}\left[\log R^{\pi,\alpha}(s_t,a_t) - \log \alpha\pi(a_t|s_t)\right]\}, \end{aligned} \tag{B.20}$$

where π in the $R^{\pi,\alpha}$ terms inside the expectations is the optimization argument. As mentioned in the main part of the paper, this facilitates proof of Lemma 2 and proof of Theorem 1, especially in Steps (B.2), (B.3), (B.5), (B.6), and (B.7). However, as explained in the main part of the paper, implementing the function $R^{\pi,\alpha}(s_t,a_t)$ with optimization argument π is difficult. Hence, we replaced $J_{\pi_{old}}(\pi)$ with $\tilde{J}_{\pi_{old}}(\pi)$ in (14) by considering the ratio function $R^{\pi_{old},\alpha}(s_t,a_t)$ for only the current policy π_{old}. Now, we prove the gradient equivalence of $J_{\pi_{old}}(\pi)$ and $\tilde{J}_{\pi_{old}}(\pi)$ at $\theta = \theta_{old}$ for parameterized policy π_θ.

Lemma 3 *For the ratio function $R^{\pi,\alpha}(s_t,a_t)$ defined in (9), we have the following:*

$$\log R^{\pi,\alpha}(s_t,a_t) - \log \alpha\pi(a_t|s_t) = \log(1 - R^{\pi,\alpha}(s_t,a_t)) - \log((1-\alpha)q(a_t|s_t)) \tag{B.21}$$

Proof. From the definition of the ratio function:

$$R^{\pi,\alpha}(s_t,a_t) = \frac{\alpha\pi(a_t|s_t)}{\alpha\pi(a_t|s_t) + (1-\alpha)q(a_t|s_t)}, \tag{B.22}$$

we have

$$1 - R^{\pi,\alpha}(s_t,a_t) = \frac{(1-\alpha)q(a_t|s_t)}{\alpha\pi(a_t|s_t) + (1-\alpha)q(a_t|s_t)}. \tag{B.23}$$

Hence, we have

$$\begin{aligned} \log \frac{1}{\alpha\pi(a_t|s_t) + (1-\alpha)q(a_t|s_t)} &= \log R^{\pi,\alpha}(s_t,a_t) - \log(\alpha\pi(a_t|s_t)) && \text{(B.24)} \\ &= \log(1 - R^{\pi,\alpha}(s_t,a_t)) - \log((1-\alpha)q(a_t|s_t)). && \text{(B.25)} \end{aligned}$$

This concludes proof. □

Theorem 2 *Consider the new objective function for policy improvement $\tilde{J}_{\pi_{old}}(\pi(\cdot|s_t))$ in (14), where the ratio function inside the expectation in (14) is the ratio function for the given current policy π_{old}. Suppose that the policy is parameterized with parameter θ. Then, for parameterized policy π_θ, the two objective functions $J_{\pi_{\theta_{old}}}(\pi_\theta(\cdot|s_t))$ and $\tilde{J}_{\pi_{\theta_{old}}}(\pi_\theta(\cdot|s_t))$ have the same gradient direction for θ at $\theta = \theta_{old}$ for all $s_t \in \mathcal{S}$, where θ_{old} is the parameter of the given current policy π_{old}.*

Proof. With the parameterized π_θ, the two objective functions are expressed as

$$\begin{aligned}
J_{\pi_{\theta_{old}}}(\pi_\theta(\cdot|s_t)) &= \beta(\mathbb{E}_{a_t\sim\pi_\theta}[Q^{\pi_{\theta_{old}}}(s_t,a_t) + \alpha\log R^{\pi_\theta,\alpha}(s_t,a_t) - \alpha\log\alpha\pi_\theta(a_t|s_t)] \\
&\quad + (1-\alpha)\mathbb{E}_{a_t\sim q}[\log R^{\pi_\theta,\alpha}(s_t,a_t) - \log\alpha\pi_\theta(a_t|s_t)]) \\
&\overset{(1)}{=} \beta(\mathbb{E}_{a_t\sim\pi_\theta}[Q^{\pi_{\theta_{old}}}(s_t,a_t) + \alpha\log R^{\pi_\theta,\alpha}(s_t,a_t) - \alpha\log\alpha\pi_\theta(a_t|s_t)] \\
&\quad + (1-\alpha)\mathbb{E}_{a_t\sim q}[\log(1-R^{\pi_\theta,\alpha}(s_t,a_t)) - \log(1-\alpha)q(a_t|s_t)]) &\text{(B.26)}\\
\tilde{J}_{\pi_{\theta_{old}}}(\pi_\theta(\cdot|s_t)) &= \beta\mathbb{E}_{a_t\sim\pi_\theta}[Q^{\pi_{\theta_{old}}}(s_t,a_t) + \alpha\log R^{\pi_{\theta_{old}},\alpha}(s_t,a_t) - \alpha\log\pi_\theta(a_t|s_t)], &\text{(B.27)}
\end{aligned}$$

where Step (1) is valid by Lemma 3. Comparing (B.26) and (B.27), we can ignore the common $Q^{\pi_{\theta_{old}}}$ and $\log\pi_\theta$ terms, and the constant terms w.r.t. θ that yield zero gradient in (B.26) and (B.27). Therefore, we only need to show

$$\nabla_\theta\{\alpha\mathbb{E}_{a_t\sim\pi_\theta}[\log R^{\pi_\theta,\alpha}] + (1-\alpha)\mathbb{E}_{a_t\sim q}[\log(1-R^{\pi_\theta,\alpha})]\} = \nabla_\theta\mathbb{E}_{a_t\sim\pi_\theta}[\alpha\log R^{\pi_{\theta_{old}},\alpha}] \tag{B.28}$$

at $\theta=\theta_{old}$. The gradient of the left-hand side (LHS) in (B.28) at $\theta=\theta_{old}$ is expressed as

$$\begin{aligned}
&\nabla_\theta\{\alpha\mathbb{E}_{a_t\sim\pi_\theta}[\log R^{\pi_\theta,\alpha}] + (1-\alpha)\mathbb{E}_{a_t\sim q}[\log(1-R^{\pi_\theta,\alpha})]\} \\
&= \nabla_\theta\left\{\alpha\int_{a_t}\pi_\theta\log R^{\pi_\theta,\alpha}da_t + (1-\alpha)\int_{a_t}q\log(1-R^{\pi_\theta,\alpha})da_t\right\} \\
&= \alpha\int_{a_t}(\nabla_\theta\pi_\theta)\log R^{\pi_\theta,\alpha}da_t + \alpha\int_{a_t}\pi_\theta(\nabla_\theta\log R^{\pi_\theta,\alpha})da_t + (1-\alpha)\int_{a_t}q\nabla_\theta\log(1-R^{\pi_\theta,\alpha})da_t \\
&= \alpha\int_{a_t}(\nabla_\theta\pi_\theta)|_{\theta=\theta_{old}}\log R^{\pi_\theta,\alpha}|_{\theta=\theta_{old}}da_t + \alpha\int_{a_t}\pi_\theta(\nabla_\theta\log R^{\pi_\theta,\alpha})da_t + (1-\alpha)\int_{a_t}q\nabla_\theta\log(1-R^{\pi_\theta,\alpha})da_t \\
&= \alpha\nabla_\theta\int_{a_t}\pi_\theta\log R^{\pi_{\theta_{old}},\alpha}da_t + \alpha\int_{a_t}\pi_\theta(\nabla_\theta\log R^{\pi_\theta,\alpha})da_t + (1-\alpha)\int_{a_t}q\nabla_\theta\log(1-R^{\pi_\theta,\alpha})da_t \\
&= \nabla_\theta\mathbb{E}_{a_t\sim\pi_\theta}[\alpha\log R^{\pi_{\theta_{old}},\alpha}] + \alpha\mathbb{E}_{a_t\sim\pi_\theta}[\nabla_\theta\log R^{\pi_\theta,\alpha}] + (1-\alpha)\mathbb{E}_{a_t\sim q}[\nabla_\theta\log(1-R^{\pi_\theta,\alpha})]. &\text{(B.29)}
\end{aligned}$$

Here, the gradient of the last two terms in the RHS of (B.29) becomes zero, as shown below:

$$\begin{aligned}
&\alpha\mathbb{E}_{a_t\sim\pi_\theta}[\nabla_\theta\log R^{\pi_\theta,\alpha}] + (1-\alpha)\mathbb{E}_{a_t\sim q}[\nabla_\theta\log(1-R^{\pi_\theta,\alpha})] \\
&= \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{R^{\pi_\theta,\alpha}}\right] + (1-\alpha)\mathbb{E}_{a_t\sim q}\left[\frac{\nabla_\theta(1-R^{\pi_\theta,\alpha})}{(1-R^{\pi_\theta,\alpha})}\right] \\
&= \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{R^{\pi_\theta,\alpha}}\right] - (1-\alpha)\mathbb{E}_{a_t\sim q}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{(1-R^{\pi_\theta,\alpha})}\right] \\
&= \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{R^{\pi_\theta,\alpha}}\right] - (1-\alpha)\mathbb{E}_{a_t\sim q}\left[\frac{\alpha\pi_\theta+(1-\alpha)q}{(1-\alpha)q}\cdot\nabla_\theta R^{\pi_\theta,\alpha}\right] \\
&= \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{R^{\pi_\theta,\alpha}}\right] - \mathbb{E}_{a_t\sim q}\left[\frac{\alpha\pi_\theta+(1-\alpha)q}{q}\cdot\nabla_\theta R^{\pi_\theta,\alpha}\right] \\
&\overset{(2)}{=} \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{R^{\pi_\theta,\alpha}}\right] - \mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\pi_\theta+(1-\alpha)q}{\pi_\theta}\cdot\nabla_\theta R^{\pi_\theta,\alpha}\right] \\
&= \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{R^{\pi_\theta,\alpha}}\right] - \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\pi_\theta+(1-\alpha)q}{\alpha\pi_\theta}\cdot\nabla_\theta R^{\pi_\theta,\alpha}\right] \\
&= \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{R^{\pi_\theta,\alpha}}\right] - \alpha\mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{\nabla_\theta R^{\pi_\theta,\alpha}}{R^{\pi_\theta,\alpha}}\right] = 0, &\text{(B.30)}
\end{aligned}$$

where we used an importance sampling technique (i.e., measure change) $\mathbb{E}_{a_t\sim q}[f(s_t,a_t)] = \mathbb{E}_{a_t\sim\pi_\theta}\left[\frac{q(a_t|s_t)}{\pi_\theta(a_t|s_t)}f(s_t,a_t)\right]$ for Step (2). By (B.29) and (B.30), $J_{\pi_{\theta_{old}}}(\pi_\theta(\cdot|s_t))$ and $J_{\pi_{\theta_{old}}}(\pi_\theta(\cdot|s_t))$ have the same gradient at $\theta=\theta_{old}$. This concludes proof. □

C. Detailed DAC Implementation

We defined the target value $\hat{V}(s_t) = \mathbb{E}_{a_t\sim\pi_\theta}[Q_\phi(s_t,a_t) + \alpha\log R_\eta^\alpha(s_t,a_t) - \alpha\log\alpha\pi_\theta(a_t|s_t)] + (1-\alpha)\mathbb{E}_{a_t\sim\mathcal{D}}[\log R_\eta^\alpha(s_t,a_t) - \log\alpha\pi_\theta(a_t|s_t)]$ in (22). However, the probability of π for actions sampled from $\mathcal{D}$ can have high variance, so we clip the term inside the expectation over $a_t\sim\mathcal{D}$ by action dimension for stable learning. Thus, the final target value is given by

$$\begin{aligned}\hat{V}(s_t) = {} & \mathbb{E}_{a_t\sim\pi_\theta}[Q_\phi(s_t,a_t) + \alpha\log R_\eta^\alpha(s_t,a_t) - \alpha\log\alpha\pi_\theta(a_t|s_t)] \\ & + (1-\alpha)\mathbb{E}_{a_t\sim\mathcal{D}}[\mathrm{clip}(\log R_\eta^\alpha(s_t,a_t) - \log\alpha\pi(a_t|s_t); -d, d)],\end{aligned} \tag{C.1}$$

where $d = \dim(\mathcal{A})$ is the action dimension and $\mathrm{clip}(x;-d,d)$ is the clipping function to fit into the range $[-d,d]$. We use (C.1) for actual implementation.

In addition, we require $R^{\pi_\theta,\alpha}\in(\epsilon,1-\epsilon)$ in the proofs of Theorems 1 and 2 so that $\log R^{\pi_\theta,\alpha}$ and $\log(1-R^{\pi_\theta,\alpha})$ appearing in the proofs do not diverge. For practical implementation, we clipped the ratio function R^α as $(\epsilon,1-\epsilon)$ for small $\epsilon>0$ since some q values can be close to zero before the replay buffer stores a sufficient amount of samples. However, π is always non-zero since we consider Gaussian policy.

To compute the gradient of $\hat{J}_\pi(\theta)$ in (17), we use the reparameterization trick proposed by (Kingma & Welling, 2013; Haarnoja et al., 2018a). Note that the policy action $a_t\sim\pi_\theta$ is the output of the policy neural network with parameter θ. So, it can be viewed as $a_t = f_\theta(\epsilon_t; s_t)$, where f is a function parameterized by θ and ϵ_t is a noise vector sampled from spherical normal distribution $\mathcal{N}$. Then, the gradient of $\hat{J}_\pi(\theta)$ is represented as $\nabla_\theta\hat{J}_\pi(\theta) = \mathbb{E}_{s_t\sim\mathcal{D},\ \epsilon_t\sim\mathcal{N}}[\nabla_a(Q_\phi(s_t,a) + \alpha\log R_\eta^\alpha(s_t,a) - \alpha\log\pi_\theta(a|s_t))|_{a=f_\theta(\epsilon_t;s_t)}\nabla_\theta f_\theta(\epsilon_t;s_t) - \alpha(\nabla_\theta\log\pi_\theta)(f_\theta(\epsilon_t;s_t)|s_t)]$.

C.1. Detailed Implementation of the α-Adaptation

In order to learn α, we parameterize α as a function of s_t using parameter ξ, i.e., $\alpha = \alpha_\xi(s_t)$, and implement $\alpha_\xi(s_t)$ with a neural network. Then, ξ is updated to minimize the following loss function of α obtained from (23):

$$\hat{L}_\alpha(\xi) = \mathbb{E}_{s_t\sim\mathcal{D}}[\mathcal{H}(q_{mix}^{\pi_\theta,\alpha_\xi}) - \alpha_\xi c] \tag{C.2}$$

In the α adaptation case, all the updates for diverse policy iteration are the same except that α is replaced with $\alpha_\xi(s_t)$. The gradient of $\hat{L}_\alpha(\xi)$ with respect to ξ can be estimated as below:

$$\begin{aligned}&\nabla_\xi\hat{L}_\alpha(\xi) = \nabla_\xi\mathbb{E}_{s_t\sim\mathcal{D}}[\mathcal{H}(q_{mix}^{\pi_\theta,\alpha_\xi}) - \alpha_\xi c]\\
=&\nabla_\xi\mathbb{E}_{s_t\sim\mathcal{D}}[\alpha_\xi\mathbb{E}_{a_t\sim\pi_\theta}[-\log(\alpha_\xi\pi_\theta + (1-\alpha_\xi)q) - c] + (1-\alpha_\xi)\mathbb{E}_{a_t\sim q}[-\log(\alpha_\xi\pi_\theta + (1-\alpha_\xi)q)]]\\
=&\mathbb{E}_{s_t\sim\mathcal{D}}[(\nabla_\xi\alpha_\xi)(\mathbb{E}_{a_t\sim\pi_\theta}[-\log(\alpha_\xi\pi_\theta + (1-\alpha_\xi)q) - c] - \mathbb{E}_{a_t\sim q}[-\log(\alpha_\xi\pi_\theta + (1-\alpha_\xi)q)])]\\
&+\mathbb{E}_{s_t\sim\mathcal{D}}[\alpha_\xi\mathbb{E}_{a_t\sim\pi_\theta}[-\nabla_\xi\log(\alpha_\xi\pi_\theta + (1-\alpha_\xi)q)] + (1-\alpha_\xi)\mathbb{E}_{a_t\sim q}[-\nabla_\xi\log(\alpha_\xi\pi_\theta + (1-\alpha_\xi)q)]]\\
=&\mathbb{E}_{s_t\sim\mathcal{D}}[(\nabla_\xi\alpha_\xi)(\mathbb{E}_{a_t\sim\pi_\theta}[-\log\alpha_\xi\pi_\theta + \log R^{\pi_\theta,\alpha_\xi} - c] - \mathbb{E}_{a_t\sim q}[\log R^{\pi_\theta,\alpha_\xi} - \log\alpha_\xi\pi_\theta])]\\
&+\mathbb{E}_{s_t\sim\mathcal{D}}\Bigg[\underbrace{\int_{a_t\in\mathcal{A}}(\alpha_\xi\pi_\theta + (1-\alpha_\xi)q)[-\nabla_\xi\log(\alpha_\xi\pi_\theta + (1-\alpha_\xi)q)]}_{=0}\Bigg]\\
=&\mathbb{E}_{s_t\sim\mathcal{D}}[(\nabla_\xi\alpha_\xi)(\mathbb{E}_{a_t\sim\pi_\theta}[-\log\alpha_\xi\pi_\theta + \log R^{\pi_\theta,\alpha_\xi} - c] - \mathbb{E}_{a_t\sim q}[\log R^{\pi_\theta,\alpha_\xi} - \log\alpha_\xi\pi_\theta])]\end{aligned} \tag{C.3}$$

Note that $R^{\pi_\theta,\alpha_\xi}$ can be estimated by the ratio function $R_\eta^{\alpha_\xi}$. Here, we use the same clipping technique as used in (C.1) for the last term of (C.3). For α-adaptation, we used regularization for α learning and restricted the range of α as $0.5\le\alpha\le0.99$ for α adaptation in order to maintain a certain level of entropy regularization and prevent saturation of R_η^α.

D. Simulation Setup

We here provide the detailed simulation setup of DAC, SAC baselines, RND, and MaxEnt(State). For fair comparison, we use the common hyperparameter setup for DAC and SAC baselines except for the parts regarding entropy or divergence.

The hyperparameter setup basically follows the setup in (Haarnoja et al., 2018a), which is given by Table D.1. Here, the entropy coefficient β is selected based on the ablation study in Section F. For the policy space Π, we considered a Gaussian policy set widely considered in usual continuous RL. Also, we provide Table D.2, which shows the environment description, the corresponding entropy control coefficient β, threshold for sparse Mujoco tasks, and reward delay D for delayed Mujoco tasks.

	SAC / SAC-Div	DAC
Learning rate δ	$3 \cdot 10^{-4}$	
Discount factor γ	0.99 (0.999 for pure exploration)	
Horizon N	1000	
Mini-batch size M	256	
Replay buffer length	10^6	
Smoothing coefficient of EMA for $V_{\bar{\psi}}$	0.005	
Optimizer	Adam	
Num. of hidden layers (all networks)	2	
Size of hidden layers (all networks)	256	
Policy distribution	Independent Gaussian distribution	
Activation layer	ReLu	
Output layer for π_θ, Q_ϕ, V_ψ , $V_{\bar{\psi}}$	Linear	
Output layer for α_ξ, R_η^α	·	Sigmoid
Regularize coefficient for α_ξ	·	10^{-3}
Control coefficient c for α-adaptation	·	$-2.0 \cdot \dim(\mathcal{A})$

Table D.1: Hyperparamter setup

	State dim.	Action dim.	β	Threshold
SparseHalfCheetah-v1	17	6	0.02	5.0
SparseHopper-v1	11	3	0.04	1.0
SparseWalker2d-v1	17	6	0.02	1.0
SparseAnt-v1	111	8	0.01	1.0
	State dim.	Action dim.	β	Delay D
HumanoidStandup-v1	376	17	1	·
DelayedHalfCheetah-v1	17	6	0.2	20
DelayedHopper-v1	11	3	0.2	20
DelayedWalker2d-v1	17	6	0.2	20
DelayedAnt-v1	111	8	0.2	20

Table D.2: State and action dimensions of Mujoco tasks and the corresponding β

In addition, we also compared the performance of DAC to two recent state-based exploration methods, RND (Burda et al., 2018) and MaxEnt(State) (Hazan et al., 2019), in Section 6. State-based exploration methods aim to find rare states to enhance exploration performance.

In order to explore rare states, RND adds an intrinsic reward based on prediction error $r_t^{int} = ||\hat{f}(s_{t+1}) - f(s_{t+1})||^2$ to the extrinsic reward r_t^{ext} so that the total reward becomes $r_t = r_t^{ext} + c^{int} r_t^{int}$, where $\hat{f}$ is a prediction network and f is a randomly fixed target network. Then, the agent goes to rare states since rare states have higher prediction errors. For our simulation, we considered MLP with 2 ReLu hidden layers of size 256 with 20-dimensional output for both networks of RND, and we used $c^{int} = 5$ that performed well for considered tasks.

On the other hand, MaxEnt(State) aims to maximize the entropy of state mixture distribution $\mathcal{H}(d^{\pi^{mix}})$ to explore rare states, where d^{π} is the state distribution of a trajectory generated from π. In order to do that, MaxEnt(State) uses the reward $r_{MaxEnt(State)}(s) = -(\log d^{\pi_{mix}}(s) + c_s)$, where c_s is a smoothing constant. MaxEnt(State) mainly considers large or continuous state space, so $d^{\pi_{mix}}$ is computed by projection/Kernel density estimation. Then, MaxEnt(State) explores the state space better than a simple random policy on various tasks in continuous state spaces. For our simulation, we use previous 100K states stored in the buffer to estimate $d^{\pi_{mix}}$. Note that MaxEnt(State) is originally designed for pure exploration, but we use its reward functional as an intrinsic reward in order to learn sparse-rewarded tasks. In this case, we found that $c^{int} = 0.02$ worked well for the considered tasks. For both RND and MaxEnt(State), we basically consider the same simulation setup with DAC and SAC baselines but use Gaussian policy with fixed standard deviation $\sigma = 0.3$ for both RND and MaxEnt(State) to make fair comparison between action-based exploration and state-based exploration.

E. More Results on Performance Comparison

We provide more numerical results in this section. In Appendix E.1, we provide the remaining learning curves and max average return tables for the performance comparisons in the main paper. In Appendix E.2, we provide the performance comparison between DAC and RND/MaxEnt(State) on SparseMujocot tasks. In Appendix E.3, we compare the DAC with α adaptation to other general RL algorithms on HumanoidStandup and DelayedMujoco tasks.

E.1. Performance Comparison with the SAC Baselines

In this subsection, we provide more performance plots and tables for the performance comparison between DAC and SAC baselines. Fig. E.1 shows the divergence D_{JS}^{α} curve ($\alpha = 0.5$) and Fig. E.2 shows the mean number of discretized state visitation curve for remaining SparseMujoco tasks. Table. E.1 shows the corresponding max average return performance on sparse Mujoco tasks. Fig. E.3 shows the scaled version of the performance plots in Fig. E.2, and Table E.2 shows the corresponding max average return performance.

Here, in order to show the tendency of state visitation in Fig. E.2, we discretized the state of each SparseMujoco task. For discretization, we simply consider 2 components of observations for each task: x, y axis position for SparseAnt, and x, z axis position for the other SparseMujoco tasks. We discretize the position by setting the grid spacing per axis to 0.01 in the range of $(-10, 10)$. For SAC/SAC-Div, the ratio function R is estimated separately by the same way with DAC.

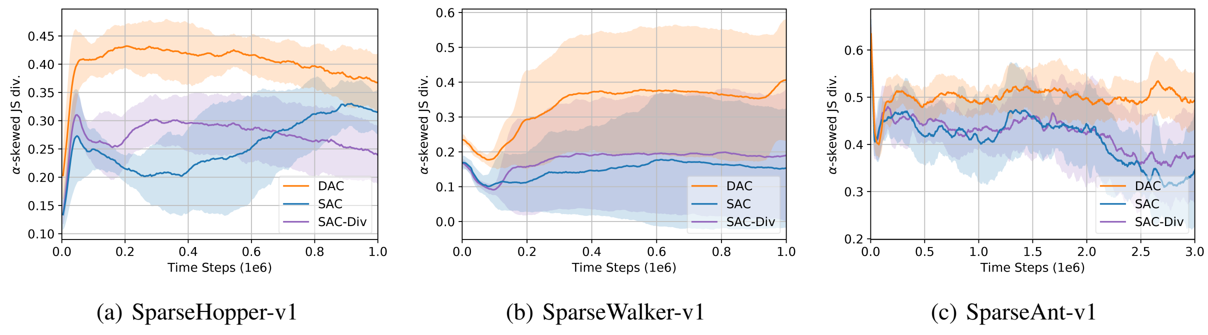


Figure E.1: α-skewed JS symmetrization of KLD D_{JS}^{α} for DAC and SAC/SAC-Div

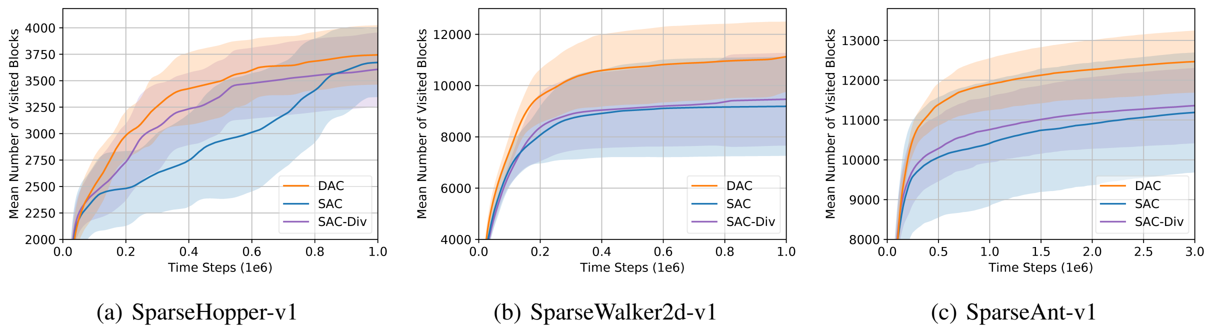
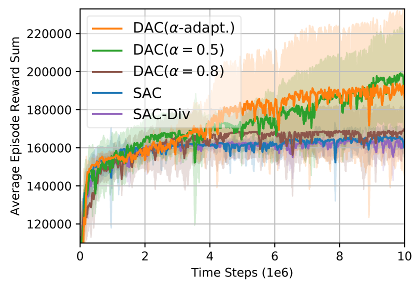
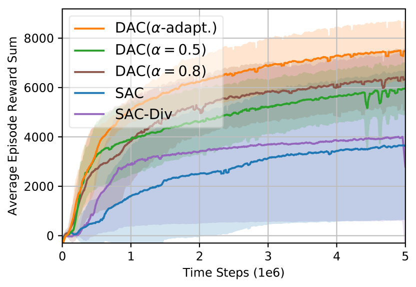


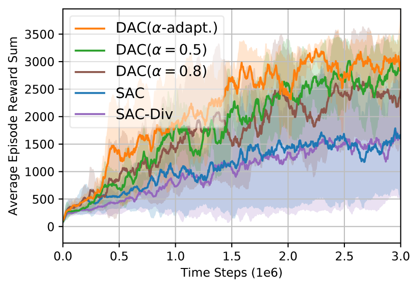
Figure E.2: The number of discretized state visitation on sparse Mujoco tasks



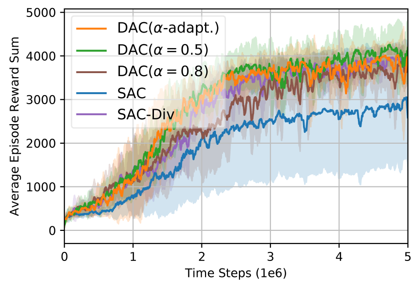
(a) HumanoidStandup-v1



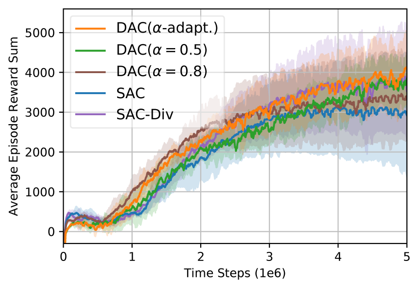
(b) Del.HalfCheetah-v1



(c) Del.Hopper-v1



(d) Del.Walker2d-v1



(e) Del.Ant-v1

Figure E.3: Performance comparison on HumanoidStandup and DelayedMujoco tasks

	DAC ($\alpha = 0.5$)	SAC	SAC-Div
SparseHalfCheetah	**915.90±50.71**	386.90±404.70	394.70±405.53
SparseHopper	**900.30±3.93**	823.70±215.35	817.40±253.54
SparseWalker2d	**665.10±355.66**	273.30±417.51	278.50±398.23
SparseAnt	935.80±37.08	**963.80±42.51**	870.70±121.14

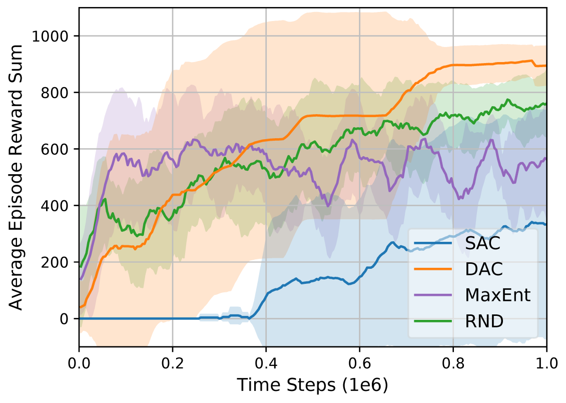
Table E.1: Max average return of DAC algorithm and SAC baselines on SparseMujoco tasks

	DAC ($\alpha = 0.5$)	DAC ($\alpha = 0.8$)	DAC (α-adapt.)	SAC	SAC-Div
HumanoidS	**202491.81 ±25222.77**	170832.05 ±12344.71	197302.37 ±43055.31	167394.36 ±7291.99	165548.76 ±2005.85
Del. HalfCheetah	6071.93±1045.64	6552.06±1140.18	**7594.70±1259.23**	3742.33±3064.55	4080.67±3418.07
Del. Hopper	3283.77±112.04	2836.81±679.05	**3428.18±69.08**	2175.31±1358.39	2090.64±1383.83
Del. Walker2d	**4360.43±507.58**	3973.37±273.63	4067.11±257.81	3220.92±1107.91	4048.11±290.48
Del. Ant	4088.12±578.99	3535.72±1164.76	**4243.19±795.49**	3248.43±1454.48	3978.34±1370.23

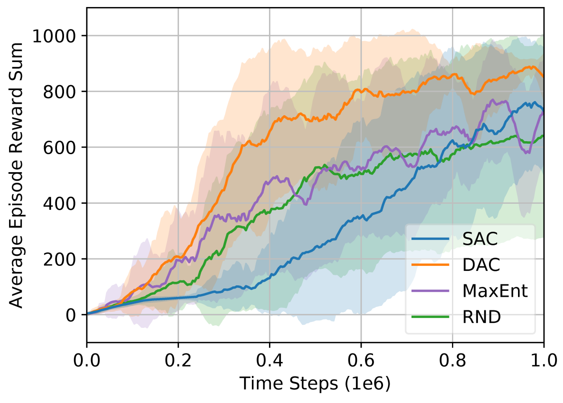
Table E.2: Max average return of DAC algorithms and SAC baselines on HumanoidStandup and DelayedMujoco tasks

E.2. Comparison to State-based Exploration Methods on Sparse Mujoco Tasks

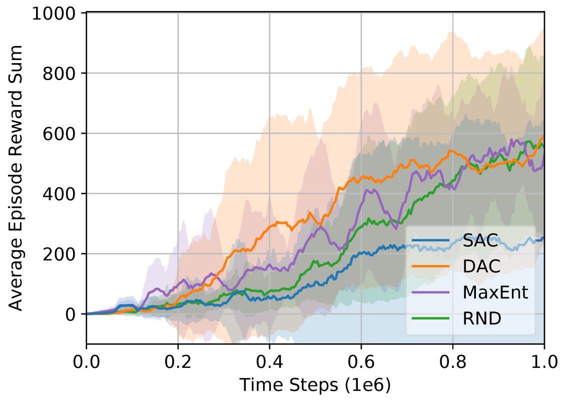
We compared the performance of DAC ($\alpha = 0.5$) with RND/MaxEnt(State) on SparseMujoco tasks, and the performance of DAC (α-adapt.) with RND/MaxEnt(State) on DelayedMujoco tasks. Fig. E.5 shows the performance learning curve, and the corresponding max average return table in Table E.3. From the results, it is seen that DAC has better performance than RND/MaxEnt(State) on most Sparse/DelayedMujoco tasks. DAC has superiority not only in pure exploration but also in learning sparse rewarded tasks as compared to recent state-based exploration methods.



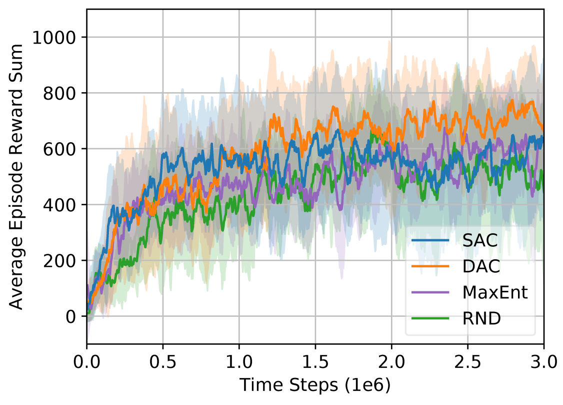
(a) SparseHalfCheetah-v1



(b) SparseHopper-v1



(c) SparseWalker2d-v1

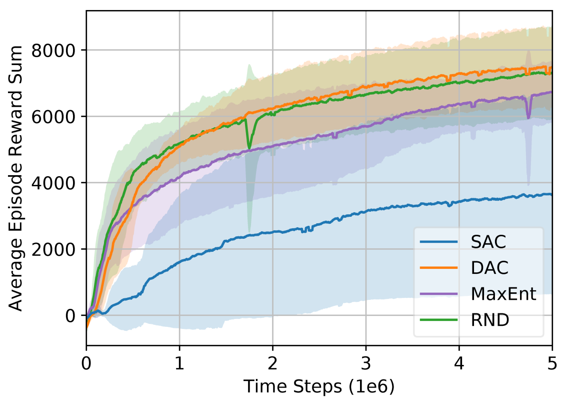


(d) SparseAnt-v1

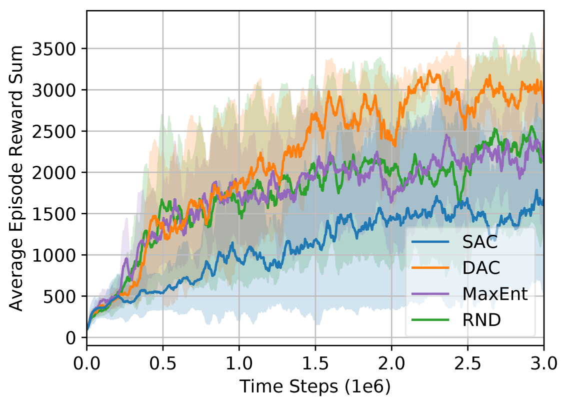
Figure E.4: Performance comparison to RND/MaxEnt(State) on SparseMujoco tasks

	DAC ($\alpha = 0.5$)	RND	MaxEnt(State)	SAC
SparseHalfCheetah	**915.90±50.71**	827.80±85.61	800.20±127.11	386.90±404.70
SparseHopper	**900.30±3.93**	648.10±363.75	879.50±30.96	823.70±215.35
SparseWalker2d	665.10±355.66	663.00±356.39	**705.30±274.88**	273.30±417.51
SparseAnt	935.80±37.08	920.60±107.50	900.00±70.02	**963.80±42.51**
	DAC (α-adapt.)	RND	MaxEnt(State)	SAC
Del.HalfCheetah	**7594.70±1259.23**	7429.94±1383.75	6823.37±882.25	3742.33±3064.55
Del.Hopper	**3428.18±69.08**	2764.06±1220.86	3254.10±30.75	2175.31±1358.39
Del.Walker2d	4067.11±257.81	3514.97±1536.04	**4430.61±347.02**	3220.92±1107.91
Del.Ant	**4243.19±795.49**	1361.36±704.69	1246.80±323.50	3248.43±1454.48

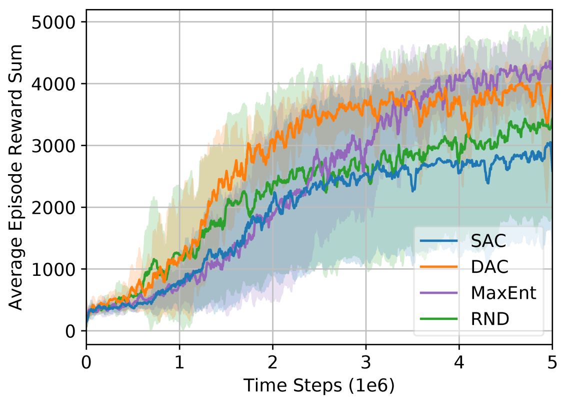
Table E.3: Max average return of DAC, RND, and MaxEnt(State)



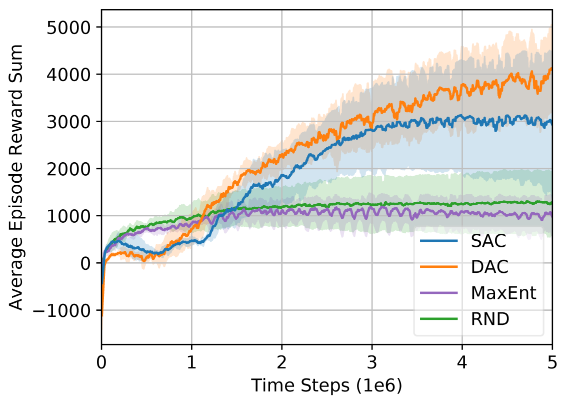
(a) Del.HalfCheetah-v1



(b) Del.Hopper-v1



(c) Del.Walker2d-v1

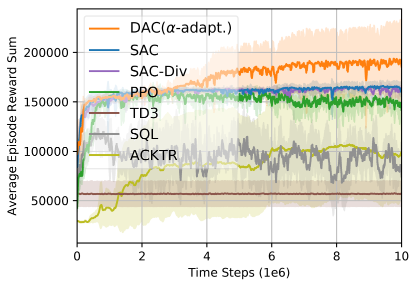


(d) Del.Ant-v1

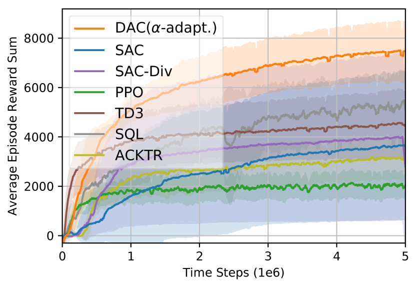
Figure E.5: Performance comparison to RND/MaxEnt(State) on DelayedMujoco tasks

E.3. Comparison to Recent General RL Algorithms

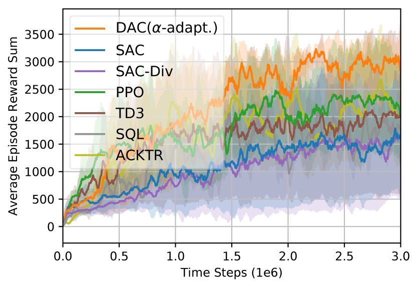
We also compare the performance of DAC with α-adaptation to other state-of-the-art RL algorithms. Here, we consider various on-policy RL algorithms: Proximal Policy Optimization (Schulman et al., 2017b) (PPO, a stable and popular on-policy algorithm), Actor Critic using Kronecker-factored Trust Region (Wu et al., 2017) (ACKTR, actor-critic that approximates natural gradient by using Kronecker-factored curvature), and off-policy RL algorithms: Twin Delayed Deep Deterministic Policy Gradient (Fujimoto et al., 2018) (TD3, using clipped double-Q learning for reducing overestimation); and Soft Q-Learning (Haarnoja et al., 2017) (SQL, energy based policy optimization using Stein variational gradient descent). We used implementations in OpenAI baselines (Dhariwal et al., 2017) for PPO and ACKTR, and implementations in author's Github for other algorithms. We provide the performance results as Fig. E.6 and Table E.4, and the results show that DAC has the best performance on all considered tasks among the compared recent RL algorithms.



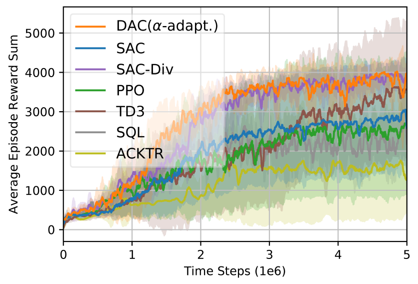
(a) HumanoidStandup-v1



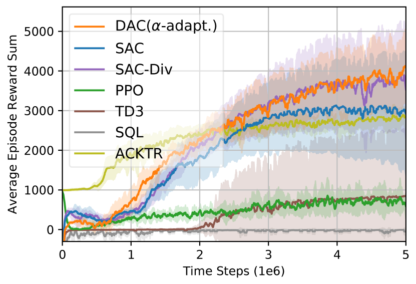
(b) DelayedHalfCheetah-v1



(c) DelayedHopper-v1



(d) DelayedWalker2d-v1



(e) DelayedAnt-v1

Figure E.6: Performance comparison to recent general RL algorithms

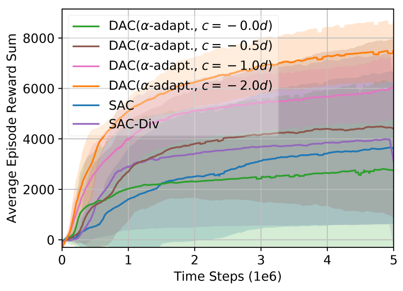
	DAC	PPO	ACKTR	SQL	TD3	SAC
HumanoidS	**197302.37 ±43055.31**	160211.90 ±3268.37	109655.30 ±49166.15	138996.84 ±33903.03	58693.87 ±12269.93	167394.36 ±7291.99
Del. HalfCheetah	**7594.70 ±1259.23**	2247.92 ±640.69	3295.30 ±824.05	5673.34 ±1241.30	4639.85 ±1393.95	3742.33 ±3064.55
Del. Hopper	**3428.18 ±69.08**	2740.15 ±719.63	2864.81 ±1072.64	2720.32 ±127.71	2276.58 ±1471.66	2175.31 ±1358.39
Del. Walker2d	**4067.11 ±257.81**	2859.27 ±1938.50	1927.32 ±1647.49	3323.63 ±503.18	3736.72 ±1806.37	3220.92 ±1107.91
Del. Ant	**4243.19 ±795.49**	1224.33 ±521.62	2956.51 ±234.89	6.59 ±16.42	904.99 ±1811.78	3248.43 ±1454.48

Table E.4: Max average return of DAC and other RL algorithms

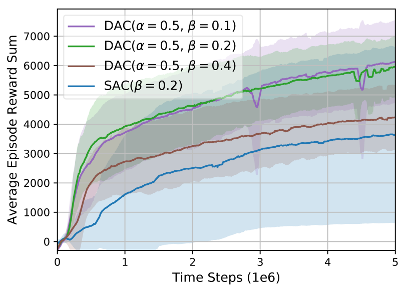
F. More Ablation Studies

In this section, we provide detailed ablation studies on the DelayedMucoco tasks. First, we focus on the DelayedHalfCheetah task because the tendencies of performance changes are similar for most environments and the performance changes on the DelayedHalfCheetah task are most noticeable. Then, we provide more ablation studies for remaining DelayedMujoco tasks in Appendix F.1.

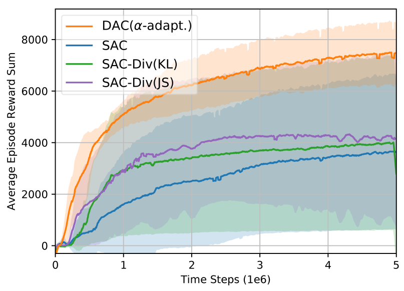
Control coefficient c in (23)**:** In the proposed α-adaptation (23) in Section 5, the control coefficient c affects the learning behavior of α. Since $\mathcal{H}(\pi)$ and D_{JS}^{α} are proportional to the action dimension, we tried a few values such as 0, $-0.5d$, $-1.0d$ and $-2.0d$, where $d = \dim(\mathcal{A})$. Fig. F.1(a) shows the corresponding performance of DAC with α-adaptation on DelayedHalfCheetah. As seen in Fig. F.1(a), the performance depends on the change of c as expected, and $c = -2.0 \cdot \dim(\mathcal{A})$



(a) Control coefficient c



(b) Entropy coefficient β



(c) JS divergence

Figure F.1: Averaged learning curve for ablation study

performs well. We observed that $-2.0d$ performed well for all considered tasks. Hence, we set $c = -2.0d$ in (C.2).

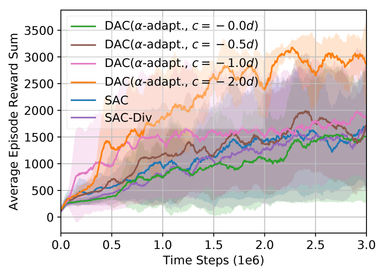
Entropy coefficient β in (3): As mentioned in (Haarnoja et al., 2018a), the performance of SAC depends on β. It is expected that the performance of DAC depends on β too. Fig. F.1(b) shows the performance of DAC with fixed $\alpha = 0.5$ for three different values of β: $\beta = 0.1$, 0.2 and 0.4 on DelayedHalfCheetah. It is seen that the performance of DAC indeed depends on β. Although there exists performance difference for DAC depending on β, the performance of DAC is much better than SAC for a wide range of β. One thing to note is that the coefficient of pure policy entropy regularization term for DAC is $\alpha\beta$, as seen in (3). Thus, DAC with $\alpha = 0.5$ and $\beta = 0.4$ has the same amount of pure policy entropy regularization as SAC with $\beta = 0.2$. However, DAC with $\alpha = 0.5$ and $\beta = 0.4$ has much higher performance than SAC with $\beta = 0.2$, as seen in Fig. Fig. F.1(b). So, we can see that the performance improvement of DAC comes from joint use of policy entropy $\mathcal{H}(\pi)$ and the sample action distribution from the replay buffer via $D^{\alpha}_{JS}(\pi||q)$.

The effect of JS divergence: In order to see the effect of the JS divergence on the performance, we provide an additional ablation study in which we consider a single JS divergence for SAC-Div by using the ratio function in Section 4.3. Fig. F.1(c) shows the performance comparison of SAC, SAC-Div(KL), SAC-Div(JS), and DAC. For SAC-Div(JS), we used $\delta_d = 0.5$ for adaptive scaling in (Hong et al., 2018). It is seen that there is no significant difference in performance between SAC-Div with JS divergence and SAC-Div with KL divergence. DAC still shows superiority to both SAC-Div(KL) and SAC-Div(JS). This shows that DAC has more advantages than simply using JS divergence.

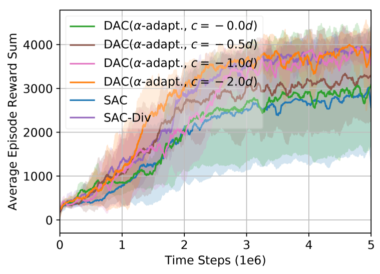
F.1. Ablation Studies for Remaining Tasks

Here, we provide more ablation studies for remaining delayed Mujoco tasks in Figure F.2, Figure F.3, and Figure F.4.

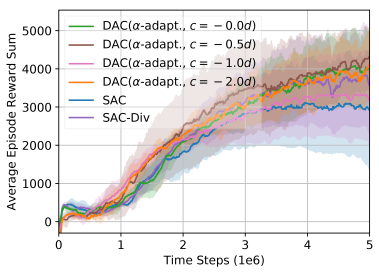
Control coefficient c



(a) DelayedHopper-v1



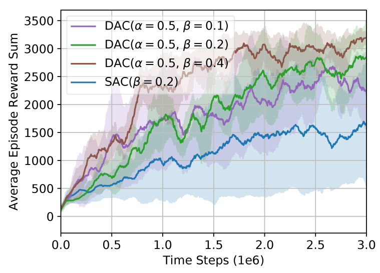
(b) DelayedWalker2d-v1



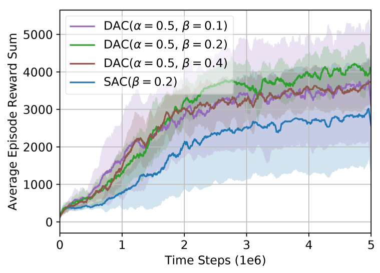
(c) DelayedAnt-v1

Figure F.2: Ablation study on c

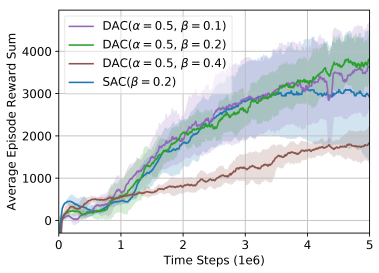
Entropy coefficient β



(a) DelayedHopper-v1



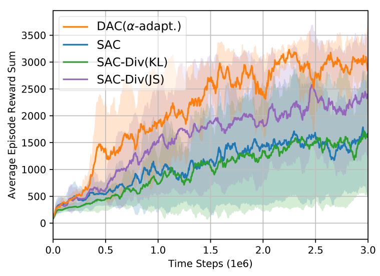
(b) DelayedWalker2d-v1



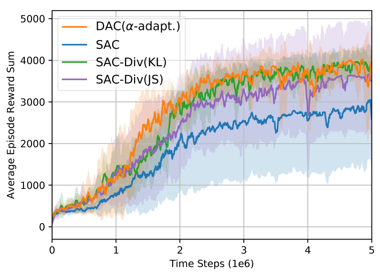
(c) DelayedAnt-v1

Figure F.3: Ablation study on β

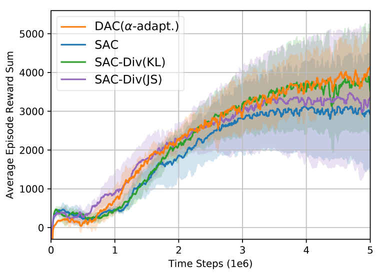
Effect of JS divergence over SAC-Div



(a) DelayedHopper-v1



(b) DelayedWalker2d-v1



(c) DelayedAnt-v1

Figure F.4: Ablation study on SAC-Div with JS divergence