processes

Article

Locality-Based Action-Poisoning Attack against the Continuous Control of an Autonomous Driving Model

Yoonsoo An [1], Wonseok Yang [2] and Daeseon Choi [2,*]

[1] Cyber Security Research Center, Graduate School of Soongsil University, Sadang-ro 50, Seoul 07027, Republic of Korea; yoonsooh@soongsil.ac.kr
[2] Department of Computer Science and Engineering, Graduate School of Soongsil University, Sadang-ro 50, Seoul 07027, Republic of Korea; diddnjstjr25@soongsil.ac.kr
* Correspondence: sunchoi@ssu.ac.kr

Abstract: Various studies have been conducted on Multi-Agent Reinforcement Learning (MARL) to control multiple agents to drive effectively and safely in a simulation, demonstrating the applicability of MARL in autonomous driving. However, several studies have indicated that MARL is vulnerable to poisoning attacks. This study proposes a 'locality-based action-poisoning attack' against MARL-based continuous control systems. Each bird in a flock interacts with its neighbors to generate the collective behavior, which is implemented through rules in the Reynolds' flocking algorithm, where each individual maintains an appropriate distance from its neighbors and moves in a similar direction. We use this concept to propose an action-poisoning attack, based on the hypothesis that if an agent is performing significantly different behaviors from neighboring agents, it can disturb the driving stability of the entirety of the agents. We demonstrate that when a MARL-based continuous control system is trained in an environment where a single target agent performs an action that violates Reynolds' rules, the driving performance of all victim agents decreases, and the model can converge to a suboptimal policy. The proposed attack method can disrupt the training performance of the victim model by up to 97% compared to the original model in certain setting, when the attacker is allowed black-box access.

Keywords: reinforcement learinng; multi-agent reinforcement learning; AI security; poisoning attack; adversarial attack

check for updates

Citation: An, Y.; Yang, W.; Choi, D. Locality-Based Action-Poisoning Attack against the Continuous Control of an Autonomous Driving Model. *Processes* **2024**, *12*, 314. https://doi.org/10.3390/pr12020314

Academic Editor: Francisco Vazquez

Received: 5 January 2024
Revised: 21 January 2024
Accepted: 26 January 2024
Published: 1 February 2024

1. Introduction

Deep reinforcement learning is a method used to approximate optimal policies for sequential decision-making problems defined as Markov Decision Process (MDP) [1], using deep neural networks that are attracting attention in various research fields such as robotics, autonomous driving, and battlefield strategies [2–4]. Due to the nature of those fields, it is essential to consider environments where many agents train and test together, or where many agents and human agents work together collaboratively, competitively or in combination, so studies on multi-agent reinforcement learning (MARL) are also in the spotlight. Recently, studies using MARL are actively conducted to control multiple agents to drive effectively and safely in simulation, showing its applicability in auotonomous driving environment. However, reinforcement learning models are vulnerable to adversarial attacks [5], including poisoning attacks [6]. Liu et al. [7] proposed an action-manipulation attack in which an attacker can manipulate actions during the training phase by changing the action signal selected by the user in a multi-armed bandit problem. This study demonstrated that an attacker can force the target agent to learn a suboptimal policy as intended by the attacker. In a subsequent study [8], an action-poisoning attack was proposed, which is an extended action-manipulation attack in multiarmed bandits that may be applied to general reinforcement learning models to investigate the potential risks of action-manipulation attacks in the MARL model with a discrete action space [9,10].

The vulnerabilities of single-agent RL models also exist in MARL models, making them susceptible to action-poisoning attacks. Despite this risk, MARL is frequently applied in autonomous driving models owing to its superior performance achieved through inter-agent cooperation compared with single-agent reinforcement learning [11]. The action space of an autonomous driving model can be discrete or continuous; however, common tasks included in such models, such as steering and acceleration, have continuous values. Given the safety-critical characteristics of autonomous driving and the risk of fatal damage, it is necessary to evaluate the vulnerability to action-poisoning attacks in MARL-based autonomous driving models with a continuous action space.

Poisoning attacks during the training of autonomous driving systems using MARL can be dangerous. These attacks disrupt the learning process, leading to unsafe driving behavior. For example, cars might learn the wrong way to handle traffic situations, increasing the chance of accidents. This is a big concern, especially since real-world driving is complex and always changing. When these attacks affect how autonomous agents make decisions together, it can cause major safety issues. This shows why having strong and reliable training methods is crucial for autonomous driving technology.

To the best of our knowledge, despite this critical importance, no previous study has investigated action-poisoning attacks on the MARL-based continuous control of autonomous driving models. Therefore, we propose a black-box action-poisoning attack against the continuous control of an autonomous driving model by using target actions that violate Reynolds' rules of a flocking algorithm [12], which implements the flocking behavior of birds. In the flocking algorithm, individuals in the flock act in consideration of their neighbors and generate collective behavior. When all agents in the MARL-based autonomous driving model are viewed as a flock, if the flock includes target agents that are not compatible with performing actions that are significantly different from those of their neighbors, all agents except the target agent may be disturbed. By identifying and analyzing the susceptibility of MARL-based autonomous models to locality-based action-poisoning attacks, our study aims to contribute to the development of more robust training methodologies. The main contributions of our study are as follows:

1. First, we execute an action-poisoning attack against the MARL-based continuous control system and force the victim model to be trained as a suboptimal policy.
2. We propose an action-poisoning attack that uses a target action that changes dynamically with the observation of the target agent.
3. We show that an attacker who allows only black-box access can disrupt training by using target actions that violate Reynolds' rules.

2. Backgrounds

2.1. Multi-Agent Reinforcement Learning

Sequential decision making in deep reinforcement learning (DRL) can be formulated as a Markov Decision Process (MDP), where at time step t, the agent observes state st and chooses action at according to policy π, and receives the reward rt and observes the next state in time step $t+1$. The cumulative reward is affected by discount factor γ. An optimal policy π^* is one that maximizes the expected value of the reward obtained by an agent when following the policy. The goal of reinforcement learning is to obtain an optimal policy through training. The discounted cumulative reward sum is obtained as follows where γ is the discount factor and $r(s_t)$is the reward at time step t:

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r(s_t) \tag{1}$$

There are two main categories of DRL algorithms. In value-based learning, the algorithm performs actions based on a value function that calculates the expected value of the reward in each state. When the optimal action-value function is learned, the optimal policy can be derived by selecting the action that can receive the highest reward according to the value function. In policy-based learning, the model trains the policy represented as a probability distribution of actions and determines a policy that maximizes the expected cumulative reward. Value-based algorithms, such as Deep Q-networks (DQN), can find it challenging to obtain the optimal action-value function, and policy-based algorithms, such as REINFORCE, may not converge well during training. Therefore, algorithms such as actor–critic have been developed to combine the advantages of both value- and policy-based learning algorithms.

Multi-agent reinforcement learning is a subfield of reinforcement learning in which multiple agents interact with each other in a shared environment and learn a policy that can obtain higher rewards through cooperative or competitive actions. MARL is more complex and challenging than single-agent reinforcement learning because it involves multiple agents in a shared environment. The dimensions of the joint action space in MARL increase exponentially, which leads to an increase in the computational complexity of MARL algorithms. In MARL, multiple agents concurrently learn and interact with the environment during training. Because the joint action of all agents affects the subsequent state that each agent observes, the agents must adapt to the changing policies of the other agents. This is one of the reasons why the stable convergence of MARL is difficult. In addition, in a partially observable environment, agents are unable to observe the full state and make decisions based on partial observation, which can lead to difficulties in learning [13,14].

Multi-agent reinforcement learning (MARL) in autonomous driving simulations is increasingly important for mimicking complex, real-world traffic scenarios. It enables autonomous vehicles, treated as individual agents, to interact with and learn from the behaviors of other vehicles, pedestrians, and traffic signals, which is vital for developing advanced, safe driving strategies. The recent trend in this field is towards creating more intricate and lifelike simulations. In these simulations, the actions of one agent can have significant effects on the entire system, promoting not only individual learning but also collaborative strategies. This aspect of MARL is particularly beneficial for enhancing traffic flow and safety. The inherent complexity and unpredictability of traffic scenarios in the real world make MARL an essential tool for training autonomous vehicles in environments that closely resemble actual driving conditions.

2.2. Adversarial Attacks against RL in Training Phase

In a poisoning attack, an attacker injects malicious samples into the training data to disrupt the models' learning process. The model learns from the malicious data and makes undesirable predictions during the inference phase. A backdoor attack [15] is a method in which an attacker inserts a backdoor trigger into the training data. When the input matches the specific trigger designated by the attacker, the model behaves as intended by the attacker. Both attacks can undermine the performance and robustness of the model. Adversarial attacks in the training phase, such as poisoning and backdoor attacks, are well-known vulnerabilities of supervised learning methods such as deep learning models [16]. Recent studies demonstrated that reinforcement learning is vulnerable to these types of attacks. Recent studies have also investigated poising attacks on RL and identified vulnerabilities. Ma et al. [17] conducted a reward poisoning attack, allowing an attacker to access the training batch data and determine the target model's algorithm. The attacker arbitrarily modifies the reward and shows that changing only the reward can sufficiently poison the policy. Rakhasha et al. [18] proposed a method in which an attacker can manipulate rewards or transition dynamics in an MDP to force the agent to execute the target policy chosen by the attacker. The proposed attack is applicable in both offline and online settings, and they also provide optimization framework for finding optimal stealthy attacks. Zhang et al. [19] studied a reward poisoning attack in which an attacker added perturbations to the environmental reward. This study also introduced adaptive attacks, which are more time-efficient than non-adaptive attacks, depending on the learning process. Sun et al. [20] indicated the unrealistic aspects of previous studies that assumed the attacker knows the MDP and proposed the Vulnerability-aware Adversarial Critic Poison (VA2C-P) algorithm, which does not require any prior knowledge of the environment. Compared to environmental-poisoning attacks, which require the direct manipulation of rewards or MDP models, action-poisoning attacks assume that an attacker can manipulate actions, limiting the ability of the model. The proposal of the attack corresponds to [7], a paper by Liu et al. proposed a new attack strategy called an action-manipulation attack in the multi-armed bandit [21] model [7]. In their study, an attacker manipulates the user's actions by changing the arm selected by the user to the target arm selected by the attacker. The attacker may or may not know the true mean reward for each arm. In an Oracle attack, when the attacker knows the true mean reward, they can decide which is the worst scenario and simply manipulate the user to pull the worst arm instead of the arm selected by the user. If the attacker does not know the true mean reward, they can estimate the mean rewards and use lower confidence bounds (LCBs) to select the arm with the smallest LCB as the non-target arm. This attack can force the user to pull the target arm, frequently at a logarithmic cost. In a subsequent study, Liu et al. proposed an attack on an RL model called an action-poisoning attack [8]. In their study, the attacker can also manipulate the action of the RL agent by assuming that the attacker is sitting between the agent and environment, and they force the agent to learn the target policy chosen by the attacker. They proposed a white-box attack called an α-portion attack that can force the RL agent to choose actions following the target policy and proposed a black-box attack called an LCB-H attack that nearly matches the performance of the α-portion attack.

The MARL model may have the vulnerabilities of other RL models; however, most studies have focused on single-agent settings, resulting in insufficient studies on poisoning and backdoor attacks on MARL. Zheng et al. [22] proposed a black-box poisoning attack method called the state noise pooling attack (SNPA) and a white-box backdoor attack method called the target action-poisoning attack (TAPA) against the MARL model with a discrete action space. SNPA modifies the observation of an agent, and TAPA injects a backdoor system by manipulating the reward function and action of an agent to trigger the target action selected by the attacker. The proposed method demonstrates that it is sufficient to manipulate only one agent to lower the performance of the c-MARL model with a discrete action space.

Several studies have shown above that MARL models are vulnerable to poisoning and backdoor attacks. Furthermore, the research about action-poisoning attacks in MARL-based autonomous driving models becomes more important due to the growing interest of MARLs in which the action of multiple agents in shared environments can significantly impact outcomes in complex systems such as autonomous driving. The growing use of MARL in safety-critical domains, such as autonomous driving, necessitates thorough security research for MARL models. As MARL models become more prevalent in these applications, their vulnerability to security risks increases. Therefore, it becomes crucial to enhance the security of these models, especially in the context of continuous action spaces common in autonomous driving. But, no previous studies have been conducted on action-poisoning attacks specifically targeting MARL models with continuous action spaces, particularly considering the continuous control systems of autonomous driving models. Therefore, we propose a locality-based action-poisoning attack against the continuous control systems of autonomous models to investigate and analyze the potential risks of autonomous driving models with safety-critical characteristics.

2.3. Social Interaction of Autonomous Driving

Various approaches exist for modeling social interactions between autonomous driving vehicles. A popular approach is a swarm/flocking-based model [23,24]. In a multivehicle traffic system, individual agents pursue their own goals while interacting with neighboring agents and strive to achieve global coordination, similar to the behavior of flocks or swarms. Flocking systems share some attributes with the particle system and follow Reynolds' rules, which dictate that each agent avoids collisions with each neighbor (separation), attempts to stay close to neighbors (cohesion), and matches the heading and velocity of its neighbors (alignment). However, unlike the flocking behavior found in birds, flocking in cooperative autonomous driving occurs in 1D environments, especially on roads, and the communication found in human drivers differs from that of birds. In this respect, Park et al. [25] proposed modified Reynolds rules for cooperative autonomous vehicles:

- Cohesion: Minimizing the distance between adjacent vehicles in the same lane improves road throughput and fuel efficiency.
- Separation: Maintaining a safe distance between adjacent vehicles in the same lane with optional lane changes.
- Alignment: Replacing desirable velocities, such as the road speed limit or the most fuel-efficient speed when heading control is not required.

A previous study discussed the importance of mimicking the social interactions and behaviors of human drivers in autonomous driving scenarios [26]. Additionally, the physical laws and natural behavior of animal swarms help improve the cooperation of autonomous vehicles on the road. By combining these two approaches, researchers can create models and algorithms that enable multiple vehicles to navigate complex driving scenarios safely and efficiently while also considering the social and behavioral aspects of human driving in the real world.

3. Preliminaries

3.1. Proximal Policy Optimization (PPO)

Proximal policy optimization (PPO) [27] represents a significant advance in policy gradient methods for reinforcement learning. It simplifies the complex computations involved in its predecessor, trust region policy optimization (TRPO), by approximating the surrogate objective function using first-order methods. This surrogate objective is designed to ensure that the policy updates do not deviate significantly from the previous policy, thus maintaining a stable learning progression. The PPO algorithm achieves this by employing a clipping mechanism in its objective function that acts as a penalty. Clipping prevents the ratio of the new to old policy probabilities from exceeding a predefined range, denoted by the hyperparameter. This range serves as a trust region, ensuring that updates are adequately significant to improve the policy but not large enough to cause performance degradation due to excessive changes. Furthermore, PPO demonstrated superior performance for various benchmarks. It achieves higher scores than its counterparts in a shorter training period, making it both efficient and effective. This efficiency is a key reason why PPO has become one of the most popular algorithms for reinforcement learning tasks. PPO uses the advantage function that is provided by Schulman et al. [27] as $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$, which allocates greater importance to relatively better actions at a state and decreases variance where s_t is the current state, a is the action, and $Q(s,a)$ is the estimated Q-value of taking action a in state s, and $V(s)$ is the estimated value of being in state s. PPO introduced its surrogate objective function in Equation (2).

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min(r(\theta)A_t, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right], \tag{2}$$

where:

- θ represents the parameters of the policy.
- $r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio of the action under the new and old policies.
- A_t is an estimator of the advantage function at time t.
- ϵ is a hyperparameter that defines the clipping range to constrain $r(\theta)$.

This objective function strikes a balance between taking larger policy steps for improvement and avoiding overly large updates that may lead to a decrease in performance. By focusing on this balance, PPO has carved out a niche in the sphere of reinforcement learning, demonstrating its capability to maintain steady advancement while avoiding the pitfalls of dramatic policy shifts. Its impact in the field is indicated by its considerable adoption and adaptation to various complex environments, highlighting its versatility and strength. Note that the exact form of the multi-agent objective function may vary based on the specific cooperative or competitive dynamics of the environment.

3.2. IPPO and MAPPO

IPPO and MAPPO are refined adaptations of the PPO algorithm designed for multi-agent situations. Schroeder et al. [28] conceptualized these algorithms to address cooperative multi-agent reinforcement learning (MARL) frameworks and to handle the complexities of Decentralized Partially Observable Markov Decision Processes (Dec-POMDP). Of the two, IPPO is noteworthy for its methodology, which allows each participating agent to independently employ a PPO algorithm. This independent application focuses on improving individual policies based on each agent's unique experiences in the environment. This approach allows agents to learn and adapt in an environment where they are unaware of their peers' actions, which offers significant advantages in non-stationary environments. The IPPO's embedded independent learning mechanism enhances the stability of the system by isolating each agent's policy shifts effectively, reducing the interdependencies between the agents. However, this approach, despite its simplicity and computational efficiency, may not fully capture the entire range of complexities linked to inter-agent interactions, particularly in situations that require high levels of cooperation or competition among agents. In IPPO, separate policies are learned for each agent, with an option for centralized critics. MAPPO is designed for situations where multiple agents work together on a shared task. It combines the advantages of Centralized Training and Decentralized Execution (CTDE) by utilizing a shared policy and critic function across all agents during training. This centralization fosters better collaboration amongst agents, as they learn to optimize collective outcomes through shared experiences. MAPPO's approach proves to be efficient in cooperative multi-agent environments wherein agents work together towards a common purpose. Both IPPO and MAPPO have showcased notable enhancements over conventional methods in various scenarios. The versatility of these algorithms is evident in their capability to adjust to numerous complexities of multi-agent interactions.

3.3. CoPO

CoPO [29] advanced the performance of the IPPO algorithm by introducing neighborhood rewards to act cooperatively with neighbor agents, inspired by Mean Field Theory [30]. CoPO introduced the Local Coordinate Factor (LCF), which controls the amount of concern neighborhood rewards. As the Local Coordinate Factor (LCF) increases, the system promotes more altruistic behaviors, while a decrease in LCF leads to more selfish actions being learned. CoPO maximizes for each agent advantages of neighborhood and individuals for each agent. Where the policy of the i-th agent is θ_i and the individual and neighborhood advantages are A^i_{Ind} and A^i_{Nei}, the objective function is as follows:

$$\begin{gathered} L_i(\boldsymbol{\theta}_i, \boldsymbol{P}) = E_{(s,a)\sim\pi_{\theta_i}(\cdot),\phi\sim P}\Big[\min(A^i r(\theta), A^i \cdot \text{clip}(1-\epsilon, 1+\epsilon, r(\theta)))\Big] \\ \text{for } i \in I \quad \text{where } A^i = A^i_{\text{Ind}}\cos(\phi) + A^i_{\text{Nei}}\sin(\phi) \end{gathered} \tag{3}$$

The LCF ϕ is derived from the local coordinate distribution $\boldsymbol{P(\mu_\phi, \sigma_\phi)}$. The parameters μ_ϕ, σ_ϕ are learned by the gradients computed from the sum of global rewards determined by $[\mu_\phi, \sigma_\phi]^T$. Note that ϕ is shared with all agents. CoPO showed better results than other DRL algorithms on Metadrive [31] multi-agent autonomous driving simulation tests in which the agents exhibited cooperative behavior. Note that ϕ is shared with all agents. CoPO showed better results than other DRL algorithms on Metadrive [31] multi-agent autonomous driving simulation tests in which the agents exhibited cooperative behavior.

4. Locality-Based Action-Poisoning Attack

We propose a novel type of adversarial attack in the training phase against MARL models. This attack is designed to be more difficult to detect and to have a significant impact with minimal manipulation. It assumes a scenario where the attacker manipulates one target agent's action among many. The attacker has only black-box access, where they cannot know the model's architecture or parameters and must execute an action-poisoning attack using only the target agent's observation. In MARL-based autonomous driving models, each agent references the position, speed, and other relevant factors of its neighboring agents to avoid collisions and navigate efficiently. Based on this fact, our proposed method assumes that an attacker manipulates the behavior of the target agent based on the behavioral information of the surrounding agent. The target action of the target agent flexibly changes with each time step according to the observation.

As shown in Figure 1, the proposed method assumes an attacker manipulates the action between the environment and the target agent. The target action is based on the action information of neighbors. This attack aims to change the action of a single agent such that it can affect the learning of all victim agents and prevent them from converging to an optimal policy π^* in the continuous control system of autonomous driving models.

The relationship between the attacker's target action and the actions of neighboring agents is shown in Figure 2. Here, the attacker selects a target action that differs significantly from the overall action of the neighboring agents in three ways. The detailed method for the proposed attack is presented in Algorithm 1. It should be noted that the success of an attack relies on the assumption that the agent references local information and that the victim agent is significantly affected by the unexpected actions of its neighbors. Therefore, target algorithms were selected for the proposed attack method, which determines the ego agent's action based on the neighboring agents' rewards or uses the mean action of neighboring agents as the action of an ego agent.

Algorithm 1: Locality-based Action-Poisoning Attack

```
Input: State space $S$, action space $\mathcal{A}$, time step $h$, set of agents $\mathcal{I}$, number of
       episodes $K$, number of timesteps $H$, agent index $i$, target policy $\pi$, LiDAR
       radius $\mathcal{L}_i$, hyperparameter $\omega$
Output: —
1  Initialize all state $s_h^i \in S$ and action $a_h^i \in \mathcal{A}$ for timestep $h$, set of agents $\mathcal{I}$ for
     episode $k$
2  for episode $k = 1$ to $K$ do
3      for timestep $h = 1, 2, \ldots, H$ do
4          forall agents $i \in \mathcal{I}$ do
5              if $i$ is the target agent then
6                  Observe state $s_h^i$, detect neighbor in $\mathcal{L}_i$
7                  if len(neighbors) $> \omega$ then
8                      Replace $a_h^i$ with poisoned action $a_h^{*i}$
9                  else
10                     $a_h^i = \pi(s_h^i)$
11             else
12                 $a_h^i = \pi(s_h^i)$
13             Environment receives action $a_h^i$ or poisoned action $a_h^{*i}$
```

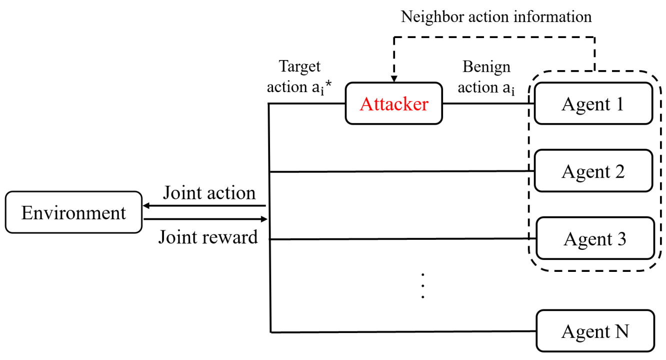


Figure 1. The attacker manipulates the action of the target agent between the environment and the agent. The attacker chooses the target action based on the action information of the neighboring agent. The neighbor shown in this figure is arbitrarily set for description. The attack action manipulated by the attacker is passed to the environment.

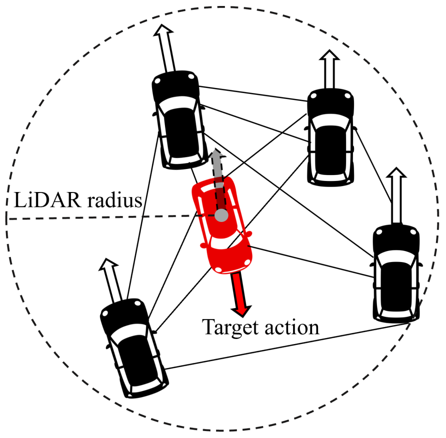


Figure 2. Here, the red car represents the target agent selected by the attacker, and the black car represents the neighbor agent detected within the radius of the target agent's LiDAR. The attacker selects the target action that deviates significantly from the average action (the gray arrow with a dotted line) of the neighboring agents.

4.1. Problem Formulation

Multiagent reinforcement learning is an effective method for training self-driving agents within a partially observable Markov decision process (POMDP) that limits observations of the environment. In real-world driving scenarios, vehicles have limited observation capabilities and must interact with other vehicles without having complete knowledge of their maps and traffic conditions. In a multiagent setting, this problem is defined as a multiagent partially observable Markov decision process (MAPOMDP). When each agent's decision making is independent, it is referred to as a decentralized POMDP (Dec-POMDP).

We define our problem as a Dec-POMDP represented as tuple $< S, \{\mathcal{A}_v, \mathcal{A}_t\} P, R, O_v, O_t, h, \gamma >$ and S is a set of all states, P is the transition probability, $R =< r_1, r_2, r_3, \cdots, r_n >$ is a tuple of rewards that each agent A_i received excluding the victim agent A_v. O_i means the observations of agent A_i, h as a time horizon, and γ is the discount factor. The model tries to find the Θ maximized objective in Equation (4) where $\mathbf{a}$ is a joint action of local or global agents, derived from policy π_θ.

$$J(\Theta) = \Sigma_{t=0}^{h} E_{s\in S, \mathbf{a}\sim\pi_\Theta(s)}(\gamma^t R(s, \mathbf{a}_t)) \tag{4}$$

Our action-poisoning attack changes the action of one target agent at each training time step, disturbing the convergence to the optimal policy π^* by reducing the rewards of its neighboring agents.

4.2. Attacker's Constraint and Capability

In an action-poisoning attack, the attacker disrupts the performance of the victim model by manipulating the actions of the target agent, which is under the attacker's full control. As the proposed method only allows black-box access, the attacker has no prior knowledge of the environment or victim model (architecture, parameters, etc.) and can know the states and actions generated during the interaction between all agents and the environment. In contrast to other attack methods that require direct access to the victim's observation space or reward function, an action-poisoning attack can effectively reduce the performance of the victim model by disrupting other agents through the actions of the target agent during the training phase. Therefore, this method has a more limited scope of requirements, as it does not require permission to access the victim agent's observations, environmental information, or the probability distribution of the victim agent's reward or actions. Consequently, the practical applicability of this attack is enhanced due to its reduced dependency on accessing extensive information about the victim agent, thereby broadening its potential utility in real-world scenarios.

4.3. Attack Actions

We defined the target actions to be used for the attack during the training phase and introduced indicators to determine the attack performance. The attacker manipulates action a^t that follows training policy π to the adversarial target action $a{*}^t$. The episode is represented by tuple $\tau =< s_h, \{a_h^v, a_h^t\}, r_h, \ldots, s_H, \{a_H^v, a_H^t\}, r_H >$ is poisoned, and the poisoned episode is represented by tuple $\tau =< s_h, \{a_h^v, a^*t_h\}, r_h, \ldots, s_T, \{a_H^v, a^*t_H\}, r_H >$. The poisoned episode data is used to train the victim model.

We propose three target actions for the MARL-based autonomous driving model based on the assertion that if an agent takes an action violating Reynolds' rules that is considered incompatible with its neighbors, it may interfere with the reward of a partial flock defined as its neighbors, lower efficiency and driving safety by considering the MARL-based autonomous driving model as a flocking model.

4.3.1. Anti-Correlated Action

Because our attack targets an algorithm that concerns the neighboring agent's reward or uses the mean action of neighboring agents, the mean action of the target agent's neighboring agent can provide important information. The neighboring agent of the target agent i^t is represented as a set of $neighbor = \{i_1, i_2, \ldots, i_N\}$ where $n = 1, 2, \ldots, N$ is the index of each neighboring agent and N is the number of neighboring agents within the LiDAR radius of the target agent. The attacker calculates the mean action of neighboring agents, consisting of the steering and acceleration values between -1 and 1 using Equations (5) and (6). $steering_n$ and $acceleration_n$ are the steering and acceleration value of each agent whose index is n, respectively.

$$steering_{mean} = \frac{\sum_{n=1}^{N} steering_{i_n}}{N} \tag{5}$$

$$acceleration_{mean} = \frac{\sum_{n=1}^{N} acceleration_{i_n}}{N} \tag{6}$$

During the training phase of the victim model, we can assume that each agent is likely to use driving velocities similar to their neighboring agents and are often positioned similarly at a specific time step t. Therefore, if an arbitrary agent executes the mean action vector of the neighboring agents $[steering_{mean},\ acceleration_{mean}]$, this strategy is less likely to significantly decrease the local reward, even if is not the optimal action, and it would be a reasonable strategy.

Based on this intuition, we propose the attacker's target action as $a_{i^t}^* = -1 \cdot [steering_{mean}, acceleration_{mean}]$, which is the opposite of the average action of neighboring agents. When the target agent performs poisoned action a', it can be expected to accelerate alone and change direction rapidly in situations where neighboring agents are driving in a similar direction without strong acceleration. This may cause the target agent to receive a reward lower than the average reward for its neighboring agents at a certain time step t.

4.3.2. Human-like Disruptive Action

The MARL-based autonomous driving model encounters human drivers on the road after training. In the presence of other drivers or pedestrians, unexpected actions from human drivers, such as turning the steering wheel sharply, sudden braking, or accelerating in anger, can be difficult for machine learning models to interpret and predict. Therefore, we propose an attacker's target action that mimics a human driver's disruptive action in order to confuse neighboring agents.

The Attacker calculates weighted sum $v_{neighbor}$ with high weight on one agent in the neighbor set $\{i_1, i_2, \ldots, i_N\}$ and with the hyperparameter α, β, ϵ chosen by the attacker, and the target action of the target agent i^t is manipulated as follows, where v_{i^t} is velocity of target agent and $v_{neighbor}$ is the velocity of its neighbors.

$$cos(\theta) = \frac{v_{neighbor} \cdot v_{i^t}}{\|v_{neighbor}\| \|v_{i^t}\|} \tag{7}$$

$$steering_{i^t} = \alpha \times cos(\theta),\ acceleration_{i^t} = \beta \times \frac{1}{1 + |cos(\theta)|} \tag{8}$$

Target action vector $a^*_{i^t} = [steering_{i^t}, acceleration_{i^t}]$ is a dynamic target action that adjusts according to the cosine similarity between v_{target} and $v_{neighbor}$ and hyperparameters. It simulates a human driver's disruptive action by accelerating more when neighbors move similarly, and steering more when neighbors move dissimilarly, to disturb the alignment of traffic flow.

4.3.3. Random Action

Since the proposed method targets the MARL-based autonomous driving model with a continuous action space, the action space is randomly selected from points divided at regular intervals for convenience. Steering and acceleration, which are components action $a_{i_n} = [steering_{i_n}, acceleration_{i_n}]$ of a agent i_n with index n, are real numbers between –1 and 1, and the interval from –1 is divided into 500 points including both endpoints. The target action of the target agent is randomly selected from 500 points, which may be the same as or similar to normal actions updated by the policy before the attacker manipulates them with a low probability, meaning the random action will have a different value from normal actions and act as an adversarial target action that can lower the local rewards. Thus, we define this as one of the target actions chosen by attacker.

5. Experiments

5.1. Environmental Settings

We selected the three victim models proposed by Peng et al. [29] for our attack: modified IPPO, MFPO, and CoPO. The IPPO algorithm maximizes only an individual's objective function, and each agent's policy is updated in a direction that takes selfish actions in autonomous driving simulations. However, our proposed attack replaces only one agent's actions with adversarial target actions to interfere with the decision making of neighboring agents and undermine the performance of the entire model. Therefore, we believe that the original IPPO algorithm, which only maximizes individual objectives, will not be affected by our attack. In contrast, Peng et al. [29] also proposed the modification of the IPPO algorithm to maximize the reward of neighboring agents within a certain radius, including individuals, and we selected this modified IPPO algorithm as one of our victim models.

The deployment of various rewards constitutes a fundamental element in training the target models. Agents receive a reward of 10 when they successfully reach their destination without any accidents. In contrast, a penalty of 10 is imposed if they collide with other agents or deviate from the road. Additionally, a safe driving reward of 1 is awarded at each time step, and a further reward of 0.1 is granted based on the speed of the agent. Therefore, each agent learns the behavior of efficiently reaching their destination without getting off the road or colliding with another agent.

The victim model was trained in a Metadrive simulator environment, which supports several maps for generalizable studies in multi-agent autonomous driving systems and provides realistic physical simulations that can be operated on lightweight hardware configurations. To conduct our attacks in various environments, we used intersection, roundabout, and bottleneck road environments implemented by MetaDrive.

- Intersection: An environment where four lanes cross in the center and there are no signs to control traffic signals. Vehicles can pass through both directions and make U-turns at intersections, and 30 agents were created when the map was initialized.
- Roundabout: Two-lane round-trip environment with four crossing roads. Agents on the road were allowed to freely join or leave the intersection, and 40 agents were created when the map was initialized. This system can accommodate a relatively large amount of traffic.
- Bottleneck: In the middle of the map, traffic flow is slowed down because of the reduced width of the road, and all vehicles must yield to each other to pass through the bottleneck. When the map was initialized, 20 agents were created.

The CoPO and modified IPPO models were trained for one million-time steps in the intersection, roundabout, and bottleneck environments, whereas the MFPO model was trained for 0.8-million-time steps in the same environments. The SGD minibatch was set to 512 with a learning rate of 0.0003, and the LiDAR radius determining the neighboring agent was set to 10 m before training, following the settings used in [32]. We designated one agent as the target agent, performed target actions in an environment with 20 to 40 agents, and trained the victim model. We then observed the performance changes in the model before and after the attack to evaluate the influence of the attack.

We set the agent with index 2 as the target agent. During the training phase, the target agent performed anti-correlated, human-like disruptive and random actions to poison the episode data whenever more than four neighboring vehicles were detected in its LiDAR. The SGD mini-batch was set to 512, the learning rate was set to 0.0003, and the determining radius of the neighboring agent was set to 10 m, similar to the settings of the victim model.

5.2. Evaluation Metric

The attacker aims to manipulate the action of a target agent and observe the resulting impact on the remaining agents and victim model. The success rate, crash rate, and out rate are used as evaluation metrics to compare the performances of victim models trained without attacks with those trained with poisoned data. The success rate, crash rate, and output rate indicate the percentage of episodes in which each agent either reached the destination or failed to do so owing to a crash or road-departure accident. The goal of the attacker is to decrease the success rate of the victim model, while increasing the crash and output rates.

5.3. Experimental Results

5.3.1. Locality-Based Action-Poisoning Attack on CoPO

The results of the locality-based action-poisoning attack on the victim model CoPO are presented in Table 1 and Figure 3, showing a success rate of up to one million-time steps. Figure 3a–c show graphs of the success rates of the CoPO model according to the learning time steps for the intersection, roundabout, and bottleneck maps, respectively. In graph (a), (b), and (c), the x-axis represents the time steps from 0 to one million, and the y-axis represents the success rate. (d), (e), and (f) display graphs showing the success, out, and crash rates at one million time steps of the victim model for the intersection, roundabout, and bottleneck maps.

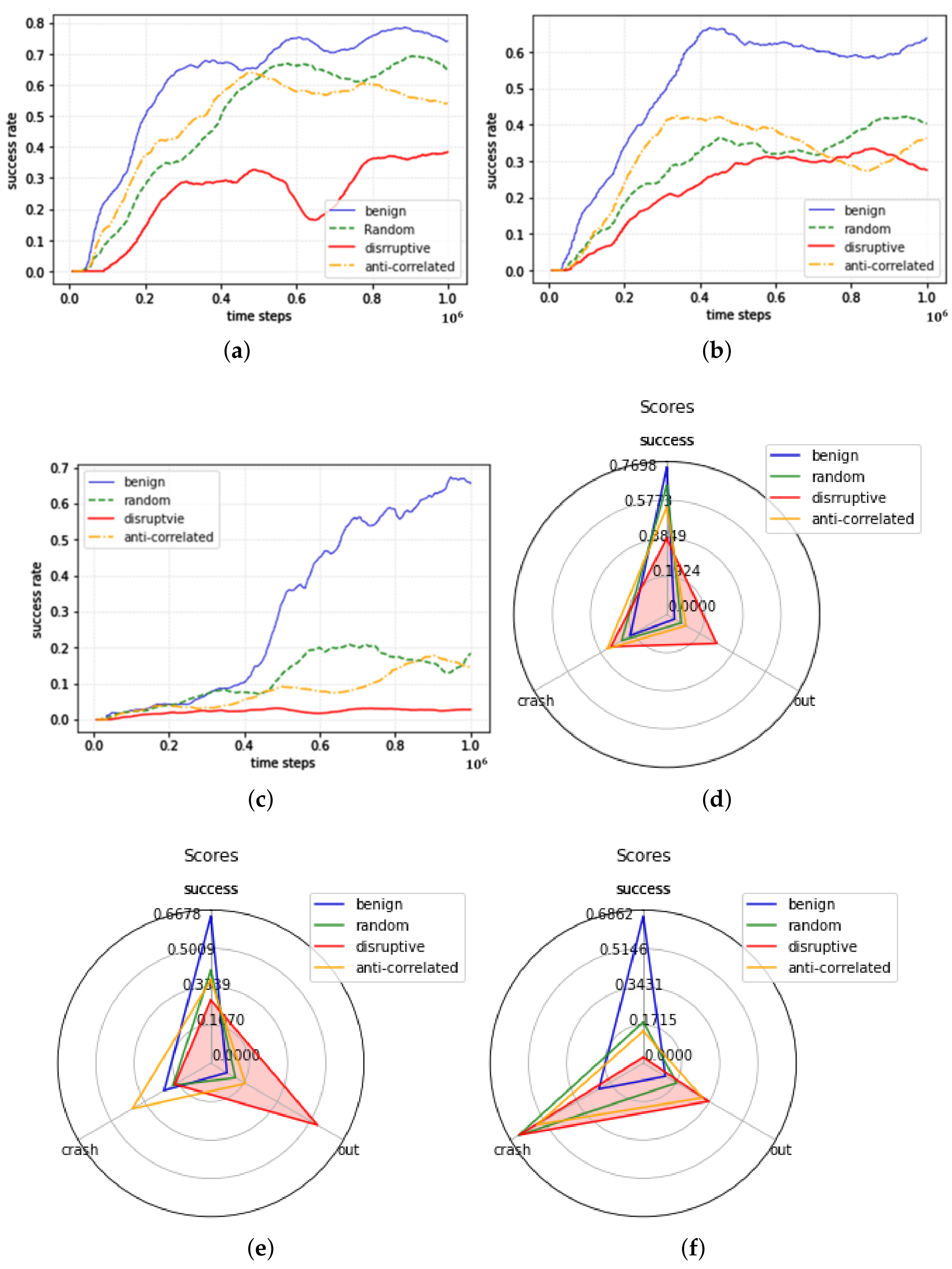


Figure 3. Performance Graphs of locality-based action-poisoning attack in CoPO.

In the case of the CoPO model trained in the intersection environment, the model exhibited the best training performance when the target agent executed a benign action and the worst training performance when the target agent executed a human-like disruptive action. Both random and anti-correlated actions were successful in attacking the model, with anti-correlated actions being slightly more effective than random actions. In addition, the human-like disruptive action resulted in the most significant reduction in the performance of the victim model, as the out rate significantly increased compared to other actions, and the anti-correlated action significantly increased the crash rate.

In a roundabout environment, the model showed the best performance in terms of the success rate when the target agent executed a benign action, and the worst training performance was observed when the target agent executed a human-like disruptive action. Because both random and anti-correlated actions successfully reduced the success rate of the victim model, all three target actions can be seen as hindering the training of the victim model.

Table 1. Locality-based action-poisoning attack performance for different target actions in CoPO.

	Intersection			**Roundabout**			**Bottleneck**		
	Success	**Crash**	**Out**	**Success**	**Crash**	**Out**	**Success**	**Crash**	**Out**
Benign Action	0.7398	0.2141	0.0458	0.6378	0.2355	0.0807	0.6562	0.2230	0.1154
Random Action	0.6501	0.2636	0.0852	0.4035	0.1907	0.1249	0.1845	0.6462	0.1739
Anti-correlated Action	0.5402	0.4578	0.1126	0.3626	0.3947	0.1746	0.1443	0.5500	0.3082
Human-like Disruptive Action	0.3838	0.3260	0.2922	0.2750	0.1817	0.5373	0.0276	0.6416	0.3397

Additionally, it can be observed that human-like disruptive actions significantly increased the out rate, whereas anti-correlated actions significantly increased the crash rate. The locality-based action-poisoning attack on the CoPO model had the greatest effect on the bottleneck environment. The performance of the original model in a bottleneck environment was significantly reduced by approximately 97% when the target agent performed a human-like disruptive action. Attacks using random and anti-correlated actions successfully reduced the training performance of the victim model. In the case of human-like disruptive actions, the crash and out rates were the highest. Because a bottleneck environment has a narrower road section than other maps, if the target agent performs a target action in a narrow lane, the impact will be more fatal than that in other environments, leading to a higher crash rate. The success, crash, and out rates achieved by the target action are shown in Figure 3, and in all of the three environments, the crash and out rates increased, whereas the success rate decreased (Figure 3).

Compared to the other target algorithms considered in this study, CoPO was more vulnerable to locality-based action-poisoning attacks. The CoPO algorithm incorporates the local coordination factor (LCF), which determines whether the agent acts in a selfish or collaborative manner during the training process. There is a possibility that the LCF may be disturbed when an action performs the target action chosen by the attacker. This finding indicates that even a well-designed algorithm can be vulnerable to the proposed attack method, underscoring the need for robust defense methods against potential attacks as agents become increasingly sensitive to the information of their neighbors.

5.3.2. Locality-Based Action-Poisoning Attack on IPPO

The results of the locality-based black-box action-poisoning attack on the modified IPPO victim model are presented in Table 2 and Figure 4, which shows the success, crash, and out rates at one million training time steps. Figure 4a–c show graphs of the success rates of the IPPO model according to the learning time steps for the intersection, roundabout, and bottleneck maps, respectively. In graph (a), (b), and (c), the x-axis represents the time step from 0 to one million, and the y-axis represents the success rate. (d), (e), and (f) display graphs showing the success, out, and crash rates at one million time steps of the victim model for the intersection, roundabout, and bottleneck maps.

In the intersection environment, the human-like disruptive action showed a slightly higher performance than the original model, with success rates of 0.6998 and 0.6627, respectively, indicating that the attack was unsuccessful for the corresponding target action. However, random and anticorrelated actions successfully attacked the original model, thus reducing its success rate. The anti-correlated action was the most effective in increasing the crash rate, whereas the random action was the most effective in increasing the out rate.

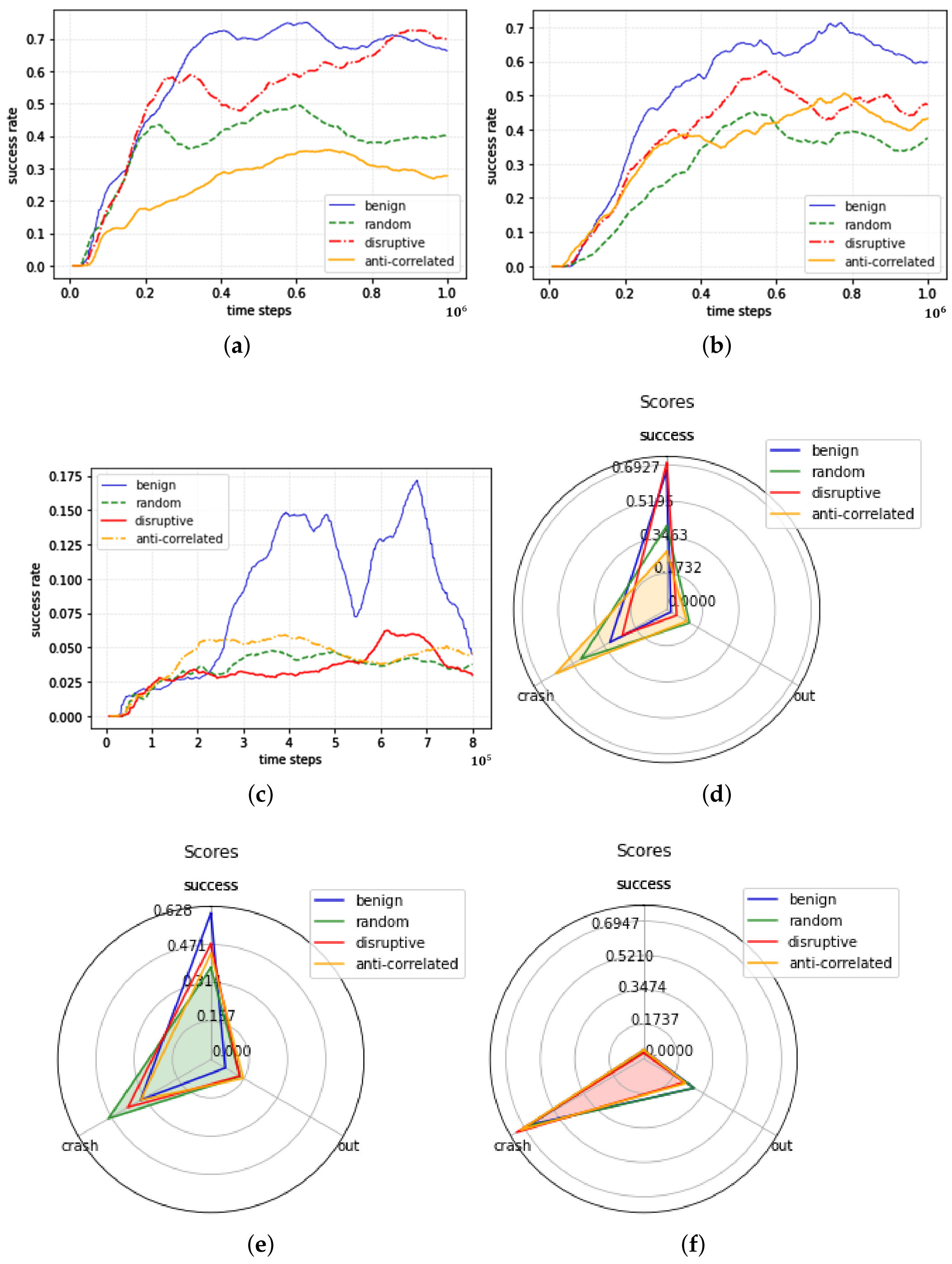


Figure 4. Performance graphs of locality-based action-poisoning attack in IPPO.

Table 2. Locality-based action-poisoning attack on IPPO.

	Intersection			**Roundabout**			**Bottleneck**		
	Success	**Crash**	**Out**	**Success**	**Crash**	**Out**	**Success**	**Crash**	**Out**
Benign Action	0.6627	0.3153	0.0226	0.5981	0.3335	0.0689	0.0452	0.6647	0.2939
Random Action	0.4002	0.4738	0.1270	0.3760	0.4850	0.1391	0.0379	0.6780	0.2937
Anti-correlated Action	0.2771	0.6137	0.1109	0.4324	0.3324	0.1545	0.0457	0.7159	0.2559
Human-like Disruptive Action	0.6998	0.2456	0.0556	0.4725	0.3929	0.1362	0.0298	0.7395	0.2357

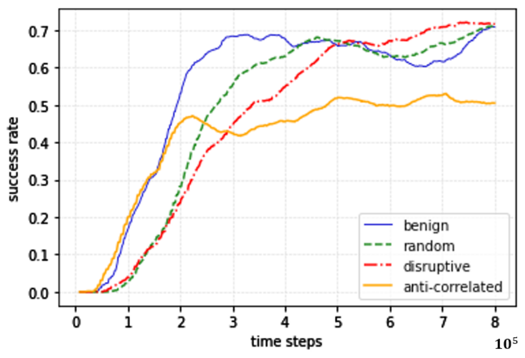
In a roundabout environment, all three target actions forced the victim model to learn a suboptimal policy, resulting in a lower success rate compared to the original model.

In a bottleneck environment, evaluating the performance of the proposed attacks was difficult because the original model trained without the attack exhibited poor performance. However, while training, we observed that the peak of the success rate of the original model were higher than those of the poisoned model. Therefore, we can infer that the original model is more likely to perform better than the poisoned model. However, considering the success rate graph, the original model is likely to find a better action compared to the poisoned model because the peak of the graph of the original model is higher than that of the poisoned model.

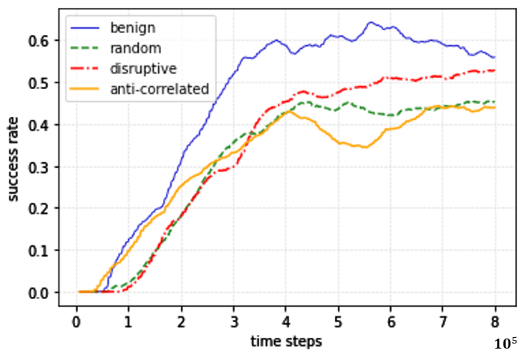
5.3.3. Locality-Based Action-Poisoning Attack on MFPO

For the locality-based black-box action-poisoning attack on the MFPO model, the success, crash, and out rates at 800,000 training time steps are shown in Table 3 and Figure 5. Figure 5a–c show graphs of the success rates of the MFPO model according to the learning time steps for the intersection, roundabout, and bottleneck maps, respectively. In graph (a), (b), and (c), the x-axis represents the time steps from 0 to 800,000, and the y-axis represents the success rate from 0 to 1. Graph (d), (e), and (f) display graphs showing the success, out, and crash rates at one million time steps of the victim model for the intersection, roundabout, and bottleneck maps. Although the number of time steps is different due to differences in the models, we trained the same number of episodes as CoPO and IPPO. In the intersection environment, the success rates of the random and human-like disruptive actions executed by the target agent were similar to those of the original model; therefore, the attack was unsuccessful. However, the anti-correlated action effectively reduced the success rate by 28.88%. For all the target actions, the out rate increased to a similar level, and the anticorrelated action significantly increased the crash rate of the victim model. In the roundabout environments, anticorrelated and human-like disruptive actions were effective attack methods. In the bottleneck environment, all three target actions can be used for effective attacks. Although the original model's performance is lower, the graph in Figure 5 indicates a higher peak, suggesting that the original model may learn more rapidly compared to the poisoned model.

In diverse driving environments like intersection, roundabout, and bottleneck, the unpredictable nature of agent interactions poses a significant challenge. The introduction of unpredictable actions of target agent in these settings adds to the complexity, making it difficult for other agents to respond effectively. This not only disrupts their immediate decision making but also hampers the overall learning process of the multi-agent system. Such disruptions impede the agents' ability to adapt and converge towards optimal driving behaviors, thereby affecting the safety of the autonomous driving system.

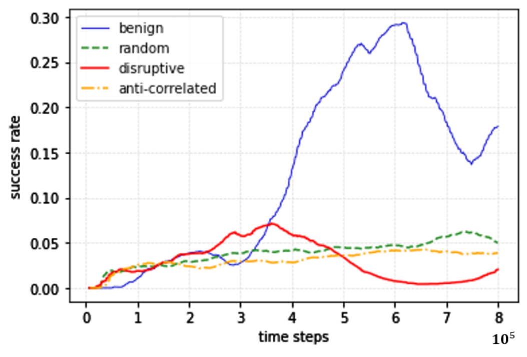


(**a**)

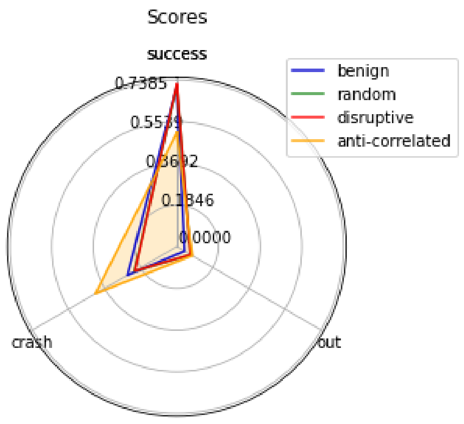


(**b**)

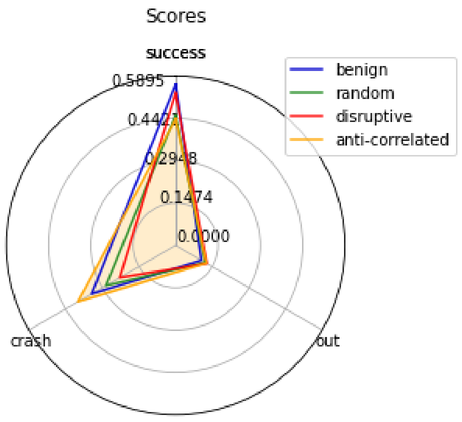
Figure 5. *Cont.*



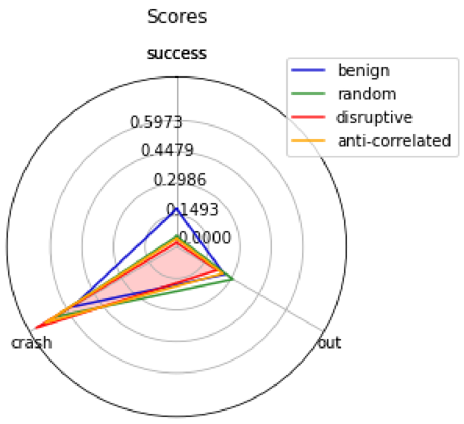
(**c**)



(**d**)



(**e**)



(**f**)

Figure 5. Performance Graphs of locality-based action-poisoning attack in MFPO.

Table 3. Locality-based action-poisoning attack performance by target actions in MFPO

	Intersection			**Roundabout**			**Bottleneck**		
	Success	**Crash**	**Out**	**Success**	**Crash**	**Out**	**Success**	**Crash**	**Out**
Benign Action	0.7085	0.2530	0.0393	0.5595	0.3372	0.1042	0.1788	0.5673	0.2619
Random Action	0.7079	0.2170	0.0714	0.4525	0.2798	0.1192	0.0499	0.6634	0.3075
Anti-correlated Action	0.5048	0.4165	0.0796	0.4394	0.3927	0.1238	0.0387	0.7163	0.2526
Human-like Disruptive Action	0.7175	0.2142	0.0737	0.5296	0.2239	0.1257	0.0205	0.7663	0.2206

6. Discussion

Our study has several limitations, including an investigation into only specific types of attack methods. therefore, it may be insufficient to capture the potential attack scenarios and risks of autonomous driving models. In addition, we did not simulate a situation in which an attacker attacks multiple agents simultaneously using the proposed method. MARL algorithms, which are less sensitive to state information from local rewards or neighboring agents, can be robust to locality-based action-poisoning attacks based on the premise that neighboring agents will be disturbed by the unexpected action of the target agent. Despite these limitations, the possibility of diminished learning performance due to attacks underscores the need for research into defensive strategies against the proposed methods.

Adversarial training is the first method to mitigate the locality-based action-poisoning attack proposed in this study. Adversarial training is a technique that intentionally adds difficult or misleading examples to the training data to make the model more robust. During this process, the AI system was exposed to adversarial examples, which are inputs specifically designed to confuse the model, alongside correct responses, so that the model can learn to withstand the types of attacks proposed in this study. By learning from these maliciously modified inputs, the model becomes less susceptible to malicious inputs that can lead to incorrect outputs. The second method is to train a robust model. Robust model training includes machine-learning algorithms and techniques that are less sensitive to noisy data. This can include incorporating regularization methods that penalize complexity to prevent overfitting to poisoned data. Robust training can also use bootstrapping methods and ensemble learning, where multiple models or multiple versions of a dataset are used. The third method is anomaly detection, which involves identifying data points or patterns within a dataset that deviate significantly from the norm. To protect AI models from poisoning attacks, anomaly-detection systems are designed to discover and flag unusual or suspicious inputs that may indicate tampering or manipulation of the data. If patterns can be identified in the behavioral data of an adversarial agent that deviate significantly from the norm for neighboring agents and that agent's data can be excluded from training, this system may be able to mitigate the risk of the proposed attack.

In addition, it is important to recognize that autonomous driving systems are vulnerable to a wider array of attacks beyond those we have focused on. These include sensor spoofing, cyber–physical system attacks, data tampering, network-based attacks, system malfunctions, and human factors, each posing significant risks to safety and reliability. Understanding and mitigating these broader risks are crucial for the advancement of autonomous driving technologies, necessitating ongoing research and development to bolster system security and resilience.

7. Conclusions

Despite the growing awareness of the vulnerability of reinforcement learning to adversarial attacks, studies on adversarial attacks against MARL model are still insufficient, particularly in the absence of previous studies on action-poisoning attacks in MARL models with a continuous action space that can be applied to the continuous control systems of autonomous driving models. Therefore, we propose a novel attack method called the "locality-based action-poisoning attack" to evaluate the vulnerability of autonomous driving models. In contrast to previous action-poisoning attack studies that only covered single-agent settings or assumed that the attacker was allowed white-box access to the victim model, we propose a black-box action-poisoning attack on the MARL model with a continuous action space using a simple idea.

The core contribution of our work lies in demonstrating that even without in-depth access to the model's internal workings, an attacker can significantly disrupt the learning process and force the target model to learn wrong action patterns. The actions we proposed—diverging from neighboring agents, mimicking disruptive human behavior, and random actions—each highlight potential weaknesses in MARL systems. This insight is pivotal, as it suggests that MARL systems, despite their complexity and sophistication, can still be susceptible to relatively straightforward adversarial strategies.

The proposed attack method was tested through a simulation using three algorithms, CoPO, IPPO, and MFPO, in three driving environments, intersections, roundabouts, and bottlenecks, to ensure some level of generalization across different driving scenarios and algorithms. Although the performance of the attack method varies depending on the target algorithm and environment, our results demonstrate that the proposed attack method is significant. We expect that this study will serve as a fundamental basis for analyzing the potential risks of autonomous driving models. As shown in related studies and experiments, the developer of a MARL model needs to develop various defense methods that consider the characteristics of the learning algorithm.

From a practical standpoint, our study acts as a pivotal alert for the autonomous driving sector, highlighting the imperative for advanced defensive mechanisms against such poisoning attacks. This underlines that security in autonomous driving necessitates not only physical safeguards but also requires equally strong digital defence systems.

Future research should focus on developing countermeasures that can detect and mitigate the effects of such attacks in real-time, ensuring that autonomous vehicles remain safe and reliable even in situations where they may be exposed to adversarial attacks. Additionally, while our study was conducted within a limited range of scenarios, this highlights the opportunity and necessity for future research to explore simulations of more diverse and sophisticated poisoning attack methods. This approach will further enhance understanding against potential vulnerabilities in autonomous driving systems.

Author Contributions: Conceptualization, Y.A., W.Y, and D.C.; methodology, Y.A, W.Y. validation, Y.A.; formal analysis, Y.A. and W.Y.; writing—original draft preparation, Y.A. and W.Y.; data curation W.Y.; writing—review and editing, Y.A.; visualization, Y.A. and W.Y.; project administration, D.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020R1A2C1014813) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A4A1029650).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Puterman, M.L. Markov decision processes. In *Handbooks in Operations Research and Management Science Handbook*; Elsevier: Amsterdam, The Netherlands, 1990; Volume 2, Chapter 8, pp. 331–434.
2. Singh, B.; Kumar, R.; Singh, V.P. Reinforcement learning in robotic applications: A comprehensive survey. *Artif. Intell. Rev.* **2021**, *55* , 945–990. [CrossRef]
3. Ravi Kiran, B.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A.A.; Yogamani, S.; Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transp. Syst.* 2022, *23*, 4909–4926. [CrossRef]
4. Hu, D.; Yang, R.; Zuo, J. Application of deep reinforcement learning in maneuver planning of beyond-visual-range air combat. *IEEE Access* **2021**, *9*, 32282–32297. [CrossRef]
5. Chen, T.; Liu, J.; Xiang, Y. Adversarial attack and defense in reinforcement learning-from AI security view. *Cybersecurity* **2019**, *2*, 1–22. [CrossRef]
6. Wang, Z.; Ma, J.; Wang, X. Threats to training: A survey of poisoning attacks and defenses on machine learning systems. *ACM Comput. Surv.* **2022**, 5, 1–36. [CrossRef]
7. Liu, G.; Lai, L. Action-manipulation attacks against stochastic bandits: Attacks and defense. *IEEE Trans. Signal Process.* **2022**, 68, 5152–5165. [CrossRef]
8. Liu, G.; Lai, L. Provably efficient black-box action poisoning attacks against reinforcement learning. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS), Online, 6–14 December 2021.
9. Böhmer, W.; Kurin, V.; Whiteson, S. Deep coordination graphs. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 12–18 July 2020; Volume 119, pp. 980–991.
10. Zemzem, W.; Tagina, M. Cooperative multi-agent systems using distributed reinforcement learning techniques. In *Procedia Comput. Sci.* **2018**, *126*, 517–5261. [CrossRef]
11. Schmidt, L.M.; Brosig, J.; Plinge, A.; Eskofier, B.M.; Mutschler, C. An introduction to multi-agent reinforcement learning and review of its application to autonomous mobility. In Proceedings of the IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), Macau, China, 8–12 October 2022; pp. 1342–1349.
12. Reynolds, C.W. Flocks, herds, and schools: A distributed behavioral model. In Proceedings of the SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, Anaheim, CA, USA, 27–31 July 1987; pp. 25–34.
13. Busoniu, L.; Babuska, R.; De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst.* **2008**, *38*, 156–172. [CrossRef]
14. Wong, A.; Bäck, T.; Kononova, A.V.; Plaat, A. Deep multiagent reinforcement learning: Challenges and directions. *Artif. Intell. Rev.* **2022**, *56*, 5023–5056. [CrossRef]
15. Li, Y.; Jiang, Y.; Li, Z.; Xia, S.T. Backdoor learning: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 5–22. [CrossRef]
16. Lin, Y.C.; Hong, Z.W.; Liao, Y.H.; Shih, M.L.; Liu, M.Y.; Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17), Melbourne, Australia, 19–25 August 2017; pp. 3756–3762.

17. Ma, Y.; Zhang, X.; Sun, W.; Zhu, J. Policy poisoning in batch reinforcement learning and control. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 8–14 December 2019.
18. Rakhsha, A.; Radanovic, G.; Devidze, R.; Zhu, X.; Singla, A. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In Proceedings of the 37th International Conference on Machine Learning (PMLR), Virtual, 13–18 July 2020; Volume 119, pp. 7974–7984.
19. Zhang, X.; Ma, Y. ; Singla, A.; Zhu, X. Adaptive reward-poisoning attacks against reinforcement learning. In Proceedings of the 37th International Conference on Machine Learning (PMLR), Virtual, 13–18 July 2020; Volume 119, pp. 11225–11234.
20. Sun, Y.; Huo, D.; Huang, F. Vulnerability-aware poisoning mechanism for online rl with unknown dynamics. In Proceedings of the Ninth International Conference on Learning Representations (ICLR), Virtual, 3–7 May 2021.
21. Mahajan, A.; Teneketzis, D. Multi-armed bandit problems. In *Foundations and Applications of Sensor Management*; Springer: Boston, MA, USA, 2008; pp. 121–151.
22. Zheng, H.; Li, X.; Chen, J.; Dong, J.; Zhang, Y.; Lin, C. One4all: Manipulate one agent to poison the cooperative multi-agent reinforcement learning. *Comput. Secur.* **2023**, *124*, 103005. [CrossRef]
23. Wang, W.; Wang, L.; Zhang, C.; Liu, C.; Sun, L. Social interactions for autonomous driving: A review and perspectives. *Found. Trends Robot.* **2018**, *10*, 198–376. [CrossRef]
24. Khan, S.; Momen, S.; Mohammed, N.; Mansoor, N. Patterns of flocking in autonomous agents. In Proceedings of the 2018 International Conference on Intelligent Autonomous Systems (ICoIAS), Singapore, 1–3 March 2018.
25. Park, Y.; Kuk, S.; Han, S.; Lim, S.; Kim, H. Flocking-based cooperative autonomous driving using vehicle-to-everything communication. *Electron. Lett.* **2019**, *55*, 535–537. [CrossRef]
26. Chater, N.; Misyak, J.; Watson, D.; Griffiths, N.; Mouzakitis, A. Negotiating the traffic: Can cognitive science help make autonomous vehicles a reality? *Trends Cogn. Sci.* **2018**, *22*, 93–95. [CrossRef]
27. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
28. de Witt, C.S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.; Sun, M.; Whiteson, S. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv* **2020**, arXiv:2003.09815.
29. Peng, Z.; Li, Q.; Hui, K.M.; Liu, C.; Zhou, B. Learning to simulate self-driven particles system with coordinated policy optimization. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS), Online, 6–14 December 2021.
30. Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; Wang, J. Mean field multi-agent reinforcement learning. In Proceedings of the the 35th International Conference on Machine Learning (PMLR), Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 5571–5580.
31. Li, Q.; Peng, Z.; Feng, L.; Zhang, Q.; Xue, Z.; Zhou, B. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *arXiv* **2023**, arXiv:2301.03461.
32. Oliehoek, F.A.; Amato, C. *A Concise Introduction to Decentralized POMDPs*; Springer: Berlin/Heidelberg, Germany, 2015.