The Feasibility of Detecting Adversarial Attacks in Multi-Agent Reinforcement Learning

October 8, 2021

Abstract

Cooperative Multi-Agent Reinforcement Learning (c-MARL) enables a team of agents to collaboratively determine the global optimal policy that maximizes the sum of their local accumulated rewards. This paper examines a common vulnerability in MARL algorithms: the adversarial perturbation. One newly-engineered adversarial attack occurs when a malicious user compromises an agent, directly controls the *compromised agent*, and subsequently pushes its cooperative agents to act off-policy. This research will investigate multiple machine learning techniques that can detect adversaries in multi-agent environments. It will present a novel stacked-LSTM ensemble approach to detecting and mitigating such attacks, comparing this new proposed detection subnetwork to other techniques. Finally, this paper will critique and undermine the feasibility of maintaining and ensuring the reliability of such detection techniques.

1 Introduction

In recent years, significant advances in the usage and implementation of reinforcement learning (RL) have occurred in a variety of different applications. Agents learning to play card games like Go and Poker, advancements in autonomous driving systems, and a multitude of robotics innovations all center around RL algorithms that optimize the training of more than one agent. Naturally, they have emerged as a subfield of RL known as multi-agent reinforcement learning (MARL) systems.

The literature has shown that many MARL policies are especially vulnerable to adversarial attacks.[6] With reinforcement learning's major implications in artificial intelligence applications ranging from computer-driven games to connected autonomous vehicles [6], the reliability and security of these algorithms is of utmost importance. Because of the common goal of cooperation or competition (or both) between agents within many MARL systems [4], agents react based on the actions of others within the same environment. For example, connected autonomous vehicles must interact with other vehicles, pedestrians, and infrastructure, considering the actions and reactions of others before acting on its own [16]. For this reason, it only takes one corrupted agent to cause major security issues. Adversaries that intentionally aim to defect targeted networks can confuse agents and lead them to make mistakes that can result in poor performance and even harm to humans that rely on these systems [14]. Often, the noise added by corrupted agents to fool MARL systems are invisible to humans [6, 14], creating the critical need for reliable machine learning algorithms to detect and mitigate these attacks. This paper assesses popular classification approaches (namely deep learning, support vector machines, k-nearest neighbors, and random forests) and their ability to detect adversarial input at test-time. Afterwards, their is discussion on the feasibility of maintaining such systems to ensure safety and reliability, concluding that many detection subnetworks are vulnerable to similar attacks as the multi-agent systems they are meant to protect. To show this, the fast gradient sign method is applied to fool the proposed ensemble model, suggesting the infeasibility of deep-learning-based detectors.

2 Related Work

2.1 Adversarial vulnerability in MARL

Research and applications surrounding MARL systems has been popularized in the literature in the past few years [3, 33, 34]. Heightened interest in RL systems has brought about a multitude of papers addressing the adversarial vulnerability evident in many cooperative [16] and competitive [14] settings. Huang et al. first presented the proof of concept that an adversary can interfere with the operation of an RL agent [14]. In research presented by Gleave et al., attacks on uncompromised agents can occur through a malicious user controlling one corrupted agent [1]. In an attempt to coordinate its actions with other agents in the system, an uncompromised agent can be directly influenced by its malicious adversaries [4]. Alqahtani et al. [2] were

able to propose three successful attacks in both white and black-box settings on two separate MARL particle environments. Using the true reward function in the white-box and an approximated state-transition model in the black-box attacks, these attacks dropped cooperative team reward by over 85 percent [2].

To detect and mitigate such attacks that are occasionally invisible to the human eye [6, 14], machine learning techniques are especially valuable. While much research involves investigations into how to prevent adversarial examples from fooling MARL policies with methods like adversarial training [1, 11, 6], data augmentation and randomization [31] and detector subnetworks [6, 19], this research will test popular machine learning approaches and their ability to detect adversarial attacks when they have already occurred.

2.2 Deep learning for detecting anomalies

Deep learning and the use of neural networks to conceptualize tasks has become a hot topic in the literature. However, due to some limitations of a feed-forward neural network's ability to remember information from previous frames of data, a Long-Short Term Memory Neural Network (LSTM) was developed using real-time recurrent learning [30]. LSTMs mitigate this vanishing-gradient problem through their implementation of various recurrent cycles from subsequent neurons to preceding ones, creating hidden layers–especially conducive to time-series data [18, 30]– to act like memory. Using input, activation, forget, and output gates, models allow for long-term memory storage [18].

LSTMs demonstrate a viable technique to predict normal time-series behavior that consequently classifies anomalous behavior without real knowledge of the domain of the data [30, 27]. Compared to other deep learning techniques, LSTM based prediction models may give better results and performance [18]. Much work has been done to prove that LSTMs are extremely effective in classifying anomalous behavior [19] in various domains ranging from medicine [30] to computer network traffic [27]. In addition, LSTM networks can go even further, differentiating different categories of anomalous behavior from normal behavior [30]. Using prediction error distributions as a baseline [18], LSTM networks are perfectly suited to analyze sequential data with temporal dynamics [30]. Stacked LSTM neural networks have been found in many works to show exceptional performance in detecting anomalies [30, 18, 21], demonstrating the promise of deep learning technologies in security applications.

Malhotra et al. [18] claim that LSTM-based prediction models, where the output is the expected action or value, may give better results than other detection subnetworks. This detection algorithm learns a threshold that maximizes the f1-score between anomalous and normal datapoints. Then, after training on normal datasets, an LSTM prediction model calculates the error vector of predicted and true values. Fitting this to a multi-variate Gaussian distribution gives an output of P^t. If $P^t < threshold$, then it will be classified as anomalous [18]. Extending this, Verner [30] found that LSTMs can differentiate between different classes of anomalous behavior.

Naseer et al. [21] proposed a simpler deep learning detector subnetwork for intrusion detection systems. Here, their binary LSTM models classified datapoints 0 as normal and 1 as anomalous. These models outperformed other machine learning classifiers. Overall, the temporal element of MARL systems clearly benefits from LSTM-based detector subnetworks.

2.3 Other machine learning techniques for detecting anomalies

Support vector machines (SVMs) are one of the most common machine classifiers for binary data. The inner workings of these machines relies on input vectors, a distance metric hyperparameter, and labeled points. Assuming that the data can be linearly separable, a linear decision surface is constructed. This decision surface is used to classify data by finding the best (largest) hyperplane that separates all data points of one class from those of another class, without interior data points. Once that decision boundary is learned, datapoints are plotted and then classified based on which half of the decision boundary it falls [7].

The majority of SVM use to detect anomalies exists in the field of network instrusion detections. Basic SVMs can reliably classify anomalous data due to its binary nature, assuming that the training data are independently and identically distributed [13, 9, 26, 29]. In Hu, Liao, and Vemuri [13], robust SVMs strengthen the case for SVM classifiers due to its independence from a clean training dataset. Using robust-SVMs, training can be deployed using real-world data that could include anomalous behavior and noisy clean behavior. Especially useful in high-dimensional input spaces, robust-SVMs address overfitting by minimizing the soft margin previously used to reduce the number of interior datapoints [13]. Other approaches combine SVMs with more complex machine learning techniques in order to reduce the number of dimensions fed into the input vector. Using principle component analysis or genetic algorithms to perform feature selection, SVMs begin to outperform standard SVMs in precision, recall, and execution time [9, 29].

Like SVM classifiers, k-nearest neighbors is another supervised learning method. Using this algorithm, predictions are made for a new instance by searching through the entire training set for the k most-similar instances (neighbors) and summarizing the most-common output label for those k neighbors. In short, the class with the highest frequency from the k-nearest neighbors will be taken as the new datapoint's predicted class. Again, a distance metric hyperparameter, as well as an integer for k must be specified prior to testing.

Again, a popular application for k-nearest neighbors is network intrusion detection. Because of the distance metric used to determine the nearest neighbors, high-dimensional vectors that are similar may not actually be regarding as "close." In addition, as the number of training datapoints increase exponentially, the probability of error is bounded [8]. To combat this, k-nearest neighbors is often combined with more sophisticated algorithms. For example, Liao and Vemuri [15] proves that using text categorization can program profiles separately, then passing them to k-nearest neighbors will largely reduce the calculation involved while effectively detecting intrusive activity with a low false-positive rate. Extending this sentiment of combining k-nearest with other techniques, Bergman, Cohen, and Hoshen [5] use a neural network for feature embedding images prior to passing them through a k-nearest neighbors classifier. Using this combination, self-supervision is used to weed out much of the anomalous data prior to even going through the k-nearest tests. Doing this allows successful scaling to large datasets for data of all kinds [5].

In other application settings, random forests have been proven useful. The basic unit of a random forest is the decision tree, an overall map of the possible outcomes of a series of related choices, usually beginning with a single node and branching into possible outcomes, repeatedly until a classification can be made by the tree . Randomly creating a forest of trees presents an ensemble that form a random forest, providing greater accuracy in numbers than stand-alone decision trees by averaging the predictions of each component tree [12, 32, 24, 34, 30]. Generally, with more trees comes greater robustness and the ability to better predict classifications on a dataset [30, 12]. In short, the capacity of random forests depends on the strength and correlation between different trees [32, 24], however, there may be a threshold of diminishing returns with too many trees [24].

In many classification problem domains such as credit card fraud detection [12] and medical anomaly detection [30], random forests can serve as beneficial due to their simplification in algorithms and flexibility in handling data of different types [32]. Some works propose that random forest approaches perform similarly to deep learning techniques in network intrusion detection on IoT devices [12, 21, 34, 24]. Hybrid approaches that use anomaly detection followed by misuse detection [34] and random forest algorithms can break the dependency on training sets with attack-free data [35] by classifying outliers found by the system as intrusions [36]. Some limitations have been uncovered, including problems imposed by imbalanced data [32], a lack of strong guidelines for hyperparameters [24, 21], and the fact that intrusions with a high degree of similarity cannot be detected as outliers in most cases [36].

3 Preliminaries

3.1 Deep reinforcement learning

The RL problem involves an agent within an environment and provides a framework for this agent to interact with its surrounding environment and receive feedback signals. RL algorithms use these feedback signals to improve the decision-making policy of the agent.

More specifically, the agent interacts with the environment in a series of discrete timesteps, indexed $t = 0, 1, 2, 3, \ldots, T$, where T is the end of a single environment trajectory, called an episode. At each timestep within an episode, the agent observes a representation of the environment, called the state, $S_t \in S$, where S is the set of all possible states. The agent then chooses an action $A_t \in A$, where A is the set of all possible actions, using its policy, π. $\pi(a|S_t)$ is the probability action a will be chosen by the agent, given state S_t. Then, the environment's transition function decides the resulting next state in timestep $t + 1$, where $p(s_{t+1}|a, S_t)$ is the probability of state s resulting from an agent taking action a in state S_t. After each action, the agent receives a reward based on the outcome of its action in S_t, $R_{t+1} = R(S_t, A_t)$. The goal of an RL agent is to maximize the expected sum of these rewards over the course of the episode, or the return denoted $G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_T$ [28].

While many problems can be formulated using this framework, not all are mathematically proven to be solvable by current RL methods. An environment has the Markov property if the reward and state at timestep t+1 are functions only of the state and action at timestep t, or $p(s', r|s, a) = Pr\{Rt{+}1 = r, St{+}1 = s'|St, At\}$ $\forall r, s', S_t$, and A_t [28]. If the environment has this property, and the state and action spaces are finite, then it is defined as a Finite Markov Decision Process (MDP), and the RL problem is proven to be solvable.

Multi-agent Reinforcement Learning (MARL) is an extension of single-agent RL, which employs $N >$

1 agents in a single environment. MARL environments are not inherently Markov since the transition probabilities $Pr\{R_{t+1} = r, S_{t+1} = s' | S_t, A_t^1, \ldots, A_t^N\}$ depends on the actions chosen by each agent, instead of just one. To each agent, the function by which the environment selects the state at $t+1$ appears non-stationary [17]. In addition, each agent receives their own observations, o_t^i, where $S_t = \{o_t^1, o_t^2, \ldots, o_t^N\}$. So, parts of the environment's state can be hidden from each agent.

MADDPG is the current state-of-the-art algorithm for solving c-MARL, as well as adversarial and mixed settings. MADDPG's main contribution is the introduction of the centralized critic-decentralized actor structure. The actor-critic structure was introduced by A2C for single-agent RL. This method uses two deep neural networks, one to learn a value representation for the expected return (the critic), and one to decide the correct action for the agent (the actor). MADDPG allows the critic network to learn a value function based on the observations of each agent combined, but the actor network must learn a policy using only the corresponding agent's own observations. The result is an agent which has a unique decision-learning policy from the others in the environment, allowing success for multiple agents with different goals. In addition, MADDPG concatenates the actions of each agent onto the state at each timestep during training. Doing so eliminates the non-stationarity of the environment, since the actions of other agents are considered part of the state. This is only done for the centralized critic network, and allows for the convergence of that network [2].

The other DRL algorithm used in this research is QMIX. QMIX [25] is a multi-agent Deep Q-Learning (DQN) algorithm based on Q-Learning with state-of-the-art performance on the StarCraft II Multi-Agent Challenge. QMIX finds the optimal joint action value function using a monotonic mixing function of per-agent utilities. In QMIX, each agent uses a recurrent-DQN network to estimate the action values based on their partial observation. DQN uses experience replay and a target network to improve stability and convergence of RL agent's training. To maximize the total team reward, QMIX estimates the joint action values through a mixing network that takes in each agent's selected action Q-value (i.e., action with max Q-value) and the current state to estimate the total team reward. With an accurate estimation of the total reward,each individual agent's Q network can be fine tuned to maximize the total team reward during execution.

3.2 Adversarial attacks on MARL systems

Following the framework provided by Gleave et al. [10], Alqahtani et al. [2] again prove the feasibility of compromising and weaponizing one agent, directing that agent to act off-policy, and subsequently creating unnatural observations for the rest of the agents within that environment. This paper will investigate various machine learning techniques and their ability to detect such attacks.

As mentioned, three different attack strategies were successfully applied to two environments in order to craft physically realistic adversarial policies for the compromised agent that, in turn, create natural adversarial observations for the cooperative agents. Alqahtani et al. [2] implemented randomly-timed, strategically-timed, and counterfactual reasoning based attacks. Here, the adversary wishes to choose a set of actions for the compromised agent, $A' = [a_{1'}, \ldots, a_{T'}]$, where T is the episode length, such that: $R_{coop}([s_1, \ldots, s_N], A') <= R_{coop}([s_1, ..., s_N], A)$ where R_{coop} represents the reward for the cooperative team. Team reward is used as a measurement of the c-MARL system's success; a successful attack should be able to decrease it quickly by injecting natural perturbations to the agents' observations using the compromised agent's adversarial actions. All three attacks were very successful at harming the systems, regardless of white or black-box knowledge [2].

3.3 Adversarial perturbations on classifiers

While most of the research on adversarial perturbations has revolved around image classifiers due to their easily-visible examples [11, 23, 22], these same attacks carry over to all deep learning classifiers. One such attack used often on image classifiers is known as Fast Gradient Sign Method (FGSM). Goodfellow, Shlens, and Szegedy [11] prove that by adding an imperceptibly small vector (noise) whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, many models will misclassify the input image using $x_{\text{adversarial}} = \epsilon * sign(\Delta_x J(\Theta, x, y))$.

One drawback of FGSM is that it perturbs every single feature in the input vector. On images, this could cause human detectability of an adversarial attack. To combat this, Papernot et al. [23] propose the Jacobian-saliency Map Method (JSMA). This attack creates a saliency map to mark the input features that have the greatest significance in classification. By only perturbing these features, the perturbation becomes less obvious. In addition, this attack can become targeted, allowing the attacker to specify to which class the model should misclassify.

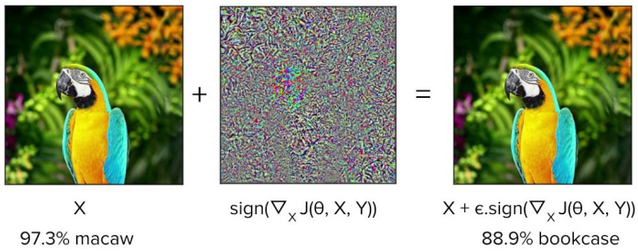

Figure 1: An example of FGSM

Though these attacks are focused on image perturbations, the same methods can be used to fool other neural-network-based classifiers. However, FGSM is more effective outside of the realm of images. In numerical datasets, it is near impossible for humans to detect FGSM perturbations. Therefore, this paper will primarily utilize FGSM to show drawbacks in the use of detection subnetworks.

4 Methodology and Approach

4.1 Environments and experimental setup

The attacks from [2] were implemented prior to this research in both the white-box and black-box settings in 2 environments of MADDPG particle environments: cooperative navigation and physical deception with variant attacking rates. In the cooperative navigation, N cooperative agents must cover L landmarks, and the agents must learn to reach separate landmarks, without communicating their observations to each other. In our experiments, we use $N = L = 3$. In the physical deception environment, N cooperative agents try to fool one adversarial agent. There are L total landmarks, with one being the 'target' landmark and only the cooperative agents know which landmark is the target one. The adversary must try to infer and reach the target landmark from the cooperative agents' positioning, and the cooperative agents must try to deceive the adversary by spacing out. The cooperative agents are rewarded as long as single member of their team reaches the target landmark. We use $N = L = 2$.

In addition, the same attacks were implemented in another, more complex environment: Starcraft II. StarCraft II is a real-time strategy game where two teams of agents can fight against each other. We use the "3m" SMAC map, which employs three "Marine" units on each team. In this scenario, a team wins by shooting the enemy team enough to where they run out of health. Our team consists of three RL agents working together in a c-Marl system to defeat the three enemy marines, which use fixed policies. For our attacks, we control one of the marines, and alter its decision-making based on the aforementioned attack strategies. We show that attacking with one of the c-MARL agents dramatically decreases the team's overall win rate. These agents were trained using QMIX. Three environments (namely cooperative navigation, physical deception, and Starcraft II) trained with two algorithms (MADDPG and QMIX) prove the generalizability of the conclusions made from this research.

4.2 Detection

To combat the adversarial vulnerability in these MARL environments, different anomalous behavior detection models were developed. Namely, a four-layer Dense deep learning, a three-layer binary LSTM, Random Forest, SVM classifier, K-nearest Neighbors classifier, and an ensemble of the binary LSTM and predictive LSTM models modeled after Malhotra et al. [18]'s. Excluding the proposed ensemble model, these classifers will be hereafter referred to as the baselines.

To ensure proper training, for each of the three white-box attacks and each of the three black-box attacks, a new detection model was generated. All of the baseline classifiers were trained using an evenly-distributed training set of approximately fifty percent anomalous and fifty percent normal datapoints. The baseline classifiers predicted abnormality based on just one single input vector of the current timestep's global observations and global actions. That is, they received the current state $\{s_0, s_1, ...s_n\}$ and actions $\{a_0, a_1, ...a_n\}$ of all agents. The binary LSTM received the same information for the three previous timesteps, as well $\left\{\{s_0, s_1, ...s_n\}, \{a_0, a_1, ...a_n\}\right\}_{t-3}^{t}$.

To form the ensemble model, the binary LSTM was combined with separate predictive LSTM networks for each of the agents. Instead of predicting normal or anomalous, these models predicted one of the possible k actions. In our experiments, the particle environments had an action space of 5, whereas Starcraft II agents had a possible action space of 9. These models corresponded with specific agents within the

environments. They were trained on clean datasets with 220,000 samples of normal datapoints generated from each environment with fully cooperative agents. We found that the ensemble model (Figure 2) achieved the highest recall across all models when tested on datasets generated by our attacks.

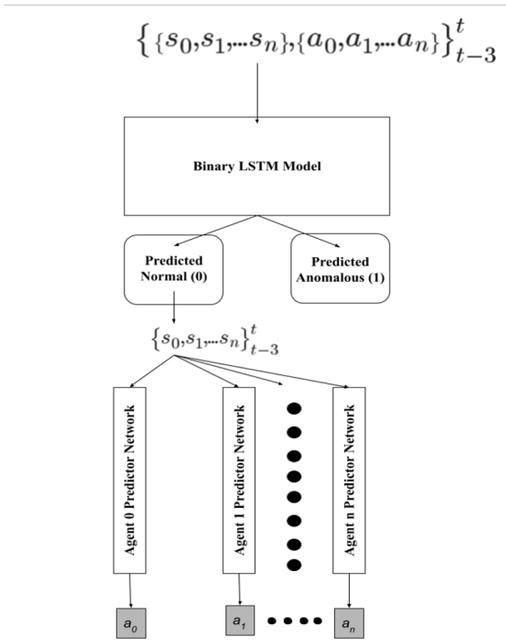

Figure 2: The ensemble model for anomalous behavior detection

In Figure 2, we first run our testing data through the Binary LSTM and if a datapoint classified as positive for anomaly then it would be investigated by a human to ensure accuracy. Any datapoint classified as negative would then be run through the predictive LSTM ensemble separated by agent. These points are tested by n models to generate the predictions based on the probability that agent i performs action $1, 2, ...k$ for k possible actions. In the particle environments, we used a threshold of 0.10 for the minimum probability for action j such that i performing j is still a relatively benign action (as suggested by the model). Threshold 0.10 was chosen because in our test environments, there are only 5 possible actions. For the model to decide that a certain action is normal, the model only needs to predict it with at least 0.20 confidence. Therefore, 0.10 has merit as a threshold. In similar fashion, Starcraft II has 9 possible actions, so for that environment, a threshold of .05 has merit. For a datapoint to be considered normal, all of the true actions must be covered by all of the predicted actions with a confidence greater than the threshold. Formally,

$$\forall\ \hat{a} \in true\{a_0, a_1, ...a_n\},\ \exists\ a \in \{predicted\{a_0, a_1, ...a_n\} \geq threshold\}\ such\ that\ \hat{a} == a. \tag{1}$$

4.3 Re-attacking the detection subnetwork

Although the proposed ensemble detection method has promising results, there are several drawbacks. Most notably are the susceptibilities of neural networks to adversarial attacks like FGSM [11] and JSMA [23]. To further argue the need for c-MARL robustness to adversarial attacks, this paper will prove the infeasibility of using detection subnetworks in detecting anomalies in reinforcement learning by applying the fast gradient sign method attack (FGSM) to fool the proposed ensemble network. The author assumes no knowledge of the interior workings of this ensemble model in an attempt to show the true implications of this attack. Recall that the FGSM equation is $x_{\text{adversarial}} = \epsilon * sign(\Delta_x J(\Theta, x, y))$.

Several steps are executed to implement this attack at a large scale. First, FGSM needs access to a replica model in order to approximate the gradient of the network used in the attack. Because we assume black-box knowledge of the ensemble, the attacker must generate samples that can query the model. By using the model's output to the attacker's queries as the true label, we can then train an approximated replica model. In other words, the attacker issues queries to the target model and labels a carefully selected dataset. For any arbitrarily chosen x the adversary obtains its label y by querying the target model f. The adversary then chooses a procedure $train'$ and model architecture f' to train a surrogate model over tuples (x, y) obtained from querying the target model. The surrogate model is used in the FGSM attack.

Next, for each environment and type of attack (ie counterfactual, random, or strategic), the most effective ϵ was found. To do this, we tested ϵ values between [0,1) at an increment of .05. For each of the tested ϵ, every anomalous datapoint that was misclassified by the surrogate model was then passed through the true ensemble model. The most effective ϵ was chosen by selecting the ϵ with the highest percentage of

misclassified anomalies by the ensemble. Though only the most-effective epsilons were found for only the whitebox randomly-timed attacks due to time consumption, it can easily be extended to all others. For the cooperative navigation, physical deception, and Starcraft II environments, the most effective ϵ were .35, .9, .7, respectively.

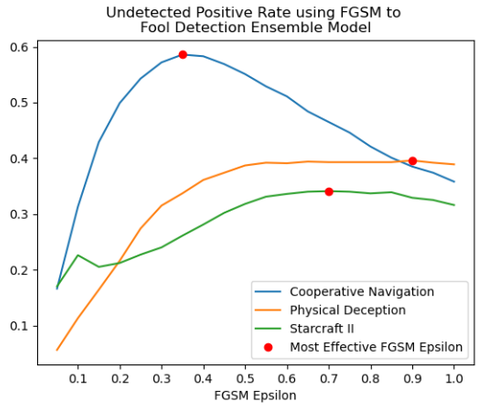

Figure 3: Finding the most effective ϵ for whitebox random attacks

To strengthen the attack of our detection subnetwork and to prove the infeasibility of deep detection subnetworks, the exact same datasets were tested for all six attacks on all environments. Except now, not only did the compromised agent attack the other MARL agents, the compromised agent also perturbed its own observations. The only perturbed observations belonged to the compromised agent, assuming it can control the observation it reports to its target agents. Keeping all else constant, the FGSM attacks severely discounted the proposed ensemble model's performance in detecting anomalies in all attacks and all environments. With Moosavi-Dezfooli et al. [20] proof that new adversarial perturbations can always be generated, these supplemental perturbations will render detection subnetworks obsolete.

5 Results and Discussion

5.1 Detection

Many believe the reason behind the vulnerability of compromised agents in the centralized-training decentralized-execution paradigm of c-MARL is due to ignoring the implicit connections of agents' actions on the others in the moment and subsequent timesteps, allowing the use of one agent as a passive adversary [2]. c-MARL agents trained with such algorithms have difficulty identifying the normal and abnormal behaviors of other agents since their policies were not trained to consider the impact of their actions on each other. To combat the adversarial attacks on MARL systems, several detection schemes were analyzed and compared by performance. Namely, a four-layer Dense deep learning, a three-layer binary LSTM, Random Forest, SVM classifier, K-nearest Neighbors classifier, and a novel stacked-LSTM ensemble architecture. Figure 2 displays the proposed ensemble method. The proposed ensemble outperformed the baselines in all three environments.

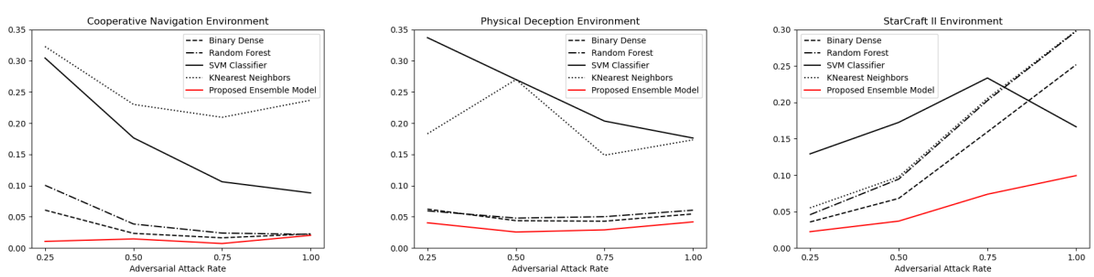

Figure 4: The percentage of undetected anomalies for each model in the whitebox randomly-timed attack strategy by attack rate

Figure 4 shows the comparison of the proposed model against other baselines. The ensemble model continued to have success in the other attack strategies across all environments based on its recall and precision metrics. Recall is the number of correctly identified anomalies in proportion to the number of true anomalies in the datasets. Precision is the proportion of correctly identified anomalies in relation to the number of predicted anomalies. Lower precision corresponds to higher false positive rates. The

sensitive nature of these attacks makes recall paramount in performance metrics. The ensemble consistently showed the best recall (percentage of anomalies detected) with some attacks only going undetected 0.1% of the time. Figure 5 emphasizes the performance of the ensemble model, particularly in the Starcraft II environment where the average recall across all attack rates and strategies in the proposed ensemble model improved by approximately seven percent. This large improvement in recall between the ensemble and other baselines in the Starcraft II environment is likely due to the high-complexity of the problem. Because of the environment's high-dimensionality, basic machine learning algorithms struggle to distinguish between normal and anomalous datapoints.

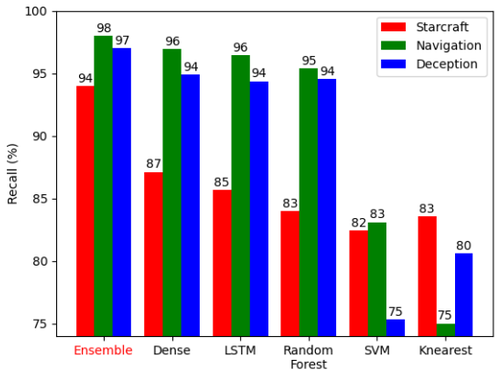

Figure 5: Average recall across all six attack strategies and attack rates for each of the detection models.

Table 1: The detection results of the attacks using the ensemble LSTM model (*precision%—recall%*) in the three environments across the different attack rates in both white and black box settings.

Cooperative Navigation												
	White Box						**Black Box**					
Attack Rate	Random		Timed		Counterfactual		Random		Timed		Counterfactual	
25%	60%	96%	64%	98%	54%	92.5%	61%	97%	62%	99%	55%	92%
50%	85%	98.5%	85.5%	98%	82%	92.5%	84.5%	98%	83.6%	98%	82.6%	92%
75%	94.5%	99%	99%	95%	95%	94%	94%	99%	94.4%	97%	95%	94%
100%	100%	98%	100%	99%	100%	96%	100%	99%	100%	95.5%	100%	96%
Physical Deception												
	White Box						**Black Box**					
Attack Rate	Random		Timed		Counterfactual		Random		Timed		Counterfactual	
25%	51%	96%	54%	97%	45%	91%	52%	97%	55%	99%	40%	97%
50%	77%	97%	77%	96%	74%	92%	78%	97%	77%	97%	72%	94%
75%	90%	97%	90%	95.5%	90%	93%	91%	98%	89%	96%	91%	88%
100%	100%	96%	100%	94.5%	100%	92%	100%	97%	100%	94%	100%	84%
Starcraft II												
	White Box						**Black Box**					
Attack Rate	Random		Timed		Counterfactual		Random		Timed		Counterfactual	
25%	44%	98%	51%	99%	44%	97.5%	44%	99%	49%	99%	44%	98%
50%	65%	96%	73%	99%	66%	99%	56%	99%	69%	99%	66%	99%
75%	82%	93%	91%	99%	78%	99%	69%	99%	70%	99%	78%	99%
100%	100%	90%	100%	99%	100%	98.5%	100%	99%	100%	97%	100%	98%

Table 1 shows the model consistently detecting close to 100% of attacks, marking the impressive performance of the proposed ensemble model. Our detection model achieves better precision when the attack rate increases across all attacks suggesting that at least 90% of the attacks can easily be traced and mitigated at the attack rate as low as of 25% of the episode time. Furthermore, the results show that the counterfactual reasoning based attack evades the detection more than other attacks with higher false negatives (i.e. smaller recalls) in both environments in the white and black box settings. Though the precision of the model leaves some room for improvement, we believe that this is to be expected from a model trained on evenly distributed data but tested on undistributed data. With higher attacking rates, the model gradually increased its precision. Like all intrusion systems, the emphasis should be on reducing the number of undetected attacks. Based on this reasoning, the outstanding recall of the stacked ensemble is crucial.

As shown in Table 1, equipping c-MARL agents with anomalous behavior detection would help them to evaluate the abnormality of each other's actions before considering their actions in the optimization process. The naive remedy once a compromised agent has been detected is to exclude it from the cooperative agents and mark it as an adversary. However, the deviation in one agent's behavior could occur due a security threat or its lack of knowledge about the environment which is reasonable considering the limited vision range and

communication bandwidth in MARL systems. For this and other reasons, anomaly detection subnetworks may not be the best defense.

5.2 Most effective epsilon attack

To further convey that detector subnetworks based on classifier machines are relatively unreliable, we re-attack our proposed ensemble model using the fast gradient sign method (FGSM) in a completely black-box setting. After querying the ensemble, training a surrogate, and finding the most-effective epsilon for the three environments, the exact same testing data was perturbed using FGSM on only the compromised agent's observations. Assuming the attacker has full control of the compromised agent, it is not unlikely to see observation perturbations to further fool the MARL system. After applying FGSM to the compromised agent's observations, the same attacks were implemented in all three environments across all six attack strategies.

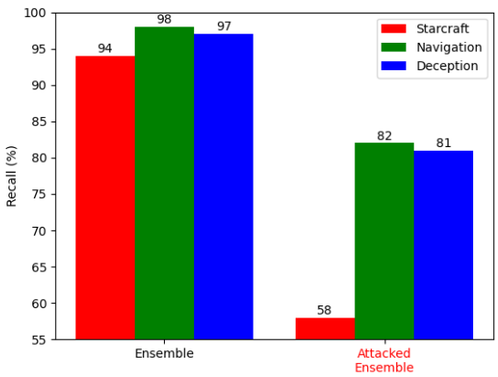

Figure 6: Average recall across all attack settings before and after FGSM re-attack. The model's recall is severely reduced.

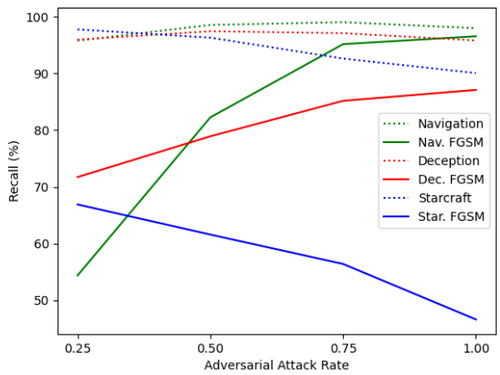

Figure 7: Recall before and after FGSM re-attack using whitebox randomly-timed setting as an example. The model's recall is severely reduced.

Figure 6 shows the drastic reduction in performance of the ensemble model across all attack rates and strategies. Prior to the attack, the model correctly identified 94, 98, and 97 percent of all adversarial attacks in the Starcraft II, cooperative navigation, and physical deception environments, respectively. After perturbing just the compromised agent's observations, the model's performance truly suffered, only identifying 58, 82, and 81 percent, respectively. The greatest detriment was seen in the Starcraft II environment, where a 36% drop in recall was observed after the FGSM re-attack. Again, this is likely due to the high-complexity of the Starcraft II environment. In this specific setting, the adversarial agent's observations accounted for one-third of the global state. Due to the perturbation of a larger proportion of the global state, the detection model suffered greatly. Out of the different attacks, the ensemble suffered most with the re-perturbation of the whitebox randomly-timed attack. Figure 7 shows the stark difference between recall before and after the FGSM re-attack in the whitebox randomly-timed setting. Clearly, the attack rate of .25 was most successful after the FGSM re-attack. This was consistent across the other attack strategies. With a lower attack rate, the difference between model recall before and after the re-perturbation was largest. In some attack settings, the recall was halved, resulting in the majority of positives going undetected.

The idea that slightly perturbing one compromised agent's observations can reduce performance at such a large scale is alarming. With the seemingly heavy dependence on small fractions of the global observations, attackers can implement similar attacks to go virtually unnoticed by detection subnetworks. By only compromising one agent, the entire system will be affected. And without reliable detection, it is clear that these attacks can be devastating.

6 Future Directions

The FGSM re-attack on the proposed ensemble model suggests that detection subnetworks are just as attackable as the MARL systems they are designed to protect. For this reason, detection may not be a reliable countermeasure against adversarial attacks in MARL. Prevention and robustness against these compromises is a better avenue to ensure safety and reliability of MARL systems.

As is common in the literature, there is much work towards the defense of adversarial attacks on machine learning techniques. However, almost none of that has been transferred to the realm of reinforcement learning. Additionally, multi-agent reinforcement learning is virtually untouched in this regard. First steps

in future directions would attempt to apply previously-proposed defenses, originally intended for other machine learning techniques, to reinforcement learning settings. Only when we have a sound understanding of the existing literature in the RL domain can we design and test novel approaches.

Once those prerequisites are met, it may be beneficial for the research community to extend the assured reinforcement learning (ARL) techniques into c-MARL by embedding formal verification in c-MARL algorithms. Real-time formal verification methods would shield the agents from reaching the dangerous states by training them to trade off between performance and security/safety in the environments containing either adversarial or dysfunctional agents. Those methods should be developed for run-time verification focusing only on the agent's time-bounded, short-term behavior. This will increase the robustness of MARL systems functioning in uncertain and unpredictable environments.

7 Conclusion

In conclusion, this research tests and compares several machine learning techniques and their ability to detect and mitigate adversarial attacks in multi-agent reinforcement learning settings. Using three different MADDPG and QMIX c-MARL environments, it was found that a stacked-LSTM ensemble detector subnetwork yielded the best results. However, due to the vulnerabilities of deep classifiers, it should be emphasized that many detection subnetworks, especially the ones utlizing neural networks, are just as attackable as the systems they are designed to protect. This sentiment is evidenced by the subsequent attack using fast gradient sign method to perturb the compromised agent's observations, drastically reducing the proposed model's recall and precision. With some environments and attack strategies experiencing a halving in identifiable positives, the evidence shows the true need for prevention and robustness against these types of attacks. This paper concludes that in real-world environments with aggressive threat actors, reliable detection of adversarial attacks in MARL systems is a far cry from reality, to say the least.

Works Cited

[1] Gleave et al. "Adversarial policies: attacking deep reinforcement learning". In: (). DOI: arXiv:1905.10615.

[2] Sarra Alqahtani et al. "Adversarial Policies and Defense in Cooperative Multi-Agent Reinforcement Learning". In: 2021.

[3] Gürdal Arslan and Serdar Yüksel. "Decentralized Q-Learning for Stochastic Teams and Games". In: (2016). arXiv: 1506.07924 [math.OC].

[4] Sean L. Barton, Nicholas R. Waytowich, and Derrik E. Asher. "Coordination-driven learning in multi-agent problem spaces". In: *CoRR* abs/1809.04918 (2018). URL: http://arxiv.org/abs/1809.04918.

[5] Liron Bergman, Niv Cohen, and Yedid Hoshen. *Deep Nearest Neighbor Anomaly Detection*. 2020. arXiv: 2002.10445 [cs.LG].

[6] Tong Chen et al. "Adversarial attack and defense in reinforcement learning-from AI security view". In: *Cybersecurity* 2 (Dec. 2019). DOI: 10.1186/s42400-019-0027-x.

[7] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: https://doi.org/10.1023/A:1022627411411.

[8] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

[9] Annie George. "Anomaly Detection based on Machine Learning Dimensionality Reduction using PCA and Classification using SVM". In: *International Journal of Computer Applications* 47 (June 2012), pp. 5–8. DOI: 10.5120/7470-0475.

[10] Adam Gleave et al. *Adversarial Policies: Attacking Deep Reinforcement Learning*. 2021. arXiv: 1905.10615 [cs.LG].

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: (2015). DOI: arXiv:1412.6572.

[12] Mahmudul Hasan et al. "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches". In: 7 (May 2019). DOI: 10.1016/j.iot.2019.10.

[13] Wenjie Hu, Yihua Liao, and Rao Vemuri. "Robust Anomaly Detection Using Support Vector Machines". In: *Proceedings of the International Conference on Machine Learning* (June 2003).

[14] Sandy H. Huang et al. "Adversarial Attacks on Neural Network Policies". In: *CoRR* abs/1702.02284 (2017). URL: http://arxiv.org/abs/1702.02284.

[15] Yihua Liao and Rao Vemuri. "Use of K-Nearest Neighbor classifier for intrusion detection". In: *Computers Security* 21 (Oct. 2002), pp. 439–448. DOI: 10.1016/S0167-4048(02)00514-X.

[16] S. W. Loke. "Cooperative Automated Vehicles: A Review of Opportunities and Challenges in Socially Intelligent Vehicles Beyond Networking". In: *IEEE Transactions on Intelligent Vehicles* 4.4 (2019), pp. 509–518.

[17] Ryan Lowe et al. *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*. 2020. arXiv: 1706.02275 [cs.LG].

[18] Pankaj Malhotra et al. "Long Short Term Memory Networks for Anomaly Detection in Time Series". In: Apr. 2015.

[19] Jan Hendrik Metzen et al. *On Detecting Adversarial Perturbations*. 2017. arXiv: 1702.04267 [stat.ML].

[20] Seyed-Mohsen Moosavi-Dezfooli et al. *Universal adversarial perturbations*. 2017. arXiv: 1610.08401 [cs.CV].

[21] S. Naseer et al. "Enhanced Network Anomaly Detection Based on Deep Neural Networks". In: *IEEE Access* 6 (2018), pp. 48231–48246. DOI: 10.1109/ACCESS.2018.2863036.

[22] Nicolas Papernot et al. *Practical Black-Box Attacks against Machine Learning*. 2017. arXiv: 1602.02697 [cs.CR].

[23] Nicolas Papernot et al. *The Limitations of Deep Learning in Adversarial Settings*. 2015. arXiv: 1511.07528 [cs.CR].

[24] R. Primartha and B. A. Tama. "Anomaly detection using random forest: A performance revisited". In: *2017 International Conference on Data and Software Engineering (ICoDSE)*. 2017, pp. 1–6. DOI: 10.1109/ICODSE.2017.8285847.

[25] Tabish Rashid et al. *QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning*. 2018. arXiv: 1803.11485 [cs.LG].

[26] O. Salem et al. "Anomaly Detection in Medical Wireless Sensor Networks using SVM and Linear Regression Models". In: *Int. J. E Health Medical Commun.* 5 (2014), pp. 20–45.

[27] Ralf C. Staudemeyer and Christian W. Omlin. "Evaluating Performance of Long Short-Term Memory Recurrent Neural Networks on Intrusion Detection Data". In: *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. SAICSIT '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 218–224. ISBN: 9781450321129. DOI: 10.1145/2513456.2513490. URL: https://doi.org/10.1145/2513456.2513490.

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[29] Taeshik Shon et al. "A machine learning framework for network anomaly detection using SVM and GA". In: *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*. 2005, pp. 176–183. DOI: 10.1109/IAW.2005.1495950.

[30] Alexander Verner. "LSTM Networks for Detection and Classification of Anomalies in Raw Sensor Data". PhD thesis. Apr. 2019. DOI: 10.13140/RG.2.2.19049.54888.

[31] Cihang Xie et al. "Mitigating Adversarial Effects Through Randomization". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=Sk9yuql0Z.

[32] S. Xuan et al. "Random forest for credit card fraud detection". In: *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*. 2018, pp. 1–6. DOI: 10.1109/ICNSC.2018.8361343.

[33] Bora Yongacoglu, Gürdal Arslan, and Serdar Yüksel. "Learning Team-Optimality for Decentralized Stochastic Control and Dynamic Games". In: (2019). arXiv: 1903.05812 [math.OC].

[34] J. Zhang and M. Zulkernine. "A hybrid network intrusion detection technique using random forests". In: *First International Conference on Availability, Reliability and Security (ARES'06)*. 2006, 8 pp.–269. DOI: 10.1109/ARES.2006.7.

[35] J. Zhang and M. Zulkernine. "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection". In: *2006 IEEE International Conference on Communications*. Vol. 5. 2006, pp. 2388–2393. DOI: 10.1109/ICC.2006.255127.

[36] Jiong Zhang and Mohammad Zulkernine. "Network Intrusion Detection using Random Forests". In: *PST*. 2005.