EPISODIC CONTROL-BASED ADVERSARIAL POLICY LEARNING IN TWO-PLAYER COMPETITIVE GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Training adversarial agents to attack neural network policies has proven to be both effective and practical. However, we observe that existing methods can be further enhanced by distinguishing between states leading to win or lose and encouraging the policy training to prioritize winning states. In this paper, we address this gap by introducing an episodic control-based approach for adversarial policy training. Our method extracts the historical evaluations for states from historical experiences with an episodic memory, and then incorporating these evaluations into the rewards to improve the adversarial policy optimization. We evaluate our approach using two-player competitive games in MuJoCo simulation environments, demonstrating that our method establishes the most promising attack performance and defense difficulty against the victims among the existing adversarial policy training techniques.

1 INTRODUCTION

It has been proved that deep reinforcement learning (DRL) policies are vulnerable to adversarial attacks (Huang et al., 2017; Kos & Song, 2017). Most existing attacks on DRL policies are executed by searching the adversarial examples and manipulating the environment (Huang et al., 2017; Kos & Song, 2017; Nguyen & Reddi, 2019). However, such adversarial examples may not be applicable in the real world (Gleave et al., 2020). Recently, training adversarial agents as attackers to DRL policies in two-player games has been proven effective and practical (Gleave et al., 2020; Wu et al., 2021; Guo et al., 2021; Bui et al., 2022). These kind of attacks first reduce the two-player environments to single-player environments by fixing the victim agents, and then train the other agent to be an adversarial agent which can be trained by conventional single-agent policy training method. Known as adversarial policy training, these attacks generate natural observations that are adversarial to the victim agents, achieving significant results.

While the aforementioned adversarial policy training methods have proven effective, there is still room to improve adversarial training by exploring how states influences game outcomes. We believe that utilizing the adversarial agent's historical experiences can enable policy training to distinguish between states that lead to win or lose and facilitate the learning of winning states through reward adjustment, leading to an improvement in the effectiveness of adversarial policy training. In this paper, we introduce a novel adversarial policy training approach that leverages the analysis of information from past episodes to assess game states and adjust rewards, thereby assisting the adversarial agent in achieving better performance.

Technically, we propose an episodic control-based adversarial policy training method for two-player competitive games. Inspired by previous works on improving the performance of DRL using episodic control (Blundell et al., 2016; Pritzel et al., 2017; Li et al., 2023), our method develops a neural network-based episodic memory to store historical experiences. We utilize this episodic memory to generate historical evaluations, which are used for reward revision. In our experiments, we evaluate our method on two-player competitive games in MuJoCo domains (Todorov et al., 2012) and compare it with state-of-the-art adversarial policy training approaches (Gleave et al., 2020; Guo et al., 2021; Wu et al., 2021). Our experimental results show that our method establishes the most promising attack performance and defense difficulty.

In summary, this paper presents four contributions. First, we propose an episodic control-based adversarial policy learning method for two-player competitive games. Second, we introduce a method for generating state evaluation from historical experiences, implemented through our proposed episodic memory. Third, we propose a reward revision approach to incorporate the historical evaluations into the rewards. Fourth, our work demonstrates that by identifying and highlighting the winning states with historical experiences, adversarial agents can achieve higher winning rates against fixed victim agents and possess the capability to integrate multiple winning strategies to defeat the victims.

2 RELATED WORK

2.1 ADVERSARIAL ATTACKS ON DRL POLICIES

Previous attacks against DRL policies mainly focus on manipulating the environment to fail the victim agents. One type of attack focuses on perturbing the victim's observations, forcing its policy network to output sub-optimal actions, and thus failed the victim agent (Russo & Proutiere, 2019; Sun et al., 2020; Zhang et al., 2021; Madry et al., 2018; Pattanaik et al., 2018; Pan et al., 2022; Zhao et al., 2020). Another kind of attack directly perturbs the trajectory of the victim, specifically actions the victim agent takes (Lee et al., 2020; Pan et al., 2022) or the rewards it receives (Ma et al., 2019; Yang et al., 2019; Lykouris et al., 2021) to effectively attack the victim. However, the above attacks are argued to be unrealistic since the real-world environment can not be manipulated (Gleave et al., 2020; Guo et al., 2021; Wu et al., 2021).

Unlike the above attacks, to simulate the real-world scenarios, Gleave *et al.* has successfully trained adversarial agents by PPO algorithm (Schulman et al., 2017) in two-player competitive games under a strict zero-sum assumption and demonstrated the effectiveness of training adversarial agents against fixed black-box victims (Gleave et al., 2020). Wu *et al.* further improved the attack performance by exploring the minimal observation differences of the shared environment to maximize deviations of the victim actions (Wu et al., 2021). Guo *et al.* relaxed the zero-sum assumptions of previous works and demonstrated that such attack could be achieved by maximizing the gap between the adversary and victim rewards which are approximated by observations of the adversarial agent (Guo et al., 2021). On the other hand, Bui *et al.* adopted imitators of the victim policies learned by imitation learning algorithms (*e.g.*, GAIL (Ho & Ermon, 2016)) to roll out the victim actions for the attacker and reached better performances (Bui et al., 2022). However, this attack is based on the specification that the victim's actions are visible and accessible to the imitators.

This paper adopts the same setting as (Gleave et al., 2020; Guo et al., 2021; Wu et al., 2021), wherein we have control solely over the adversarial agent and treat the victim agent as a black box, rendering its observations, actions, and rewards inaccessible. Meanwhile, unlike (Gleave et al., 2020; Guo et al., 2021; Wu et al., 2021), we concentrate on leveraging the historical experiences from adversarial agents to emphasize the winning states to improve the adversarial policy training.

2.2 MODEL-FREE EPISODIC CONTROL

Episodic Control (Lengyel & Dayan, 2007) has demonstrated the effectiveness of utilizing past experiences to address sample inefficiency in various tasks, such as multi-agent tasks (Zheng et al., 2021), model-based reinforcement learning (Le et al., 2021), and continuous control (Zhang et al., 2019; Kuznetsov & Filchenkov, 2021). Previous works primarily adopt a tabular episodic memory to save experiences of past scenes, leveraging the information gained during exploration and retrieving past experiences of similar scenes to expedite policy optimization (Blundell et al., 2016). This memory uses the state as a key and a measurement of the state (*e.g.*, the Q-value of the state) as the value, storing these key-value pairs. Then, a distance-based analysis (*e.g.*, KNN) is adopted to retrieve a summary statistic of similar states from the episodic memory, and the retrieved statistic can be used to guide the training process (Hansen et al., 2018).

In this paper, we adopts the episodic control to generate historical evaluations to measure the quality of the states. Unlike the episodic memory proposed by previous works, our method proposes a neural network-based episodic memory which uses state sequences as the basis of historical experiences analysis and predict historical evaluations for states based on the learned experiences.

3 Methodology

In this work, we propose an adversarial training approach for training adversarial agents. Our method utilizes the conventional adversarial policy training framework and improves the rewards used for training. The workflow of our approach is shown in Figure 1. First, the adversarial agent interacts with the environment and gathers information (states, actions, rewards) from the episodes. This information is adjusted using our proposed reward revision method, and then saved in the experience storage (*e.g.*, replay buffer). We calculate the objective function based on the revised rewards sampled from the experience storage and update the agent's policy with the objective function. In the following, we mainly elaborate on our reward revision method based on historical experiences analysis.

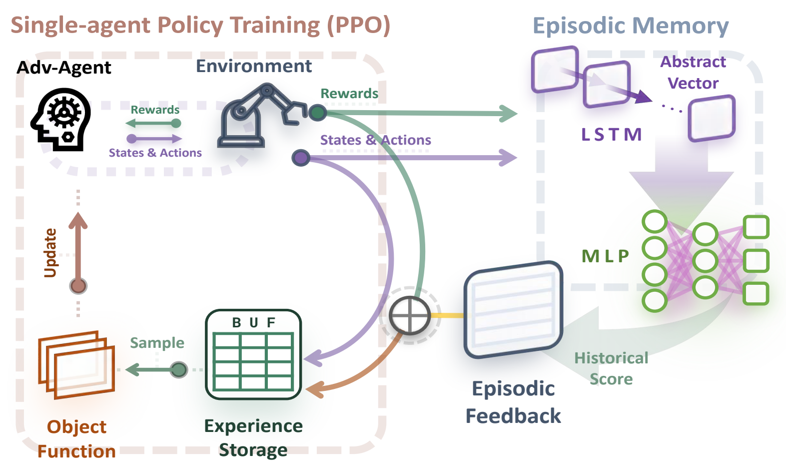


Figure 1: The workflow of our adversarial policy training.

3.1 Adversary in Two-player Markov Game Environment

A two-player Markov game environment can be modeled as $E = (S, (A_\alpha, A_\nu), T, (R_\alpha, R_\nu))$. Here, we use α and ν to represent the adversary and victim respectively. S represents a state set, and both A_α and A_ν are action sets. T denotes a joint state transition function $T : S \times A_\alpha \times A_\nu \to \Delta(S)$, where $\Delta(S)$ is a probability distribution on S. The reward function $R_i : S \times A_\alpha \times A_\nu \times S \to \mathbb{R}$ depends on the current state, actions taken by both agents and the next state.

Gleave *et al.* discovered that by fixing the victim agents, the two-player competitive games can be reduced to single-player games. In such environment, the other agent can be trained as an adversarial agent with conventional single-agent policy training methods (*e.g.*, PPO) to attack the victim (Gleave et al., 2020). The training for the adversarial agent to defeat the fixed victim agent is called **adversarial policy training**. Under this setting, the two-player game reduces to a single-player MDP: $E_\alpha = (S, A_\alpha, T_\alpha, R'_\alpha)$ as the victim policy can be treated as a part of the environment. S becomes the state set of the adversary and the state transition function and reward function change to $T_\alpha : S \times A_\alpha \times S \to \Delta(S)$ and $R'_\alpha : S \times A_\alpha \times S \to \mathbb{R}$.

Our threat model We use the same setting as Gleave *et al.* and assumes that our threat model has control over the adversarial agent and black-box access to the information of the victim agent. The adversarial agent can only interact with the environment, which is the common practice in adversarial policy training.

3.2 Reward Revision Based on Historical Experiences

The reward R'_α obtained by the adversarial agent only includes the evaluations from a single game and does not contain evaluations from the historical experiences. To enhance the adversarial policy training, our key insight is to integrate state evaluations from historical experiences into the rewards, thereby providing the adversarial policy with more comprehensive rewards for learning.

As a two-player Markov game is not deterministic, past episodes starting from the same state may include different state sequences. Therefore, we use state sequences, referred to as patterns, as the basis for performance evaluation in past episodes. Our choice of utilizing state sequence for performance evaluation aligns with existing research (Sutton & Barto, 2018; Li et al., 2023), which has proved that failures in adversarial games are often the result of a series of poor decisions rather than isolated states. Based on the qualification of these patterns from historical experiences, we can assign higher rewards to states that lead to good patterns and lower rewards to states that lead to bad patterns, thereby integrating evaluations from historical experiences into the rewards.

3.2.1 Evaluation of Pattern Performance

As mentioned above, a pattern refers to a sequence of states within an episode. For a given episode e, a *k-step pattern* starting from time step t, denoted as p_t, refers to k consecutive states in e, i.e., $p_t = s_t, \dots, s_{t+k-1}$. For example, $p_1 = s_1, \dots, s_k$ exemplifies a k-step pattern from the first state s_1. Notice that a state can be considered a special case of a pattern, specifically a 1-step pattern.

To evaluate the past performance of the patterns, we propose that a pattern can be considered **high-performing** if the past episodes containing this pattern result in more wins than losses. Conversely, a pattern can be considered **poor-performing** if the past episodes containing it result in more losses than wins. Based on this idea, we introduce a **historical score** for each pattern to quantify its past performance, defined as the average cumulative reward received in past episodes that include that pattern. The cumulative reward for an episode, which quantifies the adversarial agent's performance, tends to be higher in episodes where the agent wins than in those it loses. As the performance of adversarial agents improves, these cumulative rewards increase. Therefore, using the average cumulative reward of episodes including the pattern effectively reflects its past performance. The historical score $h\_score(p_t)$ can be calculated as:

$$h\_score(p_t) = M(p_t), \tag{1}$$

where M is an episodic memory we proposed to collect and analyze patterns in past episodes. In Section 3.3.1, we will further explain how we implement the episodic memory.

3.2.2 Conditional Reward Revision

Based on the historical scores of patterns, we incorporate the historical evaluations into rewards by reward revision. If the historical score of a pattern falls below the average cumulative rewards of all past episodes, we cannot consider such a pattern as a desirable one. Thus, we propose **episodic feedback**, which is defined as the difference between the historical score of a pattern and the average cumulative reward of all past episodes:

$$\delta(p_t) = h\_score(p_t) - \overline{\mathcal{R}}. \tag{2}$$

where $h\_score(p_t)$ is the historical score of pattern p_t and $\overline{\mathcal{R}}$ is the average cumulative reward of all past episodes.

With the episodic feedback, we conditionally add the episodic feedback of a pattern to the reward of its initial state. Specifically, for an episode e, depending on the outcome of e (whether the adversarial agent wins), we revise the rewards of the initial states of patterns in e in two cases. Assume that s_t is the initial state of pattern p_t in episode e, whose reward is r_t, and $\delta(p_t)$ is the episodic feedback for p_t, the conditional reward revision can be formulated as:

$$\hat{r}_t = \begin{cases} r_t + \delta(p_t) \times \epsilon, & \text{if the adversarial agent wins} \\ & \textit{and } \delta(p_t) > 0, \\ r_t + \delta(p_t) \times \epsilon, & \text{if the adversarial agent loses} \\ & \textit{and } \delta(p_t) < 0, \\ r_t, & \text{otherwise,} \end{cases} \tag{3}$$

where $\hat{r}_t$ is the revised reward and ϵ is a coefficient used to regulate the magnitude of encouragement and punishment. After the revision, we update r_t with $\hat{r}_t$.

The reason we only revise the reward in the two cases mentioned above is that the revision must adhere to the win-loss rules of the two-player competitive game environment, even though this environment reduces to a single-player game environment during the training process. In a competitive game, we believe only states from a winning episode of the adversarial agents should be rewarded ($\delta(p_t) > 0$), while states from the losing episode should be penalized ($\delta(p_t) < 0$). We will further analyze other cases in Section 4.3.3.

3.3 EPISODIC CONTROL-BASED ADVERSARIAL POLICY TRAINING

3.3.1 NEURAL NETWORK-BASED EPISODIC MEMORY

As we stated in Section 3.2.1, we propose an episodic memory to generate historical scores for patterns based on the historical experiences. The architecture of our episodic memory, as shown in Figure 1, includes an LSTM network followed by a multi-layer perceptron (MLP). The LSTM encodes the patterns into abstract vectors, while the MLP aggregates these encodings to produce historical scores for the patterns.

During the training, new episodes are used to update the episodic memory. Assume a newly produced episode e has a cumulative reward of $\mathcal{R}$ and contains the set of patterns P. The episodic memory M is then trained to map patterns in P to the new cumulative reward $\mathcal{R}$. Note that for a pattern in P, the learning target is the historical score of the pattern, which is not a fixed value and will change as new episodes occur. Then, during the reward revision, M predicts historical scores for the patterns as Equation 1 to generate historical evaluations for reward revision. More details of our implementation of the episodic memory can be found in Appendix A.2.

3.3.2 GROUP-BASED EPISODIC FEEDBACK

In practice, we find that later-generated episodes exhibit higher winning rates than earlier-generated episodes during the training process. To achieve better performance in finding the optimal policy, we compare the historical score of a pattern with the average cumulative reward from recently generated episodes when computing the episodic feedback. To achieve this, we divide the episodes into groups of size n, where n is a hyper-parameter, and calculate the average reward of past episodes in the group. The average reward of the ith episode of group m can be calculated by

$$\overline{\mathcal{R}}_i^m = \frac{\sum_{j=1}^{j=i} \mathcal{R}_j^m}{i},\ 1 \leq i \leq n. \tag{4}$$

Thus, we compute the episodic feedback of pattern p_t in the ith episode of group m by

$$\delta(p_t) = h\_score(p_t) - \overline{\mathcal{R}}_i^m, \tag{5}$$

which is implemented in our experiments.

3.3.3 ADVERSARIAL POLICY TRAINING WITH EPISODIC MEMORY

We implement our policy training as Algorithm 1. First, we have the adversarial agent interact with the environment and generate states, actions and rewards (Line 2-3). When an episode is ended, we extract patterns from the episode (In practical, we use sliding window) and calculate the cumulative reward of the episode (Line 5-6). We then update the episodic memory with the patterns and the cumulative reward (Line 7), and predict the historical score for each pattern with the episodic memory(Line 8). Subsequently, we calculate the average cumulative reward of the group with Equation 4 (line 9-11), and then utilize the average cumulative reward to calculate the episodic feedback for each pattern following Equation 5 (Line 12). With the episodic feedbacks, the rewards of the states could be revised by Equation 3 (Line 13) under the condition stated in Section 3.2.2. After the reward revision, we store the states, actions and the revised rewards into the experience storage (Line 14). When the update condition is satisfied (*e.g.*, storage is full), we will sample some data from the storage and calculate the objective function from our selected policy training method (Line 17-19). The adversarial agent will be updated with the objective function (Line 20). Iteration will end when the maximum training step is reached.

4 EVALUATION

In this section, we conduct a comprehensive evaluation of our approach. We compare the performances of our approach with state-of-the-art adversarial policy training techniques and show its effectiveness and efficiency in the context of two-player competitive games.

Algorithm 1 Episodic control-based Adversarial Policy Training

Input: $\mathcal{A}$: Adversarial Agent, E: Environment, M: Our episodic memory, $\mathcal{B}$: Experience Storage, O: Objective Function
Parameter: k: Pattern Length, n: Group Size, ϵ: Revision Coefficient
Output: $\mathcal{A}$: A Well-trained Adversarial Agent

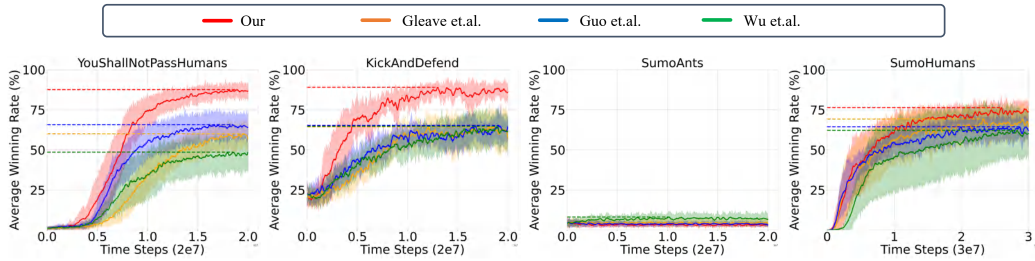
while Training does not reach the maximum step **do**
  $\mathcal{A}$ interacts with E and generate state s, action a, reward r.
  $S.add(s), A.add(a), R.add(r)$.
  if An episode ends **then**
    $P \leftarrow Pattern(k, S)$
    $R_{cum} \leftarrow Cumulative\_reward(R)$
    $M.update(P, R_{cum})$
    $H\_Score(P) \leftarrow M(P)$
    if The episode is the ith episode in Group m **then**
      $\overline{\mathcal{R}}_i^m \leftarrow Average\_Reward(i, m)$
    end if
    $\Delta(P) \leftarrow Episodic\_Feedback(H\_Score(P), \overline{\mathcal{R}}_i^m)$
    $R' \leftarrow Reward\_Revision(R, \Delta(P), \epsilon)$
    $\mathcal{B} \leftarrow S, A, R'$
    Clear S, A, R
  end if
  if Check_Update() is true **then**
    $Experiences \leftarrow Sample(B)$
    $O(Experiences)$
    $\mathcal{A}.update(O)$
  end if
end while
return $\mathcal{A}$

4.1 Experiment Setup

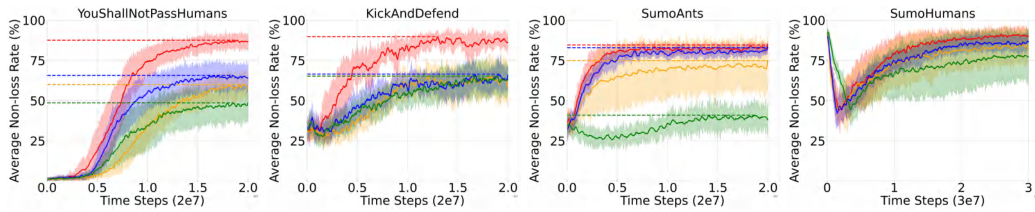
In our experiment, our method use PPO (Schulman et al., 2017) as the basic single-agent policy training method to train the adversarial agents to attack well-trained *Zoo* agents (Bansal et al., 2018). We take three state-of-the-art approaches (Gleave et al., 2020; Wu et al., 2021; Guo et al., 2021) as baselines to show the effectiveness of our method. It is important to note that Gleave *et al.* 's method fundamentally incorporates PPO for the adversarial policy training without modifying the training mechanisms of PPO. Therefore, when we draw comparisons with the outcomes achieved by Gleave *et al.* 's method, we are also comparing our method against a PPO baseline. To maintain the fairness of the experiments, we use the same two-player competitive games in the MuJoCo robotics simulator as our baselines, which are *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and *SumoAnts*, and run 5 seeds on each environment to evaluate our proposed adversarial training method. Hyper-parameters setting are shown in the Appendix A.1.

4.2 Main Results

The comparison of the winning rates and non-loss rates between our approach and the baseline approaches are summarized in Figure 2. We can observe that our proposed method reaches 88% and 89% winning rates and outperforms the baseline methods significantly in *YouShallNotPassHumans* and *KickAndDefend*. In *SumoHumans*, our method also surpasses all baselines. These results indicate that by leveraging historical experiences to highlight the high-performing states, our agents demonstrate higher sample efficiency and have more potential to discover effective adversarial policies to defeat victim agents. In *SumoAnts*, Since the winning rates of all agents against the victim are far below 50%, we use the non-loss rates to measure the effectiveness of our method. From Figure 2(b), we observe that in *SumoAnts*, the non-loss rate of our agent is still able to surpass that of agent trained by (Guo et al., 2021), which exhibits the second highest non-loss rate.



(a) The comparison of winning rates.



(b) The comparison of non-loss rates.

Figure 2: The performance of our adversarial agents and baseline adversarial agents in each environment. Dashed lines represent the highest rates of each agent. More details are shown in Table 4 and Table 5 in Appendix A.5.

Table 1: The non-loss rate of our adversarial agent and baseline adversarial agents against masked victim agents in 100 games. Each experiment has been done 4 times.'Before' and 'After' indicate before and after masking the victim agent.

Environment	Our (%)		Gleave *et al.* (%)		Guo *et al.* (%)		Wu *et al.* (%)	
	Before	After	Before	After	Before	After	Before	After
YouShallNotPassHumans	96 ± 2	73 ± 4	66 ± 2	0 ± 0	72 ± 3	0 ± 0	50 ± 2	0 ± 0
KickAndDefend	93 ± 4	7 ± 1	65 ± 1	3 ± 1	63 ± 2	5 ± 1	69 ± 4	5 ± 1
SumoAnts	83 ± 2	81 ± 2	76 ± 2	69 ± 4	81 ± 3	78 ± 1	57 ± 5	53 ± 4
SumoHumans	92 ± 1	90 ± 1	92 ± 1	91 ± 1	92 ± 0	91 ± 1	94 ± 2	92 ± 1

We share the videos of agents trained by our approach and baseline approaches in Appendix A.4 and compare their behaviors. In *KickAndDefend* and *SumoHumans*, all the adversarial agents perform similar adversarial actions to trick the victim into performing abnormal behaviors. These results align with the conclusion in (Gleave et al., 2020) that adversarial agents win by confusing the victim, instead of becoming a strong opponent. However, in *YouShallNotPassHumans*, while the three baseline agents attack the victim by convulsing on the ground, our agent simultaneously performs the adversarial action and obstructs the victim with its body. This indicates that our agent is not restricted to utilizing only one winning strategy. In fact, it is able to explore the optimal strategy by combining multiple winning strategies from historical experiences. Additionally, in *SumoAnts*, agents trained by our approach and (Guo et al., 2021) both jump out of the arena at the beginning since falling out of the arena without touching the opponent is considered a draw in this game, while agents trained by (Gleave et al., 2020) and (Wu et al., 2021) still fight with the victim. This suggests that same as (Guo et al., 2021), our agent is also capable of discovering and exploiting game imbalances.

To validate whether our approach is difficult to defend against, we conduct retraining experiments on the victim agents using the PPO algorithm. We report the results of our adversarial agent and baseline agents against the victim agents during the retraining in Figure 3. In *YouShallNotPassHumans*, our agent maintains a relatively high winning rate during the retraining of victim agents, unlike baseline agents whose winning rate quickly drops to a low level. In *KickAndDefend*, the winning rates of our agent also decreases at a slower rate compared to the baseline agents. This indicates that our approach is more difficult to defend than baseline approaches.

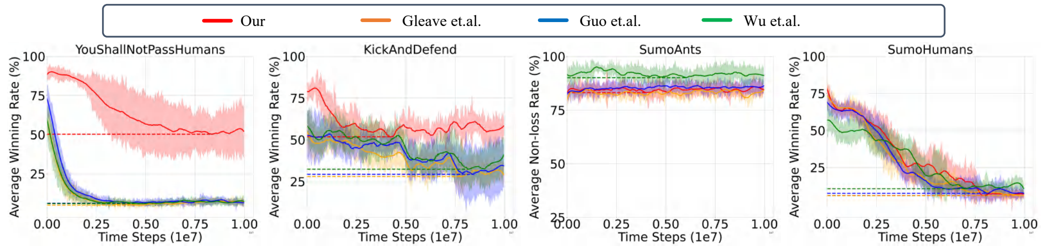


Figure 3: The comparison between our adversarial agent and baseline adversarial agents in each environment (Gleave et al., 2020; Guo et al., 2021; Wu et al., 2021) during the retraining. Specifically, we show winning rates of the agents in *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and non-loss rate in *SumoAnts*. The lowest rate of each agent is depicted with a dashed line. More details are shown in Table 6 in Appendix A.5.

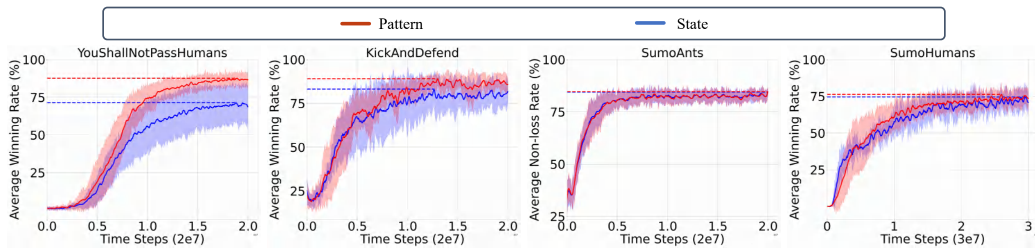


Figure 4: The comparison of performances between agents guided by pattern-based and state-based historical evaluation. We show winning rates of the agents in *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and non-loss rate in *SumoAnts*. The highest rate of each agent is depicted with a dashed line. More details are shown in Table 7 in Appendix A.5.

In order to gain a better understanding of the effectiveness of our method, following the approach in (Gleave et al., 2020), we have our adversarial agents play games against masked victim agents, whose observation of the adversary's position is set to a static value corresponding to a typical initial position so that the adversarial actions may not be effective. We show the performances of our adversarial agents and baseline agents against masked victims in Table 1. We observe a significant decline in the non-loss rates of the three baseline agents, while our agent maintains a high non-loss rate against the masked victim in *YouShallNotPassHumans*. This could be attributed to the fact that our agent not only relies on adversarial actions to attack the victim, but also incorporates non-adversarial actions like obstructing the victim with its body to win the game. Based on this finding, we demonstrate that in *YouShallNotPassHumans*, our agent can defeat the victim by performing adversarial and non-adversarial actions simultaneously.

4.3 Ablation Study

4.3.1 Patterns

As mentioned in Section 3.2.1, we use historical score of the pattern to represent historical evaluation. To show the effectiveness of utilizing pattern as the basis, we calculate the episodic feedback with historical scores of both states and 3-step patterns and then revise the rewards with two episodic feedbacks. In Figure 4, we can see pattern-guided agents hold higher winning rates than state-guided agents, which proves that patterns can provide a richer, more contextual basis for the historical evaluation.

4.3.2 Episodic Feedback

In Section 3.2.2, we mention that the episodic feedback of a pattern is computed by the difference between the historical score and the average cumulative reward of the group containing the episode. If episodic feedback is greater than 0, the pattern can be considered a better pattern than patterns in

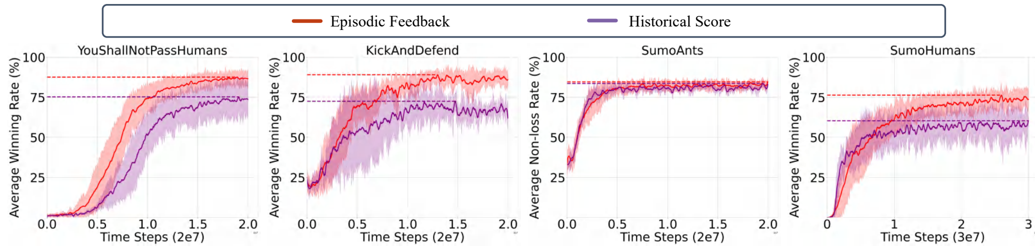


Figure 5: The comparison of performances between proposed training approach guided by episodic feedbacks and historical scores. We show winning rates of the agents in *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and non-loss rate in *SumoAnts*. The highest rate of each agent is depicted with a dashed line. More details are shown in Table 8 in Appendix A.5.

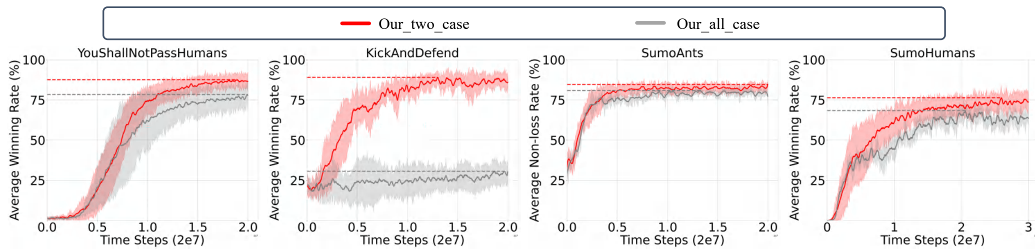


Figure 6: The comparison of performances between proposed training approach with different revision conditions. We show winning rates of the agents in *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and non-loss rate in *SumoAnts*. The highest rate of each agent is depicted with a dashed line. More details are shown in Table 9 in Appendix A.5.

recently generated episodes, thereby enabling the adversarial agent to search for an optimal strategy. To show the effectiveness of episodic feedback, we use both episodic feedback and historical score to revise the reward. Based on the results shown in Figure 5, we can find that the performances of agents trained with episodic feedback outperform agents trained with historical score, which indicates that compared to the historical score, episodic feedback is more effective in helping the agent find the optimal policy.

4.3.3 Revision Condition

As stated in Section 3.2.2, there are other cases in which we do not perform the reward revision. For example, we do not revise the reward when a state from a winning episode of the adversarial agents obtains negative episodic feedback. If we were to implement the revision for other cases, some states that lead to losses would be rewarded and some states that lead to wins would be penalized. This may make the adversarial agent learn a policy that aims at losing the episode. To better demonstrate the effectiveness, we also perform reward revisions in those cases and show the performances in Figure 6. We can observe that the winning rate drops when we revise the reward in all cases, especially in *KickAndDefend*. This proves that reward revision must comply with the rules of the two-player competitive games, even though our adversarial training method uses an single-player game environment.

5 Discussion and Conclusion

As stated in Section 4.1, our method utilizes PPO algorithm as the single-agent training method to train the adversarial agents. It is also important to note that our method is scalable and can be applied to various DRL algorithms. In the Appendix, we demonstrate the performances of our approach applied to the baseline algorithms (Wu et al., 2021; Guo et al., 2021) and compare the results with those of the original baseline algorithms in *YouShallNotPassHumans* and *KickAndDefend*. The results show that by leveraging historical evaluations to revise the rewards, the performances of all baseline approaches get improved.

In this paper, we propose an episodic control-based adversarial policy training method to train an adversarial agent more effectively and efficiently. Our method introduces an episodic memory to utilize the historical experiences to generate historical evaluations for the states and consequently revise the rewards of the states based on the evaluation, thereby integrating the historical evaluations into the rewards used for adversarial training to emphasize the high-performing states. In our experiments, we demonstrate that agents trained with our approach achieve the most promising attack performance and defense difficulty. Additionally, by comparing the behaviors of adversarial agents, we discover that our attack method can explore optimal strategies by integrating multiple winning approaches. We believe our exploration of game states and use of historical experiences advance adversarial policy training methods. Future research could focus on extracting more specific insights from these historical experiences to further enhance the effectiveness of adversarial learning.

Limitation Our method is designed for two-player competitive game environments. For other environments, the reward revision needs to be redesigned.

REFERENCES

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sy0GnUxCb.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

The Viet Bui, Tien Mai, and Thanh H Nguyen. Imitating opponent to win: Adversarial policy imitation learning in two-player competitive games. *arXiv preprint arXiv:2210.16915*, 2022.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgEMpVFwB.

Wenbo Guo, Xian Wu, Sui Huang, and Xinyu Xing. Adversarial policy learning in two-player competitive games. In *International Conference on Machine Learning*, pp. 3910–3919. PMLR, 2021.

Steven Hansen, Alexander Pritzel, Pablo Sprechmann, André Barreto, and Charles Blundell. Fast deep reinforcement learning using online adjustments from the past. *Advances in Neural Information Processing Systems*, 31, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=ryvlRyBKl.

Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

Igor Kuznetsov and Andrey Filchenkov. Solving Continuous Control with Episodic Memory. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 2651–2657, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6. doi: 10.24963/ijcai.2021/365. URL https://www.ijcai.org/proceedings/2021/365.

Hung Le, Thommen Karimpanal George, Majid Abdolshah, Truyen Tran, and Svetha Venkatesh. Model-based episodic memory induces dynamic hybrid controls. *Advances in Neural Information Processing Systems*, 34:30313–30325, 2021.

Xian Yeow Lee, Sambit Ghadai, Kai Liang Tan, Chinmay Hegde, and Soumik Sarkar. Spatiotemporally constrained action space attacks on deep reinforcement learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4577–4584, 2020.

Máté Lengyel and Peter Dayan. Hippocampal contributions to control: the third way. *Advances in neural information processing systems*, 20, 2007.

Zhuo Li, Derui Zhu, Yujing Hu, Xiaofei Xie, Lei Ma, Yan Zheng, Yan Song, Yingfeng Chen, and Jianjun Zhao. Neural episodic control with state abstraction. *arXiv preprint arXiv:2301.11490*, 2023.

Thodoris Lykouris, Max Simchowitz, Alex Slivkins, and Wen Sun. Corruption-robust exploration in episodic reinforcement learning. In *Conference on Learning Theory*, pp. 3242–3245. PMLR, 2021.

Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems*, 32, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Thanh Thi Nguyen and Vijay Janapa Reddi. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

Xinlei Pan, Chaowei Xiao, Warren He, Shuang Yang, Jian Peng, Mingjie Sun, Mingyan Liu, Bo Li, and Dawn Song. Characterizing attacks on deep reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pp. 1010–1018, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pp. 2040–2042, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International Conference on Machine Learning*, pp. 2827–2836. PMLR, 2017.

Alessio Russo and Alexandre Proutiere. Optimal attacks on reinforcement learning policies. *arXiv preprint arXiv:1907.13548*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jianwen Sun, Tianwei Zhang, Xiaofei Xie, Lei Ma, Yan Zheng, Kangjie Chen, and Yang Liu. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5883–5891, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Xian Wu, Wenbo Guo, Hua Wei, and Xinyu Xing. Adversarial policy training against deep reinforcement learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1883–1900, 2021.

Zhaoyuan Yang, Naresh Iyer, Johan Reimann, and Nurali Virani. Design of intentional backdoors in sequential models. *arXiv preprint arXiv:1902.09972*, 2019.

Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452*, 2021.

Zhizheng Zhang, Jiale Chen, Zhibo Chen, and Weiping Li. Asynchronous episodic deep deterministic policy gradient: Toward continuous control in computationally complex environments. *IEEE transactions on cybernetics*, 51(2):604–613, 2019.

Yiren Zhao, Ilia Shumailov, Han Cui, Xitong Gao, Robert Mullins, and Ross Anderson. Blackbox attacks on reinforcement learning agents using approximated temporal information. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 16–24. IEEE, 2020.

Lulu Zheng, Jiarui Chen, Jianhao Wang, Jiamin He, Yujing Hu, Yingfeng Chen, Changjie Fan, Yang Gao, and Chongjie Zhang. Episodic multi-agent reinforcement learning with curiosity-driven exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

A Appendix

A.1 Hyper-parameter Analysis

Table 2: Hyper-parameters of episodic memory used in our experiments.

Hyper-parameter	
Pattern Length k	3
Group Size n	100
Epsilon ϵ	0.1

For PPO hyper-parameter selections, we use the same parameters from (Gleave et al., 2020). The hyper-parameters for our episodic memory are listed in Table 2. As mentioned in (Li et al., 2023), 0.1 for ϵ works well for episodic control method in MuJuCo games, so we follow this setting in our experiments. We further analyze the rest two hyper-parameters.

A.1.1 Pattern Length

As stated in Section 3.2, we use state sequences, referred to as patterns, as the basis for performance evaluation in past episodes. To find out the best parameter for the length of patterns, we conduct experiments with different pattern lengths and show the results in Figure 7. We have selected three different pattern lengths in our experiments. We can see from Figure 7 that the agents have the best average winning rates when the pattern length is 3.

A.1.2 Group Size

In Section 3.3.2, we compare the historical score of a pattern with an average cumulative reward of recently generated past episodes to calculate episodic feedback. Group size n is introduced to control the number of past episodes. If the group size is too large, some states that lead to bad patterns may be erroneously rewarded, and the magnitude of rewards and penalties is reduced, thereby weakening the ability to find the optimal policy. On the other hand, if the group size is too small, it is more likely to wrongly penalize good states and reward bad states. Therefore, we conduct an analysis of 3 group sizes which are *50*, *100* and *150*. With the result shown in Figure 8, we find that the agents have the best performances when the group size is *100*. Therefore, we use *100* as the group size in our experiments.

A.2 Episodic Memory

In this section we share more details about the implementation of the episodic memory introduced in Section 3.3.1.

Table 3: The architecture of the episodic memory

Module	shape
LSTM	(64, 256, 1)
linear1	(256, 512)
Tanh	
linear2	(512, 1)

In Table 3, we give the architecture of the episodic memory. The episodic memory consists of a LSTM and a MLP (linear1, Tanh, linear2). 64 in the shape of LSTM refers to the length of the state from environment and 256 refers to the hidden length of LSTM. The LSTM is used to encode a pattern into an abstract vector so that the MLP can process. The MLP is used to output the historical score of the pattern, which evaluates the average performance of the pattern in past episodes.

In the Algorithm 2, we show the forward process of the episodic memory. The LSTM receives a pattern p as input and outputs an output sequence, the last hidden state vector and the last cell state

Algorithm 2 Forward process of the episodic memory.

Input: pattern p (shape:[3, 64])
Output: historical score h_score (shape:1)

output, hidden, cell = LSTM(p)
h_score = linear1(hidden[-1,])
h_score = Tanh(h_score)
h_score = linear2(h_score)

Algorithm 3 Update process of the episodic memory.

Input: episode e

P = Pattern(e)
R = Cumulative_Reward(e)
for p in P **do**
    h_score = Memory(p)
    loss = MSELoss(h_score, R)
    loss.backpropagation()
end for

vector of LSTM. The last hidden state vector will be processed with MLP, under the order of linear1, Tanh, linear2 and the MLP will output the historical score h_score of the input pattern.

In the Algorithm 3, we show the update process of the episodic memory. After one episode is ended, we extract patterns from the episode and calculate the cumulative reward of the episode. Then, we predict the historical score for each pattern and calculate the MSELoss between the historical score and the cumulative reward. The loss will be backpropagated to update the network.

A.3 Scalability Analysis

In Section 5, we state that our approach is scalable and can be applied to various DRL algorithms. Since Gleave *et al.* can be seen as PPO, we adopt our approach on the other two baseline attacks (Guo et al., 2021; Wu et al., 2021) and compare the performances with them. The results are shown in Figure 9. We can see the baselines adopting our episodic memory outperform the original baselines in *YouShallNotPassHumans* and *KickAndDefend*.

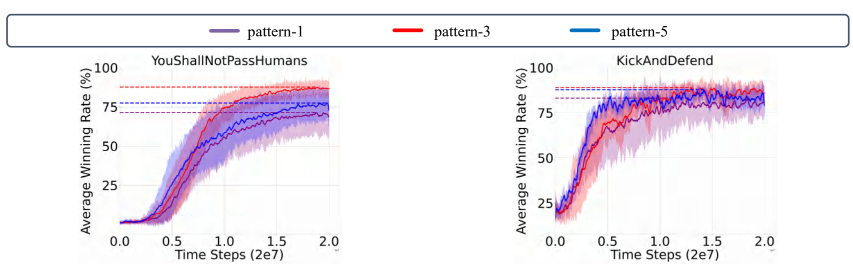


Figure 7: The comparison of winning rate between our adversarial agent with different input pattern lengths in *YouShallNotPassHumans* and *KickAndDefend*.

A.4 Videos Of Experiments

Due to the limited maximum file size for the supplementary materials, we have uploaded the videos mentioned in Section 4.2 at `https://drive.google.com/drive/folders/1lJmWA7y8-1nMs_kOwzGlIMkjkPh2QVF6?usp=drive_link`.

A.5 Main Results Supplementary

Supplementary tables of the figures in the main text are provided on the subsequent pages.

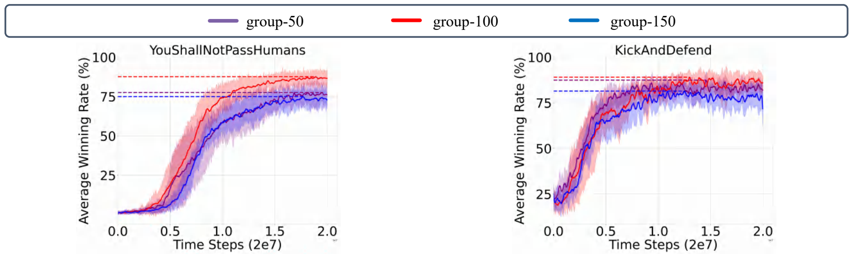
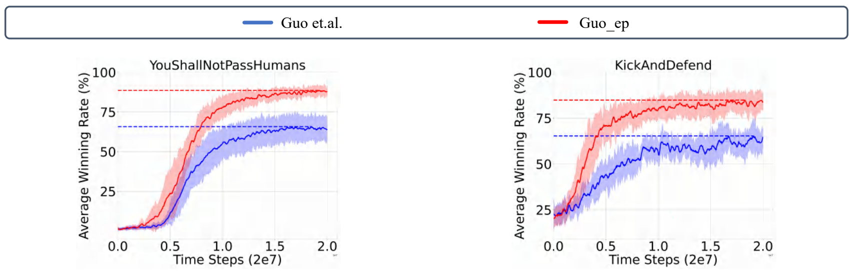
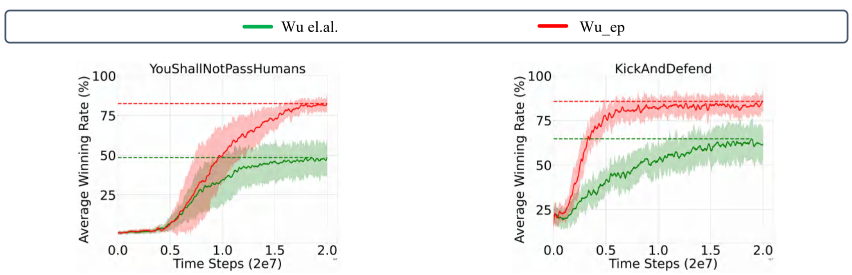


Figure 8: The comparison of winning rate between our adversarial agent with group size in *YouShallNotPassHumans* and *KickAndDefend*.



(a) The performaces of adversarial agents trained by (Guo et al., 2021) implementing with and without our episodic memory.



(b) The performaces of adversarial agents trained by (Wu et al., 2021) implementing with and without our episodic memory.

Figure 9: The performance comparison of winning rate between agents trained by (Wu et al., 2021; Guo et al., 2021) implementing with and without our episodic memory in *YouShallNotPassHumans* and *KickAndDefend*.

Table 4: The highest winning rates of our agents and agents attacks against zoo victim agents are shown in Figure 2(a).

Environment	Our (%)	Gleave *et al.* (%)	Guo *et al.* (%)	Wu *et al.* (%)
YouShallNotPassHumans	87.62 ± 7.38	60.08 ± 6.22	65.77 ± 7.60	48.60 ± 8.99
KickAndDefend	89.06 ± 7.98	64.37 ± 8.61	65.32 ± 6.92	64.76 ± 8.55
SomoAnts	5.18 ± 1.27	5.19 ± 2.10	4.70 ± 1.43	8.14 ± 2.87
SumoHumans	76.35 ± 8.29	69.24 ± 12.16	64.49 ± 7.15	62.22 ± 16.86

Table 5: The highest non-loss rates of our agents and baseline agents against zoo victim agents are shown in Figure 2(b).

Environment	Our (%)	Gleave *et al.* (%)	Guo *et al.* (%)	Wu *et al.* (%)
YouShallNotPassHumans	87.62 ± 7.38	60.08 ± 6.22	65.77 ± 7.60	48.60 ± 8.99
KickAndDefend	90.01 ± 7.56	65.17 ± 8.87	66.56 ± 7.14	65.27 ± 8.45
SomoAnts	84.66 ± 3.92	74.94 ± 16.24	82.98 ± 3.73	40.97 ± 6.65
SumoHumans	91.68 ± 7.52	91.88 ± 12.18	90.49 ± 5.48	92.55 ± 14.40

Table 6: The performances of our agents and baseline agents against retrained victim agents are shown in Figure 3. We show winning rates of the agents in *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and non-loss rate in *SumoAnts*.

Environment	Our (%)	Gleave *et al.* (%)	Guo *et al.* (%)	Wu *et al.* (%)
YouShallNotPassHumans	50.27 ± 13.03	5.00 ± 2.50	6.22 ± 2.82	5.99 ± 2.99
KickAndDefend	51.82 ± 6.84	28.02 ± 7.33	29.33 ± 9.84	32.38 ± 9.54
SomoAnts	83.15 ± 2.98	79.78 ± 2.28	82.49 ± 2.77	90.13 ± 3.42
SumoHumans	6.17 ± 7.60	6.03 ± 5.35	7.61 ± 5.88	10.71 ± 6.95

Table 7: The performances of our agents guided by pattern-based and state-based historical evaluation against zoo victim agents is shown in Figure 4. We show winning rates of the agents in *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and non-loss rate in *SumoAnts*.

Environment	Pattern (%)	State (%)
YouShallNotPassHumans	87.62 ± 7.38	71.36 ± 11.26
KickAndDefend	89.06 ± 7.98	83.15 ± 11.29
SumoAnts	84.66 ± 3.92	83.73 ± 4.15
SumoHumans	76.35 ± 8.29	74.35 ± 6.71

Table 8: The performances of our agents trained with episodic feedback and historical score against zoo victim agents shown in Figure 5. We show winning rates of the agents in *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and non-loss rate in *SumoAnts*.

Environment	Episodic Feedback (%)	historical score (%)
YouShallNotPassHumans	87.62 ± 7.38	75.28 ± 8.25
KickAndDefend	89.06 ± 7.98	72.85 ± 11.48
SumoAnts	84.66 ± 3.92	83.66 ± 2.30
SumoHumans	76.35 ± 8.29	72.15 ± 6.90

Table 9: The performances of our agents with and without revision conditions against zoo victim agents is shown in Figure 6. We show winning rates of the agents in *YouShallNotPassHumans*, *KickAndDefend*, *SumoHumans* and non-loss rate in *SumoAnts*.

Environment	Our_two_case (%)	Our_all_case (%)
YouShallNotPassHumans	87.62 ± 7.38	78.39 ± 10.74
KickAndDefend	89.06 ± 7.98	30.64 ± 9.10
SumoAnts	84.66 ± 3.92	80.44 ± 3.73
SumoHumans	76.35 ± 8.29	68.40 ± 4.14