SeRO: Self-Supervised Reinforcement Learning for Recovery from Out-of-Distribution Situations

Chan Kim[1] , **Jaekyung Cho**[1] , **Christophe Bobda**[2] , **Seung-Woo Seo**[1] and **Seong-Woo Kim**[1]
[1]Seoul National University
[2]University of Florida
{chan_kim, jackyoung96, sseo, snwoo}@snu.ac.kr, cbobda@ece.ufl.edu

Abstract

Robotic agents trained using reinforcement learning have the problem of taking unreliable actions in an out-of-distribution (OOD) state. Agents can easily become OOD in real-world environments because it is almost impossible for them to visit and learn the entire state space during training. Unfortunately, unreliable actions do not ensure that agents perform their original tasks successfully. Therefore, agents should be able to recognize whether they are in OOD states and learn how to return to the learned state distribution rather than continue to take unreliable actions. In this study, we propose a novel method for retraining agents to recover from OOD situations in a self-supervised manner when they fall into OOD states. Our in-depth experimental results demonstrate that our method substantially improves the agent's ability to recover from OOD situations in terms of sample efficiency and restoration of the performance for the original tasks. Moreover, we show that our method can retrain the agent to recover from OOD situations even when in-distribution states are difficult to visit through exploration. Code and supplementary materials are available at https://github.com/SNUChanKim/SeRO.

1 Introduction

Reinforcement learning (RL) has been used to solve challenging tasks in the field of robotics control and has achieved human-level performance [Schulman *et al.*, 2016; Heess *et al.*, 2017; Gu *et al.*, 2017; Akkaya *et al.*, 2019]. However, several limitations prevent RL from being applied in real-world environments. One of the main limitations is the unreliable actions of RL agents in out-of-distribution (OOD) states that deviate from the learned state distribution. While operating in real-world environments, agents can fall easily into OOD states because the state space is extensive and non-stationary, which makes it impossible for the agent to cover the entire space during training. Unfortunately, unreliable actions in OOD states can lead to the failure of the agent [Amodei *et al.*, 2016] because they do not ensure that the agent performs its original tasks successfully. Therefore, agents should learn how to return to learned state distribution from the time they recognize that they have fallen into an OOD state, rather than continue to take unreliable actions. Take a quadruped walking robot trained using RL as an example and suppose that the robot has never been overturned during training. If the robot collides with a person outside the robot's field of view while operating in the real world, it can overturn and unintentionally fall into an OOD state. In this situation, unreliable actions will not enable the robot to operate in its original purpose (*walking*) because the robot has never been trained for such a situation. Instead of taking unreliable actions, the robot should learn how to *turn its body over*, which enables it to return to the learned state distribution. However, the desired behavior for returning to the learned state distribution differs depending on the environment and the OOD situation. Hence, designing the corresponding reward function for each environment and OOD situation is laborious and requires prior knowledge.

Several studies in model-based RL have proposed methods to prevent agents from falling into OOD situations [Kahn *et al.*, 2017; Lütjens *et al.*, 2019; Henaff *et al.*, 2019; Kang *et al.*, 2022]. However, these methods focus on discouraging agents from visiting OOD states. They do not consider the situations where trained agents have already fallen into OOD states, which can unintentionally occur in the real world.

In this study, we propose an RL method for **Se**lf-supervised **R**ecovery from **O**OD situations (SeRO), which can retrain agents to recover from OOD situations without prior knowledge of the environment or OOD situations. Recovery from OOD situations involves both 1) learning to return to the learned state distribution from the OOD situations and 2) restoring performance for the original task after the return. Unlike previous studies that aim to *prevent* the agent from falling into OOD situations, our method aims to *recover* the agent when the trained agent has already fallen into an OOD state during operation. We emphasize that our method is orthogonal and complementary to previous studies that focus on preventing agents from falling into OOD situations.

We denote the agent's training for solving the original task as the *training phase*, whereas the *retraining phase* refers to the additional training required for returning to the learned state distribution and restoring original performance when the trained agent falls into OOD states during operation. When the agent is in OOD states, the agent cannot know the reward

function for the desired behavior of returning to the learned state distribution without prior knowledge of the environment and OOD situation. To retrain the agent to return without an explicit reward designed based on prior knowledge, we propose an intrinsically motivated auxiliary reward that increases as the agent approaches the learned state distribution. The auxiliary reward is implemented based on a metric called the uncertainty distance, which we introduce to approximate the relative distance of the state from the learned state distribution. However, learning the behaviors to return to the learned state distribution can be thought of as learning a new task that is different from the original task. This can cause the agent to forget the original task while learning to return to the learned state distribution. To prevent such a situation, we propose uncertainty-aware policy consolidation. The main contributions of our paper can be summarized as follows:

- We propose a self-supervised RL method, SeRO, which retrains the agent to recover from OOD situations in a self-supervised manner.
- We introduce a metric called the uncertainty distance, which approximately represents the relative distance of the state from the learned state distribution.
- Our in-depth experimental results demonstrate that our method substantially improves the agent's ability to recover from OOD situations in terms of sample efficiency and restoration of the performance for the original tasks.
- Moreover, we demonstrate that the proposed method can successfully retrain the agent to recover from OOD situations even when in-distribution states are difficult to visit through exploration.

2 Related Work

2.1 Preventing OOD Situations in RL

Several attempts have been made to prevent an agent from falling into OOD situations in model-based RL. Kahn *et al.* [2017] and Lütjens *et al.* [2019] proposed methods that used model-based RL with model predictive control (MPC) to prevent agents from falling into uncertain situations such as collision. In these works, the uncertainty of the learned model is calculated for all motion primitives and MPC chooses the action sequence among motion primitives by considering their cost and uncertainty. Similarly, Henaff *et al.* [2019] proposed a method that uses the uncertainty of the learned model to regularize the agent to stay in in-distribution states. Instead of using predefined motion primitives, they penalize the policy during learning when the simulated trajectory generated by the policy causes high uncertainty in the learned model. Kang *et al.* [2022] suggested combining concepts from the density model that estimates training data distribution and the Lyapunov stability [Sastry, 1999] to avoid distribution shifts when using learning-based control algorithms.

In the field of offline RL [Levine *et al.*, 2020], the policy is trained using a fixed offline dataset without additional interaction with the environment. Offline RL methods are more likely to fall into OOD situations because the distribution of the offline dataset cannot entirely cover that of the environment. To address this problem, Fujimoto *et al.* [2019] and Kumar *et al.* [2019] proposed methods to regularize the policy towards the distribution of an offline dataset, and Wu *et al.* [2019], Kumar *et al.* [2020], and Li *et al.* [2022] proposed methods to penalize the value function when the policy deviates from the distribution of an offline dataset during training.

However, these methods focus on preventing the agent from falling into OOD situations by discouraging the policy from selecting the action that leads the agent to OOD states. Unlike previous works, our method addresses situations where the agent has already fallen into OOD situations unexpectedly during operation. Because previous works focus on preventing OOD situations, they do not deal with such situations. In this study, we focus on retraining the agent to recover from such situations in a self-supervised manner, which has the concept of *recovery* rather than *prevention*. Therefore, our method is orthogonal and complementary to previous works that focus on preventing OOD situations.

2.2 Returning to a Particular State Distribution in RL

In the field of autonomous RL [Sharma *et al.*, 2022b] which aims to train the agent in non-episodic environments, the agent is trained to autonomously return to states to restart the training. Eysenbach *et al.* [2018] proposed a method to train a reset policy that leads the agent to a *predefined* initial state distribution. However, in this method, the reward for the training reset policy must be defined based on prior knowledge of the initial state distribution, e.g., in locomotion environments, the reset reward is large when the agent is standing upright. Alternatively, in order to accelerate learning, Sharma *et al.* suggested a training policy to return to states selected by a value-based curriculum [2021] or the distribution of expert demonstrations [2022a], rather than a predefined initial state. In the field of safe RL, Thananjeyan *et al.* [2021] suggested a training recovery policy to return to safe states. In this study, a safety critic that indicates the risk of the action in the given state is *pretrained* using an offline dataset that contains controlled demonstrations of constraint-violating behavior. During policy training, when the policy generates an action with high risk, a recovery policy is executed to generate an action to return to a state with low risk.

These methods all address returning from states that are included in training environments while our method addresses returning from OOD states that are completely *unseen in the training environments*. Moreover, these methods only deal with tasks where returning to a particular state distribution is symmetric to original tasks, e.g., in navigation tasks, *moving to particular positions* for returning is symmetric to *moving to goal positions*, and in manipulation tasks, *collocating objects in particular positions* for returning is symmetric to *collocating objects in goal positions*. Conversely, our method deals with tasks where returning is not symmetric to original tasks, e.g., the agent should learn to *turn its body over* to return to the learned state distribution, whereas the original task is *moving forward as fast as possible*.

3 Preliminaries

3.1 Uncertainty Estimation

Uncertainty can be categorized into two types: aleatoric and epistemic [Abdar *et al.*, 2021]. Aleatoric uncertainty, also known as data uncertainty, is caused by the inherent randomness of an input. In contrast, epistemic uncertainty, also known as model uncertainty, is caused by insufficient knowledge about the data. In this study, we focus on epistemic uncertainty and refer to it as *uncertainty* throughout the paper for the sake of simplicity. Uncertainty can be represented as the variance of the posterior distribution of network parameters W. Hence, knowledge about the posterior distribution of parameters $p(W|D)$ is required, where D refers to the data composed of input x and output y. However, the posterior distribution does not exist in a general neural network because a general neural network is optimized over the deterministic parameter. On the other hand, in Bayesian neural networks (BNN) [Goan and Fookes, 2020], $p(W|D)$ exists by assuming the prior distribution of the parameter $p(W)$. In BNN, given a deterministic input x, the output y is a random variable because the parameter W is assumed to be a random variable, and the distribution of y given D and x is as follows:

$$p(y|D,x) = \int_W p(y|x,W)p(W|D)\,dW, \tag{1}$$

When data that the network has never learned is given, the variance of the posterior distribution $p(W|D)$ increases, thereby increasing the variance of output distribution $p(y|D,x)$. This enables the estimation of uncertainty σ^u via the variance of output distribution as $\sigma^u = \mathrm{Var}_{p(W|D)}(y)$.

Gal *et al.* [2016] proposed a Monte Carlo dropout (MCD) method that uses dropout as a Bayesian approximation to estimate uncertainty. They proved that using multiple forward passes of the network with random dropout activation approximates the Bayesian inference of the deep Gaussian process. The uncertainty estimated using MCD for neural network f with input x is calculated as follows:

$$\sigma^u \approx \sum_{i=1}^{N} \frac{(f^i(x) - \bar{f})^2}{N}, \text{ with } \bar{f} = \sum_{i=1}^{N} \frac{f^i(x)}{N}, \tag{2}$$

where N is the number of forward passes, and f^i refers to the i^{th} forward pass of neural network f with dropout activation. We used MCD to estimate the uncertainty of the state, as it is empirically proven to accurately represent the uncertainty in RL [Wu *et al.*, 2021].

3.2 Soft Actor-Critic (SAC)

SAC [Haarnoja *et al.*, 2018] is an off-policy actor-critic method based on a maximum entropy RL framework, which aims to maximize expected reward while also maximizing entropy. Improvement in exploration and robustness due to maximum entropy formulation enables SAC to solve the challenges of high sample complexity and brittle convergence properties of model-free RL. In this method, soft value iteration for updating the Q-function and the policy is proposed. Soft value iteration alternates soft policy evaluation for updating the Q-function and soft policy improvement for updating the policy. It is proven that the policy provably converges to the optimal policy through soft policy iteration. The objectives for updating the parameterized Q-function Q_θ and the policy π_ϕ are as follows:

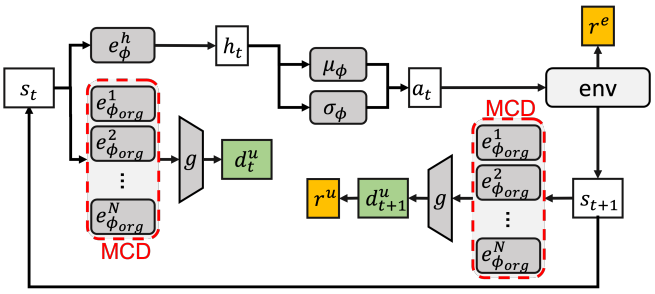


Figure 1: Environment step of SeRO in the retraining phase: All networks with subscript ϕ are the components of the policy π_ϕ. e^n_ϕ with $n = h, 1, 2, ..., N$ refers to the encoder network e_ϕ with dropout activation, and μ_ϕ and σ_ϕ refer to the networks that generate the parameters of the Gaussian action distribution. Note that, the uncertainty distance d^u_t is calculated using the fixed original policy $\pi_{\phi_{org}}$.

$$J(\theta) = \mathbb{E}_{s_t,a_t}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - y\right)^2\right], \tag{3}$$

$$J(\phi) = \mathbb{E}_{s_t,a_t}\left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t,a_t)\right], \tag{4}$$

with $y = r_t + \gamma \mathbb{E}_{s_{t+1},a_{t+1}}\left[Q_\theta(s_{t+1},a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1})\right)]$, where α is entropy coefficient, θ and ϕ are the parameters of the Q-function and the policy respectively. We refer the readers to the original paper for proof of convergence.

4 Method

In this section, we introduce the SeRO framework, which aims to retrain agents to recover from OOD situations in a self-supervised manner. SeRO is implemented by expanding SAC. Note that although our method expands SAC, it can be combined with any RL algorithm trained using the policy gradient method. We propose a novel auxiliary reward that increases as the agent approaches the learned state distribution. When the agent is in the OOD state during the retraining phase, it is trained using the auxiliary reward until it returns to the in-distribution state. Moreover, in order to prevent the agent from forgetting the original task while learning to return to the learned state distribution, we used uncertainty-aware policy consolidation. In the remainder of this section, we describe how our method is implemented in detail.

4.1 Auxiliary Reward for Recovery From OOD Situations

Because it is infeasible to directly calculate the distance of the agent's state from the learned state distribution, we approximate it using the uncertainty of the state predicted through MCD. When state s_t is given, the uncertainty of s_t is calculated by applying Eq. (2) to the encoder network e_ϕ, which is a component of policy π_ϕ, as shown in Fig. 1. The calculated uncertainty σ^u is a vector that has the same dimensions as the output because it approximates the variance of the output distribution of e_ϕ. To represent the relative distance of the state from the learned state distribution using the uncertainty vector σ^u, we propose using the mapping function $g : \mathbb{R}^d \to \mathbb{R}$,

which maps σ^u to the uncertainty distance d^u. First, each element in σ^u is normalized using min-max normalization to generalize the range of each element to between $[0, 1]$; this enables the uncertainty distance to have a generalized range for all environments. The element-wise maximum value of σ^u is saved in the policy and updated whenever a new maximum value occurs during the training as follows:

$$\langle \sigma^{max} \rangle_i = \begin{cases} -\infty & \text{if } t = 0 \\ \langle \sigma_t^u \rangle_i & \text{else if } \langle \sigma_t^u \rangle_i > \langle \sigma^{max} \rangle_i \\ \langle \sigma^{max} \rangle_i & \text{otherwise} \end{cases}, \forall i \in \{1, 2..., d\}, \tag{5}$$

where $\langle \cdot \rangle_i$ is the i^{th} element of the vector, d is the dimension of the σ^u, and t is the time step of the training. Because MCD approximates the variance of the output distribution of e_ϕ, the element-wise minimum value of σ^u is set to zero which is the lower bound of the variance. Finally, the uncertainty distance d^u is calculated by normalizing each element of σ^u and taking the element-wise weighted average of the normalized uncertainty vector as follows:

$$d_t^u = g(\sigma_t^u) = \sum_{i=1}^{d} w_i \frac{\langle \sigma_t^u \rangle_i}{\langle \sigma^{max} \rangle_i}, \text{ where } w_i = \frac{\frac{\langle \sigma_t^u \rangle_i}{\langle \sigma^{max} \rangle_i}}{\sum_{i=1}^{d} \frac{\langle \sigma_t^u \rangle_i}{\langle \sigma^{max} \rangle_i}}. \tag{6}$$

We used the weighted average to prevent an element with high uncertainty from being offset by one with low uncertainty. As each element of the uncertainty vector is guaranteed to be in the range of $[0, 1]$, the uncertainty distance is also in the range of $[0, 1]$. The uncertainty distance is close to 1 when the state of the agent has high uncertainty, which means the state is far from the learned state distribution. In contrast, the uncertainty distance is close to 0 when the state of the agent has low uncertainty, which means the state is close to the learned state distribution. We designed an auxiliary reward as a negative uncertainty distance as follows:

$$r^u(s_t, a_t, s_{t+1}) = -d_{t+1}^u = -g(\sigma_{t+1}^u). \tag{7}$$

When the agent chooses action a_t at state s_t and reaches the next state s_{t+1}, it receives a reward according to the uncertainty distance of the next state. When the agent takes an action that minimizes the uncertainty distance of the next state, which means approaching the learned state distribution, it receives a high auxiliary reward. Accordingly, by maximizing the expected cumulative auxiliary reward, the agent can learn how to return to the learned state distribution.

4.2 Uncertainty-Aware Policy Consolidation (UPC)

Catastrophic forgetting for the original task is problematic in terms of the sample efficiency of the retraining phase because the agent should relearn the original task after learning how to return to the learned state distribution. In order to prevent the agent from forgetting the original task during the retraining phase, we proposed UPC loss as follows:

$$\mathcal{L}_{con}^{\pi} = (1 - d_t^u) D_{KL} \left(\pi(a_t|s_t) || \pi_{org}(a_t|s_t) \right), \tag{8}$$

where π_{org} is the policy for the original task, which is the fixed policy after the training phase. When the uncertainty distance is small, which indicates that the agent is close to the learned state distribution, the effect of the UPC is increased to regularize the policy to take action that is similar to the original policy to solve the original task. In contrast, when the agent's state is far from the learned state distribution, the effect of the UPC is reduced, which enables the agent to learn new behavior to return to the learned state distribution.

4.3 Self-Supervised RL for Recovery From OOD Situations

As SeRO expands SAC, we consider the parameterized Q-function $Q_\theta(s_t, a_t)$ and the policy $\pi_\phi(a_t|s_t)$. The policy and Q-function are updated based on soft value iteration using experience sampled from the replay buffer $\mathcal{D}$. The Q_θ is updated by minimizing Eq. (3) where the reward function r_t is defined as follows:

$$r_t = \begin{cases} r_t^e & \text{if } s_t \in \mathcal{S}_{in} \\ \lambda r_t^u & \text{else if } s_t \in \mathcal{S}_{OOD} = (\mathcal{S}_{in})^c \end{cases}, \tag{9}$$

where r^e is an environmental reward for the original tasks, λ is the weight coefficient, and $\mathcal{S}_{in}$ and $\mathcal{S}_{OOD}$ correspond to in-distribution state space and OOD state space, respectively. The objective for updating the policy π_ϕ is based on Eq. (4) and augmented by UPC loss to regularize the policy. The augmented objective for training π_ϕ is as follows:

$$J(\phi) = \mathbb{E}_{(s_t, d_t^u) \sim \mathcal{D}, a_t \sim \pi_\phi} \left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t) + \mathcal{L}_{con}^{\pi_\phi} \right]. \tag{10}$$

By updating the Q-function and the policy alternately, the agent is trained to return to the learned state distribution in the OOD states using the proposed auxiliary reward, and trained to solve the original tasks using an environmental reward when the agent returns to the learned state distribution. A detailed explanation of the overall retraining procedure can be found in the supplementary material.

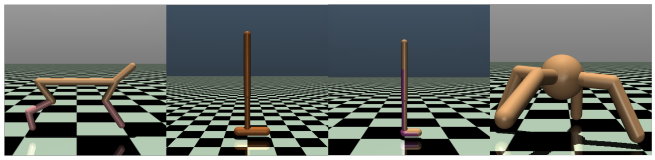
5 Experiments

Our in-depth experiments were designed to answer the following questions: 1) Is retraining for recovery from OOD situations necessary? 2) Can the proposed uncertainty distance successfully represent the relative distance of the state from the learned state distribution? 3) Can our method improve the agent's ability to recover from OOD situations compared to the baseline? 4) Can our method self-recognize whether the agent is in an OOD state and retrain the agent to recover from OOD situations? 5) What is the effect of each component of the proposed method?

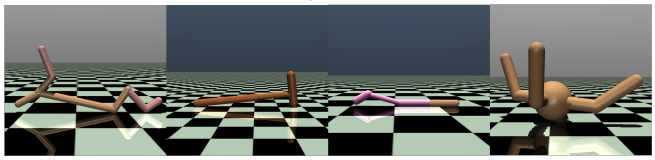
We conducted experiments on four OpenAI gym's MuJoCo environments [Brockman *et al.*, 2016] to answer the above questions. To evaluate the improvement of the agent's ability to recover from OOD situations, we used SAC as a baseline because our method was implemented by expanding it. In the remainder of this section, we explain how we implemented the environments for the experiments and the result of the experiment for each question in detail. Note that the experiments for the second and fifth questions can be found in the supplementary material.

Method	HalfCheetah-v2		Hopper-v2		Walker2D-v2		Ant-v2	
	HalfCheetahNormal-v2	HalfCheetahOOD-v2	HopperNormal-v2	HopperOOD-v2	Walker2DNormal-v2	Walker2DOOD-v2	AntNormal-v2	AntOOD-v2
SAC	11895.96±591.54	-635.38±112.47	3411.02±146.90	758.47±366.47	4069.70±432.95	576.48±527.65	5555.20±866.35	0.86±8.49

Table 1: Average returns computed over 100 episodes after the training phase.



(a) Training environments



(b) Retraining environments

Figure 2: Training environments (top) and retraining environments (bottom). From left: HalfCheetah-v2, Hopper-v2, Walker2D-v2, and Ant-v2.

5.1 Environments

We used HalfCheetah-v2, Hopper-v2, Walker2D-v2, and Ant-v2 from the gym's MuJoCo environments. To evaluate our method, we modified the original environments to implement training environments (e.g., HalfCheetahNormal-v2, HopperNormal-v2, Walker2DNormal-v2, and AntNormal-v2) and retraining environments (e.g., HalfCheetahOOD-v2, HopperOOD-v2, Walker2DOOD-v2, and AntOOD-v2) separately as shown in Fig. 2. In the retraining environments, we emphasize that a trained agent is spawned in a state that is *unseen in the training phase*. In the experiments, we denoted states in the training environments as the in-distribution states, and the states belonging to the rest of the state space as the OOD states for the sake of clarity. We want to note that our method solely addressed OOD states that can be recovered by the agent based on its physical characteristics.

HalfCheetah-v2: In HalfCheetahNormal-v2, the episode is terminated when the agent flips over. Whether the agent flips over is determined by whether the angle of the agent's front tip is out of a certain range. In HalfCheetahOOD-v2, the agent is spawned upside down. To return to the learned state distribution, the agent should learn how to *turn its body over*.

Hopper-v2 & Walker2D-v2: In HopperNormal-v2 and Walker2DNormal-v2, the episode is terminated when the agent falls down. Whether the agent falls down is determined by whether the angle of the agent's top part or the height is out of a certain range. The agent is spawned lying face up and lying face down on the floor in HopperOOD-v2 and Walker2DOOD-v2, respectively. To return to learned state distribution, the agent should learn how to *stand up*.

Ant-v2: In AntNormal-v2, the episode is terminated when the agent flips over. Whether the agent flips over is determined when the pitch or roll angle of the agent's torso is out of a certain range. In AntOOD-v2, the agent is spawned upside down. To return to the learned state distribution, the agent should learn how to *turn its body over*.

We refer the readers to the supplementary material for a more detailed explanation of the environments.

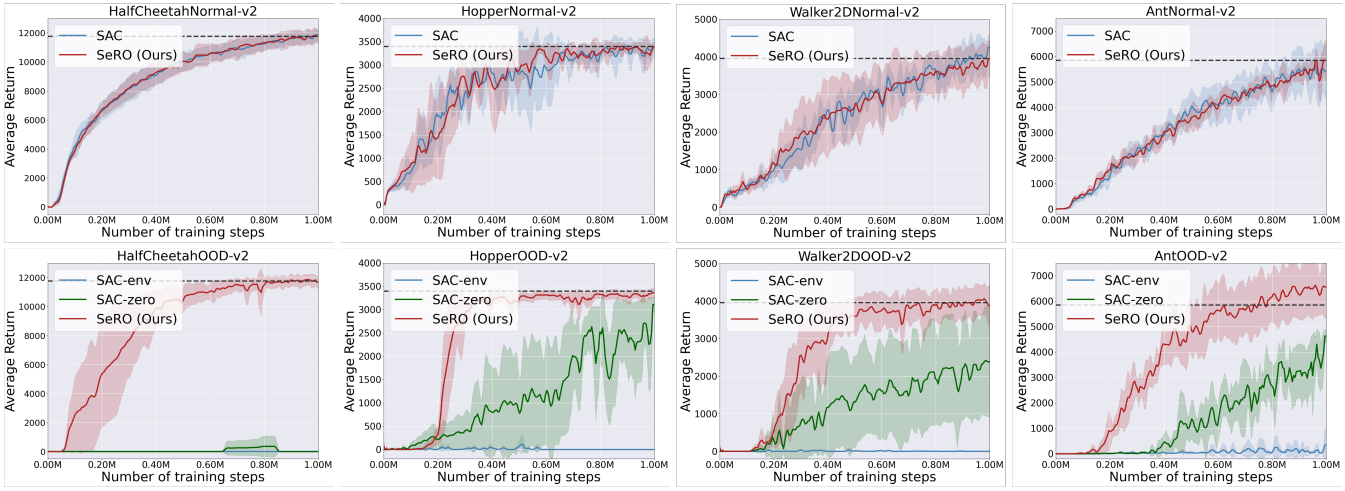
5.2 Analysis of the Necessity of Retraining

In this subsection, we analyze whether the retraining of the trained agent is necessary for OOD situations. If agents trained for solving the original tasks can perform well in the OOD states without retraining, retraining for recovery from OOD situations may not be needed. To verify the necessity of retraining, we first trained the agents in the training environments for 1 million steps using SAC, and we then evaluated the trained agents in the training environments and retraining environments respectively. Table 1 displays the average returns of the environmental reward for the original tasks computed over 100 episodes on five random seeds. As shown in the table, the average return of the agent was significantly lower in the retraining environments than in the training environments. This result suggests that the trained agent could not perform the original tasks well when it fell into an OOD state, and therefore, retraining for recovery from OOD situations is necessary.

5.3 Retraining for Recovery From OOD Situations

We conducted an experiment to compare our method and SAC to answer the question that whether our method can improve the agent's ability to recover from OOD situations. Both methods are trained in the training phase for 1 million steps. Subsequently, the trained agents are retrained to recover from OOD situations in the retraining phase. We retrained the SAC agent in two different ways; One is SAC–env which receives *environmental rewards* for the original tasks in OOD states, and the other is SAC–zero which receives *zero rewards* in OOD states. Both methods receive environmental rewards for the original tasks in in-distribution states. In the case of SAC–zero, the agent is guided to return to the learned state distribution because the agent has a chance to receive a high reward if it visits in-distribution states while receiving zero rewards if it stays in OOD states. Fig. 3 shows learning curves acquired in the training phase and retraining phase for each environment. The learning curves represent the average return of the environmental reward for the original tasks. We emphasize that the learning curves for the retraining phase are calculated by setting the reward in OOD states to *zero* to intuitively represent whether the agent successfully learns to return to in-distribution states so that a higher average return means that the agent returned to the learned state distribution and performed the original tasks successfully.

As shown in the figure, both methods exhibited comparable performance in the training environments. However, in the retraining phase, SAC–env agent failed to recover from OOD situations in all environments. We found that the agent falls into a local optimum and tries to perform the original tasks without returning to the learned state distribution, e.g., in AntOOD-v2, the agent moves without turning their body. Further analysis and qualitative results of the SAC–env after



(a) HalfCheetah-v2 (b) Hopper-v2 (c) Walker2D-v2 (d) Ant-v2

Figure 3: Learning curves for the training environments (top) and the retraining environments (bottom), calculated for five episodes of evaluation at every 5000 steps of training. The darker-colored lines and shaded areas represent the average returns and standard deviations computed over five random seeds. The black dashed horizontal line represents the average return of the SeRO agent after the training phase.

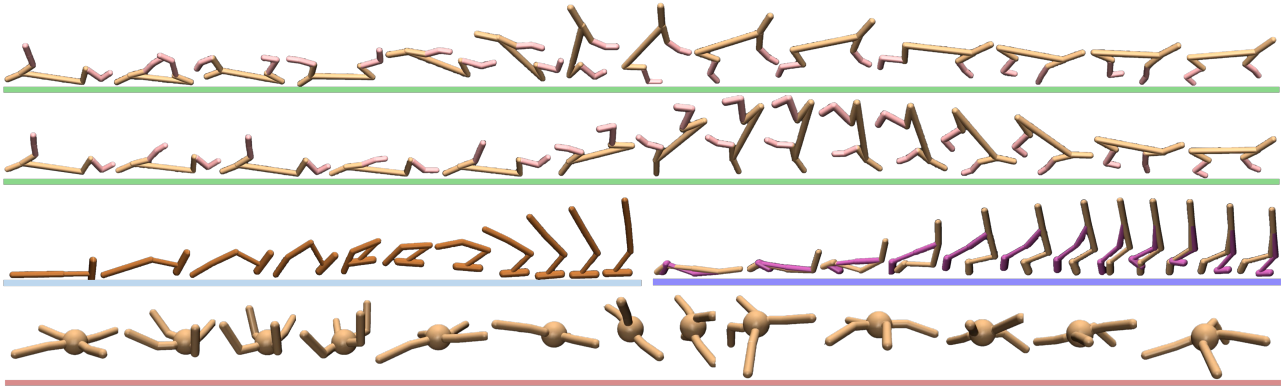
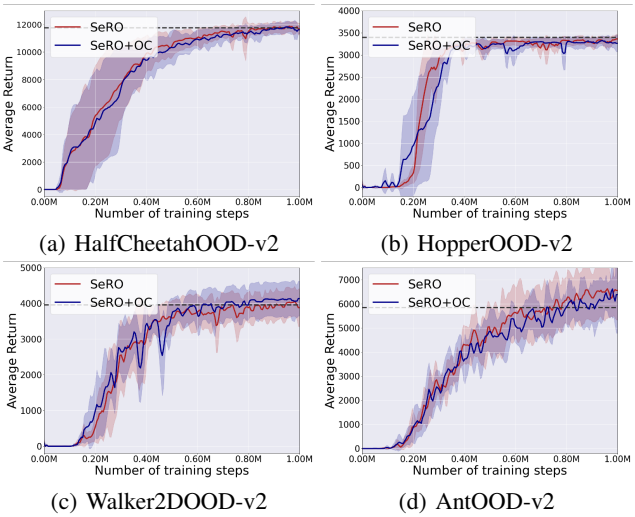


Figure 4: Qualitative results of SeRO after the retraining phase. The figure shows the agent's motion for a time increasing from left to right in each environment.

the retraining phase can be found in the supplementary material. When comparing SeRO and SAC–zero, SeRO showed much higher sample efficiency and average return than SAC–zero. Moreover, while SAC–zero failed to restore the original performance, SeRO successfully restored the original performance for all environments considering that the average return converges to that reached in the training phase. These results are due to the two characteristics of SeRO. First, the auxiliary reward enables SeRO agents to distinguish the values of OOD states according to their distance from the learned state distribution even before visiting in-distribution states. We refer the readers to the experiment for the second question in the supplementary material for whether the uncertainty distance can approximate the distance of the state from the learned state distribution. In contrast, SAC–zero agents cannot distinguish the values of OOD states until it visits in-distribution states because it receives constant zero rewards in OOD states. Therefore, SeRO agents can quickly learn how to return to learned state distribution, while SAC–zero agents cannot learn until they first visit in-distribution states through exploration. We found that SAC–zero failed to return to the learned state distribution in HalfCheetahOOD-v2 where in-distribution states are difficult to visit through exploration, while our method succeeded for all seeds. Second, UPC prevents SeRO agents from forgetting the original tasks during learning to return to the learned state distribution and also regularizes agents to take action similar to what they learned in the training phase once they return to in-distribution states. Therefore, once the agent learns to return to the learned state distribution, SeRO converges quickly to the average return comparable to that reached in the training phase. To summarize the results, the reward for the original tasks cannot lead the agent to return to the learned state distribution and therefore the reward for recovery should be defined separately. The constant zero rewards in OOD states are also not suitable because they cannot guarantee the agent to return to the learned state distribution in environments where in-distribution states are difficult to visit through exploration. In contrast, the proposed auxiliary reward successfully leads the agent to return even in such environments. Moreover, the proposed UPC enables the agent to restore the original performance for the original tasks quickly by preventing catastrophic forgetting.



(a) HalfCheetahOOD-v2 (b) HopperOOD-v2

(c) Walker2DOOD-v2 (d) AntOOD-v2

Figure 5: Learning curves for the retraining environments of SeRO and SeRO+OC, calculated for five episodes of evaluation at every 5000 steps of training. The darker-colored lines and shaded areas represent the average returns and standard deviations computed over five random seeds. The black dashed horizontal line represents the average return of the SeRO agent after the training phase.

As a result, our method improves the agent's ability to recover from OOD situations in terms of sample efficiency and restoration of the original performance for the original tasks.

Fig. 4 visualizes the qualitative results of the agent trained using SeRO after the retraining phase. In HalfCheetahOOD-v2, the agent first shakes its legs to create rotational force and then hits the floor and tumbles to turn over. As shown in the figure, because our auxiliary reward is not designed explicitly for the desired behavior, agents learn diverse ways to return to the learned state distribution, such as tumbling forward or backward. In HopperOOD-v2, the agent first bends its top part to raise the body and then stands up by pushing the floor using the top part. In Walker2DOOD-v2, the agent first raises its top part, steps on the floor with one knee, and then stands up by pushing the floor with the knee. In AntOOD-v2, the agent turns its body over with momentum by hitting the floor with its legs. Although there is no explicit reward based on the prior knowledge of the OOD situations and environments, the agent trained using SeRO successfully learns the desired behavior to return to the learned state distribution in a self-supervised manner.

5.4 Retraining for Recovery From OOD Situations With the Agent's Own Criterion

In the previous subsections, we defined the states in the training environments as in-distribution states $s_{in} \in \mathcal{S}_{training}$ and states belonging to the rest of the state space as the OOD states $s_{OOD} \in (\mathcal{S}_{training})^c$ for the experiments. Accordingly, the reward function for retraining SeRO in Section 5.3 can be formulated as follows:

$$r_t = \begin{cases} r_t^e & \text{if } s_t \in \mathcal{S}_{training} \\ \lambda r_t^u & \text{otherwise} \end{cases}, \tag{11}$$

However, determining whether the current state is in $\mathcal{S}_{training}$ requires prior knowledge of $\mathcal{S}_{training}$, which is infeasible in real-world environments. The reason we used explicit criterion for distinguishing in-distribution states and OOD states in this way was for a fair comparison with the baseline (SAC–zero), which could not recognize whether the current state is the in-distribution state or the OOD state itself. To retrain the agent completely without prior knowledge or human intervention, the agent should recognize whether the current state is the OOD state or the in-distribution state based on its own criterion. In this experiment, we retrained SeRO with its own criterion based on the uncertainty distance. The corresponding reward function for retraining SeRO can be formulated as follows:

$$r_t = \begin{cases} r_t^e & \text{if } d_t^u < \epsilon \\ \lambda r_t^u & \text{otherwise} \end{cases}, \tag{12}$$

where ϵ is the threshold. When the uncertainty distance is smaller than ϵ, which means that the agent's state is close to the learned state distribution, the agent is trained using environmental rewards to solve the original task. Otherwise, the agent is trained using auxiliary rewards to return to the learned state distribution.

We first trained the SeRO agent in the training environments for 1 million steps. Subsequently, the trained agent is retrained in the retraining environments with the reward function Eq. (12). Fig. 5 shows the learning curves of the SeRO trained in Section 5.3 and the SeRO trained with its own criterion which we refer to as SeRO+OC. Note that the same trained agent is used for both algorithms in the retraining phase. For a fair comparison with SeRO, the average return of SeRO+OC in the plots is calculated in the same way as in Section 5.3 using an explicit criterion based on the prior knowledge of $\mathcal{S}_{training}$, although SeRO+OC is actually trained using the reward function in Eq. (12) with its own criterion. As shown in the figure, SeRO+OC shows comparable performance to SeRO for all environments. The results demonstrate that our method can learn to return to the learned state distribution and perform original tasks successfully with its own criterion by self-recognizing whether the current state is the OOD state or the in-distribution state according to the uncertainty distance, which enables self-supervised recovery without prior knowledge about environments and OOD situations.

6 Conclusion

In this study, we addressed the situation in which a trained agent has already fallen into an OOD state, which can happen unintentionally in real-world environments. We introduced SeRO, a self-supervised RL method for recovery from OOD situations. In particular, we proposed an auxiliary reward that guides the agent to the learned state distribution. We also proposed uncertainty-aware policy consolidation to prevent the agent from forgetting the original task while learning how to return to the learned state distribution. We evaluated our method on OpenAI gym's MuJoCo environments and demonstrated that the proposed method can improve the agent's ability to recover from OOD situations. Moreover, we also showed that our method successfully learns to recover from OOD situations even when in-distribution states are difficult to visit through exploration.

Acknowledgments

This work was supported by the MOTIE (Ministry of Trade, Industry, and Energy) in Korea, under the Fostering Global Talents for Innovative Growth Program (P0008747) supervised by the Korea Institute for Advancement of Technology (KIAT), and in part by the National Research Foundation of Korea (NRF) through the Ministry of Science and ICT under Grant 2021R1A2C1093957. Cho was majoring in the Integrated Major in Smart City Global Convergence, Seoul National University when he was participating in this research. The Institute of Engineering Research at Seoul National University provided research facilities for this work.

References

[Abdar *et al.*, 2021] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

[Akkaya *et al.*, 2019] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[Amodei *et al.*, 2016] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[Eysenbach *et al.*, 2018] Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations*, 2018.

[Fujimoto *et al.*, 2019] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 2019.

[Gal and Ghahramani, 2016] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, 2016.

[Goan and Fookes, 2020] Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. In *Case Studies in Applied Bayesian Data Science*, pages 45–87. Springer International Publishing, 2020.

[Gu *et al.*, 2017] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, 2018.

[Heess *et al.*, 2017] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[Henaff *et al.*, 2019] Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *International Conference on Learning Representations*, 2019.

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

[Kang *et al.*, 2022] 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*, pages 10708–10733. PMLR, 2022.

[Kumar *et al.*, 2019] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[Kumar *et al.*, 2020] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020.

[Levine *et al.*, 2020] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[Li *et al.*, 2022] Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Dealing with the unknown: Pessimistic offline reinforcement learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1455–1464. PMLR, 08–11 Nov 2022.

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

[Sastry, 1999] Shankar Sastry. *Lyapunov Stability Theory*, pages 182–234. Springer New York, New York, NY, 1999.

[Schulman *et al.*, 2016] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[Sharma *et al.*, 2021] Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34:18474–18486, 2021.

[Sharma *et al.*, 2022a] Archit Sharma, Rehaan Ahmad, and Chelsea Finn. A state-distribution matching approach to non-episodic reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 19645–19657. PMLR, 17–23 Jul 2022.

[Sharma *et al.*, 2022b] Archit Sharma, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn. Autonomous reinforcement learning: Formalism and benchmarking. In *International Conference on Learning Representations*, 2022.

[Thananjeyan *et al.*, 2021] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.

[Wu *et al.*, 2019] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[Wu *et al.*, 2021] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11319–11328. PMLR, 18–24 Jul 2021.