

Appendices

A PROOFS

A.1 THE PROOF OF LEMMA 4.3

Lemma 4.3 (weak bisimulation metric). *The weak-bisimulation metric is a contraction mapping w.r.t the L^∞ norm on $\mathbb{R}^{S \times S}$, and there exists a fixed-point d_W^π for \mathcal{F}_W^π .*

Proof. To prove that the \mathcal{F}_W^π function has a fixed point d_W^π , we first need to prove that it is a contraction mapping.

As Definition 4.2, given the fixed expectation of $\epsilon(s_i, s_j)$, weak-bisimulation metric is,

$$\mathcal{F}_W^\pi(d)(s_i, s_j) = \epsilon(s_i, s_j) - \gamma \mathcal{W}_1(d) \left(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi \right) \quad (12)$$

Consider $d, d' \in \mathbb{M}$, we derive,

$$\begin{aligned} & |\mathcal{F}_W^\pi(d)(s_i, s_j) - \mathcal{F}_W^\pi(d')(s_i, s_j)| \\ &= \epsilon(s_i, s_j) + \mathcal{W}_1(d) \left(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi \right) - \epsilon(s_i, s_j) - \mathcal{W}_1(d') \left(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi \right) \\ &= \left| \gamma \left(\mathcal{W}_1(d) \left(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi \right) - \mathcal{W}_1(d') \left(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi \right) \right) \right| \\ &= \left| \gamma \sum_{s'_i, s'_j} \pi(a_i | s_i) \pi(a_j | s_j) \mathcal{P}_{s_i}^{a_i}(s'_i) \mathcal{P}_{s_j}^{a_j}(s'_j) (d - d')(s'_i, s'_j) \right| \\ &\leq \gamma \|d - d'\|_\infty. \end{aligned} \quad (13)$$

Therefore, $\mathcal{F}_W^\pi(d)$ is a contraction mapping w.r.t. the L_∞ norm and there exists a unique fixed-point d_W^π for $\mathcal{F}_W^\pi(d)$. \square

A.2 PROOFS OF THEOREM 4.4 AND THEOREM 4.5

Before proving Theorem 4.4 and Theorem 4.5, we need to present a weak assumption regarding the reward expectation under sparse reward settings. Then, we set the hyperparameter $\mu_c \geq |\mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} - \mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j}|$, where μ_c is the mean of the trainable Gaussian distribution $\epsilon(s_i, s_j)$ in the weak bisimulation metric, i.e., $\mu_c = \mathbb{E}_{s_i, s_j \sim \mathcal{P}} [\epsilon(s_i, s_j)]$.

Assumption A.1 (sparse-reward expectation). *Given a reward function $r_{s_i}^{a_i} = r(s_i, a_i)$ with sparse reward settings, the expectation of $r_{s_i}^{a_i}$ before obtaining the success reward satisfies $\mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} \leq C_1$, where C_1 is a sufficiently small constant.*

Taking the verification environment of this work with sparse rewards as an example, experience shows that almost all transition rewards are usually minimal or even zero before reaching the goal. Therefore, the above Assumption A.1 is a weak assumption that is easy to satisfy in a sparse reward environment.

Then, based on the above weak assumptions, we can easily obtain,

$$0 \leq \left| \mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} - \mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j} \right| \leq 2C_1. \quad (14)$$

In order to make $|\mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} - \mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j}| \leq \mu_c$ hold, we only need to satisfy $C_1 \leq 1/2\mu_c$ in the actual implementation, where the hyperparameters C_1 and μ_c can be seen in Appendix C.2.

In light of the above conclusions, we then prove Theorem 4.4 and Theorem 4.5 respectively.

Theorem 4.4 (Value difference bound). *Given states s_i and s_j , and a policy π , we have,*

$$|V^\pi(s_i) - V^\pi(s_j)| \leq d_W^\pi(s_i, s_j) \quad (15)$$

Proof. We follow the work (Castro, 2020) to prove the value function difference bound. To prove the above theory, we first introduce the standard value function $V_{n+1}^\pi(s_i) = \mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s_i}^\pi(s') V_n^\pi(s')$ and the update operation with the property of contraction mapping (see Lemma 4.3),

$$d_{W,n+1}^\pi(s_i, s_j) = \epsilon(s_i, s_j) + \gamma \mathcal{W}_1(d_n^\pi)(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi) \quad (16)$$

with initial $V_0^\pi \equiv 0$ and $d_{W,0}^\pi \equiv 0$.

Then, we proceed it by induction. For initial case $n = 0$, obviously $|V_0^\pi(s_i) - V_0^\pi(s_j)| \leq d_{W,0}^\pi(s_i, s_j)$ holds, and we then suppose true up to case n . Therefore, we have,

$$\begin{aligned} & |V_{n+1}^\pi(s_i) - V_{n+1}^\pi(s_j)| \\ &= \left| \mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s_i}^\pi(s') V_n^\pi(s') - \left(\mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s_j}^\pi(s') V_n^\pi(s') \right) \right| \\ &\leq \left| \mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} - \mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j} \right| + \left| \gamma \sum_{s' \in \mathcal{S}} V_n^\pi(s') (\mathcal{P}_{s_i}^\pi(s') - \mathcal{P}_{s_j}^\pi(s')) \right| \\ &\leq \left| \mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} - \mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j} \right| + \left| \gamma \mathcal{W}(d_{W,n}^\pi)(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi) \right| \\ &= \left(\left| \mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} - \mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j} \right| - \epsilon(s_i, s_j) \right) + \left(\epsilon(s_i, s_j) + \left| \gamma \mathcal{W}(d_{W,n}^\pi)(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi) \right| \right) \\ &\leq \epsilon(s_i, s_j) + \left| \gamma \mathcal{W}(d_n^\pi)(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi) \right| = d_{W,n+1}^\pi(s_i, s_j) \end{aligned} \quad (17)$$

where the second inequality comes from the dual representation of the Wasserstein distance (Villani, 2021), and the last inequality is true when satisfying $C_1 \leq 1/2\mu_c$ based on the Assumption A.1. By the above steps, we can summarize that, $\forall n \in \mathbb{N}, \forall s_i, s_j \in \mathcal{S}$, $|V_{n+1}^\pi(s_i) - V_{n+1}^\pi(s_j)| \leq d_{W,n+1}^\pi(s_i, s_j)$ holds. \square

Theorem 4.5 (relaxed value difference bound). *Given states s_i and s_j , and a policy π , we have,*

$$d_B^\pi(s_i, s_j) \leq d_W^\pi(s_i, s_j) \quad (18)$$

Proof. To prove $d_B^\pi(s_i, s_j) \leq d_W^\pi(s_i, s_j)$, we just need to prove $d_W^\pi(s_i, s_j) - d_B^\pi(s_i, s_j) \geq 0$. Clearly, given $C_1 \leq 1/2\mu_C$, we have,

$$\begin{aligned} & d_W^\pi(s_i, s_j) - d_B^\pi(s_i, s_j) \\ &= \left(\gamma \mathcal{W}_1(d)(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi) + \epsilon(s_i, s_j) \right) \\ &\quad - \left(\left| \mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} - \mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j} \right| + \gamma \mathcal{W}_1(d)(\mathcal{P}_{s_i}^\pi, \mathcal{P}_{s_j}^\pi) \right) \\ &= \epsilon(s_i, s_j) - \left| \mathbb{E}_{a_i \sim \pi} r_{s_i}^{a_i} - \mathbb{E}_{a_j \sim \pi} r_{s_j}^{a_j} \right| \geq 0 \end{aligned} \quad (19)$$

Then, $d_B^\pi(s_i, s_j) \leq d_W^\pi(s_i, s_j)$ holds. \square

B BACKGROUND SUPPLEMENT

B.1 BISIMULATION RELATION

Bisimulation Relations can be applied to group states in Markov Decision Processes (MDPs) that are behaviorally equivalent, aiding in state space reduction and efficient policy learning. Bisimulation Relation is defined as follows.

Definition B.1 (Bisimulation Relations (Givan et al., 2003)). *Given an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r)$, a relation $E \subseteq \mathcal{S} \times \mathcal{S}$ is a bisimulation relation, if whenever $(s_i, s_j) \in E$ the following properties hold,*

$$\mathcal{R}(s_i, a) = \mathcal{R}(s_j, a) \quad \forall a \in \mathcal{A} \quad (20)$$

$$\mathcal{P}(G|s_i, a) = \mathcal{P}(G|s_j, a) \quad \forall a \in \mathcal{A}, \quad \forall G \in \mathcal{S}_B \quad (21)$$

where \mathcal{S}_B is the partition of \mathcal{S} defined by the relation E (the set of all groups G of equivalent states), and $\mathcal{P}(G|s, a) = \sum_{s' \in G} \mathcal{P}(s'|s, a)$.

B.2 REPRESENTATIONS IN SPARSE REWARDS

Reward signals can implicitly represent specific or abstract regions in observations that cause rewards (Yarats et al., 2021a). Recently, some work has modeled reward signals to obtain useful representation information from observations, such as bisimulation metric representation based on reward differences (Wang et al., 2024). However, when faced with real tasks with sparse rewards, metrics tend to converge to the zero-fixed point due to continuous zero rewards, i.e., representation collapse (Liao et al., 2023). Although prior work has shown that modeling reward sequences can collect richer reward signals (Kemertas & Aumentado-Armstrong, 2021; Yang et al., 2022), the above problem is still difficult to avoid. Instead, it is more important to balance the dependence on reward utilization in sparse reward environments.

C IMPLEMENTATIONS

C.1 ACTOR-CRITIC

Following the work Yarats et al. (2021b), we employ the Actor-Critic (AC) algorithm (Konda & Tsitsiklis, 1999) as the backbone framework of DrQ-v2, where the DrQ-v2 is the basic structure of the above experimental approaches. In general, AC is an off-policy reinforcement learning algorithm for continuous control, consisting of a Critic network with a value function $Q_{\vartheta}(s_t, a_t)$ and an Actor network with a policy function $\pi_v(s_t)$. Similar to Barth-Maron et al. (2018), this setting uniformly uses the n -step return value to enhance the Temporal-Difference error estimation, and uses double $Q_{k=0}$ and $Q_{k=1}$ to alleviate the overestimation bias. Therefore, we train the Critic network using the following loss:

$$\mathcal{L}^Q(\vartheta_k, \phi_\omega) = \mathbb{E}_{\mathcal{D}} \left[Q_{\vartheta_k}(\phi_\omega(s_t), a_t) - \left(\sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \min_{k=0,1} Q_{\bar{\vartheta}_k}^{tgt}(\phi(s_{t+n}), a_{t+n}) \right) \right]^2 \quad (22)$$

where $Q_{\bar{\vartheta}_k}^{tgt}$ is the target Q function with frozen network parameters $\bar{\vartheta}_k$, and $\bar{\vartheta}_k$ is updated from the exponential moving average (EMA) of the trainable parameters ϑ_k . \mathcal{D} represents the experience replay buffer in off-policy DRL.

In addition, since DrQ-v2 adopts a deterministic policy, I train the Actor network with parameter v with the following loss,

$$\mathcal{L}^\pi(v) = -\mathbb{E}_{\mathcal{D}} \left[\min_{k=0,1} Q_{\vartheta_k}(\phi_\omega(s_t), \pi_v(\phi_\omega(s_t))) + \varepsilon \right] \quad (23)$$

where the action noise $\varepsilon \sim \text{clip}(\mathcal{N}(0, \sigma^2))$ is used to ensure the stochastic nature in the deterministic policy. In the interactive phase, $\varepsilon \sim \mathcal{N}(0, \sigma_t^2)$, where σ_t is scheduled standard deviation for the exploration noise. Notably, the encoder parameters ϕ_ω will not be trained by the Actor gradient, see DrQ-v2 for details.

C.2 HYPERPARAMETERS

For the approaches involved in the experimental section, the main encoder network consists of 4 convolutional layers with the same filter sizes 3×3 and strides 2, 1, 1 and 1, respectively. In the Actor and Critic networks, an encoder trunk network is independently set up to map the encoded convolution output to a 50-dimensional feature vector, thereby serving the learning of policies and value functions. For the settings of the SRL representation module, we set a 2-step transition distribution difference to achieve a trade-off between the utilization of the dynamics model and the deviation caused by its accuracy. In addition, we set the sparse reward expectation $C_1 = 0.1$ in Assumption A.1. To satisfy the condition $C_1 \leq 1/2\mu_c$, we conservatively set the mean $\mu_c = 0.5$ of the Gaussian distribution function $\mathcal{N}(\mu_c, \Sigma_\theta)$. It is worth noting that the replay buffer size in this work is set to $2e^5$, which is 20% of the previous capacity. This can effectively verify the efficiency of approaches while avoiding huge resource consumption. In fact, the experimental results also show that the replay buffer size of $2e^5$ may be sufficient. Please see the table below for detailed hyperparameters.

Table 3: Networks dimensions and hyperparameters.

Hyperparameter	Shared Setting
Training steps	1 ~ 4M
Seed frames	4000
Exploration steps	2000
Evaluation episodes	10
Replay buffer capacity	$2e^5$
Episode length	1000 for DMControl, 500 for MetaWorld, 200 for pen and 100 for hammer
Batch size	512 for walker_run and walker_walk, otherwise 256
Frame stack	1 for Adroit, otherwise 3
Discount factor γ	0.97 for MetaWorld, otherwise 0.99
State dims	$9 \times 84 \times 84$
Encoder conv kernels	[32,32,32,32]
Encoder conv filter size	$[3 \times 3, 3 \times 3, 3 \times 3, 3 \times 3]$
Encoder conv strides	[2,1,1,1]
Hidden dims	1024
Latent state dims	50
Action repeat	2
n -step returns	3
Optimizer	Adam
Learning rate	$1e^{-4}$
τ_Q	0.01
Gradient training frequency	2
Exploration temperature	0.1
Hyperparameter	SRL Setting
State transition step T	2
Sparse reward expectation C_1	0.1
Trainable Gaussian distribution mean μ_c	0.5

C.3 LEARNING ALGORITHM

We describe the main algorithm steps as follows.

Algorithm 1 Scalable Representation Learning (SRL)

- 1: Initialize a replay buffer \mathcal{D} with size N , encoder ϕ , policy π , latent model $\hat{\mathcal{P}}_\psi$.
 - 2: **for** $m \leftarrow 1$ to (# epochs) **do**
 - 3: **for** $i \leftarrow 1$ to (# episodes per epoch) **do**
 - 4: Encode state $z_t = \phi_\omega(s_t)$
 - 5: Execute action $a_t \sim \pi_\phi(z_t) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$.
 - 6: Run a step in environments $s_{t+1} \sim \mathcal{P}(\cdot | s_t, a_t)$
 - 7: Collect data $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, r_{t+1}, s_{t+1}\}$
 - 8: **end for**
 - 9: **for** $k \leftarrow 1$ to (# gradient steps) **do**
 - 10: Sample batch $\mathcal{B}_i \sim \mathcal{D}$
 - 11: Rearrange batch $\mathcal{B}_j = \text{Rearrange}(\mathcal{B}_i)$
 - 12: Sample a trainable Gaussian noise $\tilde{\epsilon} = \mu_c + \sigma_1 f_\theta(\phi_\omega(s_j), \phi_\omega(s_j))$, where $\sigma_1 \sim \mathcal{N}(0, 1)$
 - 13: Train encoder $\mathbb{E}_{\mathcal{B}_i, \mathcal{B}_j} [\|\phi_\omega(s_i) - \phi_\omega(s_j)\|_2 - \mathcal{T}^{(T)}(s_i, s_j; \bar{\omega}, \theta)]$ with $\hat{\mathcal{P}}_\psi^\pi$ in Equation (10)
 - 14: Train the Actor-Critic: $\mathbb{E}_{\mathcal{B}_i} [\mathcal{L}^Q(\vartheta_{k=0,1}, \phi_\omega) + \mathcal{L}^\pi(\pi)]$ in Equation (22) and Equation (23)
 - 15: Train latent model $\mathbb{E}_{\mathcal{B}_i} \|\hat{\mathcal{P}}_\psi^\pi(\cdot | \bar{z}_t, a_t) - \bar{z}_{t+1}\|_2$ with frozen latent states
 - 16: Update target critics: $\mathbb{E}_{\mathcal{B}_i} Q_{\vartheta_k}^{tgt} \leftarrow \tau_Q Q_{\vartheta_k} + (1 - \tau_Q) Q_{\vartheta_k}^{tgt}, \forall k = 0, 1$
 - 17: **end for**
 - 18: **end for**
-

D EXPERIMENTAL SUPPLEMENT

D.1 THE BEHAVIOR SIMILARITY UNDER ENCODER SPACES

To quantitatively analyze the accuracy of the converged SRL and DBC encoders in terms of behavioral similarity metrics, we calculate the encoding distances $d = \|\phi_\omega(s_i) - \phi_\omega(s_j)\|_2$, ($0 \leq i, j < 500$) of their pairwise combinations over 500 states and show the two pairs of states with the smallest encoding distance in Figure 8. It is worth noting that to avoid combinations with too close state transition steps, we set $|i - j| \geq 250$ to ensure the effectiveness of the encoding distance. $\|\hat{a}_i(s_i, s_{i+1}) - \hat{a}_j(s_j, s_{j+1})\|$

As depicted in Figure 8, on the left and right sides, we display the two closest groups of encoded distances for each DMC task under the SRL and DBC encoders, respectively. We can easily observe that the two sets of encoding states with the closest distance identified by the SRL encoder, also have very similar behavioral manifestation or task features intuitively. For example, in the `walker_run`, the differences between the states in each group are almost indistinguishable. In contrast, although the states classified by the DBC method have certain similarities, there are obvious differences in the detailed behaviors, so the effect is not good enough. In summary, the above results show that the weak bisimulation metric can effectively cluster similar states, and this ability actually requires SRL to accurately extract task-relevant features in the state, which strongly proves that SRL has powerful representation learning performance.

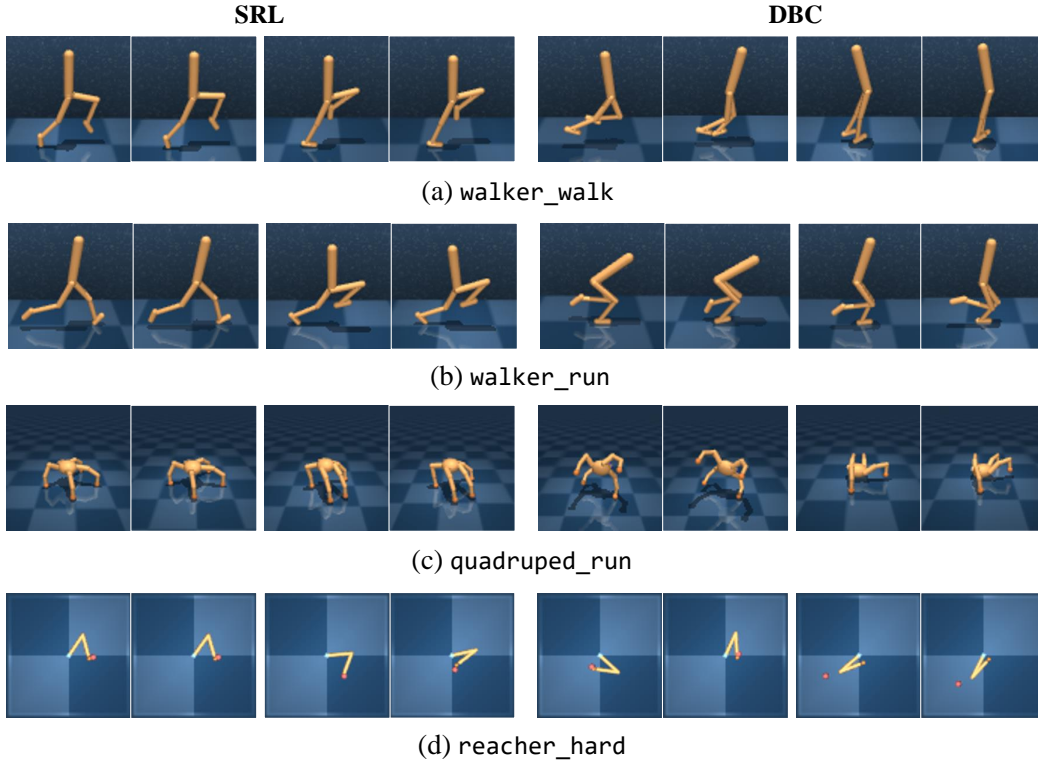


Figure 8: Comparison of the behavioral similarity of state pairs under the latent space of SRL and DBC encoders. **Left:** SRL, **Right:** DBC.

D.2 TASK-RELEVANT VISUALIZATIONS

As shown in Figure 9, we visualize the regions (green) of interest learned by convergent SRL and DBC encoders in the MetaWorld environment. Overall, we can observe that the features extracted by the SRL encoder (left side) are generally more complete and unambiguous. Interestingly, upon closer inspection, we notice that the SRL encoder may have also learned to recognize and emphasize potential safety boundaries, such as the operation boundaries on the desktop in the `box-close`

task. In contrast, the encoding abilities of DBC (right) are bad, as they either only notice partial components (e.g., cups and pallets), while failing to capture potential task components (e.g., buttons and target points), or result in disorganized and unclear visualized regions, such as the `box-close` task. More importantly, the task relationships between components, the abstract features of logic, and the scope of the components themselves seem difficult to represent. Due to the lack of an overall understanding of the task, it is difficult for the DBC to perform each necessary operation in an orderly manner and complete the task. To sum up, SRL can more systematically and clearly extract the entity features and abstract features required to support decision-making tasks compared to the baseline, and demonstrates strong representation learning capabilities in sparse reward environments.

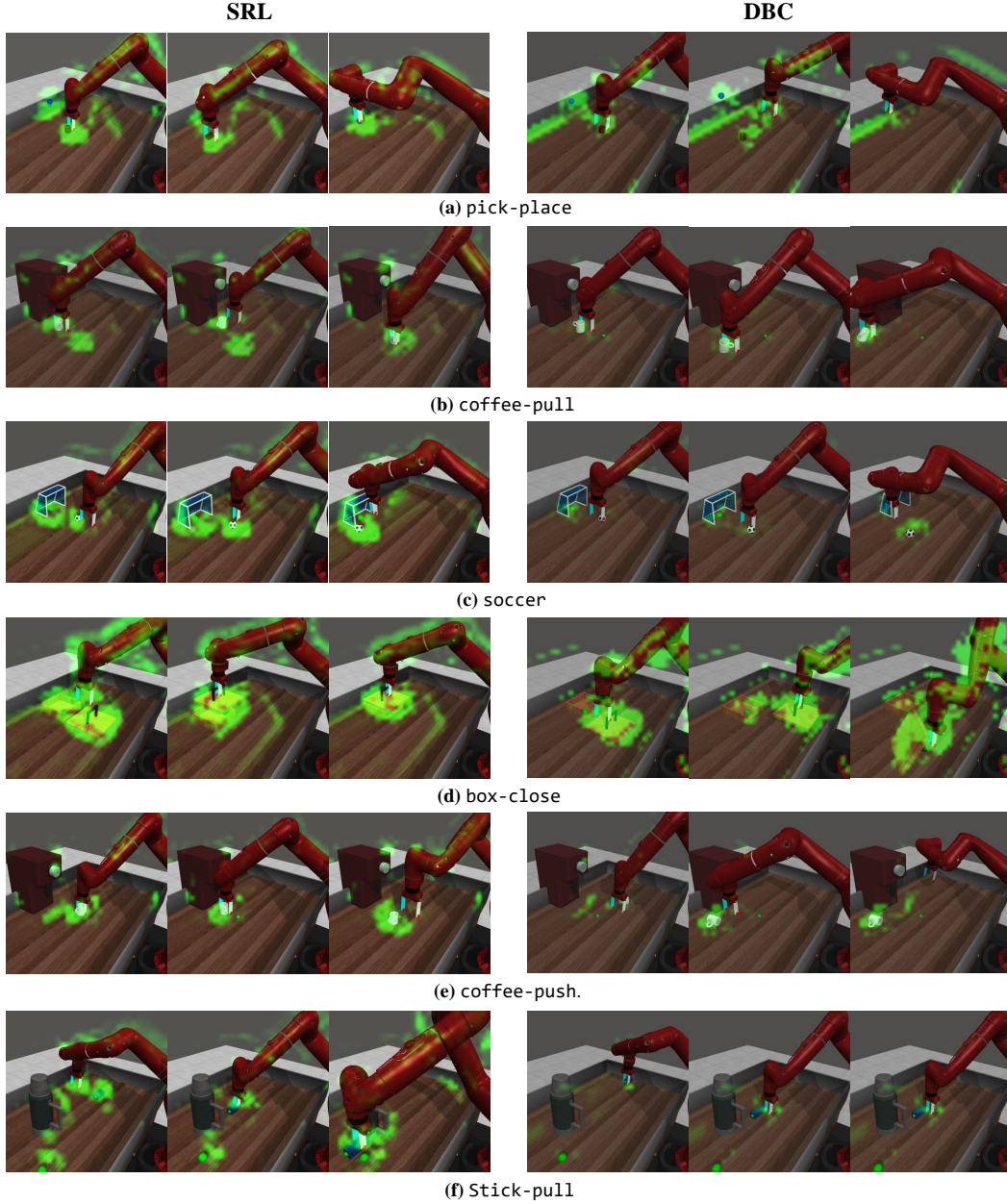


Figure 9: Visualization comparison between task-relevant features captured by the SRL and DBC encoders. **Left:** SRL, **Right:** DBC, with green heatmaps representing the task-relevant regions of interest as identified by the final convolutional layer of each encoder. Note that we employed consistent color parameters across both visualizations to ensure a fair comparison.

D.3 REWARD COMPARISON

As shown in Figure 10 and Figure 11, we report the comparison curves of mean episode rewards between SRL and baselines in the MetaWorld and Adroit environments, respectively. The highest episode reward results are recorded in Table 4. We can observe that, despite the difficulty of the task settings, our method achieves the highest rewards across all tasks. As a key baseline for bisimulation-based methods, DBC performed poorly in almost all tasks, and in some cases, it struggled to achieve dense rewards within the limited training frames.

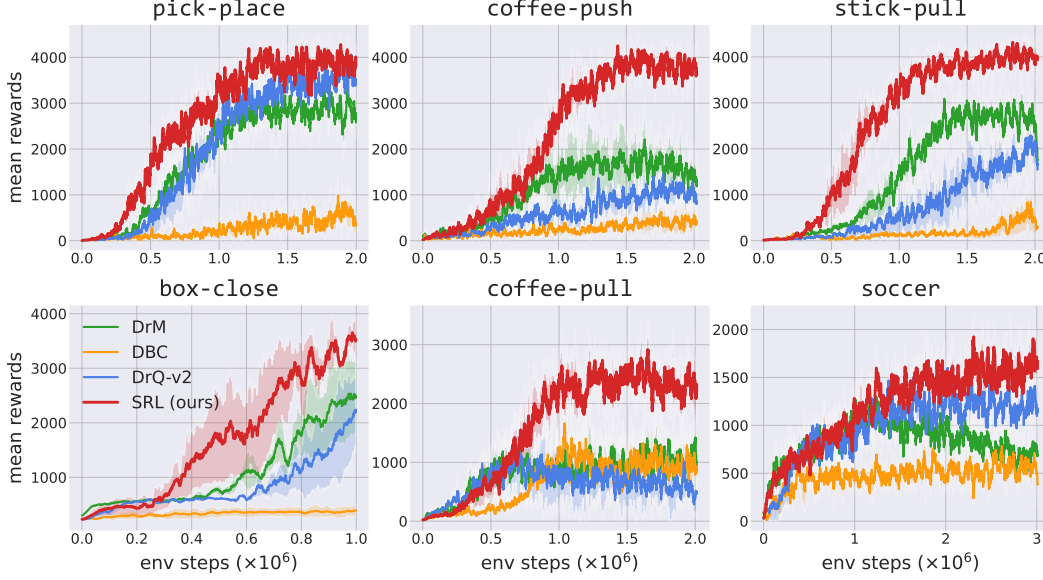


Figure 10: Learning curves of rewards for SRL and baselines on 6 complex tasks with sparse rewards in MetaWorld. Each curve represents the average of three random seeds, with the shaded regions indicating the standard deviation.

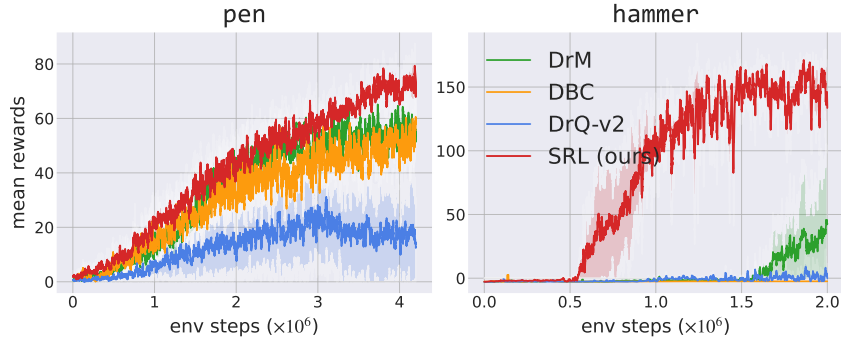


Figure 11: Learning curves of rewards for SRL and baselines on two complex tasks with sparse rewards in Adroit. Each curve represents the average of three random seeds, with the shaded regions indicating the standard deviation.

Table 4: Comparison results of the best mean episode reward on complex MetaWorld and Adroit tasks with sparse rewards.

Methods	pick-place	coffee-push	stick-pull	box-close	coffee-pull	soccer	pen	hammer
DrM	3345 \pm 80	2213 \pm 898	3091 \pm 168	2538 \pm 747	1424 \pm 253	1298 \pm 280	65.2 \pm 7.0	45.8 \pm 59.5
DBC	986 \pm 479	630 \pm 383	843 \pm 647	395 \pm 122	1665 \pm 58	784 \pm 241	60.6 \pm 3.7	2.7 \pm 0.0
DrQ-v2	4061 \pm 76	1460 \pm 59	2294 \pm 421	2242 \pm 788	1225 \pm 572	1598 \pm 168	31.2 \pm 20.0	9.1 \pm 13.6
SRL (ours)	4281\pm194	4254\pm73	4319\pm110	3650\pm203	2916\pm559	1925\pm211	79.4\pm8.0	171.3\pm13.6

D.4 SUPPLEMENTARY COMPARATIVE EXPERIMENT

We show the comparison results between SRL and the method of Kemertas et al. (Kemertas & Aumentado-Armstrong, 2021) in Figure 12 as their work also aims to solve the problem of representation collapse within bisimulation metrics caused by sparse reward settings. Specifically, we insert SRL into their framework to maintain completely consistent parameter and environment settings. Furthermore, we selected "ContinuousCartpole-v0" and "SparsePendulum-v0" with reward sparse modification as test tasks, and completed three sets of comparative experiments with the $N_m = \{1, 2, 3\}$ distractor noise dimensions respectively. First of all, we can easily see that the SRL (Ours) method achieves better performance in the "ContinuousCartpole-v0" task with state dimension $\dim(\mathcal{S}) = 4$, and significant performance improvements in the "SparsePendulum-v0" task with $\dim(\mathcal{S}) = 6$. In addition, according to the their work, the improvement brought by embedding normalization, i.e., "norm" is very critical, so we also show the performance of their method with the "without norm" setting. In contrast, SRL without embedding normalization can also achieve the best performance.

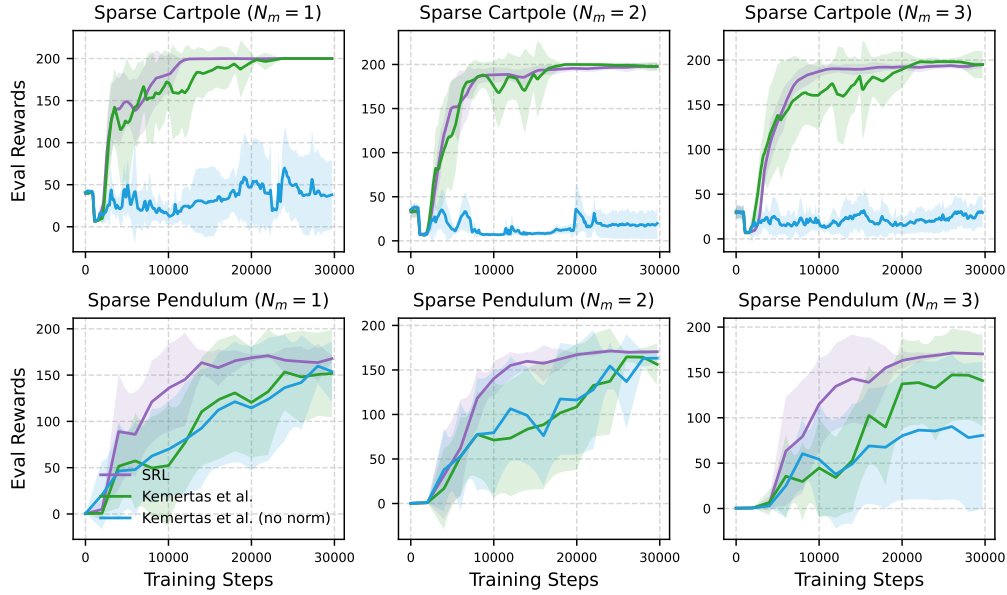


Figure 12: Experimental results on the modified Gym tasks: Cartpole (top row) and Pendulum (bottom row). Each curve is averaged over 10 seeds with one standard deviation shaded in the default setting.