ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSP48485.2024.10447216

SELF-SUPERVISED REINFORCEMENT LEARNING FOR OUT-OF-DISTRIBUTION RECOVERY VIA AUXILIARY REWARD

Yufeng Xie[1,2] *Yinan Wang*[1,2] *Han Wang*[1,2,†] *Qingshan Li*[1,2]

[1] Xi'an Key Laboratory of Intelligent Software Engineering, Xidian University, Xi'an, China
[2] School of Computer Science and Technology, Xidian University, Xi'an, China

ABSTRACT

Recently, the real-world applications of reinforcement learning (RL) have seen the problem of taking actions in an out-of-distribution (OOD) state. However, most existing research is limited to take actions to narrow the visited training distribution and OOD, and does not consider the efficiency to choose such actions. In this paper, we propose a novel approach, called Self-Supervised Reinforcement Learning for OOD recovery via Auxiliary Reward (SRL-AR), to address this issue. By leveraging cumulative reward, we force the representations to discriminate state-action pairs with different returns as auxiliary task. Then, the auxiliary reward calculated from the auxiliary loss is used to generate a new policy that can effectively handle OOD situations. Moreover, we show that our method outperforms prior works in terms of asymptotic performance and sample efficiency on MuJoCo tasks.

Index Terms— Self-Supervised Learning, Reinforcement Learning, Out-of-Distribution Recovery, Auxiliary Reward

1. INTRODUCTION

Through trial and error interaction with the environment, reinforcement learning (RL) provides a promising method for solving problems in the field of decision-making and control. In recent years, RL has achieved remarkable success in various domains such as autonomous driving [1, 2], robotics [3, 4], continuous control tasks [5, 6], and multi-agent systems [7, 8]. However, RL algorithms often struggle when deployed in real-world scenarios due to the presence of out-of-distribution (OOD) states, which are states that significantly deviate from the training distribution. OOD states can arise from unforeseen environmental changes, novel situations, and adversarial attacks. Addressing the challenge of OOD recovery is crucial to ensure the robustness and generalization of RL agent [9], as their performance can deteriorate when encountering OOD states.

This work was supported by the National Natural Science Foundation of China (U21B2015) and the Fundamental Research Funds for the Central Universities (XJSJ23033).

[†]Corresponding author: Han Wang (wanghan@xidian.edu.cn).

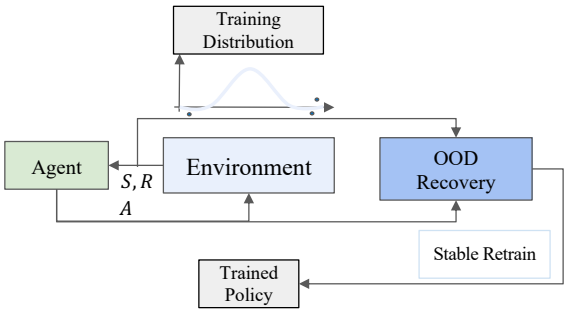


Fig. 1: OOD recovery scenario. RL agent recovers from OOD situations.

Recently, several works have proposed methods to prevent agents from encountering OOD states in RL. Kahn et al. [10] and Lütjens et al. [11] proposed RL with model predictive control (MPC) to prevent uncertain scenarios like collisions. They calculated model uncertainty for motion primitives and used MPC to choose actions based on cost and uncertainty. Henaff et al. [12] penalized high uncertainty trajectories during learning to keep the agent within the desired distribution. Kang et al. [13] combined density modeling and Lyapunov stability to mitigate distribution shifts in learning-based control. Castaneda et al. [14] proposed to learn a task-agnostic policy filter that prevents the system from entering OOD states. However, the main focus of these methods is to prevent the agents from encountering OOD states, failing to solve the problem of trained agents that have already found themselves in OOD states. Even though there are currently a few studies [9, 15] addressing this aspect by taking actions to narrow the visited training distribution and OOD, they do not consider the efficiency to choose such actions.

In this paper, we introduce a self-supervised RL approach, called Self-Supervised Reinforcement Learning for OOD recovery via Auxiliary Reward (SRL-AR), which aims to improve the agent's ability to recover from OOD situations. By incorporating auxiliary rewards and leveraging self-supervised learning, SRL-AR provides a stable learning signal, enabling RL agent to learn a policy that handles OOD states more efficiently. As shown in Figure 1, this is the sce-

nario we focus on. Our main contributions are summarized as follows:

- We introduce the self-supervised state-action abstraction based on return distribution, which effectively forces the learned representations to discriminate state-action pairs with different returns.
- We propose a self-supervised reinforcement learning approach, SRL-AR, which incorporates standard RL algorithms with the auxiliary self-supervised state-action abstraction learning task, enabling RL agent to handle OOD states.
- We present experimental results on MuJoCo tasks. The results show that SRL-AR effectively enables the agent to recover from OOD situations, and outperforms prior works in the sample efficiency.

2. PRELIMINARY

We consider RL problem as a Markov Decision Process (MDP), which is a tuple $< \mathcal{S}, \mathcal{A}, P, R, \mu, \gamma >$. Such tuple specifies the state space $\mathcal{S}$, the action space $\mathcal{A}$, the state transition probability $P(s_{t+1}|s_t, a_t)$, the reward function $R(r_t|s_t, a_t)$, the initial state distribution $\mu \in \Delta(S)$ and the discount factor γ. The policy $\pi(a_t|s_t)$ specifies the action selection probability on each state. We use $p^\pi(\tau)$ to denote the probability distribution over trajectories for policy π. Episodes have T steps, which we summarize as a trajectory $\tau = (o_1, a_1, \cdots, s_T, a_T)$. Without loss of generality, we can assume that rewards are undiscounted, as any discount can be addressed by modifying the dynamics to transition to an absorbing state with probability $1 - \gamma$. The standard RL objective is: $\arg\max_\pi \mathbb{E}_{\tau\sim p^\pi(\tau)}[\sum_{t=1}^{T} r(s_t, a_t)]$.

3. METHODOLOGY

In this section, we present our method SRL-AR, from both theoretical and empirical perspectives. First, we propose self-supervised state-action abstraction $\phi(s)$ based on return distribution. Then we consider an auxiliary reward design method, which enables us to calculate auxiliary reward r^{aux} from the samples collected using the state-action abstraction. After that, we introduce return-based contrastive representation learning for RL that incorporates standard RL algorithms with the auxiliary self-supervised state-action abstraction learning task. Figure 2 illustrates the proposed method.

3.1. Self-Supervised State-Action Abstraction

We extend the Z^π-irrelevance observation-action pair's abstraction [16] based on observation dataset $\mathcal{D}$ with a contrastive loss (see Algorithm 1). We can sample state-action pairs from the trajectories generated by the policy π. Given a policy π, Z^π-irrelevance abstraction is denoted as $\phi(s, a)$ such that, for any $(s_1, a_1), (s_2, a_2) \sim \mathcal{D}$ with $\phi(s_1, a_1) = \phi(s_2, a_2)$, we have $Z^\pi(s_1, a_1) = Z^\pi(s_2, a_2)$. Thus, there exists a function $Q : \phi(s, a) \to \mathbb{R}$, such that $|Q(\phi(s, a)) - Q^\pi(s, a)| \leq (R_{max} - R_{min})/K$.

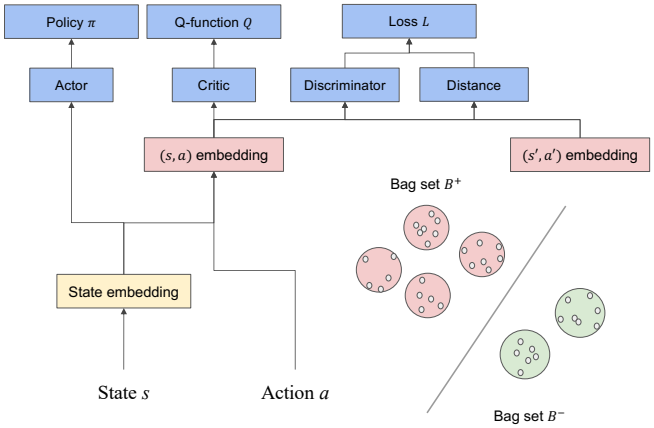


Fig. 2: An overview of SRL-AR. The auxiliary task is based on the state-action embedding.

Algorithm 1 Self-Supervised State-Action Abstraction

Require: Given the policy π_{pre}, the encoder ϕ, the discriminator w, and the number of bins for the return K
Roll-out the current policy π and store the samples to the replay buffer $\mathcal{B}$
$\mathcal{D} = \emptyset$
for $i = 1 \cdots n$ **do**
  Draw a batch of samples $\mathcal{D}$ from the replay buffer $\mathcal{B}$
  Draw state-action pairs $\{(s_1, a_1), (s_2, a_2)\} \sim \mathcal{D}$
  Calculate $R_1 = Z^\pi(s_1, a_1)$ and $R_2 = Z^\pi(s_2, a_2)$
  $\mathcal{D} = \mathcal{D} \cup \{((s_1, a_1, s_2, a_2), \mathbb{1}[b(s_1, a_1) \neq b(s_2, a_2)])\}$
end for
Calculate the loss function $\mathcal{L}(\phi_\theta, w_\nu; \mathcal{D}_{OOD})$
Update ϕ and w with loss function

We maintain the binary label y for this state-action pair, which indicates whether the two returns belong to the same bin. Besides, we introduce a distance-constraint item to enforce the Z^π-irrelevance abstraction $\phi(s, a)$ in bins with larger rewards to be closer. We denote the bins with larger rewards as $\mathcal{B}^+$. Thus, the rest bins are $\mathcal{B}^- = \mathcal{B}/\mathcal{B}^+$, which may contain OOD situations. The hierarchical contrastive loss is defined as follows:

$$\begin{aligned}\min_{\phi,w} \mathcal{L}(\phi, w; \mathcal{D}) :=& \mathbb{E}_{(s,a,y)\sim\mathcal{D}} \left[(w(\phi(s_1, a_1), \phi(s_2, a_2)) - y)^2\right] \\ &+ \lambda_1 dis(\phi^+(s_1, a_1), \phi^+(s_2, a_2)) \\ &- \lambda_2 dis(\phi^+(s_1, a_1), \phi^-(s_2, a_2)) \qquad (1)\end{aligned}$$

3.2. Auxiliary Reward

Principled on SAC [17], we consider a parameterized Q-function $Q_\theta(s_t, a_t)$. By minimizing the objectives in Equation 1, the agent is retrained to return to the learned state distribution from an OOD state by using the proposed auxiliary reward while being regularized by KL-divergence not to forget the original tasks.

The agent may receive an auxiliary reward when taking an action that minimizes the distance between next state and states in $\mathcal{B}^+$, shown in Equation 2. Accordingly, by maximizing the expected cumulative auxiliary reward, the agent can learn how to stay away from OOD states and return to the learned state distribution.

$$r^{aux}(s_t, a_t) = \alpha \cdot \frac{dis(\phi(s_t, a_t), \mathcal{B}^-)}{dis(\phi(s_t, a_t), \mathcal{B}^+)} \tag{2}$$

3.3. Retraining with Auxiliary Task

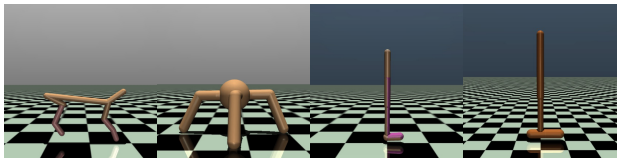
The auxiliary task based RL algorithm is called SRL-AR and shown in Algorithm 2. While Z^π-irrelevance learning relies on a dataset sampled by rolling out the trained policy π_{pre}, SRL-AR constructs such a dataset using the samples collected by the base RL algorithm and therefore does not require additional samples.

Algorithm 2 Retraining

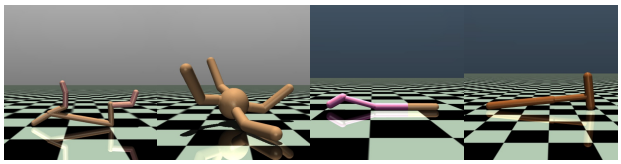
Require: Policy π_{pre}, Q-function Q_θ, replay buffer $\mathcal{B}$, and the embedding ϕ
for each iteration **do**
Rollout the current policy π and store the samples to the replay buffer $\mathcal{B}$
Draw a batch of samples $\mathcal{D}$ from the replay buffer $\mathcal{B}$
Update the policy π_{pre} with auxiliary rewards and KL-loss
end for

During retraining, for each sample in replay buffer, we first draw N positive anchor state-action pairs from each $b \in \mathcal{B}^+$ randomly to calculate $dis(\phi(s_t, a_t), \mathcal{B}^+) = \sum_{|\mathcal{B}^+|} \sum_{i=1}^{N} dist(\phi(s_t, a_t), \phi(s_i, a_i))$. Afterwards, we generate a OOD sample by drawing state-action pairs randomly from $b \in \mathcal{B}^-$ and calculate $dis(\phi(s_t, a_t), \mathcal{B}^-)$. When the ϕ network assigns similar representations to similar state-action pairs, the update for one state-action pair is representative for the updates for other similar state-action pairs, which improves sample efficiency. Meanwhile, state-action pairs with higher rewards indicate the direction to explore, which effectively improve the recovery performance. In order to prevent the agent from unstable learning during the retraining phase, we adopt KL-loss as follows:

$$\mathcal{L}_{stable} = \beta \cdot D_{KL}(\pi(s, a) \| \pi_{pre}(s, a)) \tag{3}$$



(a) Training environments



(b) Retraining environments

Fig. 3: Training environments (a) and retraining environments (b). From left: HalfCheetah-v2, Ant-v2, Walker2D-v2, and Hopper-v2.

4. EXPERIMENTS

In this section, we conduct extensive experiments on MuJoCo tasks [18], which provide RL environments related to robotics and continuous control tasks, to analyze the effectiveness of the proposed method.

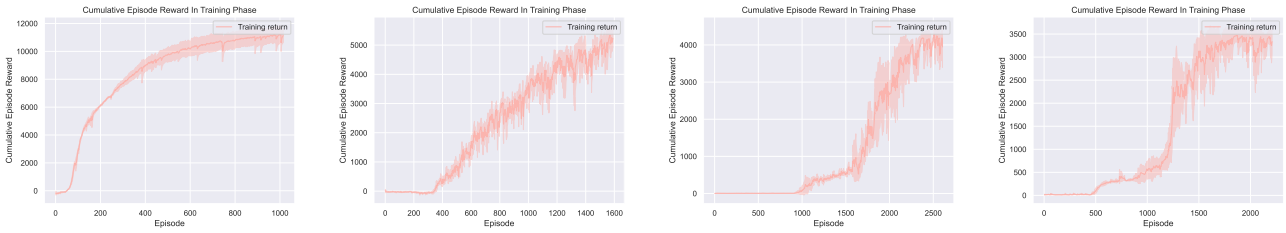
4.1. Experiment Setting

Environments: To evaluate our method, we modify the original environments to implement training environments and retraining environments separately as shown in Figure 3. In the retraining environments, we specifically highlight that a trained agent is initialized in a state not encountered during the training phase. For clarity, in our experiments we refer to states in the training environment as normal states, and states outside this defined set as OOD states, representing the remaining space of possible states. In the training phase, the episode ends if the agent flips over. In the retraining phase, the agent is initially spawned in an upside-down position. To return to the trained state distribution, the agent should learn how to turn its body over.

Baselines: To evaluate the improvement of the agent's ability to recover from OOD situations, We compare our method to several RL algorithms for continuous control tasks, including SeRO [15] and SAC [17]. Note that SRL-AR is implemented by expanding SAC.

4.2. Main Results

Figure 4 shows the cumulative rewards of the SAC algorithm on all tasks during the training phase. However, we are more concerned about whether the policy can effectively handle OOD states in the early stage of retraining and transition to normal states. As shown in Figure 5, it illustrates the changes



(a) HalfCheetahNormal-v2 (b) AntNormal-v2 (c) Walker2DNormal-v2 (d) HopperNormal-v2

Fig. 4: Learning curves in the training environments.

(a) HalfCheetahOOD-v2 (b) AntOOD-v2 (c) Walker2DOOD-v2 (d) HopperOOD-v2

Fig. 5: Learning curves in the retraining environments.

in policy rewards during the early stages of retraining environments. We emphasize that the learning curve in the retraining phase is computed by setting the reward in the OOD state to zero. When we apply the trained policy to OOD situations (retraining phase), the results show SAC agent fails to recover from OOD states on all tasks. Meanwhile, both SRL-AR and SeRO can effectively recover from OOD states, but SRL-AR shows higher sample efficiency and average cumulative reward than SeRO. Table 1 shows the quantitative results of all the methods. The convergence value of SRL-AR is close to the value in the training phase. We attribute this to our approach's ability to limit the extent of policy updates while ensuring effective rewards are obtained.

Table 1: Cumulative reward achieved by SRL-AR and baselines (SeRO, SAC) after the retraining phase.

Task	SRL-AR	SeRO	SAC
HalfCheetah-v2	**11377.96 ± 371.65**	11467.76 ± 764.58	371.59 ± 452.76
Ant-v2	**5862.46 ± 802.36**	5479.12 ± 877.46	151.37 ± 97.14
Walker2D-v2	**3832.76 ± 332.19**	3648.29 ± 412.32	137.49 ± 82.48
Hopper-v2	**3531.93 ± 219.75**	3354.76 ± 378.24	114.83 ± 76.56

4.3. Ablation Studies

To analyze the effect of each component of our method, we perform an ablation study to assess two variants in the retraining phase, including the proposed method that only uses auxiliary reward and is regularized by KL-loss, the results of which are presented in Table 2. Note that the regularization receives zero rewards instead of auxiliary rewards in OOD states. The results show that although the method only implementing auxiliary reward does not achieve the best performance, it can also return to within the learned state distribution, and the cumulative reward can converge to a high value. It is difficult to return to the learned state distribution using regularization alone. However, according to the previous experimental results, regularization plays a certain role in maintaining the learning stability of SRL-AR.

Table 2: Cumulative reward achieved of the ablation studies.

Task	Auxiliary Reward	Regularization
HalfCheetah-v2	9374.96 ± 513.65	2113.48 ± 602.16
Ant-v2	4367.59 ± 753.47	2479.23 ± 865.52
Walker2D-v2	3814.35 ± 319.31	1732.60 ± 523.79
Hopper2D-v2	2932.81 ± 428.20	2468.59 ± 921.46

5. CONCLUSION

In this paper, we propose SRL-AR, a self-supervised reinforcement learning approach for recovering from OOD situations. By incorporating standard RL algorithms with the auxiliary self-supervised state-action abstraction learning task, SRL-AR enables RL agent to handle OOD states. Our experimental results demonstrate that SRL-AR effectively enables the agent to recover from OOD situations, and outperforms prior works in the sample efficiency.

6. REFERENCES

[1] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 2019, pp. 2765–2771.

[2] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7153–7162.

[3] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al., "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[4] Michael Neunert, Abbas Abdolmaleki, Markus Wulfmeier, Thomas Lampe, Tobias Springenberg, Roland Hafner, Francesco Romano, Jonas Buchli, Nicolas Heess, and Martin Riedmiller, "Continuous-discrete reinforcement learning for hybrid control in robotics," in *Conference on Robot Learning*. PMLR, 2020, pp. 735–751.

[5] Chayan Banerjee, Zhiyong Chen, and Nasimul Noman, "Improved soft actor-critic: Mixing prioritized off-policy samples with on-policy experiences," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[6] Duo Xu and Faramarz Fekri, "Improving actor-critic reinforcement learning via hamiltonian monte carlo method," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4018–4022.

[7] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al., "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[8] Yue Jin, Yaodong Zhang, Jian Yuan, and Xudong Zhang, "Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2897–2901.

[9] Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3145–3153.

[10] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[11] Björn Lütjens, Michael Everett, and Jonathan P How, "Safe reinforcement learning with model uncertainty estimates," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8662–8668.

[12] Mikael Henaff, Alfredo Canziani, and Yann LeCun, "Model-predictive policy learning with uncertainty regularization for driving in dense traffic," *arXiv preprint arXiv:1901.02705*, 2019.

[13] Katie Kang, Paula Gradu, Jason J Choi, Michael Janner, Claire Tomlin, and Sergey Levine, "Lyapunov density models: Constraining distribution shift in learning-based control," in *International Conference on Machine Learning*. PMLR, 2022, pp. 10708–10733.

[14] Fernando Castañeda, Haruki Nishimura, Rowan Thomas McAllister, Koushil Sreenath, and Adrien Gaidon, "In-distribution barrier functions: Self-supervised policy filters that avoid out-of-distribution states," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 286–299.

[15] Chan Kim, JaeKyung Cho, Christophe Bobda, Seung-Woo Seo, and Seong-Woo Kim, "Sero: Self-supervised reinforcement learning for recovery from out-of-distribution situations," in *IJCAI*, 2023, vol. 2023, pp. 3884–3892.

[16] Guoqing Liu, Chuheng Zhang, Li Zhao, Tao Qin, Jinhua Zhu, Jian Li, Nenghai Yu, and Tie-Yan Liu, "Return-based contrastive representation learning for reinforcement learning," *arXiv preprint arXiv:2102.10960*, 2021.

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

[18] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.