A Additional Details of Algorithms

A.1 Psuedocode of Q^2-learning

We apply Q^2-learning to continuous state and action spaces as shown in Alg. 1. The overall procedure is similar to TD3 (Fujimoto, Hoof, and Meger 2018). The interval update of the actor is borrowed from it. Noticeably, the way we compute D_φ and D_ψ for every pair of states follows (Castro 2020). According to Alg. 1, the size of the matrix formed by our batch of samples is $N \times 1$, where we suppose the state is one-dimensional feature. We "square" the matrix to get a new one whose size is $N^2 \times 1$. Each state in this new matrix appears twice. We use this squared version matrix to compute D_φ and D_ψ for every pair, which means that all computations are performed on matrices without looping. Thanks to efficient computing power of GPU to matrices, this process does not consume a lot of time.

Algorithm 1: Q^2-learning for continuous action space

Initialize D^*_{SA} critic φ, target network $\hat{\varphi}$, D^*_{SA} actor ψ
for $t = 1$ **to** T **do**
    Sample N transitions (x, a, r^a_x, x') from $\mathcal{B}$
    Compute $D_\varphi(x, y)$ and $D_\psi(x', y')$ for every pair of states
    Update the D^*_{SA} critic by $\varphi \leftarrow \varphi - \lambda_\varphi \nabla_\varphi \mathcal{L}_\varphi$
    if $t \bmod actor\_update\_interval == 0$ **then**
        Compute $D_\psi(x, y)$ for every pair of states
        Update the D^*_{SA} actor by $\psi \leftarrow \psi - \lambda_\psi \nabla_\psi \mathcal{L}_\psi$
        Update the target network by $\hat{\varphi} \leftarrow \tau\varphi + (1-\tau)\hat{\varphi}$
    end if
end for

When applied to RL, the detailed learning procedure is shown in Alg. 2. The reason why we can concurrently update the encoder in line 8 is that the input of D is given by the output of the encoder, which corresponds to the gradient backprop problem discussed in Section 3.5. For buffer data collection, we use actions computed from Q-head rather than D-head to interact with environments. Additionally, once we combine Q^2-learning with RL algorithms, the inherent exploration problem in Q^2-learning has no need to be dealt with separately. This is because we can directly regard the policy that an RL algorithm is learning as the "behavioral policy" in Q^2-learning.

A.2 Training Details of MICo-SA

For MICo-SA, the way to represent the metric is similar to Q^2-learning, using the function approximator of the architecture like the D^*_{SA} critic. The learning procedure is similar to MICo, using the TD update rule. MICo-SA updates the encoder with gradients from that part of the network regarding state-action similarity. Specifically, we approximate D^π_{SA} and D^π_S in Eqs. 8 and 9 as: $D_{\varphi'} = ||\varphi'(x, a) - \varphi'(y, b)||_1$ and $D_{\psi'} = ||\psi'(x) - \psi'(y)||_1$, where φ' and ψ' are networks' parameters. We can view $D_{\varphi'}$ and $D_{\psi'}$ as action-value function and state-value function, respectively. Thus accordingly, the following TD objectives are

---

Algorithm 2: Policy training with Q^2-learning

1: Initialize encoder network E_ϕ, target network $E_{\hat{\phi}}$, Q-value network Q_θ, target network $Q_{\hat{\theta}}$, D^*_{SA} critic D_φ, target network $D_{\hat{\varphi}}$, D^*_{SA} actor D_ψ, policy network π_ω, replay buffer $\mathcal{B}$, temperature coefficient α, and target entropy $\overline{\mathcal{H}}$
2: **for** $t = 1$ **to** T **do**
3:     Sample N transitions (o, a, r^a_o, o') from $\mathcal{B}$
4:     Encode observations into latent states: $x = E_\varphi(o)$, $\hat{x}' = E_{\hat{\varphi}}(o')$ (the corresponding reward is denoted as r^a_x)
5:     Update the Q-value network and the encoder by $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \mathcal{L}_\theta$, $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi \mathcal{L}_\theta$
6:     Update the policy network by $\omega \leftarrow \omega - \lambda_\omega \nabla_\omega \mathcal{L}_\omega$
7:     Adjust the temperature coefficient by $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha(-\alpha \log \pi_\omega(x) - \alpha\overline{\mathcal{H}})$
8:     Update D_φ, $D_{\hat{\varphi}}$, and D_ψ according to Alg. 1 and the encoder concurrently by $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi \mathcal{L}_\varphi$
9:     Update the target network by $\hat{\theta} \leftarrow \tau\theta + (1-\tau)\hat{\theta}$, $\hat{\phi} \leftarrow \tau\phi + (1-\tau)\hat{\phi}$
10: **end for**

---

minimized to train them:

$$\mathcal{L}_{\varphi'} = \mathbb{E}_{\substack{(x,a,x')\sim\mathcal{B} \\ (y,b,y')\sim\mathcal{B}}}[(||\varphi'(x,a) - \overline{\varphi'}(y,b)||_1 - |r^a_x - r^b_y| - \gamma||\hat{\psi}'(x') - \hat{\psi}'(y')||_1)^2],$$

$$\mathcal{L}_{\psi'} = \mathbb{E}_{\substack{x\sim\mathcal{B} \\ y\sim\mathcal{B}}}[(||\psi'(x) - \overline{\psi'}(y)||_1 - ||\overline{\varphi'}(x, \pi(x)) - \overline{\varphi'}(y, \pi(y))||_1)^2],$$

where $\hat{\psi}'$ is the target network and its parameters, $\overline{\varphi'}$ means that gradients will not pass through φ', and the policy π is learnt online. Thus, it is still not a completely policy-independent behavioral metric. The reason why we maintain a parameterized function $D_{\psi'}$ for D^π_S is to track the bootstrapped estimate of the TD target for D^π_{SA}. Though in principle there is no need to do this, as the target could be estimated directly from $D_{\varphi'}$ with respect to the current policy, we find it can bring more stable and efficient learning process. Similar approaches were also adopted in (Haarnoja et al. 2018; Dadashi et al. 2021).

B Proofs

B.1 Proof of Lemma 3.1

Proof. The case of Eq. 3 has been proven in previous work (Kemertas and Aumentado-Armstrong 2021). As for Eq. 4, the proof is straightforward as:

$$\begin{aligned} D^\pi(x, y) &= |r^\pi_x - r^\pi_y| + \gamma \mathbb{E}_{x'\sim P^\pi_x, y'\sim P^\pi_y}[D^\pi(x', y')] \\ &\leq \sup_{x,y} |r^\pi_x - r^\pi_y| + \gamma \mathrm{diam}(\mathcal{X}; D^\pi), \forall (x, y) \in \mathcal{X}. \end{aligned}$$

Then, $\mathrm{diam}(\mathcal{X}; D^\pi) = \sup_{x,y} D^\pi(x, y) \leq \sup_{x,y} |r^\pi_x - r^\pi_y| + \gamma \mathrm{diam}(\mathcal{X}; D^\pi) \leq \frac{1}{1-\gamma} \sup_{x,y} |r^\pi_x - r^\pi_y|$. □

B.2 Proof of Lemma 3.3

Proof.

$$D_S^*(x,y) = \max_{a\in\mathcal{A}} |r_x^a - r_y^a| + \gamma\mathbb{E}_{x'\sim P_x^a, y'\sim P_y^a}[D_S^*(x',y')]$$
$$\leq |r_{\max} - r_{\min}| + \gamma \text{diam}(\mathcal{X}; D_S^*),\ \forall(x,y)\in\mathcal{X}.$$

Then, $\text{diam}(\mathcal{X}; D_S^*) = \sup_{x,y} D_S^*(x,y) \leq \frac{1}{1-\gamma}|r_{\max}-r_{\min}|$. □

B.3 Proof of Proposition 3.4

Proof. The proof is by induction. We define value iteration sequence $\{Q^n\}_{n\in\mathbb{N}}$ for all state-action pairs $(x,a)\in\mathbb{R}^{\mathcal{X}\times\mathcal{A}}$. The initial one $Q^0(x,a) = 0$, $\forall(x,a)$. It is obvious that $|Q^0(x,a) - Q^0(y,b)| \leq D_{SA}^*([x,a],[y,b])$. Assume that $|Q^n(x,a) - Q^n(y,b)| \leq D_{SA}^*([x,a],[y,b])$, we now check the case $n+1$:

$$\begin{aligned}
&|Q^{n+1}(x,a) - Q^{n+1}(y,b)| \\
&= |r_x^a - r_y^b + \gamma\mathbb{E}_{\substack{x'\sim P_x^a,\\ y'\sim P_y^b}}[\max_{a'} Q^n(x',a') - \max_{b'} Q^n(y',b')]| \\
&\leq |r_x^a - r_y^b| + \gamma\mathbb{E}_{\substack{x'\sim P_x^a,\\ y'\sim P_y^b}}|\max_{a'} Q^n(x',a') - \max_{b'} Q^n(y',b')| \\
&\leq |r_x^a - r_y^b| + \gamma\mathbb{E}_{\substack{x'\sim P_x^a,\\ y'\sim P_y^b}}\max_{a'} |Q^n(x',a') - Q^n(y',a')| \\
&\leq |r_x^a - r_y^b| + \gamma\mathbb{E}_{\substack{x'\sim P_x^a,\\ y'\sim P_y^b}}\max_{a'} D_{SA}^*([x',a'],[y',a']) \\
&= D_{SA}^*([x,a],[y,b]).
\end{aligned}$$

The penultimate inequality is due to the fact that $|\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)|$. The last inequality is the use of induction hypothesis. The limit of sequence $\{Q^n\}_{n\in\mathbb{N}}$ is Q^*, so we have $|Q^*(x,a) - Q^*(y,b)| \leq D_{SA}^*([x,a],[y,b])$. □

B.4 Proof of Proposition 3.5

Proof. Let $D_{SA}^*, D_{SA}^{*\prime} \in \mathbb{R}^{\mathcal{X}^2\times\mathcal{A}^2}$, and $[x,a],[y,b]\in\mathcal{X}\times\mathcal{A}$, there is:

$$\begin{aligned}
&|F_C(D_{SA}^*)([x,a],[y,b]) - F_C(D_{SA}^{*\prime})([x,a],[y,b])| \\
&= \gamma|\mathbb{E}_{\substack{x'\sim P_x^a,\\ y'\sim P_y^b}}[\max_{a'} D_{SA}^*([x,a'],[y,a']) - \\
&\qquad \max_{a'} D_{SA}^{*\prime}([x,a'],[y,a'])]| \\
&\leq \gamma\mathbb{E}_{\substack{x'\sim P_x^a,\\ y'\sim P_y^b}}\max_{a'} |D_{SA}^*([x,a'],[y,a']) - D_{SA}^{*\prime}([x,a'],[y,a'])| \\
&\leq \gamma\mathbb{E}_{\substack{x'\sim P_x^a,\\ y'\sim P_y^b}}\max_{a',b'} |D_{SA}^*([x,a'],[y,b']) - D_{SA}^{*\prime}([x,a'],[y,b'])| \\
&\leq \gamma\|D_{SA}^* - D_{SA}^{*\prime}\|_\infty.
\end{aligned}$$

Since this holds for any state-action pair in $\mathbb{R}^{\mathcal{X}^2\times\mathcal{A}^2}$, we have that $\|F_C(D_{SA}^*) - F_C(D_{SA}^{*\prime})\|_\infty \leq \gamma\|D_{SA}^* - D_{SA}^{*\prime}\|_\infty$. And $\gamma < 1$, so F_C is a contraction mapping. □

B.5 Proof of Lemma 3.8

Proof. Given a base MDP $\mathcal{M} = \langle\mathcal{X},\mathcal{A},P,r,\gamma\rangle$, we construct an auxiliary MDP $\tilde{\mathcal{M}}_C = \langle\tilde{X},\tilde{A},\tilde{P},\tilde{r},\gamma\rangle$, where $\tilde{\mathcal{X}} = \mathcal{X}\times\mathcal{X}$, $\tilde{\mathcal{A}} = \mathcal{A}\times\mathcal{A}$, $\tilde{P}_{\tilde{x}}^{\tilde{a}} = P_x^a P_y^b$, and $\tilde{r}_{\tilde{x}}^{\tilde{a}} = \tilde{r}_{(x,a),(y,b)} = |r_x^a - r_y^b|$ for every $\tilde{x} = (x,y)$, $\tilde{x}' = (x',y') \in \tilde{X}$, and $\tilde{a} = (a,b)\in\tilde{A}$, where $x,y,x',y'\in\mathcal{X}$ and $a,b\in\mathcal{A}$. There is

$$\begin{aligned}
&F_C(D_{SA})([x,a],[y,b]) \\
&= |r_x^a - r_y^b| + \gamma\mathbb{E}_{\substack{x'\sim P_x^a\\ y'\sim P_y^b}}\max_{a'} D_{SA}([x',a'],[y',a']) \\
&= \tilde{r}_{\tilde{x}}^{\tilde{a}} + \gamma\mathbb{E}_{\tilde{x}'\sim\tilde{P}_{\tilde{x}}^{\tilde{a}}}\max_{\tilde{a}'} D_{SA}(\tilde{x}',\tilde{a}') \\
&= B^*(D_{SA})(\tilde{x},\tilde{a}).
\end{aligned}$$

So, the operator F_C is the Bellman optimality operator for $\tilde{\mathcal{M}}_C$. □

B.6 Proof of Proposition 3.9

Proof. We construct the auxiliary MDP $\tilde{\mathcal{M}}$ as we did in Lemma 3.8. The relationship between the action-value function and the state-value function in $\tilde{\mathcal{M}}$ is expressed as:

$$\tilde{Q}(\tilde{x},\tilde{a}) = \tilde{r}_{\tilde{x}}^{\tilde{a}} + \gamma\mathbb{E}_{\tilde{x}'\sim\tilde{P}_{\tilde{x}}^{\tilde{a}}}[\tilde{V}(\tilde{x}')],$$
$$\tilde{V}(\tilde{x}) = \mathbb{E}_{\tilde{a}\sim\tilde{\pi}}[\tilde{Q}(\tilde{x},\tilde{a})],$$

where $\tilde{\pi}$ is defined as a policy in $\tilde{\mathcal{M}}$. It maps states in $\tilde{\mathcal{X}}$ to actions in $\tilde{\mathcal{A}}$. And the policy only outputs a single action instead of two different actions. Notice that a single action $\tilde{a}$ still belongs to $\tilde{\mathcal{A}}$ because it can be rewritten as $\tilde{a} = (a,a)$. Thus, the corresponding optimal equation is:

$$\tilde{Q}^*(\tilde{x},\tilde{a}) = \tilde{r}_{\tilde{x}}^{\tilde{a}} + \gamma\mathbb{E}_{\tilde{x}'\sim\tilde{P}_{\tilde{x}}^{\tilde{a}}}[\tilde{V}^*(\tilde{x}')],$$
$$\tilde{V}^*(\tilde{x}) = \max_{\tilde{a}=(a,a)}[\tilde{Q}^*(\tilde{x},\tilde{a})].$$

Expanding the formula, it is exactly the same as Eqs. 5 and 6. □

B.7 Proof of Proposition 3.10

Proof. We first prove that for environments with deterministic transitions, if D_{SA} is a pseudometric, then $F_C(D_{SA})$ is a pseudometric. For any $(x,a),(y,b)$, we assume that they transition to x',y', respectively. Thus, we have

- zero self-distance: $F_C(D_{SA})([x,a],[x,a]) = |r_x^a - r_x^a| + \gamma\max_{a'} D_{SA}([x',a'],[x',a']) = 0$,
- symmetry:

$$\begin{aligned}
&F_C(D_{SA})([x,a],[y,b]) \\
&= |r_x^a - r_y^b| + \gamma\max_{a'} D_{SA}([x',a'],[y',a']) \\
&= |r_y^b - r_x^a| + \gamma\max_{a'} D_{SA}([y',a'],[x',a']) \\
&= F_C(D_{SA})([y,b],[x,a]),
\end{aligned}$$

- triangular inequality:

$$
\begin{aligned}
&F_C(D_{SA})([x,a],[y,b]) \\
&= |r_x^a - r_y^b| + \gamma \max_{a'} D_{SA}([x',a'],[y',a']) \\
&\leq |r_x^a - r_z^c| + |r_z^c - r_y^b| + \\
&\quad \gamma \max_{a'} \left(D_{SA}([x',a'],[z',a']) + D_{SA}([z',a'],[y',a'])\right) \\
&\leq |r_x^a - r_z^c| + \gamma \max_{a'} D_{SA}([x',a'],[z',a']) + \\
&\quad |r_z^c - r_y^b| + \gamma \max_{a'} D_{SA}([y',a'],[z',a']) \\
&= F_C(D_{SA})([x,a],[z,c]) + F_C(D_{SA})([z,c],[y,b]).
\end{aligned}
$$

So the statement is satisfied. And then according to Corollary 3.6, repeated execution of the operator F_C makes D_{SA} converge to the fixed point D^*_{SA}. So starting from a function that is zero everywhere, the fixed point of F_C, i.e., D^*_{SA}, is a pseudometric. □

C Implementation Details for DeepMind Control Suite

Our encoder network is based on (Yarats et al. 2021), and modified a little bit for faster training speed according to (Stooke et al. 2021). The encoder has 4 convolution layers with 3×3 kernels, and the strides are all set to 2 except the last layer. The activation function is ReLU except the last layer, which is tanh instead. The output of encoder is a 50-dimensional vector.

The hidden layer of SAC's Q-value function and its corresponding actor function is a fully-connected layer with 1024 neurons. The hidden layer of the D^*_{SA} critic and actor functions is also a fully-connected layer with 1024 neurons, and the output dimension of the D^*_{SA} critic is 32. The learning rates of these two functions are the same as that of encoder. The $actor\_update\_interval$ is shown in Alg. 1 is 1 for all tasks except for the Finger-Spin and Walker-walk, in which the hyperparameter is 2. The transition model used in DBC methods is the same as the one in the original paper. For DBC, the way to compute the D_φ and D_ψ mentioned in Appendix A is also adopted in other-metric methods to compute corresponding distances between every pair of states.

We implement all algorithms based on PyTorch (Paszke et al. 2019). Main experiments are run on NVIDIA GeForce RTX 2080 Ti with 11GB memory. For faster training speed and fitting our machine memory, we choose smaller batch size than (Zhang et al. 2021) used in their source code and correspondingly scale down the learning rate linearly referring to (Goyal et al. 2017). For a fair comparison, we keep these common training hyperparameters consistent across all methods. So the results of DBC are slightly inconsistent with the results in the original paper. Specifically, we obtain lower returns on Walker-Walk while achieving higher returns on Cheetah-Run. It should be noticed that even if the original performance of DBC is achieved, they are still lower than that of Q^2-learning. See Figs. 13 and 14 for detailed results.

The coefficient used in MICo to compute the distance is 0.1, the same as in the original paper. Original MICo also sets a weight coefficient to balance representation loss and policy loss, however, we argue that a robust method should be insensitive to such a coefficient, which means that we don't need to tune it carefully, thus we just treat these two losses equally, which is also adopted in Q^2-learning to keep consistent. There is only exception. On Cartpole-Swingup_sparse, we set this coefficient to 1×10^{-5}, which is the same as used in the MICo source code. The reasons are: a) for equal weight, both MICo and our method cannot learn effectively; and b) it can be an evidence that our method can also benefit from the adjustment of this coefficient.

All methods use the following training hyperparameters:

- Observation size: 84
- Frame stack : 3
- Action repeat: 4 for Cheetah-Run and Cartpole-Swingup_sparse, 8 for Pendulum-Swingup, 2 for others
- Discount factor γ: 0.99
- Initial steps: 1000
- Replay buffer size: 10^6
- Batch size: 128
- Critic's learning rate λ_θ: 3×10^{-4}
- Actor's learning rate λ_ω: 3×10^{-4}
- Actor log STD bounds: $[-10, 2]$
- Alpha's learning rate λ_α: 1×10^{-4}
- Encoder's learning rate λ_ϕ: 3×10^{-4}
- Initial temperature α_0: 0.1
- Critic's and Encoder's soft update rate τ: 0.005
- Optimizer: Adam

D Additional Experiment Results

D.1 Per-task Performance Comparisons

We test the performance in low data regime of 500K environment steps for each task. The episodic return is evaluated every 8K environment steps. The term episodic return is the sum of the undiscounted rewards the agent collect in one episode. We obtain this value by averaging 10 episodes. The environment step is the actual time steps of the simulator (Laskin, Srinivas, and Abbeel 2020). Per-task results of the clean background and the clutter background are shown in Figs. 13 and 14, respectively.

In the clean setting, other methods cannot guarantee to improve the performance of the baseline SAC and even worse than it on some tasks, e.g., MICo on Finger-Spin. This phenomenon is also reflected in MICo's original paper. However, Q^2-learning breaks the dilemma. It consistently improves SAC's performance on all tested tasks and what's more, it can achieve the highest return even if other methods can be good at those tasks. In terms of sample efficiency, Q^2-learning also performs better than others on most tasks.

In the clutter setting, Q^2-learning still stays competitive against other methods on most tasks. Though DBC and DBC-normed are more sample efficient on Cheetah-Run and Finger-Spin, Q^2-learning achieves the closest performance at the end of 500K steps compared to MICo and SAC. However, Q^2-learning fails on Cartpole-Swingup_sparse, which is a sparse reward task. We guess the reason behind it is that reward sparsity plus complex distractors increase the optimization difficulty of deterministic policy gradient, on which Q^2-learning highly relies on, resulting in poor encoder learning consequently. We leave the sparse reward challenge for metric-based methods as a future work.

According to (Agarwal et al. 2021), we additionally use the optimality gap, the amount by which a method fails to meet a desired score (here is 1.00), to summarize the final performance, as shown in Fig. 15. Computing this uses all runs across tasks.

One more thing to illustrate is that the vanilla SAC we implemented is much better than the one in previous papers (e.g., (Yarats et al. 2021)), which we guess is because we have a different training setup. That is, after the initial data collection phase (using a random policy), we only update models once, while the code provided by previous papers will update them many times, which may cause heavy primacy bias (Nikishin et al. 2022) and negatively affecting the rest of the learning process.

D.2 Per-task Results of Ablations

In Fig. 16, we display per-task results of four algorithms, including two ablation methods of Q^2-learning. Q^2-learning-a cannot work effectively. MICo-SA is comparable to MICo on most tasks. It shows that only considering policy-independent state similarity or policy-dependent state-action similarity cannot bring about significant performance gain.

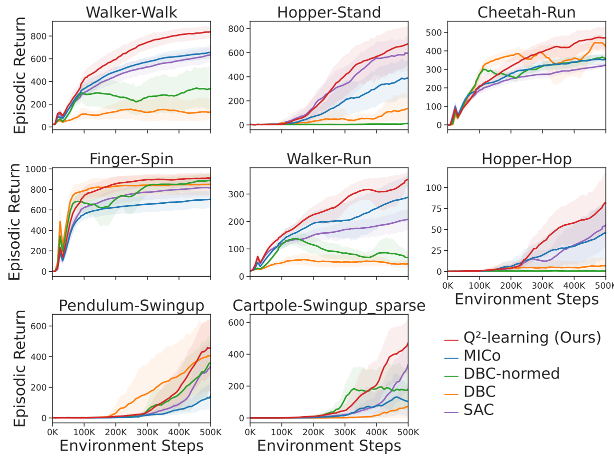

Figure 13: Performance comparison on the DeepMind control tasks with the clean background. 10 random seeds are used. The shaded areas are 95% stratified bootstrap CIs.

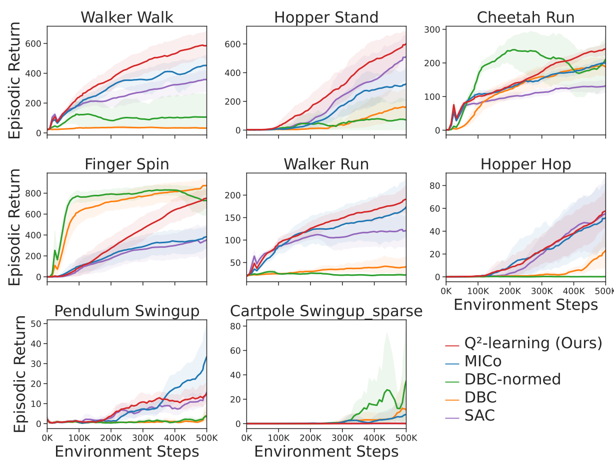

Figure 14: Performance comparison on the DeepMind control tasks with the clutter background. 10 random seeds are used. The shaded areas are 95% stratified bootstrap CIs.

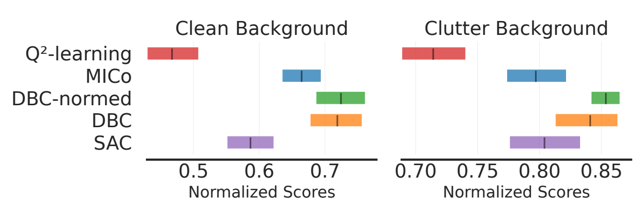

Figure 15: The optimality gap on the 500K-th step. Interval estimates show 95% stratified bootstrap CIs.

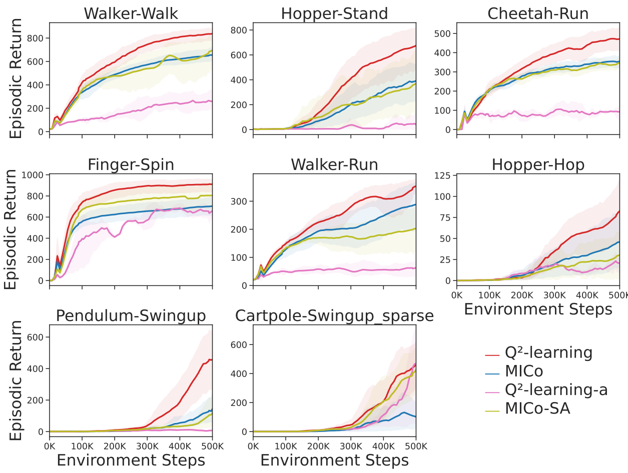

Figure 16: Per-task performance of variants of different algorithms. 10 seeds are used. The shaded areas are 95% CIs.

References

Agarwal, R.; Schwarzer, M.; Castro, P. S.; Courville, A. C.; and Bellemare, M. 2021. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Castro, P. S. 2020. Scalable methods for computing state similarity in deterministic Markov decision processes. In *AAAI Conference on Artificial Intelligence (AAAI)*, 10069–10076.

Dadashi, R.; Rezaeifar, S.; Vieillard, N.; Hussenot, L.; Pietquin, O.; and Geist, M. 2021. Offline reinforcement learning with pseudometric learning. In *International Conference on Machine Learning (ICML)*, 2307–2318.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 1587–1596.

Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 1861–1870.

Kemertas, M.; and Aumentado-Armstrong, T. 2021. Towards robust bisimulation metric learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Laskin, M.; Srinivas, A.; and Abbeel, P. 2020. CURL: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 5639–5650.

Nikishin, E.; Schwarzer, M.; D'Oro, P.; Bacon, P.-L.; and Courville, A. 2022. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 16828–16847.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 8024–8035.

Stooke, A.; Lee, K.; Abbeel, P.; and Laskin, M. 2021. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning (ICML)*, 9870–9879.

Yarats, D.; Zhang, A.; Kostrikov, I.; Amos, B.; Pineau, J.; and Fergus, R. 2021. Improving sample efficiency in model-Free reinforcement learning from images. In *AAAI Conference on Artificial Intelligence (AAAI)*, 10674–10681.

Zhang, A.; McAllister, R. T.; Calandra, R.; Gal, Y.; and Levine, S. 2021. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations (ICLR)*.