A Appendix: Algorithms

Algorithms 1 and 2 contain the pseudo-code of action replacement mechanism and complete optimization procedure in MBHI, respectively.

Algorithm 1 Action Replacement Mechanism in MBHI

Inputs: Initial state s_0, MPC horizon H, number of CEM iterations T, number of gradient steps I, transition ensemble $\hat{T}_\xi$, reward ensemble $\hat{R}_\phi$, Block ensemble $\hat{B}_\mu$
for $t = 0$ to $T-1$ **do:**
  Sample K action sequences $\{a^{(t)}_{0:H-1}\}^K_{k=1} \sim \text{CEM}(\cdot)$
  for $i = 1$ to I **do:**
    Calculate imaginary rollouts $\{s^{(t)}_{1:H}\}_k = \hat{T}_\xi(\{a^{(t)}_{0:H-1}\}_k, s_0)$
    Calculate MPC returns for each action sequence by Eq (11)
    Update the action sequence $\{a^{(t)}_{0:H-1}\}^K_{k=1}$ via gradient ascent
  end
  Recalculate imaginary rollouts and MPC returns
  Update CEM$(\cdot)$ distribution with top N action sequences sorted by MPC returns
end
Output: The first action a^*_0 from optimal action sequence $a^*_{0:H-1}$

Algorithm 2 Actual algorithm of MBHI

Inputs: Number of initial samples N_0, interacting step T, length of safety detection H, safe threshold ε, scaling factors c_l and c_h, Block ensemble $\hat{B}_\mu$
Initialize: random initialize policy π_θ, value ensemble $\hat{Q}_\varphi$, transition ensemble $\hat{T}_\xi$, reward ensemble $\hat{R}_\phi$, termination model $\hat{D}_\psi$
Collect N_0 samples with random policy, initialize replay buffer $D = \{(s_t, a_t, r_t, s_{t+1}, d_t)\}^{N_0}_{t=1}$ and pre-train $\{\hat{T}_\xi, \hat{R}_\phi, \hat{D}_\psi\}$ N_0 times
for each epoch **do:**
  Get initial state s_0
  for $t = 1$ to T **do:**
    Check the safety p_t of the current policy π_θ with Eq (9) in imaginary rollouts of depth H starting from state s_t
    if $p_t \geq \varepsilon$ **then** Compute action a_t using Algorithm 1
    else Compute $a_t \sim \pi_\theta(a_t|s_t)$ with policy network
    Execute a_t to the real world and add transition to D
    Train the dynamics model $\{\hat{T}_\xi, \hat{R}_\phi, \hat{D}_\psi\}$ on buffer D with Eq (2)
    Update π_θ and $\hat{Q}_\varphi$ with model-based learning methods on buffer D with Eq (13)
  end
end

B Appendix: Experiment Setup

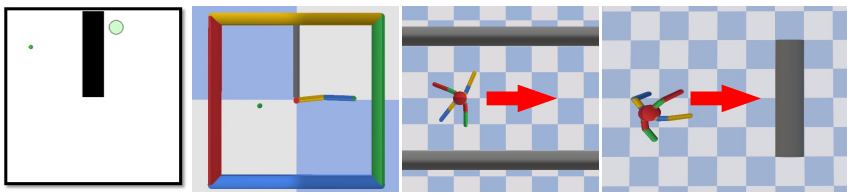


Figure 1: Visualization of the experimental environments. From left to right: PuckWorld, Reacher, Ant-Limit and Ant-Block.

- **PuckWorld-L:** A randomly generated puck has to reach a given random target in two-dimensional space. The maximum speed of the puck is 0.025, and the maximum acceleration is 0.002. Therefore, it needs to decelerate multiple steps in advance before the puck can stop. The constraint is that the puck cannot enter the black area in the middle. It's a catastrophe if the puck enters the black area.
- **PuckWorld-H:** Same as PuckWorld-L, except that the maximum acceleration is 0.025. It means that the puck can be stopped immediately. The catastrophe is the same as PuckWorld-L.
- **Reacher:** A two-link arm has to reach a given random target in two-dimensional space. The arm is not allowed to touch the vertical bar. It's a catastrophe if the arm collides with the vertical obstacle.
- **Ant-Limit:** An ant with four legs needs to run as fast as possible along the positive X-axis in three-dimensional space. The ant can only move between two parallel boundaries. It's a catastrophe if the ant's centroid crosses the boundary.
- **Ant-Block:** The basic environment is the same as Ant-Limit. The difference is that a vertical obstacle blocks the motion path of the ant. It's a catastrophe if the ant's centroid collides with the obstacle.

In PuckWorld and Reacher, 100 reward is given when the agent successfully reaching the target area, -100 when causing a catastrophe, and 0 in the rest of time. In Ant, -100 reward is given when the agent causing a catastrophe, and the reward is the same as the original environment Ant at other times. The episode will be terminated immediately if the agent violates the safety constraints.

C Appendix: Detailed Analysis of The Human Oversight Phase

In all experiments, only the obstacle was labeled as unsafe, which greatly reduces the workload of human labors (no need to deduce) and can quickly evaluate the proposed method. Since the input of the Blocker is (s_t, a_t, s_{t+1}) in practice, the transition can also be intercepted and labeled in advance. In this case, the labeled unsafe transition means that once it is visited, the agent cannot be rescued.

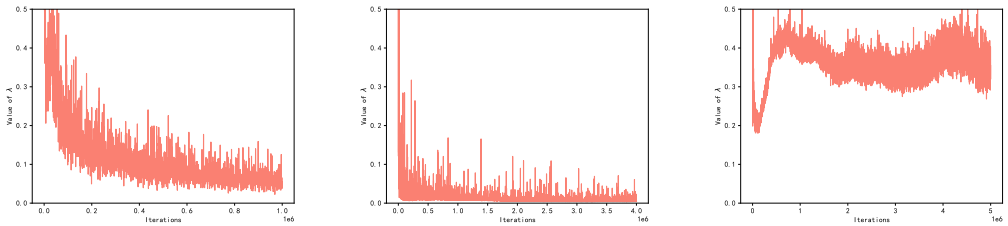
The human time-cost can be roughly formulated as follows [12]:

$$C = t_{human} \times N_{all}, \tag{1}$$

where C is the total time-cost of the oversight phase, t_{human} is the time-cost per human label, and N_{all} is the number of training samples. In our experiments, since the judgment of catastrophe is very simple, t_{human} is about 0.1 seconds. Therefore, the main way to reduce C is to reduce N_{all}. We fixed this problem through two methods. One is to shield the negative rewards for catastrophes from the environment to prevent the agent from quickly learning to avoid obstacles. The second is to initialize the agent near the obstacle with certain probability. Beacuse the catastrophe is not complicated, the Human Oversight phase of PuckWorld and Reacher lasted for about 3 hours, and Ant lasted for abount 5 hours. Finaly, we believe that the available historical logs containing catastrophes will be able to effectively alleviate the problem of time-cost.

D Appendix: Visualization of The Parameter λ

The value of parameter λ defines the relative weighting of exploitation and exploration. With $\lambda = 0$ that training a policy only maximizes the expected return, and $\lambda = 1$ traing a policy that only be encouraged to visit unfamiliar states. Fig 2 shows the plot of λ during policy training. In PuckWorld and Reacher, with the convergence of the policy and the full exploration of the environment, the value of λ gradually decreases to zero. While in Ant, λ does not converge to a small value, beacuse the space in Ant is infinite, and Ant encourages the agent to walk as far as possible, instead of completing the task as soon as possible in a limited space. Therefore, during the training process, the agent in Ant is constantly visiting new states. We notice that the λ in Ant has a significant decrease in the initial of training. This is because the agent has not learned to move in the positive direction of X-axis, and has been walking near the origin, resulting in full exploration of this area.



(a) (b) (c)

Figure 2: Visualization of the active learning parameter λ in Eq (13) during training. (a) PuckWorld, (b) Reacher, (c) Ant. The x-axis denotes the environment step.

E Appendix: Ablations

E.1 Scaling Factors

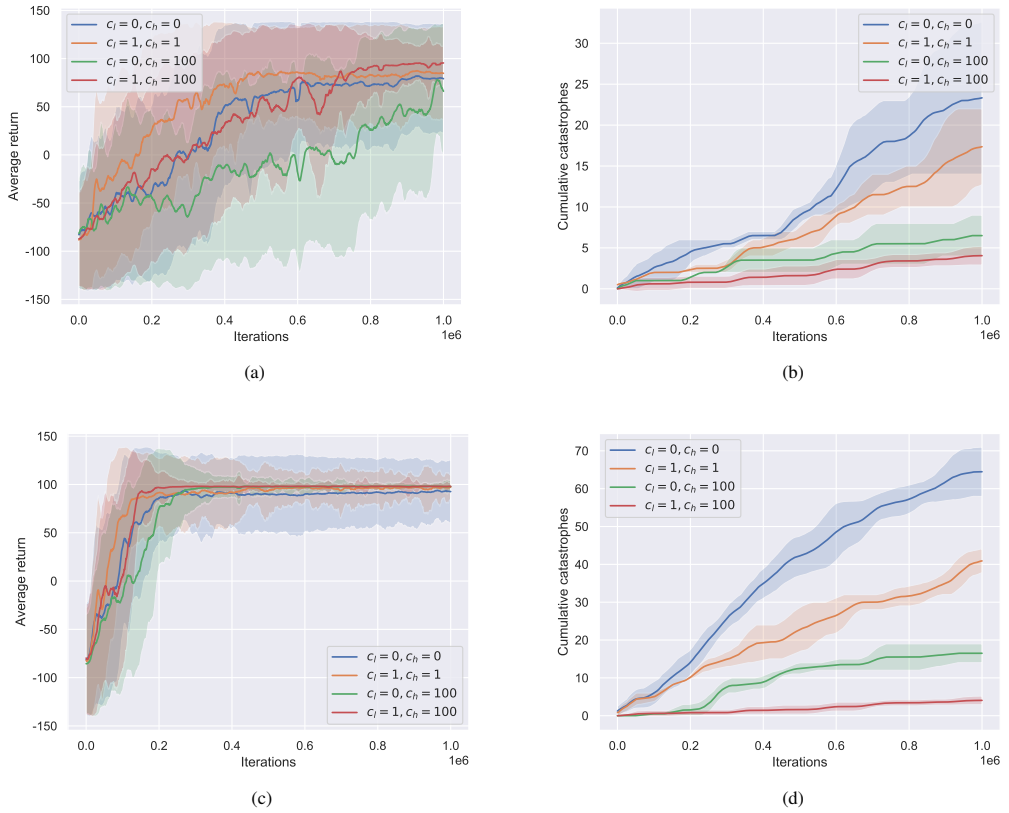


Figure 3: Evaluation of how scaling factors c_l and c_h affect MBHI performance in PuckWorld-L and PuckWorld-H. (a) and (b) are the performance and cumulative catastrophe of environment PuckWorld-L. (c) and (d) are the learning curves and cumulative catastrophe of environment PuckWorld-H.

An ablation study is performed on scaling factors c_l and c_h in Fig 3. $(c_l = 0, c_h = 0)$ means that when the agent's action are intercepted by the Blocker, it cannot obtain a negative reward to perceive potential dangers. When the model is evaluated in the PuckWorld, ideally, there is about 25% probability that the agent will pass through the dangerous area, that is, the expectation of the cumulative reward is $\mathbb{E}[G] = \frac{3}{4} \times r_{target} + \frac{1}{4} \times r_{catastrophe} = 50$. But in practice, due to the Blocker's prediction error, the agent will visit the dangerous area during the training process and thus have the opportunity to learn about catastrophes. As shown in Fig 3a and Fig 3c, the learned

policy's performance is better than 50, but the standard deviation of the cumulative reward is larger than other ablations, and it cannot converge to the optimal policy.

When the scaling factor c_l or c_h is non-zero, the knowledge of catastrophes is introduced to the agent by means of intrinsic reward, and helps agent avoid dangerous areas. Furthermore, in sparse reward environment, this kind of intrinsic reward can also accelerate convergence. c_h controls the penalties of the area in the immediate vicinity of the catastrophe. We make c_h much larger than c_l, which can strongly correct the behavior of the agent near the unsafe region. As can be seen in Fig 3, larger c_h can further reduce the number of cumulative catastrophe in the training phase. However, the larger c_h also increases the complexity of the environment and makes policy learning more difficult.

E.2 Safety Bound

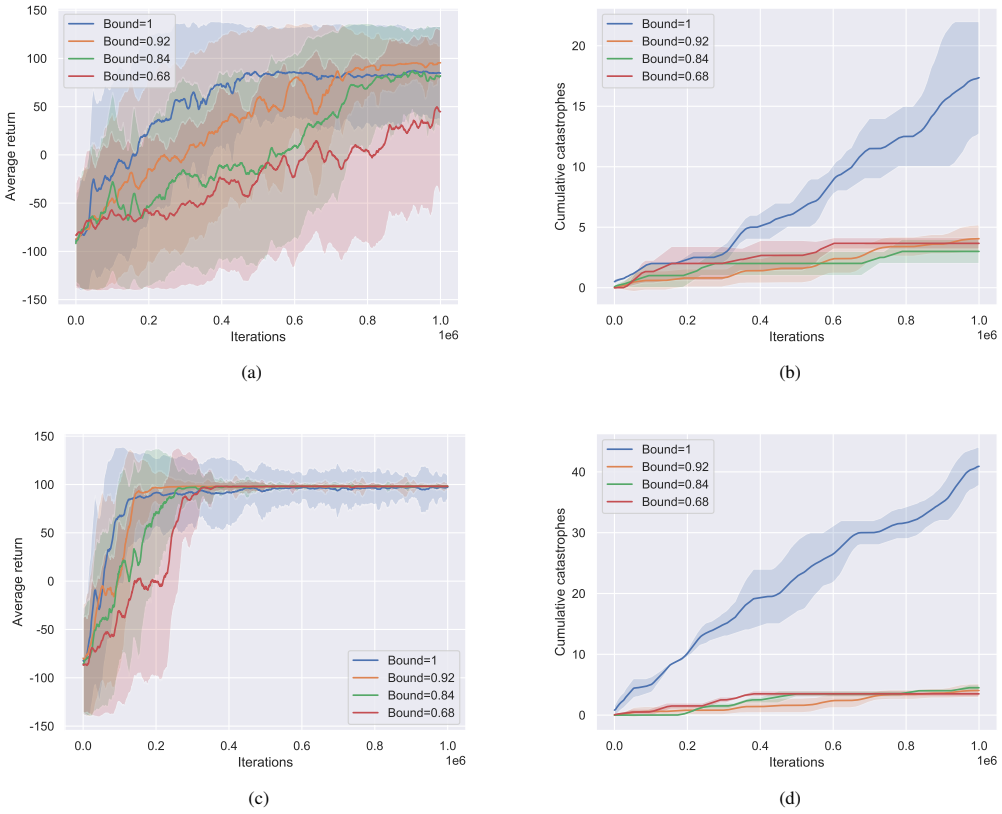


Figure 4: Evaluation of how safety bound affect MBHI performance in PuckWorld-L and PuckWorld-H. (a) and (b) are the performance and cumulative catastrophe of environment PuckWorld-L. (c) and (d) are the learning curves and cumulative catastrophe of environment PuckWorld-H.

We also conduct ablation studies about safety bound *Bound*, which are demonstrated together in Fig 4. $Bound = 1$ is equivalent to $(c_l = 1, c_h = 1)$ described in Sec E.1. The introduction of the safety-aware intrinsic reward improves the sampling efficiency. However, due to insufficient punishment near the catastrophe, the agent will still occasionally try to cause catastrophes. From Fig 4c we can clearly find that the variance of $Bound = 1$ is much larger than other ablations, which means that it has not learned the optimal safety policy.

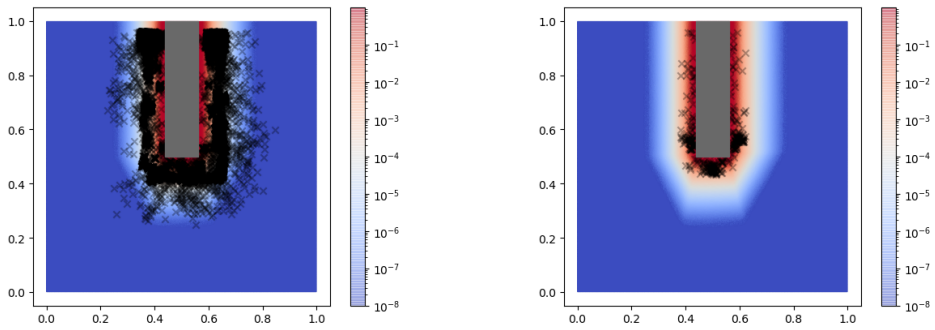
As shown in Fig 4a and 4c, as the value of *Bound* decreases, the speed of policy learning becomes slower and slower. This is because smaller *Bound* means the larger subjective disaster area (i.e. areas with large negative intrinsic reward). This will result in a reduction in the feasible state space of the agent, and increase the complexity of the task. As a consequence, the agent has to learn a more conservative policy. Moreover, Fig 4b and Fig 4d indicate that too small $Bound$ can not further reduce the number of catastrophe, but will greatly affect the speed of policy convergence.

F Appendix: Discussion of Safety-critical Components

In many natural scenarios, security constraints are subjective and difficult to be formulated explicitly. In addition, sub-optimal trajectories or demonstrations are not available, and humans can only judge whether the current state is safe or not. In this case, human-in-the-loop seems to be the only way to guarantee the safety of RL systems during training [12]. We studied whether the human-imitator could avoid both "local" and "non-local" catastrophes when enhanced with environmental dynamics approximators. As shown below, we summarize the key components to ensure safety:

- **Look before leap:** Imaging in the learning dynamics helps the agent to correct the catastrophic policy in advance. As shown in Fig 4, When the momentum of the agent in a certain direction cannot be decayed to zero immediately, it is necessary to block the dangerous action in advance. This is like the braking distance of a car. We must depress, hold down the brake pedal and turn the wheel in advance to avoid collisions.
- **Separated replay buffer:** Neural networks are always suffer from catastrophic forgetting [33], especially in reinforcement learning, where old samples are constantly replaced by new ones. Since there are very few unsafe samples compared with safe samples, it is easy to be forgotten. Splitting the replay buffer into safe and unsafe categories can effectively alleviate the problem of catastrophic forgetting by over-sampling unsafe samples. Similar to PER [34], importance-sampling method needs to be used to compensate the bias introduced by this prioritization.
- **Safety-aware intrinsic reward:** As presented in Appendix E, using the predicted catastrophe probability as an intrinsic reward can make the agent perceive the dangerous area during training. Furthermore, Larger intrinsic reward leads to a more secure and conservative policy, but makes the environment more complicated and the policy is more difficult to converge.
- **Model-based RL:** Model-based methods can significantly improve the sampling efficiency, so as to learn the safe policy faster. Moreover, the reward function and termination function in dynamics can also model the catastrophe from unsafe samples, thereby further improve the safety during policy learning. Besides, different from [7, 14, 18], approximating the environmental dynamics make the catastrophe prediction network only need to focus on the safety of the current state-action pair, rather than a sequence, making it easier to train.

G Appendix: Visualization of The Agent Motion



(a) (b)

Figure 5: Visualization of state points intercepted by the Blocker. (a) PuckWorld-L, (b) PuckWorld-H. The gray rectangle indicates the obstacle, which is labeled as unsafe when training the Blocker. Red corresponds to a high probability of catastrophe predicted by the Blocker (normalized on a log scale), while blue to a lower.

The visualization of interception points in environments PuckWorld-L and PuckWorld-H is shown in Fig 5. The interception region in PuckWorld-L is much larger than that in PuckWorld-H. In

PuckWorld-L, due to the low acceleration and the long braking-distance of the agent, it is necessary to intercept dangerous actions multiple steps in advance. But, it is enough to replace dangerous actions one step ahead in PuckWorld-H.

As shown in Fig 5a, the interception points on the outside are significantly less than the areas close to the catastrophe. This is because the MBHI's interception strategy does not only depend on the current state, but also considers the current behavioral policy. Therefore, in the same state, with different behavioral polices, MBHI's interception results may also different. That is, if the agent's policy can avoid catastrophes by itself in the future, MBHI will not intercept it. In Fig 5, we also observe that there are more interception points at the lower end of the obstacle, mainly because it is more difficult to learn to bypass the obstacle than to avoid it.

The visualization of motion sequences is shown in Figure 6, MBHI executes the current behavioral policy in the imagination to decide whether to block the action.

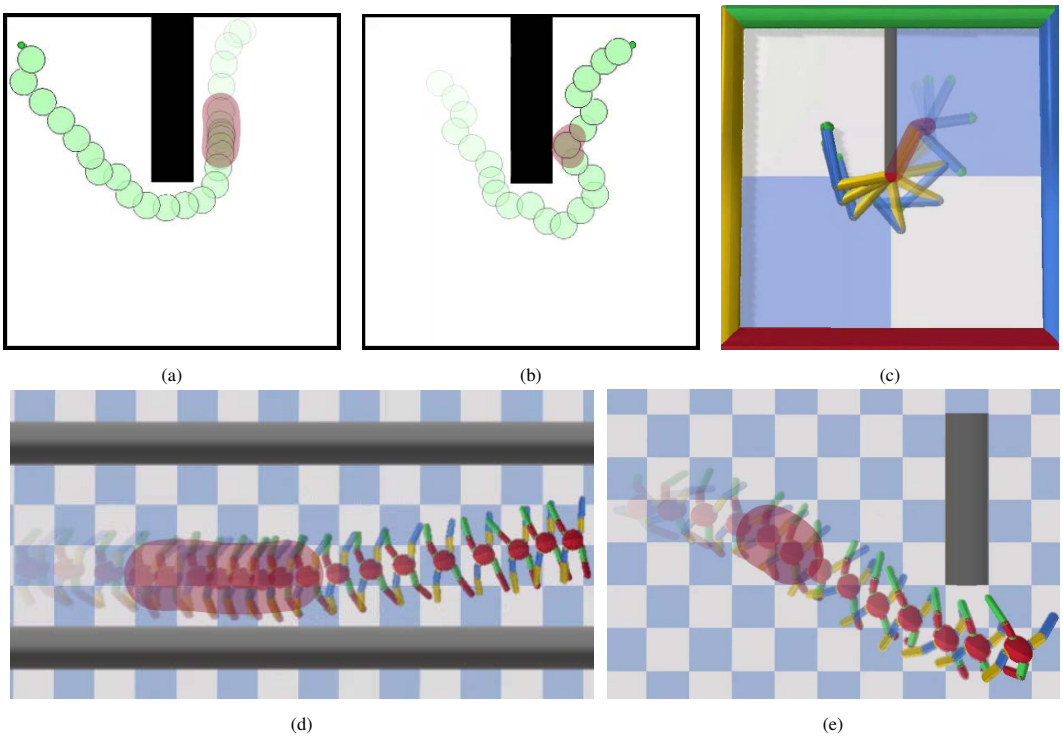


Figure 6: The Visualization of motion sequences in experiments. (a) PuckWorld-L, (b) PuckWorld-H, (c) Reacher, (d) Ant-Limit, (e) Ant-Block. The red area indicates that the action predicted by the agent's behavioral policy is blocked and replaced by the MPC controller. The final display is the safe trajectory monitored by the Blocker and corrected by MPC.

H Appendix: Implementation Details

In this work, we follow the same network structure and policy learning method as STEVE. All models are fully connected neural networks optimized by Adam with learning rate of $3e-4$. The value network, policy network, reward model, termination model and Blocker each has 4 layers of size 128. The transition model has 8 layers of size 512.

The replay buffer size is $1e6$. The first $1e5$ frames are sampled by the agent interacting with the environment through random actions. After that, the dynamics model is pre-trained $1e5$ times, and then 1 model update and 1 policy update are performed for each frame sampled from the environment. The mini-batch size of all models is 512. We use soft update for the target network instead of hard update used in STEVE. Note that the replay buffer is divided into two parts, safe and unsafe, and sampled from them in equal proportions. This prioritization can introduce bias, which we correct

with importance sampling method. When correcting the dangerous action by MPC controller, the Gradient + CEM method uses 5 iterations plus 5 gradient steps to sample 128 candidate actions.

The policy network and value network of DDPG and PPO are also fully connected neural networks with 4 layers of size 128. The hyper-parameters of DDPG are the same as STEVE, but there is no model ensemble and dynamics. For the hyper-parameters of PPO, the clipping parameter is set to 0.2, the GAE Lambda is selected as 0.95, the learning rate is $3e-4$, and the mini-batch size is 512.

Table 1: Experiment hyper-parameters.

Hyperparameter Name	PuckWorld-Low	PuckWorld-High	Reacher	Ant-Limit	Ant-Block
Safety detection length	10	1	8	10	10
MPC horizon	10	10	10	10	10
Safety weight	1	1	1	1	1
Active learning coefficient	5e4	10	1e3	10	10
Safe threshold	0.96	0.96	0.96	0.99	0.99
Intrinsic reward bound	0.92	0.92	0.92	0.95	0.95
Scaling factor c_l	1	1	1	0.5	0.5
Scaling factor c_h	100	100	100	10	10