Multi-Task Federated Reinforcement Learning with Adversaries

Aqeel Anwar [1] **Zishen Wan** [1] **Arijit Raychowdhury** [1]

Abstract

Reinforcement learning algorithms pose a serious threat from adversaries. The adversaries can manipulate the learning algorithm resulting in non-optimal policies. In this paper, we analyze the Multi-task Federated Reinforcement Learning algorithms, where multiple collaborative agents in various environments are trying to maximize the sum of discounted return, in the presence of adversarial agents. We argue that the common attack methods are not guaranteed to carry out a successful attack on Multi-task Federated Reinforcement Learning and propose an adaptive attack method with better attack performance. Furthermore, we modify the conventional federated reinforcement learning algorithm to address the issue of adversaries that works equally well with and without adversaries. Experimentation on reinforcement learning problems of different scales shows that the proposed attack method outperforms other general attack methods and the proposed modification to federated reinforcement learning algorithm was able to achieve near-optimal policies in the presence of adversarial agents.

1. Introduction

In the past decade, Reinforcement Learning (RL) has gained wide popularity in solving complex problems in an online fashion for various problem sets such as game playing (Mnih et al., 2015; Krishnan et al., 2022), autonomous navigation (Anwar & Raychowdhury, 2018; Wang et al., 2019; Wan et al., 2021), robotics (Kober et al., 2013) and network security (Xiao et al., 2018). With a boom in Internet of Things devices, we have a lot of compute power at our disposal. The problem, however, is that the compute is distributed. Distributed algorithms have been studied to take advantage of these distributed compute agents. Federated Learning (Bonawitz et al., 2019; 2017) is a distributed approach to machine learning tasks enabling model training on large sets of decentralized data by individual agents. The key idea behind federated learning is to preserve the privacy of the data to the local node responsible for generating it. Federated learning has also been considered in the context of reinforcement learning problem for both multi-agent RL (Kumar et al., 2017; Zhuo et al., 2019; Palmer et al., 2017) and multi-task RL (Lim et al., 2020; Liu et al., 2019; Zeng et al., 2020; Wan et al., 2022) where multiple RL agents either in a single or multiple environments try to jointly maximize the collective of individual discounted returns, respectively.

While machine learning (ML) algorithms have proven to provide superior accuracy over conventional methods, they pose a threat from adversarial manipulations. Common attack methods include data-poisoning (Huang et al., 2017; Kos & Song, 2017) and model poisoning (Blanchard et al., 2017; Bhagoji et al., 2019) where the adversary tries to manipulate the input data or directly the learned model. Such vulnerabilities can significantly degrade the performance. In this paper, we are interested in mathematically formulating the vulnerabilities of multi-task federated reinforcement learning (MT-FedRL) problems under model poisoning attacks. Various works have addressed the adversarial attacks in RL and Federated ML by adversarial training, feature de-noising, modifying the federated aggregation operator (Rodríguez-Barroso et al., 2020; Mandlekar et al., 2017; Pattanaik et al., 2017), etc. However, not much research is done in addressing the vulnerabilities of MT-FedRL systems where multiple agents are collectively maximizing the sum of discounted return. The key motivation for this work is to rigorously analyze MT-FedRL under an adversarial lens so that it can be made immune to adversarial attacks.

The contributions of this paper are as follows

- We provide a clean mathematical formulation of the MT-FedRL problem in the presence of adversaries and analyze the vulnerabilities of such systems.
- We argue that the general adversarial methods are not good enough to create an effective attack on MT-FedRL, and propose a model-poisoning attack methodology *AdAMInG* based on minimizing the information

[1]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA. Correspondence to: Aqeel Anwar <aqeel.anwar@gatech.edu>, Zishen Wan <zishenwan@gatech.edu>, Arijit Raychowdhury <arijit.raychowdhury@ece.gatech.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

gained during training.

- The key motivation is to make MT-FedRL systems prone to adversarial manipulations, hence we address the adversarial attack issue by proposing a modification to the general FedRL algorithm, `ComA-FedRL`, that works equally well with and without adversaries.
- We conduct an in-depth adversarial attack study on the MT-FedRL system from small (grid-based task) to large (drone autonomous navigation task) computing scales. Our results demonstrate the effectiveness of our proposed method *AdAMInG* and `ComA-FedRL` under different parameters and scenarios.

2. Related Work

The effects of adversaries in machine learning algorithms were first discovered in (Szegedy et al., 2013) where it was observed that a small l_p norm perturbation to the input of a trained classifier model resulted in confidently misclassifying the input. The adversary here acts in the form of specifically creating adversarial inputs to produce erroneous outputs to a learned model (Kurakin et al., 2016; Tramèr et al., 2017). For supervised learning problems, where the network model has already been trained, attacking the input is the most probable choice for an adversary. Our work focuses on attack methods for the problem of reinforcement learning where the system is trained on the go.

In RL, the adversary can act either in the form of data-poisoning attacks, such as creating adversarial examples (Huang et al., 2017; Kos & Song, 2017), or can directly attack the underlying learned policy (Huang & Zhu, 2019; Ma et al., 2019) either in terms of malicious falsification of reward signals, or estimating the RL dynamics from a batch data set and poisoning the policy. Authors in (Gleave et al., 2019), attack an RL agent by selecting an adversarial policy acting in a multi-agent environment as a result of creating adversarial observations. Their results on a two-player zero-sum game show that an adversarial agent can be trained to interact with the victim winning reliably against it. Such attacks, however, can not be directly extended to multi-task RL problem, which is the scope of this paper.

In federated RL, alongside the data-poisoning and policy-poisoning attacks, we also have to worry about the model-poisoning attacks. Since we have more than one learning agents, a complete agent can take up the role of an adversary. In model poisoning attacks the adversary tries to modify the learned model parameters directly by feeding false information purposely poisoning the global model (Blanchard et al., 2017; Bhagoji et al., 2019). Since federated learning uses an average operator to merge the local model parameters learned by individual agents, such attacks can severely affect the performance of the global model. Such model poisoning attacks have been extensively studied for supervised federated learning problems. In this paper, we address model poisoning attacks for MT-FedRL problems which have not been studied in much detail.

Adversarial training can be used to mitigate the effects of such adversaries. (Xie et al., 2019) showed that the classification model can be made much more robust against the adversarial examples by feature de-noising. The robustness of RL policies has also been analyzed by the adversarial training (Pinto et al., 2017; Tessler et al., 2019). (Tessler et al., 2019) show that the data-poisoning can be made a part of RL training to learn more robust policies. They feed perturbed observations during RL training for the trained policy to be more robust to dynamic changing conditions during test time. (Rodríguez-Barroso et al., 2020) shows that the data-poisoning attacks in federated learning can be resolved by modifying the federated aggregation operator based on induced ordered weighted averaging operators (Yager & Filev, 1999) and filtering out possible adversaries.

To the best of our knowledge, this is the first work on MT-FedRL in the presence of adversaries. We address the effects of model poisoning attacks on the MT-FedRL problem by mathematically analyzing the vulnerabilities. In the light of the observed vulnerabilities, we then propose a modification to the existing algorithm addressing the issue of adversaries.

3. Multi-task Federated Reinforcement Learning (MT-FedRL)

We consider a Multi-task Federated Reinforcement Learning (MT-FedRL) problem with n agents. Each agent operates in its own environment which can be characterized by a different Markov Decision Process (MDP). The goal of MT-FedRL is to learn a unified policy, which is jointly optimal across n environments. Each agent shares its information with a centralized server. We consider policy gradient methods for RL. The MDP at each agent i can be described by $\mathcal{M}_i = (\mathcal{S}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{R}_i, \gamma_i)$ where $\mathcal{S}_i$ is the state space, $\mathcal{A}_i$ is the action space, $\mathcal{P}_i$ is the MDP transition probabilities, $\mathcal{R}_i : \mathcal{S}_i \times \mathcal{A}_i \to \mathbb{R}$ is the reward function, and $\gamma_i \in (0, 1)$ is the discount factor. Let V_i^π be the value function, induced by policy π, at the state s in the i-th environment, we have

$$V_i^\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma_i^k \mathcal{R}_i(s_i^k, a_i^k) \,|\, s_i^0 = s\right]. \tag{1}$$

where $a_i^k \sim \pi(\cdot|s_i^k)$. We denote by ρ_i the initial state distribution over the action space of i-th environment. The goal is to find a unified policy π^* that maximizes the sum of long-term discounted return for all the environments i.e.

$$\max_\pi V(\pi; \boldsymbol{\rho}) \triangleq \sum_{i=0}^{n-1} \mathbb{E}_{s_i \sim \rho_i} V_i^\pi(s_i), \quad \boldsymbol{\rho} = \begin{bmatrix} \rho_0 \\ \vdots \\ \rho_{n-1} \end{bmatrix} \tag{2}$$

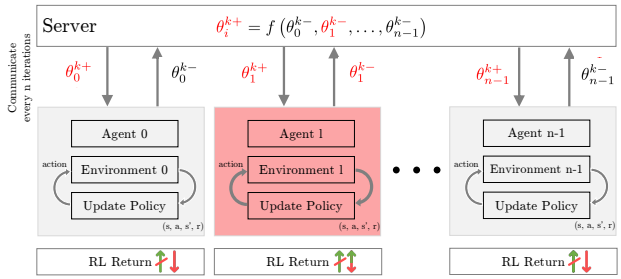

Figure 1. Adversaries can impact the unified policy by providing adversarial policies to the server. This results in negatively impacting the achieved discounted return on the environments.

We use θ to model the family of policies $\pi_\theta(a|s)$, considering both the tabular method and neural network-based function approximation. The goal of the MT-FedRL problem is to find θ^* satisfying

$$\theta^* = \arg\max_\theta V(\theta; \boldsymbol{\rho}) \triangleq \sum_{i=0}^{n-1} \mathbb{E}_{s_i \sim \rho_i} V_i^{\pi_\theta}(s_i). \quad (3)$$

In MT-FedRL, the data is not shared with the server due to privacy concerns. The data remains at the local agent and instead, the policy parameter θ_i is shared with the server. Each agent i utilizes its locally available data $\mathcal{D}_i$ to train the policy parameter θ_i by maximizing its local value function V_i^π through SGD. After the completion of each episode k, the agents share their policy parameter θ_i^{k-} with a centralized server. The server carries out a smoothing average and generates N new sets of parameters $\theta_i^{k+} = \alpha^k \theta_i^{k-} + \beta^k \sum_{j \neq i} \theta_j^{k-}$ for each agent, where $\alpha^k, \beta^k = \frac{1-\alpha}{n-1} \in (0, 1)$ are smoothing average weights.

The goal of this smoothing average is to achieve a consensus among the agents' parameters, $\lim_{k\to\infty} \theta_i^{k+} \to \theta^*, \forall i \in \{0, \ldots, n-1\}$. As the training proceeds, the smoothing average constants converge to $\alpha^k, \beta^k \to \frac{1}{n}$. The conditions on α^k, β^k to guarantee the convergence can be found in (Zeng et al., 2020).

4. MT-FedRL with adversaries

Let $\mathcal{L}$ denote the set of adversarial agents in a $n - agent\ MTFedRL$ problem. The smoothing average at the server can be decomposed based on the adversarial and non-adversarial agent as follows

$$\theta_i^{k+} = \alpha^k \theta_i^{k-} + \beta^k \sum_{j \neq i, j \notin \mathcal{L}} \theta_j^{k-} + \beta^k \sum_{l \in \mathcal{L}} \theta_l^{k-} \quad (4)$$

where $i \notin \mathcal{L}$. θ_i^{k+} is the updated policy parameter for agent i calculated by the server at iteration k. In an adversarial MT-FedRL, the goal of the adversarial agent is to prevent the MT-FedRL from achieving the unified θ^* by purposely providing adversarial policy parameters θ_l^{k-} (Fig. 1).

Parameters that effect learning: Using gradient ascent, each agent updates its own set of policy parameter locally according to the following equation,

$$\theta_i^{k-} = \theta_i^{(k-1)+} + \delta_i \nabla_{\theta_i} V_i^{\pi_{\theta_i}}(\rho_i) \quad (5)$$

where δ_i is the learning rate for agent i. Using Eq. 5 in the smoothing average Eq. 3 yields

$$\theta_i^{k+} = \left(\alpha^k \theta_i^{(k-1)+} + \beta^k \sum_{j \neq i, j \notin \mathcal{L}} \theta_j^{(k-1)+} \right) + \left(\alpha^k \delta_i \nabla_{\theta_i} V_i^{\pi_{\theta_i}}(\rho_i) + \beta^k \sum_{j \neq i, j \notin \mathcal{L}} \delta_j \nabla_{\theta_j} V_j^{\pi_{\theta_j}}(\rho_j) \right) + \left(\beta^k \sum_{l \in \mathcal{L}} \theta_l^{k-} \right) \quad (6)$$

The server update of the policy parameter can be decomposed into three parts.

- The weighted sum of the previous set of policy parameters $\theta_i^{(k-1)+}$ shared by server with agents.
- The agent's local update, which tries to shift the policy parameter distribution towards the goal direction.
- The adversarial policy parameter which aims at shifting the policy parameter away from achieving the goal.

If the update carried out by the adversarial agent is larger than the sum of each agent's policy gradient update, the policy parameter will start diverging from the desired consensus θ^*. The success of the adversarial attack hence depends on the nature of adversarial policy parameter θ_l^{k-}, non-adversarial agent's local learning rate δ_i, and the number of non-adversarial agents $n - |\mathcal{L}|$.

Threat Model: In MTFed-RL, the adversarial agent can control the convergence through the policy parameter θ_l^{k-} that it shares with the server. The adversarial agent needs to share the policy parameter that moves the distribution of the average of non-adversarial agents either to a uniform distribution or in the direction that purposely yields bad actions. Generally shifting the distribution to uniform distribution will require less energy than to shift it to a non-optimal action distribution. This requires that the adversary cancel out the information gained by all the other non-adversarial agents hence not being able to differentiate between good and bad actions, leaving all the actions equally likely.

We will assume the following threat model. At iteration k, each adversarial agent l shares the following policy parameter with the server $\theta_l^{k-} = \lambda^k \theta_{adv}^k$. Hence the threat model is defined by the choice of the attack model θ_{adv} and $\lambda^k \in \mathbb{R}$ which is a non-negative iteration-dependant scaling

factor that will be used to control the norm of the adversarial attack. To make the scaling factor more meaningful, we will assume that

$$\|\theta_{adv}\|^2 \approx \frac{1}{(n-|\mathcal{L}|)} \sum_{i \notin \mathcal{L}} \|\theta_i\|^2 \tag{7}$$

5. Attack Models - AdAMIn

Making the MT-FedRL systems secure against adversarial attacks requires us to analyze the vulnerabilities of such systems. Hence, in this section, we propose an attack model mathematically for MT-FedRL systems.

Two common attack models are Random Policy Attack (Rand) and Opposite Goal Policy Attack (OppositeGoal), which details can be found in Appendix A. Even though the adversarial choice of opposite goal makes an intuitive sense as the best attack method, Section 7 shows that it's not. We propose an attack method, *AdAMInG*, that takes into account the nature of MT-FedRL smoothing averaging and devises the best attack given the information available locally. The goal of *AdAMInG* is to devise an attack that uses a single adversarial agent with a small scaling factor by forcing the server to gradually forget what it learns from the non-adversarial agents.

For smoothing average at the server to lose all the information gained by other non-adversarial agents we should have

$$\theta_l^{k-} = -\frac{1}{\beta^k |\mathcal{L}|} \left(\alpha^k \theta_i^{k-} + \beta^k \sum_{j \neq i,l} \theta_j^{k-} \right) \tag{8}$$

Using the above equation in Eq. 4 will result $\theta_i^{k+} = \mathbf{0}$, hence losing the information gained by θ_i^{k-}. The problem in Eq. 8 is that the adversarial agents do not have access to the policy parameter shared by non-adversarial agents $\theta_i^{k-}, \forall i \neq l$ and hence the smoothing average of non-adversarial agents is unknown. The attack model then is to estimate the smoothing average of non-adversarial agents.

The adversarial agent has two available information: (1) The previous set of policy parameter shared by the adversarial agent to the server $\theta_l^{(k-1)-}$. (2) The federated policy parameter shared by the server to the adversarial agent $\theta_l^{(k-1)+}$.

The adversarial agent can estimate the smoothing average of the non-adversarial agents from these quantities. The *AdAMInG* attack shares the following policy parameter

$$\theta_l^{k-} = \lambda^k \left(\frac{\alpha^k \theta_l^{(k-1)+} - \theta_l^{(k-1)-}}{\beta^k} \right) \tag{9}$$

The smoothing average at the server for $i \in \{0, \ldots, n-1\}, i \neq l$ becomes

$$\theta_i^{k+} = \left(\alpha^k \theta_i^{k-} - \frac{\lambda^k}{n-1} \beta^k \theta_i^{(k-1)-} \right) + \sum_{j \neq i,l} \left(\beta^k \theta_j^{k-} - \frac{\beta^k \lambda^k}{n-1} \theta_j^{(k-1)-} \right) \tag{10}$$

We want $\theta_i^{k+} \to \mathbf{0}$, $\forall i \in \{0, n-1\}, i \neq l$. This means forcing the two terms inside the parenthesis to $\mathbf{0}$. If the initialization of all the agents are same, i.e. $\theta_i^{0-} = \theta^0 = \mathbf{0}, \forall i$ and the learning rate is small, we have $\|\theta_i^{k-} - \theta_i^{(k-1)-}\| < \epsilon$. Hence $\theta_i^{k+} \to \mathbf{0}$ can be achieved by the scaling factor $\lambda^{k*} = argmin_{\lambda^k}\, g(\lambda^k, n)$, where

$$g(\lambda^k, n) = \left| \alpha^k - \beta^k \frac{\lambda^k}{n-1} \right| + \left| \beta^k \left(1 - \frac{\lambda^k}{n-1} \right) (n-2) \right|$$

For simplicity we have not shown the dependence of α^k, β^k in $g(\lambda^k, n)$ as they directly depend on k. Solving this optimization problem yields $\lambda^* = n-1, (n \geq 3)$.

This means that the scaling factor should be equal to the number of non-adversarial agents and is independent of the iteration k. For $\lambda^k < \lambda^*$ we still can achieve a successful attack if the learning rate δ is not too high.

As the training proceeds, the values of the smoothing constants α^k, β^k approach their steady-state value of $\frac{1}{n}$. At that point, the steady-state value of $g(\lambda^k, n)$ defined as $g_{ss}(\lambda^k, n)$ is given by $g_{ss}(\lambda^k, n) = \frac{n-1-\lambda}{n}$. The steady-state value $g_{ss}(\lambda^k, n)$ signifies how effective/successful the *AdAMInG* attack will be for the selected parameters (λ^k, n). A steady-state value of 0 signifies a perfect attack, where the policy parameter shared by the server loses all the information gained by the non-adversarial agents. A steady-state value of 1 indicates a completely unsuccessful attack. The smaller the $g_{ss}(\lambda^k, n)$, the better the *AdAMInG* attack. A detailed explanation on how $g(\lambda^k, n)$ affects the MT-FedRL can be found in Appendix B.

Unlike the OppositeGoal attack, we can guarantee that the *AdAMInG* attack will yield a successful attack if the scaling factor is equal to the number of non-adversarial agents. We will see in the results section that the scaling factor does not need this high if the learning rate δ is not high. We will be able to achieve a good enough attack even if $\lambda^k < n-1$. The only down-side with the *AdAMInG* attack method is that it requires twice the amount of memory as compared to that of OppositeGoal or Rand attack method. *AdAMInG* attack method needs to store the both the adversary shared policy parameter $\theta_l^{(k-1)-}$ and the server shared policy parameter $\theta_l^{(k-1)+}$ from the previous iteration to compute the new set of policy parameters to be shared θ_l^{k-} as shown in Eq. 9.

6. Detecting Attacks - ComA-FedRL

Adversary can severely affect the performance of the unified policy in MT-FedRL systems. Based on the analysed vulnerabilities in the *AdAMInG* attack method, we propose Communication Adaptive Federated RL (`ComA-FedRL`) to address the adversarial attacks on a Federated RL algorithm. Instead of communicating the policy parameter from all agents at a fixed communication interval, we assign different communication intervals to agents based on the confidence of them being an adversary. An agent, with higher confidence of being an adversary, is assigned a large communication interval and vice-versa.

`ComA-FedRL` begins with a pre-train phase, where each agent tries to learn a locally optimistic policy independent of others. After every certain number of episodes, the server randomly assigns a policy to all the environments without replacement for evaluation, and the cumulative reward achieved by this policy is recorded. Based on the nature of the policy and the environment it is cross-evaluated in, we have four cases as shown in Table 2. When the policy locally learned by a non-adversarial agent is evaluated in the environment of a non-adversarial agent, it generally performs better than a random policy because of the correlation of the underlying tasks. Hence we get a slightly higher cumulative reward compared to other cases. If an adversarial policy is cross-evaluated on a non-adversarial agent's environment, it generally performs worse because of the inherent nature of the adversary, giving a low cumulative reward. When the policies are evaluated on the adversarial agent's environment, the adversary can present a secondary attack in faking the cumulative reward. It intentionally reports a low return with the hopes of confusing the server to mistake a non-adversarial agent with an adversarial one. Since the adversarial agent has no way of knowing if the policy shared by the server belongs to an adversarial or a non-adversarial agent, it always shares a low cumulative return.

At the end of the pre-train phase, the cumulative rewards are averaged out for a given policy and are compared to a threshold. If the averaged reward of the policy is below (above) this threshold, the policy is marked as *possibly-adversarial* (*possibly-non-adversarial*). The *possibly-adversarial* agents are assigned a higher communication interval, while *possibly-non-adversarial* agents are assigned a smaller communication interval. The agents are constantly re-evaluated after a certain number of iterations. After re-evaluation, if an already marked possible-adversary agent is re-marked as *possibly-adversary*, the agent's communication interval is doubled, signifying a higher probability of it being an adversary and making it contribute even lesser towards the server smoothing average. Hence, as the training proceeds, the adversarial agent's contribution to the server smoothing average becomes smaller and smaller.

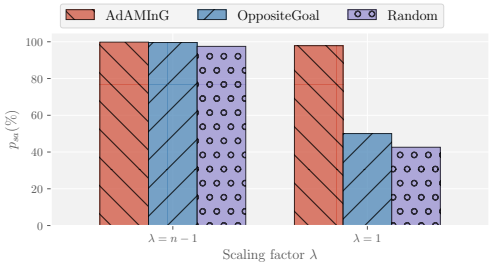

Figure 2. [GridWorld] Probability of successful attack $p_{sa}(\%)$ under different attack models. The greater the p_{sa} the better the performance of the adversary.

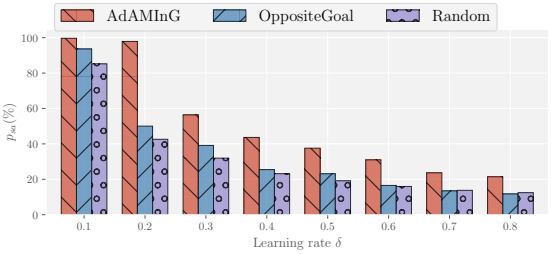

Figure 3. [GridWorld] Effect of learning rate (δ) on the performance of attack methods with $\lambda = 1$ and $n = 12$.

Further details on `ComA-FedRL` can be found in the Appendix C.

7. Experimental Results

We evaluate the proposed attack and detection schemes on both simpler tabular-based MT-FedRL problem (GridWorld) and complex neural network-based MT-FedRL problem (AutoNav). We use policy gradient methods for both cases.

7.1. GridWorld - Tabular RL

We begin our experimentation with a tabular-based problem of GridWorld (Fig. 8). We characterize the performance of the adversarial attack by the probability of successful attack p_{sa}. The detailed GridWorld problem description and metric definition can be found in Appendix D.1.

Effect of Adversaries We first evaluate the effect of the attack models. Fig. 2 reports the p_{sa} for the three attack methods with the scaling factor of $n-1$ and 1 (and a learning rate $\sigma = 0.2$). With the optimal scaling factor of $n-1$, it can be seen that all the three attack methods were able to achieve a good enough attack ($p_{sa} > 96\%$). For a scaling factor of 1, however, only *AdAMInG* attack method was able to achieve a successful attack ($p_{sa} = 98\%$).

As mentioned in Section 4, for a scaling factor of 1, the performance of the attack method depends on the learning rate (δ) and the number of non-adversarial agents ($n - |\mathcal{L}|$). Fig. 3 reports p_{sa} of the attack methods with varying learning rates. It can be seen that the greater the learning rate, the poorer the performance of the attack method. For a higher learning rate, the local update for each agent's

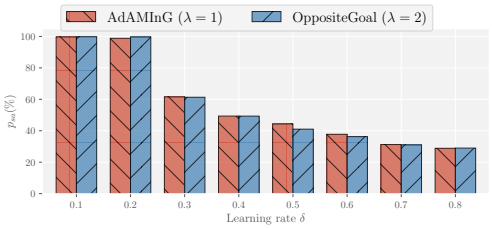

Figure 4. [GridWorld] Comparing attack performance for $n = 12$ between *AdAMInG* with $\lambda = 1$ and OppositeGoal with $\lambda = 2$.

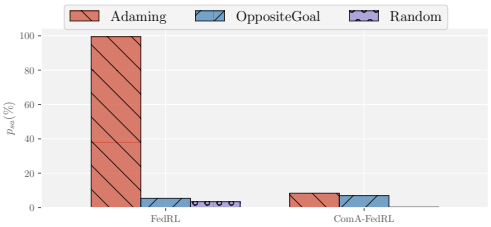

Figure 5. [AutoNav] Comparison of successful attack $p_{sa}(\%)$ under different attack models for `FedRL` and `ComA-FedRL`.

Table 1. [AutoNav] MSF (m) for different attack methods

	AdAMInG	Opposite Goal	Random	No Adv
FedRL	6	1076	1098	1137
ComA-FedRL	1042	1028	1134	1156

policy parameter has more effect than the update of the server carried out with the adversary, and hence poorer the performance of the attack. A detailed analysis of the learning rate on *AdAMInG* is provided in Appendix E.

Another thing to observe is that as the learning rate increases the relative performance of the OppositeGoal attack compared to the Random policy attack becomes poorer even becoming worse than the Random policy attack. The reason behind this is that the observable states across environments are not completely overlapping. The environment available to the adversary for devising OppositeGoal attack from might not have access to the states observable in other environments. Hence the OppositeGoal policy attack can not modify the policy parameter related to those states. OppositeGoal attack method either require a large scaling factor or more than one adversary to attack the MT-FedRL with performance similar to *AdAMInG* with single-adversary and unity scaling factor λ, as shown in Fig. 4.

A similar trend can be observed with varying the number of non-adversarial agents. It can be seen in Fig. 9 that for a smaller number of non-adversarial agents (equivalently smaller number of total agents if the number of the adversarial agents is fixed), it is easier for the adversary to attack with a high p_{sa}. The reason behind this is that the local update in Eq. 6 is proportional to the number of non-adversarial agents. With a smaller number of non-adversarial agents, the local update is smaller compared to the update by the adversary. Among the three attack methods, *AdAMInG* is the most resilient to these two parameters (λ, n), hence making it a better choice for an adversarial attack in MT-FedRL.

Resolving Adversaries: We implement the N-agent single-adversary MT-FedRL problem using `ComA-FedRL` to address the high p_{sa} of the conventional FedRL algorithm. Fig. 10 compares the performance of FedRL and `ComA-FedRL` for different attack methods. By assigning a higher communication interval to the probable adversary, `ComA-FedRL` was able to decrease the probability of successful attack p_{sa} in the presence of adversary to as low as $< 10\%$. The mean communication interval for adversarial and non-adversarial agents is shown in Fig. 11. It can be seen that Random policy attack has a slightly higher communication interval. The reason behind this is one of the non-adversarial agents was incorrectly marked as a probable adversarial agent at the beginning of training, but later that was self-corrected to a *possibly-non-adversarial* agent.

7.2. AutoNav - NN based RL

We also experiment on a more complex problem of drone autonomous navigation in 3D realistic environments (Fig. 16). The effectiveness of MT-FedRL-achieved unified policy is quantified by drone Mean Safe Flight (MSF). Similar as GridWorld, the performance of the adversarial attack is characterized by the probability of successful attack p_{sa}. The detailed AutoNav problem description and metric definition can be found in Appendix D.2.

Effect of Adversaries: For each experiment, the MT-FedRL problem is trained for 4000 episodes using the REINFORCE algorithm with a learning rate of 1e-4 and $\gamma = 0.99$. Training hyper-parameters are listed in the Appendix D.2 in detail. Table 1 reports the MSF achieved by the AutoNav problem for various attack methods. It can be seen that except for the *AdAMInG* attack, the rest of the attack methods achieve MSF comparable to the one achieved in the absence of an adversary. Fig. 5 shows the p_{sa} for different attack methods. It can be seen that *AdAMInG* achieves a p_{sa} of $\sim 99.5\%$ while all the other attack methods achieve a p_{sa} of $< 6\%$. The trend is similar to the GridWorld task.

Resolving Adversaries: We implement the N-agent single-adversary MT-FedRL problem using `ComA-FedRL` to address the low MSF of FedRL. The results are reported in Table 1. It can be seen that the decrease in MSF due to adversary was recovered using `ComA-FedRL`. Fig. 5 plots the p_{sa} for various attack methods with `ComA-FedRL` and compares it with `FedRL`, showing that with `ComA-FedRL` we have $p_{sa} < 10\%$. Hence `ComA-FedRL` was able to address the issue of adversaries in a MT-FedRL problem.

8. Conclusion

In this paper, we analyze the Multi-task Federated Reinforcement Learning algorithm with an adversarial perspective. We analyze the attacking performance of some gen-

eral attack methods and propose an adaptive attack method *AdAMInG* that devises an attack taking into account the aggregation operator of federated RL. The *AdAMinG* attack method is formulated and its effectiveness is studied. Furthermore, to address the issue of adversaries in the MT-FedRL problem, we propose a communication adaptive modification to conventional federated RL algorithm, `ComA-FedRL`, that varies the communication frequency for the agents based on their probability of being an adversary. Results on the problems of various scales show that the *AdAMInG* attack outperforms other attack methods almost every time. Moreover, `ComA-FedRL` can recover from the adversarial attack resulting in near-optimal policies.

References

Anwar, A. and Raychowdhury, A. Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning. *IEEE Access*, 8:26549–26560, 2020.

Anwar, M. A. and Raychowdhury, A. Navren-rl: Learning to fly in real environment via end-to-end deep reinforcement learning using monocular images. In *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pp. 1–6. IEEE, 2018.

Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pp. 634–643. PMLR, 2019.

Blanchard, P., Guerraoui, R., Stainer, J., et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pp. 119–129, 2017.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, H. B., et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., and Russell, S. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Huang, Y. and Zhu, Q. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *International Conference on Decision and Game Theory for Security*, pp. 217–237. Springer, 2019.

Kober, J., Bagnell, J. A., and Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

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

Krishnan, S., Lam, M., Chitlangia, S., Wan, Z., Barth-Maron, G., Faust, A., and Janapa Reddi, V. Quarl: Quantization for sustainable reinforcement learning. *Transactions on Machine Learning Research*, 2022.

Kumar, S., Shah, P., Hakkani-Tur, D., and Heck, L. Federated control with hierarchical multi-agent deep reinforcement learning. *arXiv preprint arXiv:1712.08266*, 2017.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lim, H.-K., Kim, J.-B., Heo, J.-S., and Han, Y.-H. Federated reinforcement learning for training control policies on multiple iot devices. *Sensors*, 20(5):1359, 2020.

Liu, B., Wang, L., and Liu, M. Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems. *IEEE Robotics and Automation Letters*, 4(4):4555–4562, 2019.

Ma, Y., Zhang, X., Sun, W., and Zhu, J. Policy poisoning in batch reinforcement learning and control. In *Advances in Neural Information Processing Systems*, pp. 14570–14580, 2019.

Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., and Savarese, S. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3932–3939. IEEE, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control

through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Palmer, G., Tuyls, K., Bloembergen, D., and Savani, R. Lenient multi-agent deep reinforcement learning. *arXiv preprint arXiv:1707.04402*, 2017.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

Rodríguez-Barroso, N., Martínez-Cámara, E., Luzón, M., Seco, G. G., Veganzones, M. Á., and Herrera, F. Dynamic federated learning model for identifying adversarial clients. *arXiv preprint arXiv:2007.15030*, 2020.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tessler, C., Efroni, Y., and Mannor, S. Action robust reinforcement learning and applications in continuous control. *arXiv preprint arXiv:1901.09184*, 2019.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Wan, Z., Anwar, A., Hsiao, Y.-S., Jia, T., Reddi, V. J., and Raychowdhury, A. Analyzing and improving fault tolerance of learning-based navigation systems. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 841–846. IEEE, 2021.

Wan, Z., Anwar, A., Mahmoud, A., Jia, T., Hsiao, Y.-S., Reddi, V. J., and Raychowdhury, A. Frl-fi: Transient fault analysis for federated reinforcement learning-based navigation systems. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 430–435. IEEE, 2022.

Wang, C., Wang, J., Shen, Y., and Zhang, X. Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 68(3):2124–2136, 2019.

Xiao, L., Wan, X., Dai, C., Du, X., Chen, X., and Guizani, M. Security in mobile edge caching with reinforcement learning. *IEEE Wireless Communications*, 25(3):116–122, 2018.

Xie, C., Wu, Y., Maaten, L. v. d., Yuille, A. L., and He, K. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

Yager, R. R. and Filev, D. P. Induced ordered weighted averaging operators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(2):141–150, 1999.

Zeng, S., Anwar, A., Doan, T., Romberg, J., and Raychowdhury, A. A decentralized policy gradient approach to multi-task reinforcement learning. *arXiv preprint arXiv:2006.04338*, 2020.

Zhuo, H. H., Feng, W., Xu, Q., Yang, Q., and Lin, Y. Federated reinforcement learning. *arXiv preprint arXiv:1901.08277*, 2019.

Appendix

A. Common Attack Models

A.1. Random Policy Attack (Rand)

This attack will be used as a baseline for the other attack methods. In a Random policy attack, the adversarial agent maintains a set of random policy parameter sampled from a Gaussian distribution with mean 0 and standard deviation $\sigma \in \mathbb{R}$ i.e. for each element $\theta_{adv,j}$ of θ_{adv}

$$\theta_{adv,j} \sim \mathcal{N}(0, \sigma) \tag{11}$$

A.2. Opposite Goal Policy Attack (OppositeGoal)

This attack method assumes that a sample environment is available for the agent to devise the attack. In this attack method, the adversary l learns a policy $\pi^{OG}_{\theta_{adv}}$ utilizing its local environment with the goal of minimizing (instead of maximizing) the long term discounted return i.e. the objective function to maximize is

$$J(\theta_{adv}) = -V_l^{\pi_{\theta_{adv}}}(\rho_l) \tag{12}$$

B. Mathematical Analysis of the *AdAMInG* Attack

Fig. 6 plots $g(\lambda^k, n)$ as a function of the number of agents n for a scaling factor of 1 ($\lambda^k = 1$). It can be seen that as the number of agents increases, the steady-state value g_{ss} becomes closer to 1 making it difficult for *AdAMInG* to have a successful attack with a scaling factor of 1. As the number of agents increases, the update carried out by the non-adversarial agent has a more significant impact on the smoothing average than the adversarial agent making it harder for the adversarial agent to attack. Fig. 7 plots $g(\lambda^k, n)$ as a function of the scaling factor λ^k for $n = 100$. It can be seen that the scaling factor has a linear impact on the success of the *AdAMInG* attack. The performance of the *AdAMInG* attack increases linearly with the increase in the scaling factor. The best *AdAMInG* attack is achieved when $\lambda^k = n - 1$. Section 7 demonstrates that a non-zero steady-state value can still result in a successful attack for a small learning rate δ.

It is safe to assume that if we do not change the learning rate (and it is small enough), we can find the scaling factor required to achieve the same attacking performance by increasing the number of agents n. The steady-state relationship between n and λ in Section 5 lets us analyze the relative attacking performances by varying the number of agents n. Let's say that for n_1 number of agents and a given learning rate that is small, we were able to achieve a successful attack with λ_1. Now to achieve the same successful attack for n_2 number of agents we need

$$\lambda_2 = \frac{n_2(1 + \lambda_1)}{n_1} - 1 \tag{13}$$

Table 2. Cross-evaluation of policies in `ComA-FedRL` in terms of cumulative return

Evaluated Policy	Environment	Cumulative reward
Non-adv	Non-adv	High
Non-adv	Adv	Low (secondary attack)
Adv	Non-adv	Low
Adv	Adv	Low (secondary attack)

C. Training Details

Policy gradient methods for RL is used to train both the GridWorld and AutoNav RL problems. The `ComA-FedRL` algorithm can be seen in Alg. 1. For `ComA-FedRL`, we use a base communication $base\_comm$. In the pre-train phase, the communication interval for each agent is assigned this base communication i.e.

$$comm[i] = base\_comm \quad \forall i \in \{0, \ldots, n-1\}$$

This means that in the pre-train phase, the agents learn only on local data, and after every $base\_comm$ number of episodes, the locally learned policies are shared with the server for cross-evaluation. This cross-evaluation runs n policies, each on a randomly selected environment and the cumulative reward is recorded. We also take into account the fact that the adversarial agent can present a secondary attack in terms of faking the cumulative reward that it return when evaluating a policy. In the `ComA-FedRL` implementation, we assume that the adversarial agent returns a cumulative reward of -1, meaning that it fakes the policy being evaluated as adversarial.

At the end of the pre-train phase, the cross evaluated rewards are used to assign communication intervals to all the agents. There are various choices for the selection of this mapping. The underlying goal is to assign a higher communication interval for agents whose policy performs poorly when cross-evaluated and vice versa. We use the mapping shown in Alg. 2. A reward threshold r_{th} is used to assign agents different communication intervals. If the cumulative reward of a policy in an environment is below r_{th}, it is assigned a high communication interval of $high\_comm$ episodes (marked as a possible adversary), otherwise it is assigned a low communication interval of low_comm episodes (marked as a possible non-adversary). The assigned communication interval also depends on the one-step history of communication intervals. If an agent was previously assigned a higher communication interval and is again marked as a possible adversary, the communication interval assigned to such an agent is doubled. The complete list of hyperparameters used for GridWorld and AutoNav can be seen in Table 4.

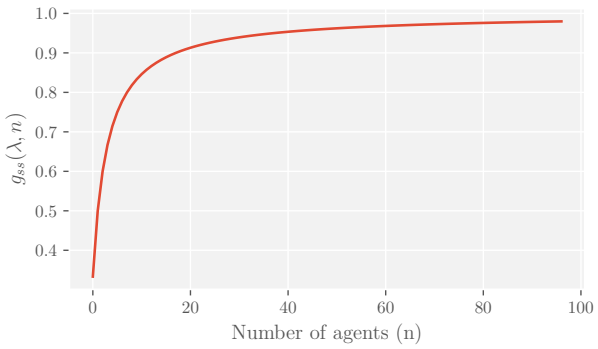

Figure 6. $g\left(\lambda^k, n\right)$ as a function of $\lambda^k = 1$ and n

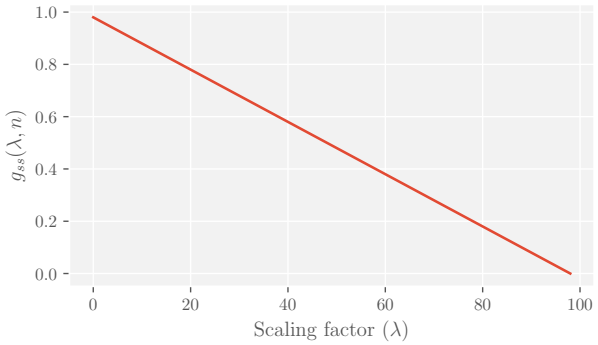

Figure 7. $g\left(\lambda^k, n\right)$ as a function of λ^k and $n = 100$

D. Problem Description of MT-FedRL GridWorld and AutoNav

D.1. MT-FedRL GridWorld Problem Description

The environments of GridWorld are mazes of size 10×10 as seen in Fig. 8. Each cell in the maze is characterized into one of the following 4 categories: *hell-cell* (red), *goal-cell* (yellow), *source-cell* (green), and *free-cell* (white). The agent is initialized at the *source-cell* and is required to reach the *goal-cell* avoiding getting into the *hell-cell*. The *free-cells* in the maze can be occupied without any consequences. The agent can take one of the following 4 actions $\mathcal{A} = \{\texttt{move-up}, \texttt{move-down}, \texttt{move-right}, \texttt{move-left}\}$ which corresponds to the agent moving one cell in the respective direction. At each iteration, the agent observes a one-step SONAR-based state $s \in \mathcal{R}^4$ corresponding to the four cells (up, down, right, left) around the agent. At each iteration, the agent samples an action from the action space and based on the next state, observes a reward. The reward is -1, 1, 0.1, or -0.1 if the agent crashed into hell-cell, reached the goal, moved closer to or away from the goal, respectively. The effectiveness of the MT-FedRL unified policy is quantified by the win ratio (WR) defined by

$$WR = \frac{1}{n-1} \sum_{i \neq l} \frac{\text{\# of times agent } i \text{ reached goal state}}{\text{total \# of attempts in environment } i}$$

In this 12-agent MT-FedRL system, agent 0 is assigned the adversarial role ($l = 0$). The goal for agent 0 is to decrease WR. We characterize the performance of the adversarial attack by the probability of successful attack p_{sa} given by

$$p_{sa} = 1 - \frac{WR_{adv}}{WR_{no-adv}}$$

where WR_{adv} is the win ratio with an adversary, while WR_{no-adv} is the win ratio without any adversary.

D.2. MT-FedRL AutoNav Problem Description

We use PEDRA (Anwar & Raychowdhury, 2020) as the drone navigation platform. The drone is initialized at a starting point and is required to navigate across the hallways of the environments. There is no goal position, and the drone is required to fly avoiding the obstacles as long as it can. At each iteration t, the drone captures an RGB monocular image from the front-facing camera which is taken as the state $s_t \in \mathbb{R}^{(320 \times 180 \times 3)}$ of the RL problem. Based on the state s_t, the drone takes an action $a_t \in \mathcal{A}$. We consider a perception based probabilistic action space with 25 actions ($|\mathcal{A}| = 25$). A depth-based reward function is used to encourage the drone to stay away from the obstacles. We use neural network-based function approximation to estimate the action probabilities based on states. The used network is shown in Appendix Fig. 17. We consider 4 indoor environments (indoor-twist, indoor-frogeyes, indoor-pyramid, and indoor-complex) hence we have $n = 4$. These environments can be seen in Fig. 16.

The effectiveness of MT-FedRL-achieved unified policy is quantified by Mean Safe Flight (MSF) defined as

$$MSF = \frac{1}{n-1} \mathbb{E}\left[\sum_{i \neq l} d_i\right]$$

where d_i is the distance traveled by the agent in the environment i before crashing. In this 4-agent MT-FedRL system, the agent in the environment indoor-complex is assigned the adversarial role ($l = 3$). The goal for the adversarial agent is to decrease this MSF. We will characterize the performance of the adversarial attack by the probability of successful attack p_{sa} given by

$$p_{sa} = 1 - \frac{MSF_{adv}}{MSF_{no-adv}}$$

where MSF_{adv} is the mean safe flight of the MT-FedRL system in the presence of the adversary, while MSF_{no-adv} is the mean safe flight of the MT-FedRL system in the absence of the adversary. The greater the p_{sa} the better the attack method in achieving its goal.

The C3F2 neural network architecture used in AutoNav is shown in Fig. 17.

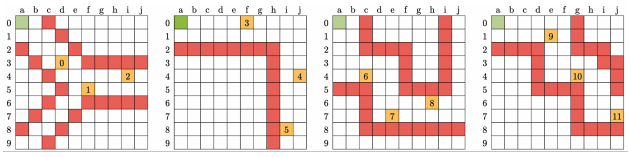

Figure 8. [GridWorld] The 12 environments used

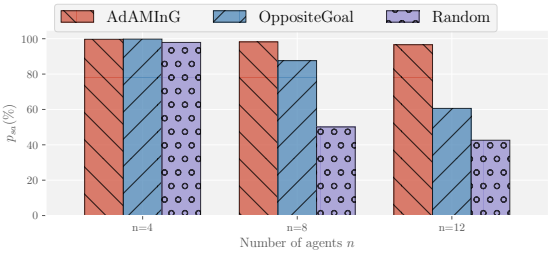

Figure 9. [GridWorld] Effect of number of agents (n) on the performance of attack methods with $\lambda = 1$ and $\delta = 0.2$.

E. Experimental Analysis of the *AdAMInG* Attack

We carry out a detailed analysis of the *AdAMInG* attack method. The smoothing average (Eq. 6) in the presence of an adversary carries out two updates - the local update which moves the policy parameter in a direction to maximize the collective goal, and the adversarial update which tries to move the policy parameter away from the consensus. When the training begins, the initial set of policy parameters θ_i is farther away from the consensus θ^*. Gradient descent finds a direction from the current set of policy parameters to the consensus. This direction has a higher magnitude when the distance between the current policy parameter and the consensus is high. As the system learns, the current policy parameter gets closer to the consensus, and hence the magnitude of the direction of update decreases. So even if we have a static learning rate δ, the magnitude of local update $\delta_j \nabla_{\theta_j} V_j^{\pi_{\theta_j}}(\rho_j)$ in Eq. 6 will, in general, decrease as the system successfully learns. There will be a point in training where the local update will become equal but opposite to the update being carried out by the *AdAMInG* adversary. From that point onwards, the current policy parameter won't change much. This can be seen in Fig. 12. The greater the learning rate δ, the earlier in training we will get to the equilibrium point, and hence poorer the attack performance which can be seen in terms of the achieved discounted return in Fig. 13. A greater standard deviation of the consensus policy parameter indicates a better differentiation between good and bad actions for a given state. Fig. 14 plots the standard deviation of the consensus policy parameter for different learning rates δ. It can be seen that for higher learning rates, the consensus has a higher standard deviation hence being able to perform better than the consensus achieved under lower learning rates.

We also compare the performance of the *AdAMInG* attack in

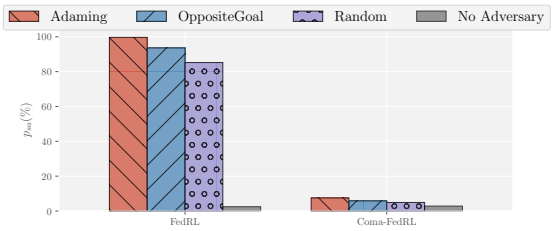

Figure 10. [GridWorld] Comparison of probability of successful attack $p_{sa}(\%)$ under different attack models for `FedRL` and `ComA-FedRL`.

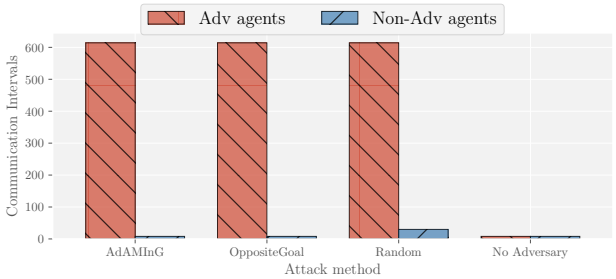

Figure 11. [GridWorld] Average communication intervals for adversarial and non adversarial agents in `ComA-FedRL`

relation to the scaling factor λ and the number of agents n. According to Eq. 13 an increase in the number of agents can be compensated by increasing the scaling factor λ to achieve the same attacking performance. We analyse the *AdAMInG* attack for the following two configurations: $(\lambda = 1, n = 8)$ and $(\lambda = 2, n = 12)$. Table 3 reports the p_{sa} and the standard deviation of the consensus policy parameter θ^*. It can be seen that both the configurations generate similar numbers. The same trend can be observed temporally, in Fig. 15, for the achieved discounted return during each episode in training.

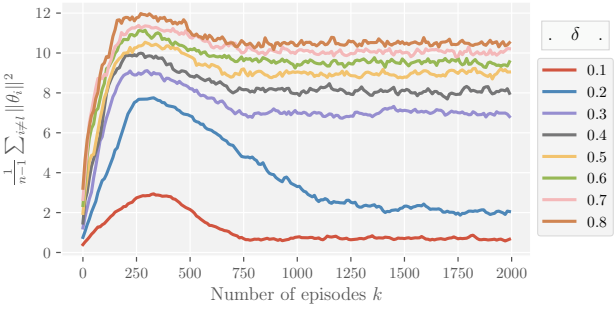

Figure 12. [GridWorld] Based on the learning rate, the consensus gets converged to an intermediate value

Algorithm 1 Communication Aware Federated RL

Initialization: Number ofagents n, $\theta_i^0 \in \mathbb{R}^d$, step size δ^k, $base\_comm \in \mathbb{R}$, $comm[i] = base\_comm \ \forall i \in \{0, n-1\}$, $wait\_comm \in \mathbb{R}$

for $k = 1, 2, 3, \ldots$ **do**
  % Pre-train phase
  if $k \leq wait\_comm$ **then**
    if $k\%base\_comm = 0$ **then**
      for each agent i **in parallel do**
        $\theta_i^{(k+1)-} \leftarrow \texttt{ClientUpdate}\left(i, \theta_i^{k+}\right)$
      $r \leftarrow \texttt{CrossEvalPolicies}\left(r, \theta^{(k+1)-}\right)$
  else
    Calculate smoothing average parameters α_k, β_k
    $comm \leftarrow \texttt{UpdateCommInt}(r, comm)$

    for each agent i **in parallel do**
      if $k\%comm[i] = 0$ **then**
        $\theta_i^{(k+1)-} \leftarrow \texttt{ClientUpdate}\left(i, \theta_i^{k+}\right)$
    $num\_active\_agents = 0$

    for each agent i **do**
      if $k\%comm[i] = 0$ **then**
        num_active_agents+=1

$$\theta_i^{(k+1)+} = \alpha^k \theta_i^{(k+1)-} + \beta^k \sum_{i \neq j} \theta_j^{(k+1)-}$$

        Send $\theta_i^{(k+1)+}$ back to client i
    if $num\_active\_agents = n$ **then**
      $r \leftarrow \texttt{CrossEvalPolicies}\left(r, \theta^{(k+1)-}\right)$

function $\texttt{CrossEvalPolicies}(r, \theta)$
  for each agent i **in parallel do**
    Randomly assign each agent i another agent j without replacement
    $r_j.append(\texttt{ClientEvaluate}(i, \theta_j))$
  return r

function $\texttt{ClientUpdate}(i, \theta)$
  Compute the gradient of the local value function $\frac{\partial V_i^{\pi_\theta}(\rho_i)}{\partial \theta_{s_i, a_i}}$ based on the local data;
  Update the policy parameter

$$\theta^- = \theta + \delta^k \frac{\partial V_i^{\pi_\theta}(\rho_i)}{\partial \theta_{s_i, a_i}}$$

  return θ^-

function $\texttt{ClientEvaluate}(i, \theta_j)$
  Evaluate the policy θ_j on agent i and return the cumulative reward ret
  return ret

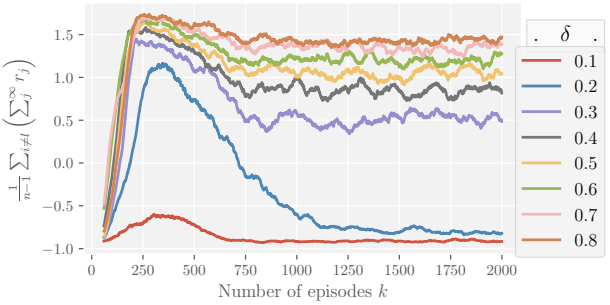

Figure 13. [GridWorld] Cumulative return (moving average of 60) for different learning rate (δ) and $n = 12$

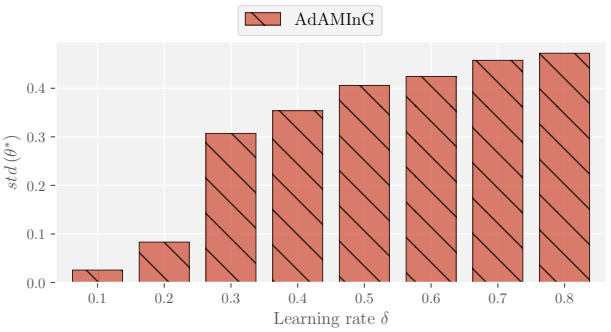

Figure 14. [GridWorld] Standard deviation of the consensus policy parameter

Table 3. [GridWorld] Relationship between λ and n for same attack performance with *AdAMInG*

Configuration	Learning rate δ	$p_{sa}\%$	std
$\lambda = 1, n = 8$	0.2	99.75%	0.036
$\lambda = 2, n = 12$	0.2	99.49%	0.031

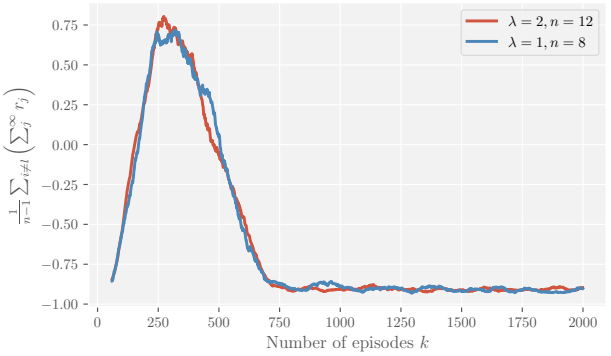

Figure 15. [GridWorld] Relationship between λ and n for same *AdAMInG* attack performance. $(\lambda = 1, n = 8)$ and $(\lambda = 2, n = 12)$ follows the same discounted return across episodes which is in accordance with Eq. 13

Table 4. Training hyper-parameters for GridWorld and AutoNav

HyperParameter	GridWorld	AutoNav
Functional Mapping	Tabular	Neural Network
Number of agents	4, 8, 12	4
Algorithm	REINFORCE	REINFORCE
Max Episodes	1000	4000
Gamma	0.95	0.99
Learning rate	Variable	1e-4
$base\_comm$	8	8
$wait\_train$	600	1000
Gradient clip norm	None	0.1
Optimizer type	ADAM	ADAM
Entropy Regularizer	None	0.5
Training station	GTX1080	GTX1080
Training Time	9 hours	35 hours

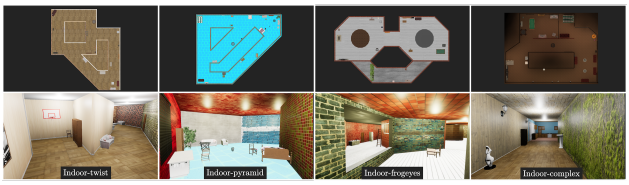

Figure 16. [AutoNav] Floor plan and screenshot of the four 3-D environments used (from left to right: Indoor-twist, Indoor-pyramid, Indoor-frogeyes, Indoor-complex).

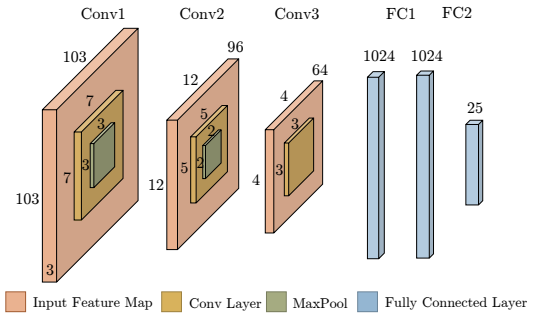

Figure 17. [AutoNav] C3F2 neural network used to map states to action probabilities

Algorithm 2 Update Communication Intervals

function `UpdateCommInt`($r_{m\times n}$, $comm$)
  Initialize low_comm, $high\_comm$, r_{th}
  for each agent i **do**
    Average the rewards across episodes

$$r_{avg} \leftarrow \frac{1}{m}\sum_{j=0}^{m-1} r[:, i]$$

  if $r_{avg} \geq r_{th}$ **then**
    $comm[i] = low\_comm$
  else if $r_{avg} < r_{th}$ **then**
    if $comm[i] \neq low\_comm$ **then**
      $comm[i] = 2 * comm[i]$
    else
      $comm[i] = high\_comm$