Supplementary Material for 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@ufl.edu

1 Retraining procedure of SeRO

As SeRO expands the SAC, we consider a parameterized Q-function, $Q_\theta(s_t, a_t)$, and a policy, $\pi_\phi(a_t|s_t)$, where θ and ϕ are the parameters of each network. The retraining procedure can be divided into two steps: 1) the environment step, for collecting experience through interaction with the environment, and 2) the gradient step, for updating the policy and Q-function using the collected experience. Algorithm 1 describes the overall retraining procedure. When the state s_t is given, the state is encoded into a hidden vector h_t using encoder network e_ϕ (line 6), and the uncertainty distance d_t^u is calculated using Monte Carlo Dropout (MCD) and the mapping function g (line 8). Note that, during the retraining phase, the uncertainty distance is calculated using the fixed original policy $\pi_{\phi_{org}}$ and the mapping function saved in the original policy. Subsequently, h_t is entered into the network to generate the action distribution (line 9), and the action a_t is sampled from the action distribution (line 10). The action is then taken in the environment and the agent receives the next state s_{t+1} and the reward r^e from the environment (line 11). Finally, the uncertainty distance of s_{t+1} and auxiliary reward r^u are calculated (line 13, 14). In the gradient step, the policy and Q-function are updated by soft value iteration [Haarnoja *et al.*, 2018a] using experience sampled from the replay buffer $\mathcal{D}$ (line 17). The Q-function is updated using soft policy evaluation (line 18) and the objective for training Q_θ is as follows:

$$J(\theta) = \mathbb{E}_{(s_t,a_t,s_{t+1},r^e,r^u)\sim\mathcal{D},a_{t+1}\sim\pi}\left[\frac{1}{2}(Q_\theta(s_t,a_t)-y)^2\right], \text{ with } y = r_t + \gamma(Q_{\bar{\theta}}(s_{t+1},a_{t+1}) - \alpha\log\pi(a_{t+1}|s_{t+1})), \quad (1)$$

$$r_t = \begin{cases} r_t^e & \text{if } s_t \in \mathcal{S}_{in} \\ \lambda r_t^u & \text{else if } s_t \in \mathcal{S}_{OOD} = (\mathcal{S}_{in})^c, \end{cases} \quad (2)$$

where r^e is an environmental reward for the original tasks, λ is the weight coefficient, $\mathcal{S}_{in}$ and $\mathcal{S}_{OOD}$ correspond to in-distribution and out-of-distribution (OOD) state space, respectively, α is an entropy coefficient, and $\bar{\theta}$ is a moving average of the parameterized Q-function's weights, which stabilizes training [Mnih *et al.*, 2013]. This objective function is identical to that of SAC except that the reward function has been changed to consider the auxiliary reward in OOD states. The policy π_ϕ is updated using soft policy improvement to guarantee the result of an improvement in terms of its soft value (line 19). However, the objective is augmented by adding uncertainty-aware policy consolidation (UPC) loss $\mathcal{L}_{con}^{\pi_\phi}$ to prevent the agent from forgetting the original tasks during learning to return to the learned state distribution. The objective for training π_ϕ is as follows:

$$J(\phi) = \mathbb{E}_{(s_t,d_t^u)\sim\mathcal{D},a_t\sim\pi_\phi}\left[\alpha\log\pi_\phi(a_t|s_t) - Q_\theta(s_t,a_t) + \mathcal{L}_{con}^{\pi_\phi}\right]. \quad (3)$$

By minimizing the objectives, 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 UPC not to forget the original tasks. When the agent returns to the learned state distribution, the agent is retrained to solve the original tasks using an environmental reward while being regularized by UPC to generate an action similar to that of the original policy.

Algorithm 1 Retraining procedure of SeRO

Load the policy π_ϕ, and initialize Q-function Q_θ and replay buffer $\mathcal{D}$
Fix the original policy $\pi_{\phi_{org}} \leftarrow \pi_\phi$
Observe initial state $s_t \leftarrow s_0$
for each iteration **do**
  for each environment step **do**
    Encode state $h_t = e^h_\phi(s_t) = \text{dropout}(e_\phi(s_t))$
    with no gradient calculation:
      Compute uncertainty distance $d^u_t = g(\text{MCD}(e_{\phi_{org}}(s_t)))$
    Compute parameters for action distribution $\mu_{a_t} = \mu_\phi(h_t)$, $\sigma_{a_t} = \sigma_\phi(h_t)$
    Sample action $a_t \sim \mathcal{N}(\mu_{a_t}, \sigma_{a_t})$
    Execute action a_t in the environment, and observe reward r^e and next state s_{t+1}.
    with no gradient calculation:
      Compute uncertainty distance $d^u_{t+1} = g(\text{MCD}(e_{\phi_{org}}(s_{t+1})))$
    Compute auxiliary reward $r^u = -d^u_{t+1}$
    Save experience to replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, d^u_t, a_t, s_{t+1}, r^e, r^u)\}$
  for each gradient step **do**
    Sample experience from replay buffer $(s_t, d^u_t, a_t, s_{t+1}, r^e, r^u) \sim \mathcal{D}$
    Update the Q-function $\theta \leftarrow \theta - \lambda_Q \nabla_\theta J(\theta)$
    Update the policy $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J(\phi)$

2 Experimental details

2.1 Implementation details

In this subsection, we describe the implementation details of our method and the baseline (SAC). For all environments, the policies and Q-functions of both methods are implemented as a two-layer multi-layer perceptron (MLP) with 256 hidden units with a rectified linear unit (ReLU) activation for each layer. Both methods are updated using the Adam optimizer [Kingma and Ba, 2014]. Table 1 presents the hyperparameters used for the experiments. We determined the hyperparameters for the networks by using a coarse grid search, e.g., a policy learning rate over {0.00001, 0.00003, 0.0001, 0.0003, 0.001}, and a mini-batch size over {16, 32, 64, 128, 256}. We used automated entropy adjustment proposed in [Haarnoja *et al.*, 2018b] for an entropy coefficient α. Note that the same hyperparameters are applied to both methods.

HYPERPARAMETER	VALUE
DISCOUNT FACTOR	0.99
ADAM β_1	0.9
ADAM β_2	0.999
POLICY LEARNING RATE λ_π	0.0003
Q-FUNCTION LEARNING RATE λ_Q	0.0003
MINI-BATCH SIZE	256
REPLAY BUFFER SIZE	1M
NUMBER OF TRAINING STEPS	1M
NUMBER OF RETRAINING STEPS	1M
NUMBER OF EVALUATION EPISODES	5
EVALUATION INTERVAL	5000
DROPOUT RATE (SERO)	0.1
UNCERTAINTY THRESHOLD ϵ (SERO+OC)	0.4

Table 1: Hyperparameters.

2.2 Environmental details

HalfCheetah-v2

The goal of the HalfCheetah-v2 is to make a 2-dimensional cheetah robot run as fast as possible in the x-axis direction. For the experiments, we implemented HalfCheetahNormal-v2 for the training phase and HalfCheetahOOD-v2 for the retraining phase by modifying the termination condition and the initial state of the agent of the original environment, respectively. In HalfCheetahNormal-v2, the episode is terminated when the agent flips over. To implement such a termination we checked the

ENVIRONMENT	STATE / ACTION	TERMINATION CONDITION	MAXIMUM STEP
HALFCHEETAHNORMAL-V2	17 / 6	$\lvert\theta_{front}\rvert > \pi/2$	1000
HOPPERNORMAL-V2	11 / 3	$h < 0.7 \vee \lvert\theta_{top}\rvert > 0.2$	1000
WALKER2DNORMAL-V2	17 / 6	$(h < 0.9 \vee h > 2.0) \vee \lvert\theta_{top}\rvert > 0.3$	1000
ANTNORMAL-V2	27 / 8	$\max\left(\lvert\theta_{pitch}\rvert, \lvert\phi_{roll}\rvert\right) > \pi/2$	1000

Table 2: Training environments

ENVIRONMENT	STATE / ACTION	INITIAL STATE	MAXIMUM STEP	COEF. REWARD (λ)
HALFCHEETAHOOD-V2	17 / 6	$\theta_{front} = -11\pi/12$	1000	3.0
HOPPEROOD-V2	11 / 3	$h = 0.1 \wedge \theta_{top} = -5\pi/9$	1000	0.5
WALKER2DOOD-V2	17 / 6	$h = 0.1 \wedge \theta_{top} = \pi/2$	1000	1.0
ANTOOD-V2	27 / 8	$\theta_{pitch} = \pi$	1000	1.0

Table 3: Retraining environments

angle of the agent's front tip θ_{front} at every step and terminated the episode when it is out of the range of $[-\pi/2, \pi/2]$, which means that the agent starts to flip. In the HalfCheetahOOD-v2 environment, the agent is spawned upside down, in which θ_{front} equals $-11\pi/12$. Because the episode in the training phase is terminated when θ_{front} is below $-\pi/2$, the initial state of the agent in HalfCheetahOOD-v2 is never visited during the training phase, and therefore, this is an OOD state for the agent trained in HalfCheetahNormal-v2. To return to the learned state distribution, the agent should learn how to *turn its body over*.

Hopper-v2

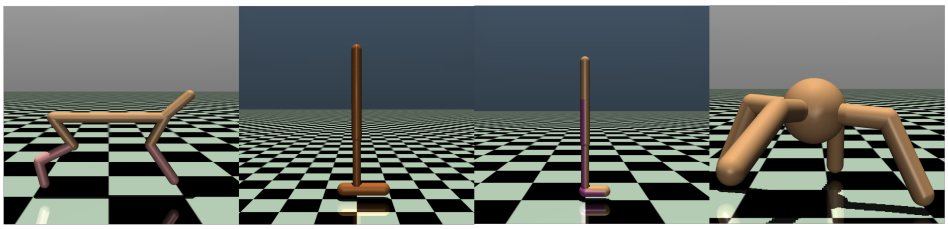
The goal of the Hopper-v2 is to make a 2-dimensional one-legged hopping robot hop as fast as possible in the x-axis direction. For the experiments, we used the original environment as HopperNormal-v2 for the training phase and implemented HopperOOD-v2 for the retraining phase by modifying the initial state of the agent of the original environment. In HopperNormal-v2, the episode is terminated when the height of the agent (z-coordinate of the top part) h is below 0.7 or when the angle of the agent's top part θ_{top} is out of the range of $[-0.2, 0.2]$, which means the agent starts to fall down. In the HopperOOD-v2 environment, the agent is spawned lying face-up on the floor, in which θ_{top} equals $-5\pi/9$ and h equals 0.1. Because the episode in the training phase is terminated when h is below 0.7 or θ_{top} is below -0.2, the initial state of HopperOOD-v2 is never visited during the training phase, and therefore, this is the OOD state for the agent trained in HopperNormal-v2. To return to the learned state distribution, the agent should learn how to *stand up*.

Walker2D-v2

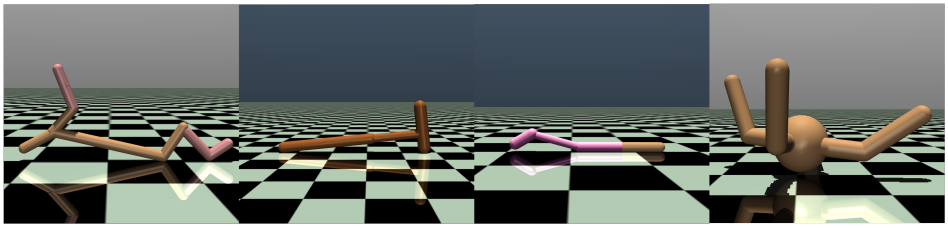
The goal of the Walker2D-v2 is to make a 2-dimensional two-legged walking robot run as fast as possible in the x-axis direction. For the experiments, we implemented Walker2DNormal-v2 for the training phase and Walker2DOOD-v2 for the retraining phase by modifying the termination condition and the initial state of the agent of the original environment, respectively. In Walker2DNormal-v2, the episode is terminated when the height of the agent (z-coordinate of the top part), h, is out of the range of $[0.9, 2.0]$, or the angle of the agent's top part θ_{top} is out of the range of $[-0.3, 0.3]$, which means the agent starts to fall down. In the Walker2DOOD-v2 environment, the agent is spawned lying face down on the floor, in which θ_{top} equals $\pi/2$ and h equals 0.1. Because the episode in the training phase is terminated when h is below 0.9 or θ_{top} is above 0.3, the initial state of Walker2DOOD-v2 is never visited during the training phase, and therefore, this is the OOD state for the agent trained in the Walker2DNormal-v2. In Walker2DOOD-v2, the agent is spawned lying face down on the floor. To return to the learned state distribution, the agent should learn how to *stand up*.

Ant-v2

The goal of the Ant-v2 is to make a 3-dimensional four-legged ant robot run as fast as possible in the x-axis direction. For the experiments, we implemented AntNormal-v2 for the training phase and AntOOD-v2 for the retraining phase by modifying the termination condition and the initial state of the agent of the original environment, respectively. In AntNormal-v2, the episode is terminated when the agent flips over. To implement such a termination condition, we calculate the pitch θ_{pitch} and the roll angles θ_{roll} of the agent's torso using quaternion included in the agent's state and terminate the episode if either of the two angle's absolute values exceeds $\pi/2$, which means that the agent starts to flip. In the AntOOD-v2 environment, the agent is spawned flipped, in which θ_{pitch} equals π. Because the episode in the training phase is terminated when θ_{pitch} is above $\pi/2$, the initial state of the agent in AntOOD-v2 is never visited during the training phase, and therefore, this is the OOD state for the agent trained in AntNormal-v2. However, because of the constraint on the maximum angle of the ant robot's leg joint, it

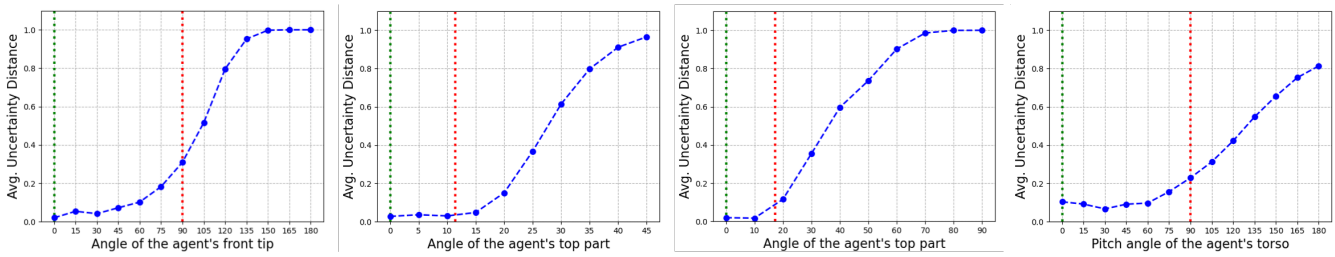


(a) Training environments



(b) Retraining environments

Figure 1: Training environments (top) and retraining environments (bottom). From left: HalfCheetah-v2, Hopper-v2, Walker2D-v2, and Ant-v2.



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

Figure 2: Uncertainty distance according to the state of the agent. Each marker in the blue plot represents the average uncertainty distance of the selected states. The green dotted vertical line corresponds to the initial state of training environments. The red dotted vertical line corresponds to the state where the episodes of the training environments are terminated, and therefore states existing on the right side of the red dotted line are OOD states which cannot be visited in the training phase.

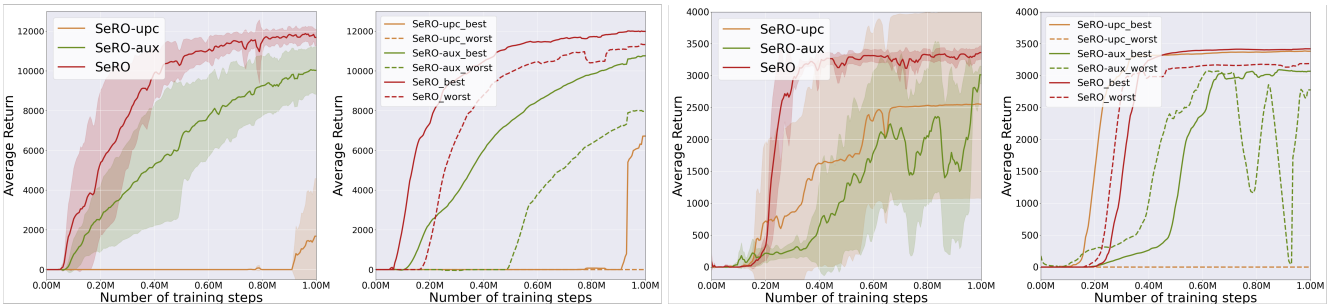
is almost impossible for the robot to turn its body over in the flipped position. To make this possible, we slightly relieved the constraint on the maximum angle of the leg joints in our environments. To return to learned state distribution, the agent should learn how to *turn its body over*.

3 Additional experiments

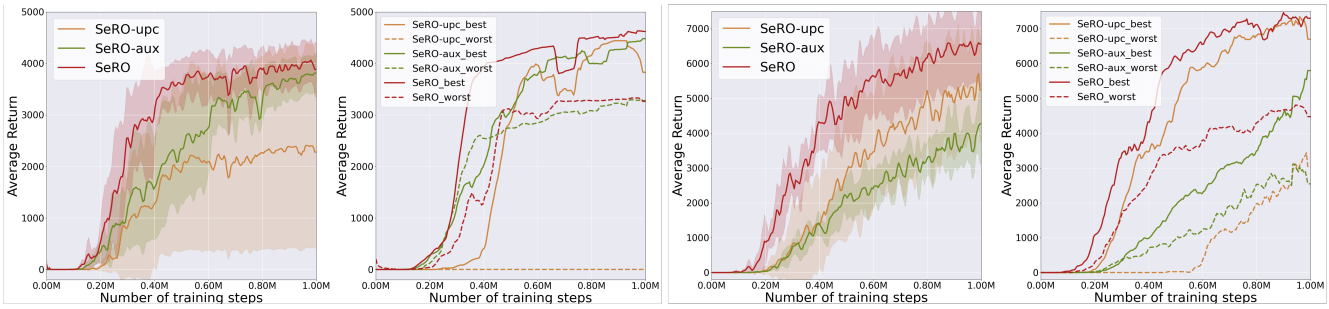
3.1 Analysis of the uncertainty distance

We investigated whether the proposed uncertainty distance could successfully represent the relative distance of the state from the learned state distribution. First, we trained the agent using SeRO in the training environments for 1 million steps, and we then measured the uncertainty distance of several different states using the trained agent. In HalfCheetah-v2, we selected the states according to the angle of the agent's front tip, because this is the criterion for determining whether the agent flips over, which distinguishes states that can be visited and states that cannot be visited in the training phase. For a similar reason, we selected the states according to the angle of the agent's top part in Hopper-v2 and Walker2D-v2, and the pitch angle of the agent's torso in Ant-v2. Note that all other elements are the same for the selected states.

Fig. 2 shows the uncertainty distance of the selected states. The uncertainty distance was measured 1000 times for each state, and the average value was plotted on a graph. In HalfCheetah-v2, the uncertainty distance was close to 0 for the in-distribution states that were close to the initial state of the agent in the training phase. As the angle of the agent's front tip increased in the direction of the criterion state where the episodes of the training environments are terminated, the uncertainty distance increased and approached 1. Similarly, in Hopper-v2, Walker2d-v2, and Ant-v2, the uncertainty distance was close to 0 for the in-distribution states that were close to the initial state, and it increased as the angle of the agent's top part and the pitch angle of



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



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

Figure 3: Learning curves for the retraining environments of SeRO–upc, SeRO–aux, and SeRO. They are calculated for five episodes of evaluation at every 5000 steps of training. The left part of each subfigure is computed over five random seeds, and the darker-colored lines and shaded areas denote the average returns and standard deviations. The right part of each subfigure represents the best and worst case of SeRO–upc, SeRO–aux, and SeRO among five random seeds.

the agent's torso increase in the direction of the criterion state, respectively. These results suggest that the proposed uncertainty distance approximately represented the relative distance of the state from the learned state distribution.

3.2 Analysis of each component of SeRO

To analyze the effect of each component of our method, we conducted an ablation study by evaluating SeRO–aux and SeRO–upc in the retraining phase, which is the proposed method that only uses auxiliary reward and UPC, respectively. Note that SeRO–upc receives *zero rewards* instead of auxiliary rewards in OOD states like SAC–zero. We first trained the SeRO agent in the training environments for 1 million steps and then retrained each method using the same trained agent. Fig. 3 shows the average learning curves of five random seeds and the worst and the best case among five random seeds for each retraining environment. As shown in the figure, even the worst case of SeRO–aux successfully learned how to return to the learned state distribution in all environments including HalfCheetahOOD-v2, where in-distribution states are difficult to visit through exploration; this means that SeRO–aux successfully learns to return to in-distribution states for all seeds. However, SeRO–aux showed a lower average return compared to SeRO for all environments, which suggests that SeRO–aux forgot original tasks during the retraining phase. In the case of SeRO–upc, we observed that the average return for the original tasks rapidly increases once it learned how to return to a learned state distribution, which can be thought of as the effect of the UPC loss that prevents the agent from forgetting the original tasks and consolidates the policy towards the original policy after the agent's returning to the learned state distribution. However, it failed to learn to return to a learned state distribution for one or more seeds for most environments. As shown in the figure, the worst case of SeRO–upc failed to learn to return to the learned state distribution in HalfCheetahOOD-v2, HopperOOD-v2, and Walker2DOOD-v2. To summarize the results, the agent cannot be guaranteed to return to the learned state distribution without the proposed auxiliary reward, and the UPC prevents the agent from forgetting the original task and accelerates the restoration of the original performance.

3.3 Further analysis of SAC–env

For further analysis, we qualitatively evaluated the SAC–env agent after the retraining phase, which receives environmental rewards for the original tasks in OOD states. Fig. 4 shows the qualitative results of the SAC–env agent in the retraining environments. In HalfCheetahOOD-v2, the agent should learn how to *turn its body over* to return to the learned state distribution. However, the agent moves in the x-axis direction without turning its body. In HopperOOD-v2, the agent first rolls forward and then moves in the x-axis direction while lying face down, instead of *standing up* to return to the learned state distribution. Sim-

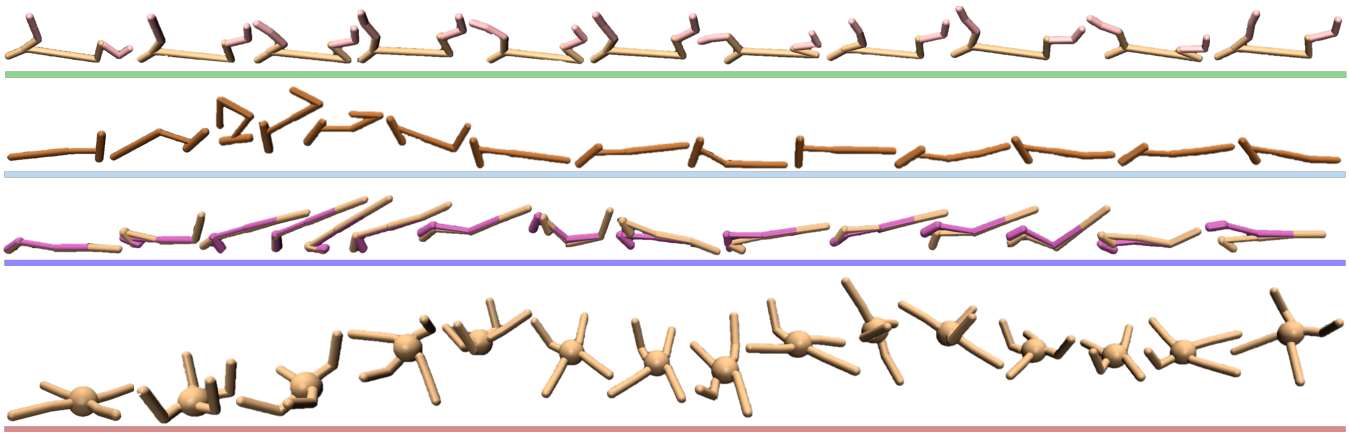


Figure 4: Qualitative analysis of SAC–env after the retraining phase.

ilarly, the agent moves in the x-axis direction while lying face down in Walker2DOOD-v2. In AntOOD-v2, the agent moves forward without *turning its body*, which suggests that the agent fails to learn how to return to the learned state distribution. These can be thought of as the case that the agent falls into a local optimum. This happens because the agent received an environmental reward for the original tasks, which guided the agent to move in the x-axis direction even in OOD states. This situation does not happen in SAC–zero which receives zero rewards in OOO states, because the agent receives zero rewards even if it moves in the x-axis direction in OOD states. The results suggest that using environmental rewards for the original tasks in OOD states does not help the agent to return to a learned state distribution as the agent can fall into a local optimum, therefore, the reward for recovery from OOD situations should be defined separately.

References

[Haarnoja *et al.*, 2018a] 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.

[Haarnoja *et al.*, 2018b] 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.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.