COORDINATED ATTACKS AGAINST FEDERATED LEARNING: A MULTI-AGENT REINFORCEMENT LEARNING APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a model-based multi-agent reinforcement learning attack framework against federated learning systems. Our framework first approximates the distribution of the clients' aggregated data through cooperative multi-agent coordination. It then learns an attack policy through multi-agent reinforcement learning. Depending on the availability of the server's federated learning configurations, we introduce algorithms for both white-box attacks and black-box attacks. Our attack methods are capable of handling scenarios when the clients' data is independent and identically distributed and when the data is independent but not necessarily identically distributed. We further derive an upper bound on the attacker's performance loss due to inaccurate distribution estimation. Experimental results on real-world datasets demonstrate that the proposed attack framework achieves strong performance even if the server deploys advanced defense mechanisms. Our work sheds light on how to attack federated learning systems through multi-agent coordination.

1 INTRODUCTION

Federated learning (FL) is a powerful machine learning framework that allows a server to train machine learning models across multiple workers that hold local data samples, without exchanging them. Unfortunately, federated learning systems are vulnerable to threats (Lyu et al., 2020) such as model poisoning attacks (Fang et al., 2020; Bagdasaryan et al., 2020; Bhagoji et al., 2019), data poisoning attacks (Baruch et al., 2019; Fung et al., 2018; Gu et al., 2017), and inference attacks (Melis et al., 2019; Hitaj et al., 2017; Zhu et al., 2019). These attacks are effective even when the server applies robust aggregation rules such as coordinate-wise median (Yin et al., 2018), trimmed mean (Yin et al., 2018), Krum (Blanchard et al., 2017), or Bulyan (Mhamdi et al., 2018). Nevertheless, a recent study (Cao et al., 2020) shows that the server can collect a small clean training dataset to bootstrap trust to defend a variety of attacks (Fang et al., 2020; Bagdasaryan et al., 2020). Their results show that the FLTrust (Cao et al., 2020) defense is capable of achieving a high level of robustness against a large fraction of adversarial attackers. While these defense mechanisms focus on limiting the influence of the attackers in a single round, there also exists more adaptive defense algorithms such as Safeguard (Allen-Zhu et al., 2020) and Centered Clipping (Karimireddy et al., 2021), both of which incorporate historical gradient information into the aggregation rule. These adaptive defense mechanisms are shown to be capable of overcoming time-couple attacks. In this work, we propose a novel *multi-agent reinforcement learning* (MARL) attack framework that is effective even if the server deploys the state-of-the-art defense mechanism such as FLTrust (Cao et al., 2020), Safeguard (Allen-Zhu et al., 2020) and Centered Clipping (Karimireddy et al., 2021).

Learning effective attack policies against federated learning is challenging. A major reason is that it typically requires attackers to obtain sufficient information about the federated learning environment and training dynamics that includes both the server's behavior and the normal workers' data and behavior. However, except for the model parameters, attackers often have rather limited information about the FL dynamics in practice. To address this problem, we propose to leverage the power of model-based reinforcement learning by integrating *distribution learning* and *policy learning* in our MARL attack framework. In MARL, the attackers first cooperatively learn a model of the aggregated data distribution through gradient matching, and then simulate the behavior of the server and the benign workers using the learned distribution. In doing so, the attackers jointly learn an attack policy through multi-agent reinforcement learning. Our MARL attack framework allows attackers to

implement attack policies in both white-box (i.e., attacks have some information about the server's algorithm) and black-box (i.e., attacks have no access to the server's algorithm) settings.

While there is much research on adversarial attacks against federated learning, a majority of existing attack methods (e.g., (Bhagoji et al., 2019; Fang et al., 2020; Xie et al., 2020b)) typically craft myopic attack strategies based on heuristics. Further, most of them consider independent attackers without coordination. Recently, a distributed backdoor attack (DBA) method is proposed in (Xie et al., 2020a) where a global trigger pattern is manually decomposed into local patterns that are embedded to different attackers. Compared with DBA, our MARL method enables the attackers to jointly learn an attack policy through coordinating the behavior of attackers in distribution learning, policy learning and attack execution, while their distributed backdoor attacks are coordinated in attack execution (i.e., trigger injection) only.

Our model-based multi-agent reinforcement learning attack framework distinguishes from existing work (Sun et al., 2019; Zhang et al., 2020) on reinforcement learning based adversarial attacks by considering a more realistic threat model where the attacker might not always be selected due to subsampling nor does it have prior information about the distribution of the aggregated data. The attackers need to efficiently learn the distribution along the federated learning process in real time. We also consider the scenario when the attackers have no access to the server's configurations. In contrast, previous works typically assume more powerful attackers that can attack at any time and have full knowledge about the environment.

Our contributions. We advance the state-of-the-art in the following aspects. First, we propose a novel multi-agent reinforcement learning attack framework against federated learning systems by integrating distribution learning and policy learning through multi-agent coordination. Second, we propose algorithms for both white-box attacks and black-box attacks. Third, we theoretically quantify the effect of inaccurate distribution learning on the optimality of policy learning. Fourth, our experiments on real-world datasets demonstrate that our proposed MARL method consistently outperforms existing model poisoning attacks (Bhagoji et al., 2019; Fang et al., 2020; Xie et al., 2020b) due to multi-agent coordination in distribution learning, policy learning and attack execution.

2 BACKGROUND

In this section, we describe the federated learning setting used in our work and present the threat model.

Federated learning. We consider an FL setting that is similar to *federated averaging (FedAvg)* (McMahan et al., 2017). The FL system consists of a server and K workers (also known as workers or clients) in which each worker has some private data. Coordinated by the server, the set of workers cooperate to train a machine learning model within $\mathcal{T}$ epochs by solving the following problem: $\min_\theta f(\theta)$ where $f(\theta) := \sum_{k=1}^{K} p_k F_k(\theta)$ where $F_k(\cdot)$ is the local objective of worker k and p_k is the weight assigned to worker k and satisfies $p_k \geqslant 0$ and $\sum_k p_k = 1$. The local objective $F_k(\theta)$ is usually defined as the empirical risk over worker k's local data with model parameter $\theta \in \Theta$. That is, $F_k(\theta) = \frac{1}{N_k} \sum_{j=1}^{N_k} \ell(\theta; (x_{jk}, y_{jk}))$, where N_k is the number of data samples available locally on worker k, $\ell(\cdot, \cdot)$ is the loss function, and $(x_{jk}, y_{jk}) := z_{jk}$ is the jth data sample that is drawn *i.i.d.* from some distribution P_k. It is typical to set $p_k = \frac{N_k}{N}$, where $N = \sum_k N_k$ the total number of data samples across workers.

If all the local workers' data distributions are the same (i.e., $P_k = P_{k'}$ for all $k, k' \in [K]$), we call the workers' data are *i.i.d.*; otherwise, the data are *non-i.i.d.*. We write $\widehat{P}_k$ as the empirical distribution of the N_k data samples drawn from P_k, and let $\widehat{P} := \sum_{k=1}^{K} \frac{N_k}{N} \widehat{P}_k$ denote the mixture empirical distribution across workers.

The FL algorithm (see Algorithm 1 in Appendix B.2) works as follows: at each time step t, a random subset $\mathcal{S}^t$ of size w is uniformly sampled without replacement from the workers set $[K]$ by the server for synchronous aggregation (Li et al., 2019). The process of selecting workers for aggregation is called *subsampling*. Let $\kappa = w/K$ denote the *subsampling rate*. Each selected worker $k \in [w]$ then samples a minibatch b_k of size B from its local data distribution P_k. The worker then calculates the average local gradient $g_k^{t+1} \leftarrow \frac{1}{B} \sum_{z \in b_k} \nabla_\theta \ell(\theta^t; z)$ and sends the gradient to the server. The server then uses an aggregation rule to compute the aggregated gradient $g^{t+1} \leftarrow Aggr(g_{k_1}^{t+1}, ..., g_{k_w}^{t+1})$ where $k_i \in \mathcal{S}^t$, and updates the global model parameters $\theta^{t+1} \leftarrow \theta^t - \eta g^{t+1}$ where η is the learning

rate. The newly updated model parameters θ^{t+1} are then sent to the selected workers to perform the next FL iteration.

Threat Model. We assume that among the K workers, $M(1 \leqslant M < K)$ of them are malicious. Let $\mathcal{A}$ denote the set of malicious attackers. Further, these attacker are fully cooperative and share the same goal of compromising the FL system. We consider untargeted model poisoning attacks where the M cooperative attackers send crafted local updates $\{\widetilde{g}_k^t\}_{k\in\mathcal{A}}$ to the server in order to maximize the empirical loss, i.e., $\max_\theta f(\theta)$. They are coordinated either by one leading attacker or an external agent. We refer such agent as a *leader agent*.

We assume the attackers know the global model parameters received from the server, $\{\theta^t\}$, the local training algorithm (including the batch size B) and their local data distributions $\{\widehat{P}_k\}_{k\in\mathcal{A}}$. In *white-box* attacks, we assume that the attackers further obtain information about the server's training algorithm. This information includes the server's learning rate η, the subsampling rate κ, the total number of workers K, and the aggregation rule $Aggr$. In *black-box* attacks, the attackers have no access to the server's training algorithm. They must estimate these parameters instead.

In practice, the attackers may communicate with each other to share their local information such as local data distributions, the status of whether being selected by the server or not, their unique identifiers, and each attacker's action. Such internal communication allows each selected attacker i to obtain the following information at each time step t: the number of attackers being selected by the server $m^t(0 \leqslant m^t \leqslant M)$, and its rank information $\sigma_i^t \in \{0, ..., m^t - 1\}$ in the set of the selected attackers $\mathcal{A}^t$. The rank is obtained by sorting the selected attackers according to their identifiers in ascending order. The rank of a selected malicious worker plays a similar role as a unique identifier does. It distinguishes the malicious worker from the other selected malicious workers. In our work, we use the ranks instead of the unique identifiers to accelerate policy training in the presence of worker subsampling. In addition, we assume the attackers obtain a lower bound of the total number of training epochs $\mathcal{T}$.

3 MARL Attack Framework Against Federated Learning

3.1 Overview

In this work, our goal is to design a non-myopic coordinated attack against federated learning. To this end, we formulate the attacker's problem as a fully cooperative Markov game (Section 3.2) by viewing the FL training process as the environment. The main obstacle for solving the game, however, is that benign agents' local data distributions are unknown to the attacker, making it difficult to evaluate the effectiveness of a joint attack policy, which is necessary to compute the rewards to the attackers. An important observation of our approach is that although the joint empirical distribution $\{\widehat{P}_k\}_{k\in[K]}$ is unknown, the attackers can learn an approximation of the mixture distribution $\widehat{P} = \sum_{k=1}^K \frac{N_k}{N}\widehat{P}_k$, denoted by $\widetilde{P}$, from model updates shared by the server. In the black-box setting, the attackers can further utilize the model updates to infer FL training parameters, which together with $\widetilde{P}$ are often sufficient to simulate the behavior of benign agents and the server. Thus, our model-based reinforcement learning attack framework naturally consists of the following three stages (see Figure 1), all of which happen while federating learning is ongoing.

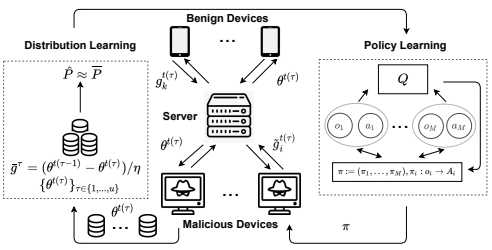

Figure 1: An overview of the MARL attack framework against federated learning.

Distribution learning. Initially, the set of attackers jointly learn a mixture distribution $\widetilde{P}$ from the model updates $\{\theta^t\}$ using a gradient leakage based inference attack (Section 3.3). In the black-box setting, they further infer key FL training parameters from the model updates (Section 3.5).

Policy learning. The learnt parameters are used to build a fully cooperative Markov game for the attackers, which is solved using centralized training to identify the set of decentralized attack policies that are distributed to the attackers (Section 3.3).

Attack execution and update. The set of attackers coordinate to attack the FL system. In the black-box setting, the attack policies are updated when new model updates are received (Section 3.5).

3.2 ATTACKERS' PROBLEM AS A FULLY COOPERATIVE MARKOV GAME

We formulate the attackers' optimization problem as a fully cooperative Markov game, denoted by $\mathcal{M} = (S, A, T, r, H)$, where

- S is the state space. Let $\tau \in \{0, 1, ...\}$ denote the index of the attack step and $t(\tau) \in [\mathcal{T}]$ the corresponding FL epoch when at least one attacker is selected by the server. The state at step τ is defined as $s^\tau := (\theta^{t(\tau)}, \mathcal{A}^{t(\tau)})$ where $\mathcal{A}^{t(\tau)}$ is the set of attackers selected at time $t(\tau)$. We further define the *observation* of attacker i at τ as $o_i^\tau := (\theta^{t(\tau)}, m^{t(\tau)}, \sigma_i^{t(\tau)})$, which can be derived from s^τ. Let O_i denote the space of o_i^τ.
- A is the space of the attackers' joint actions. If attacker i is selected at τ, its action $a_i^\tau := \widetilde{g}_i^{t(\tau)+1} \in \mathbb{R}^d$ is the local update that attacker i sends to the server at time step $t(\tau)$, where d is the dimension of the model parameters. The only action available to an attacker not selected at $t(\tau)$ is $\perp$, indicating that the attacker does not send any information in that step. Let A_i denote the domain of attack i's actions, we have $A := A_1 \times \cdots \times A_M$.
- $T : S \times A \to \mathcal{P}(S)$ is the state transition function that represents the probability of reaching a state $s' \in S$ from the state $s \in S$ when attackers choose actions $a_1^\tau, ..., a_M^\tau$, respectively.
- $r : S \times A \times S \to \mathbb{R}_{\geqslant 0}$ is the reward function. We define the reward at step τ as $r^\tau := f(\theta^{t(\tau+1)}) - f(\theta^{t(\tau)})$, which is determined by the joint actions of attackers and shared by all the attackers. Both the transition probability T and the reward function r are determined by the joint empirical distribution across workers $\{\widehat{P}_k\}_{k\in[K]}$ (fixed but unknown to the attackers), the number of workers K, the number of workers w selected for each time step, the size of local minibatch B, the algorithm used by each worker, the aggregation rule used by the server, and the attackers' actions $a_1^\tau, ..., a_M^\tau$.
- H is the number of attack steps in each episode. H is a hyperparameter that can be adjusted, but we require $t(H) < \mathcal{T}$ so that the attackers have time to execute attacks.

The attackers' goal is to jointly find an attack policy $\pi = (\pi_1, ..., \pi_M)$ that maximizes the expected total rewards over H attack steps, i.e., $\mathbb{E}[\sum_{\tau=0}^{H-1} r^\tau]$, where $\pi_i : O_i \to \mathcal{P}(A_i)$ denotes a stationary policy of attacker i that maps its observation o_i to a probability measure over A_i. Using the definition of r^τ, this objective is equivalent to finding a policy π that maximizes $\mathbb{E}_{\theta^{t(H)}}[f(\theta^{t(H)})]$.

We note that conceptually our problem can be formulated as a single agent RL problem as the set of attackers share a common goal. However, this approach leads to an action space that grows exponentially with the number of attackers, making the direct application of standard RL algorithms difficult. In particular, existing single-agent RL algorithms cannot easily exploit the inherent symmetry in our problem where the ordering of attack actions does not matter. By viewing the problem as a fully cooperative multi-agent Markov game, the symmetry can be naturally incorporated by using the rank information as part of individual attacker's observation, leading to a more scalable solution.

3.3 WHITE-BOX MARL ATTACKS

In this section, we focus on the white-box attack setting where the attackers obtain information about the server's training algorithm and defer the discussion of the black-box attack to Section 3.5.

Distribution learning. Initially, the attackers do not perform model-poisoning attacks. Instead, they jointly learn a mixture distribution $\widetilde{P}$ from the model updates $\{\theta^t\}$ using a gradient leakage based inference attack (Geiping et al., 2020; Zhu et al., 2019). This gives rise to a new Markov game $\widetilde{\mathcal{M}} = (S, A, T', r', H)$ where T' and r' are derived from $\widetilde{P}$.

Note that from any two consecutive model updates received from the server, the attackers can estimate a batch-level gradient $\bar{g}^\tau := (\theta^{t(\tau-1)} - \theta^{t(\tau)})/(\eta(t(\tau) - t(\tau-1)))$. A gradient leakage attack works by starting with (randomly generated) dummy data and feeding it into the model to get dummy gradients. The dummy data is then iteratively updated until the dummy gradients get close to the real gradients.

Various gradient leakage attacks have been proposed in the literature. In this work, we adapt the *inverting gradients* (IG) method (Geiping et al., 2020) to distribution learning (see Appendix B.2 for the detailed description of our adaptation). The IG method reconstructs data samples by optimizing a loss function based on the angles (i.e., cosine similarity) of gradients to find data samples that lead to a similar change in model prediction as the ground truth. Note that the goal of IG is to reconstruct the original data samples, which is more ambitious than what the attackers need in our setting. On the

other hand, recent works on gradient leakage including IG have focused on the server side, where the true gradients of each individual worker can be easily obtained from model updates. In contrast, the attackers only obtain approximated batch-level gradients due to model aggregation and subsampling. Despite these differences, our experiment results show that the IG method is very effective in learning $\widetilde{P}$ (see Figure 2 in Appendix C.2).

In addition to the IG method, several other existing methods can also be utilized. For instance, the *deep leakage from gradients* method (DLG) (Zhu et al., 2019), the latent projection method (Karras et al., 2020), the DeepInversion method (Yin et al., 2020) and the *GradInversion* method (Yin et al., 2021). It is worth noting that the GradInversion method is capable of reconstructing individual images with high fidelity from averaging gradients even for complex datasets like ImageNet (Deng et al., 2009), deep networks, and large batch sizes. These approaches can potentially be adopted to learn $\widetilde{P}$ in more challenging settings.

Policy learning. With the distribution $\widetilde{P}$ estimated in the distribution learning stage, the leader attacker can build an approximate Markov game $\widetilde{\mathcal{M}} = (S, A, T', r', H)$ by simulating the FL training environment as follows. We consider the white-box setting where the leader attacker knows the total number of workers and FL training parameters. To simulate benign workers' behavior in each FL epoch, the leader attacker assumes that each benign worker has the same amount of data (i.e., $p_k = \frac{1}{K}$), and samples a minibatch that is *i.i.d.* drawn from the same learned distribution $\widetilde{P}$ for each benign worker and computes the respective gradients given the current model $\theta^{t(\tau)}$. To simulate the server's behavior, the leader agent applies the same aggregation rule as the server to compute the next model update.

As the attackers are fully cooperative and the leader agent has all the information needed to simulate system dynamics, we adopt the framework of centralized training with decentralized execution (CTDE). The centralized training can be implemented by the leader agent and no communication with other attackers is needed during policy training. The learned policy is shared with all the attackers at the end of policy training and executed in a decentralized way during attacks.

We use the *multi-agent deep deterministic policy gradient* (MADDPG) method (Lowe et al., 2017) to train attack policies in our experiments, mainly because it allows continuous actions and it is an off-policy algorithm and typically is more sample efficient than on-policy learning algorithms. However, our attack framework is general to incorporate other MARL methods such as the multi-agent TD3 (Ackermann et al., 2019) or the multi-agent PPO (Yu et al., 2021).

Since the attackers share the same reward function and differ in the observations only in our setting, it suffices to train a single centralized action-value function $Q^{\mu}(s, a_1, ..., a_M)$ for all the attackers and a shared deterministic policy $\mu : O \to A$ where $O = \bigcup_i O_i$ and $A = \bigcup_i A_i$. We adapt the MADDPG algorithm to our fully cooperative setting with subsampling and name it CMADDPG (see Algorithm 3 in Appendix B.2).

Attack execution. After policy learning, each selected attacker i sends the crafted gradients according to the learned policy μ and its local observation o_i in the remaining FL epochs. Since attacker i's local observation depends on its rank information, the set of attackers need to communicate with each other in each epoch to share the number of selected attackers as well as their unique identifiers.

3.4 Design Considerations

Space reduction. When we train a small neural network with the federated learning system, it is natural to use $(\theta^{t(\tau)}, \mathcal{A}^{t(\tau)})$ as the state, and the gradient $\widetilde{g}_i^{t(\tau)+1}$ as the action. When we use the federated learning system to train a large neural network, however, this approach does not scale as it results in extremely large search space that requires both large runtime memory and long training time, which is usually prohibitive. To solve this problem, we propose to approximate the state $(\theta^{t(\tau)}, \mathcal{A}^{t(\tau)})$ and the action $g_i^{t(\tau)+1}$ for high-dimensional data. To approximate the state, we use parameters of the last hidden layer of the current neural network model to replace $\theta^{t(\tau)}$ in state $(\theta^{t(\tau)}, \mathcal{A}^{t(\tau)})$. This is because because the last hidden layer passes on values to the output layer and typically carries information about important features of the model (Sun et al., 2014). Note that the true state is still the full FL model that determines transition probabilities and rewards. We further define the action a_i^{τ} as a one-dimension scaling factor $a_i^{\tau} \in [-1, 1]$, and set $\widetilde{g}_i^{t(\tau)+1} = a_i^{\tau} \times g_i^{t(\tau)+1}$, which is used to compute the next state and reward.

Gradient rescaling. To reduce the chance that the gradients from the attackers are filtered out by the defense mechanisms (e.g., coordinate-wise median, Krum, or FLTrust), we further rescale the gradients. When computing the gradients to be sent to the server, the attack method first calculates the maximum value $\zeta_{\max}$ and the minimum value $\zeta_{\min}$ that occur in any dimension of $\nabla_\theta l(\theta^{t(\tau)}; z)$ for any z in the data samples generated in the distribution learning stage. The original scaling factor a^τ is in $[-1, 1]$. The rescaled scaling factor $\tilde{a}^\tau = a^\tau \times (\zeta_{\max} - \zeta_{\min})/|\zeta_{\max}| \times 0.5 + (\zeta_{\max} + \zeta_{\min})/|\zeta_{\max}| \times 0.5$.

Training efficiency. The attackers' total training time (including distribution learning and policy learning) should be significantly less than the total FL training time so that the attackers have time to execute the attacks. In real-world FL training, the server usually must wait for some time (typically ranging from 1 minute to 10 minutes) before it receives responses from the clients (Yang et al., 2018; Bonawitz et al., 2019; Kairouz et al., 2019). In contrast, the leader agent does not incur such time cost in training attackers' policies using a simulated FL environment. Therefore, an epoch in policy learning is typically much shorter than an FL epoch, making it possible to train the attack policy with a large number of episodes. In addition, the leader agent is usually equipped with GPUs, or other parallel computing facilities and can run multiple training episodes in parallel (Clemente et al., 2017).

Non-i.i.d. data. In the basic approach described above, we propose to simulate benign works' behavior using mini-batches $i.i.d.$ sampled from the same distribution $\widetilde{P}$. When the clients' data are known to be *non-i.i.d.*, we further consider the following approach that generates data samples from distributions that are within a Wasserstein ball centered at $\widetilde{P}$ to obtain a more robust attack policy. More concretely, the data samples are *i.i.d.* drawn from $\mathcal{P} = \{P : W_1(P, \widetilde{P}) \leqslant \max_{k \in \mathcal{A}} W_1(\widehat{P}_k, \widetilde{P})\}$, where $\widehat{P}_k$ is attacker k's empirical distribution. In practice, attackers may not obtain prior information about the data heterogeneity of an FL system. In such case, the attackers may measure the Wasserstein distances between the empirical data distributions of any two attackers to decide whether to adopt the *non-i.i.d.* sampling method. If the Wasserstein distances are similar, the attackers may conclude that the workers' data distributions are *i.i.d.*; otherwise, the workers' data are *non-i.i.d.*.

3.5 Black-box MARL Attacks

In our white-box MARL attack described above, we assume that the attackers have access to the server's algorithm including the aggregation rule $Aggr$ and parameters such as the learning rate η, the worker subsampling rate κ, and the total number of workers K. In practice, such knowledge might be unavailable to the attackers. They must estimate these parameters in order to learn an effective attack policy. We call this attack scenario *black-box attacks*. Built upon our white-box attack, we present a black-box MARL attack method. Due to space limitations, we only provide a summary of the high level ideas in this section and defer the detailed discussion to Appendix B.4.

We first introduce methods to estimate the three missing parameters using global model updates and the attackers' local gradients. In particular, we estimate the server's learning rate η using the model difference in two consecutive steps when at least one attacker is sampled divided by the average gradient computed from the attackers' local data. We use the empirical subsampling rate of the malicious workers as an estimate of the server's subsampling rate κ. We estimate the number of workers w sampled in each time step by comparing the variance of the average gradients across all sampled workers' and that of an individual attacker. The estimates of κ and w together give an estimate of K.

We then propose a method to learn a non-linear approximation of the server's aggregation rule. In particular, we use a small neural network with two inputs that model the average gradients of normal workers and that of malicious workers, respectively. The former is estimated from the learned mixture distribution $\widetilde{P}$, while latter is computed using the attackers' local data.

After parameter estimation, the leader attacker first learns an initial attack policy similar to the white-box attack. To further improve the quality of the estimated aggregation rule and subsequently improve the performance of the attack policy, we apply the Adaptation augmented Model-based Policy Optimization (AMPO) (Shen et al., 2020) method to continuously adapt the aggregation rule and learn the attack policy. The AMPO method was originally designed to address the distribution mismatch problem for better policy optimization in Dyna-style model-based reinforcement learning (Sutton, 1990). We adapt the original AMPO by explicitly requiring the model adaptation procedure to learn a non-linear representation of the aggregation rule. The black-box MARL attack method has the same three stages as the white-box MARL attacks: distribution learning, policy learning and attack

execution. The key difference is that the learned policy will be continuously adapted along with attack execution.

4 IMPACT OF INACCURATE DISTRIBUTION AND DATA HETEROGENEITY

Our MARL attack employs the estimated data distribution $\widetilde{P}$ to simulate the behavior of benign workers, which can suffer from two types of errors. First, $\widetilde{P}$ can be far away from the true mixture distribution $\widehat{P}$ due to inaccurate distribution learning. Second, benign workers may vary in their local data distributions $\widehat{P}_k$, which cannot be fully captured by a single mixture distribution. In this section, we study how the attack performance is affected by these two factors, which provides insights into properly distributing resources between the three stages of our attack. To simplify the discussion, we focus on the white-box setting and assume that data samples are drawn from $\widehat{P}$ when simulating benign workers in our analysis, ignoring the further optimization discussed in Sections 3.4 and 3.5.

Our analysis is adapted from recent works that study the impact of model inaccuracy on the performance of model-based reinforcement learning (Yu et al., 2020; Luo et al., 2018; Zhang et al., 2020) by addressing two key challenges. First, we need to establish the connection between the inaccuracy in data distribution $\widetilde{P}$ and the inaccuracy in the corresponding Markov game model as both the reward function and the transition dynamics depend on $\widetilde{P}$. Second, although there are different ways to measure the distance between two models (Yu et al., 2020), it makes more sense to use the Wasserstein distance to measure the distance between two data distributions. This, however, requires bounding the Lipschitz constant of the optimal value function (Yu et al., 2020). Although this is a challenging task for general RL tasks, we are able to show that this is indeed the case in our setting under the following assumptions. The first assumption models the inaccuracy of distribution learning as well as the heterogeneity of benign worker's local data .

Assumption 1. *$W_1(\widetilde{P}, \widehat{P}_k) \leqslant \delta$ for any benign worker k.*

We further need the following standard assumptions on the loss function.

Assumption 2. *Let Z denote the domain of data samples across all the workers. For any $s_1, s_2 \in S$ and $z_1, z_2 \in Z$, the loss function $\ell : S \times Z \to \mathbb{R}$ satisfies:*

1. $|\ell(s_1, z_1) - \ell(s_2, z_2)| \leqslant L\|(s_1, z_1) - (s_2, z_2)\|_2$ *(Lipschitz continuity w.r.t. s and z);*
2. $\|\nabla_s \ell(s_1, z_1) - \nabla_s \ell(s_1, z_2)\|_2 \leqslant L_z\|z_1 - z_2\|_2$ *(Lipschitz smoothness w.r.t. z);*
3. $\ell(s_2, z_1) \geqslant \ell(s_1, z_1) + \langle \nabla_s \ell(s_1, z_1), s_2 - s_1\rangle + \frac{\alpha}{2}\|s_2 - s_1\|_2^2$ *(strongly convex w.r.t. s);*
4. $\ell(s_2, z_1) \leqslant \ell(s_1, z_1) + \langle \nabla_s \ell(s_1, z_1), s_2 - s_1\rangle + \frac{\beta}{2}\|s_2 - s_1\|_2^2$ *(strongly smooth w.r.t. s);*
5. *$\ell(\cdot, \cdot)$ is twice continuously differentiable with respect to s.*

where $\|(s_1, z_1) - (s_2, z_2)\|_2^2 = \|s_1 - s_2\|_2^2 + \|z_1 - z_2\|_2^2$. For simplicity, we further make the following assumption on the FL environment, although our analysis can be readily applied to more general settings.

Assumption 3. *The server adopts FedAvg without subsampling ($w = K$). All workers have same amount of data ($p_k = \frac{1}{K}$) and the local minibatch size $B = 1$. In each epoch of federated learning, each normal worker's local minibatch is sampled independently from the local empirical data distribution $\widehat{P}_k$.*

Let $\mathcal{M} = (S, A, T, r, H)$ denote the true Markov game for the attackers, and $\widetilde{\mathcal{M}} = (S, A, T', r', H)$ the simulated Markov game when the local distribution of any benign worker is estimated as $\widetilde{P}$. Let $T(s'|s, a)$ and $r(s, a, s')$ denote the transition dynamics and reward function for $\mathcal{M}$, respectively. The following theorem captures the attack performance loss due to inaccurate distribution learning (see Appendix D for the proof).

Theorem 1. *Let $\mathcal{J}_{\mathcal{M}}(\pi) := \mathbb{E}_{\pi,T,\mu_0}[\sum_{t=0}^{H-1} r(s^t, a^t, s^{t+1})]$ denote the expected return over H attack steps under $\mathcal{M}$, policy π and initial state distribution μ_0. Let π^* and $\widetilde{\pi}$ be the optimal policies for $\mathcal{M}$ and $\widetilde{\mathcal{M}}$ respectively, with the same initial state distribution μ_0. Then,*

$$|\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\mathcal{M}}(\widetilde{\pi})| \leqslant 2H\epsilon\delta[(L + L_v)L_z\eta + 2L],$$

where $\epsilon = \frac{K-M}{K}$ is the fraction of benign nodes, $L_v \leqslant \sum_{t=0}^{H-1}(K_F)^t(L + LK_F)$ and $K_F \leqslant \epsilon \max\{|1 - \eta\alpha|, |1 - \eta\beta|\}$.

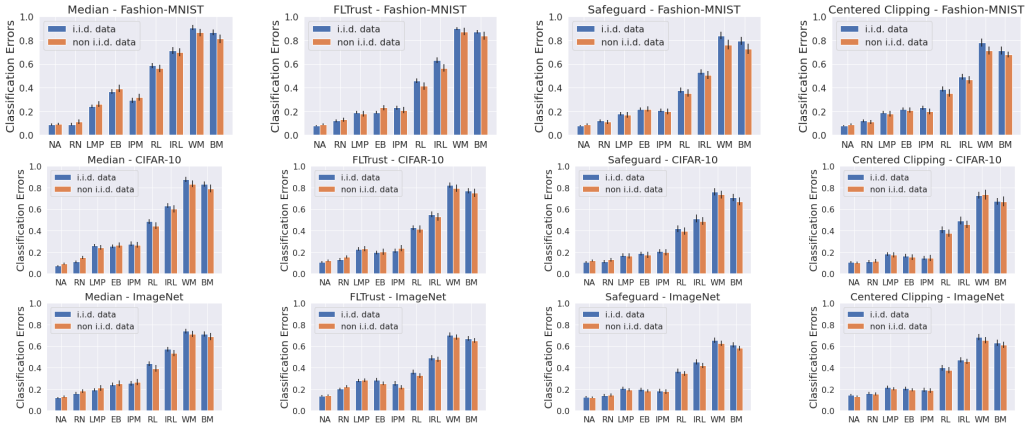

Figure 2: A comparison of average classification error rates for FL with four different aggregation rules (coordinate-wise median, FLTrust, Safeguard, and Centered Clipping) on three different datasets (Fashion-MNIST, CIFAR-10 and ImageNet) for both i.i.d. data and non-i.i.d. data. Key parameters: number of workers = $1,000$, number of attackers = 50, subsampling rate = 20%. All other parameters are set as default. Error bars indicate the standard deviations. The results for Krum follow similar trends and are omitted to save space.

In practice, the learning rate η is typically small enough so that $\max\{|1-\eta\alpha|, |1-\eta\beta|\} \leqslant 1$. In this case, L_v is bounded by $\frac{L(1+K_F)}{1-K_F} \leqslant \frac{L(1+\epsilon)}{1-\epsilon}$. Therefore, we have $|\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\mathcal{M}}(\widetilde{\pi})| = O(H\frac{\epsilon}{1-\epsilon}\eta\delta)$. To ensure convergence, we typically have $\eta = O(\frac{1}{\sqrt{H}})$ (Polyak, 1987), thus $|\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\mathcal{M}}(\widetilde{\pi})| = O(\frac{\epsilon}{1-\epsilon}\delta\sqrt{H})$.

Our theoretical result indicates the impact of imperfect information on attack performance, which can potentially be utilized to derive a proactive defense that manages to increase the attackers' uncertainty.

5 Experiments

We conduct extensive experiments on three real-world datasets: Fashion-MNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009). We compare the performance of different FL attack methods: no attack (NA), random attack (RN), inner product manipulation (IPM) (Xie et al., 2020b), local model poisoning attack (LMP) (Fang et al., 2020), explicit boosting (EB) (Bhagoji et al., 2019). We further consider three variants of our RL-based methods, namely, RL with a single attacker (RL), RL with multiple independent attackers (IRL), and the proposed MARL attacks, to understand the advantage of multi-agent coordination in attacking FL systems. For our MARL attacks, we consider both white-box attack (WM) and black-box attack (BM). We consider the FL settings where the server is equipped with one the following defense mechanisms: Krum (Blanchard et al., 2017), coordinated-wise median (Yin et al., 2018), FLTrust (Cao et al., 2020), Safeguard (Allen-Zhu et al., 2020) and Centered Clipping (Karimireddy et al., 2021). See Appendix C for details of the datasets, experimental setups, and additional results.

Results. Due to no-myopic policy learning, our model-based RL methods (RL, IRL, and MARL) consistently outperform existing methods that craft model updates myopically with heuristics such as LMP, EB and IPM by a large margin (see Figure 2). For instance, both the white-box MARL attack and the black-box MARL attack achieve misclassification rates of above 0.80 for Fashion-MNIST dataset (both i.i.d. and non-i.i.d.) when the server is equipped with FLTrust. In comparison, the misclassification errors of all the existing baselines are below 0.30. Adaptive defenses such as Safeguard and Centered Clipping provide a higher degree of robustness due to their capability of limiting time-coupled attacks. However, they are still vulnerable to our reinforcement learning based attacks. As shown in Figure 2, the MARL attacks achieved a classification error rate of over 60% for all the three datasets, while that of all other baseline attacks are below 25% under these two defenses.

Impact of distribution learning. We expect that higher quality of distribution learning allows the attackers to learn better attack policies, which is confirmed in Figure 3(a), where the misclassification rates increase as the threshold ν of distribution learning reduces. On the other hand, even when the attackers use random data samples as the dummy data (DLR), the misclassification rates still reach

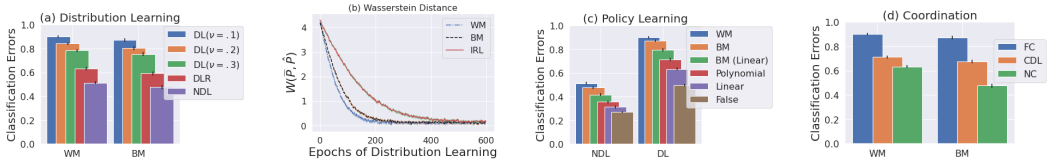

Figure 3: A comparison of average classification error rates and Wasserstein distance by varying configurations of RL-based methods for FL with FLTrust defense rule on Fashion-MNIST with *i.i.d.* data. Error bars indicate the standard deviations. In (a) and (c), DL: default distribution learning; NDL: no distribution learning; DLR: distribution learning with random dummy data. In (d), FC: coordination in all the stages; NC: no coordination; CDL: coordinated distribution learning and uncoordinated policy learning.

above 0.55 for both white-box attacks and black-box attacks. In comparison, without distribution learning, the misclassification rates of the RL-based attack methods (NDL) drop to a level below 0.5. Further, coordinated distribution learning allows the proposed MARL methods to learn the mixture distribution substantially more efficiently than RL and IRL. As shown in Figure 3(b), it takes 206 FL epochs and 224 epochs for WM and BM to learn an mixture distribution $\widetilde{P}$ that is within $\nu = 0.1$ Wasserstein distance from the empirical distribution $\widehat{P}$. Nevertheless, it takes 458 FL epochs for IRL to learn the same quality of mixture distribution.

White-box vs. Block-box attacks. The attackers' knowledge of the server's aggregation rule also largely influences attack performance (see Figure 3(c)). The attack performance increases from 0.497 when the attackers use a completely false aggregation rule (False) such as the average aggregation rule to 0.90 when the attackers knows the true aggregation rule (WM). When the attackers use the black-box attacks (BM) with policy adaptation, the classification error rate reaches 0.873, which is on par with the white-box attacks. This is partly due to the effectiveness of our proposed method for estimating server's parameters in the $i.i.d.$ setting, such as learning rate (actual $\eta = 0.001$ vs. estimate $\tilde{\eta} = 0.0012$), the subsampling rate (actual $\kappa = 20\%$ vs. estimate $\tilde{\kappa} = 19.7\%$), and the number of total workers (actual $K = 1,000$ vs. estimate $\tilde{K} = 1,083$). More importantly, the continuous model adaptation significantly improves the quality of the estimated non-linear aggregation rule and consequently the quality of policy learning. Even the black-box attack that learns a linear aggregation rule (BM(Linear)) outperforms the polynomial attack policy (Polynomial) and the linear attack policy (Linear) without model adaptation. This indicates the striking usefulness of model adaption in policy learning.

Importance of coordination. Among the three RL-based methods, the proposed MARL methods (WM and BM) perform the best due to better coordination among the attackers in distribution learning, policy learning and attack execution. To further understand the importance of multi-agent coordination in learning effective attacks, we investigate the use of coordination in different stages of our model-based RL attack method (see Figure 3 (d)). With full coordination (FC) in distribution learning, policy learning and attack execution, the MARL methods achieve a misclassification rate of above 0.80. This is in comparison to below 0.60 when no coordination (NC) involved in the RL-based methods. When the RL-based methods has coordinated distribution learning and uncoordinated policy learning (CDL), the attack performance lies in between at around 0.70. These results demonstrate the important role of multi-agent coordination in attacking federated learning systems.

6 Conclusion

We propose a multi-agent reinforcement learning attack framework to learn a non-myopic attack policy that can effectively compromise FL systems even with advanced defense mechanisms applied. Our attack framework can be extended to incorporate advances in meta-learning (Finn et al., 2017; Gupta et al., 2018) to generalize the learned policy to different training tasks. For instance, the leader attacker can first learn a distribution over different tasks and train a model-agnostic policy with the learned distribution by utilizing model-agnostic meta-learning (MAML) (Finn et al., 2017). Given a new task, the leader attacker can then train an attack policy fast by adapting from the model-agnostic policy with few shots. We envision that this approach can be especially useful when similar tasks are trained over different sets of clients. While we focus on untargeted attacks against FL systems in this paper, our attack framework can be extended to targeted attacks or backdoor attacks. Another direction is to investigate novel methods to defend our adaptive attack methods. One possible solution would be to dynamically adjust FL parameters such as the subsampling rate or the aggregation rule.

REFERENCES

Johannes Ackermann, Volker Gabler, Takayuki Osa, and Masashi Sugiyama. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465*, 2019.

Zeyuan Allen-Zhu, Faeze Ebrahimian, Jerry Li, and Dan Alistarh. Byzantine-resilient non-convex stochastic gradient descent. *arXiv preprint arXiv:2012.14368*, 2020.

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pp. 243–252. PMLR, 2017.

Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 264–273. PMLR, 2018.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948. PMLR, 2020.

Moran Baruch, Gilad Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *arXiv preprint arXiv:1902.06156*, 2019.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pp. 634–643. PMLR, 2019.

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

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. FLTrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.

YU Chao, A VELU, E VINITSKY, et al. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

Alfredo V Clemente, Humberto N Castejón, and Arjun Chandra. Efficient parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1705.04862*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th USENIX Security Symposium*, pp. 1605–1622, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 603–618, 2017.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, pp. 5311–5319. PMLR, 2021.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

Lingjuan Lyu, Han Yu, and Qiang Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy*, pp. 691–706, 2019.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927*, 2018.

Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Boris T. Polyak. *Introduction to optimization*. Optimization Software, 1987.

Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

Jian Shen, Han Zhao, Weinan Zhang, and Yong Yu. Model-based policy optimization with unsupervised model adaptation. *Advances in Neural Information Processing Systems*, 33, 2020.

Aman Sinha, Hongseok Namkoong, and John Duchi. Certifying some distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018.

Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1891–1898, 2014.

Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Node injection attacks on graphs via reinforcement learning. *arXiv preprint arXiv:1909.06543*, 2019.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pp. 216–224. Elsevier, 1990.

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

Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020a.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence*, pp. 261–270. PMLR, 2020b.

Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. *arXiv preprint arXiv:2104.07586*, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, pp. 11225–11234. PMLR, 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, pp. 14774–14784, 2019.

APPENDIX

A APPENDIX TO SECTION 1: INTRODUCTION

A.1 BROADER IMPACT

To study the vulnerabilities of federated learning, we propose a model-based multi-agent reinforcement learning attack framework. Our work shows that non-myopic attacks with coordinated attackers can break federated learning systems even when they are equipped with sophisticated defense rules such as FLTrust (Cao et al., 2020), Safeguard (Allen-Zhu et al., 2020) and Centered Clipping (Karimireddy et al., 2021). This reveals the urgent need of developing more advanced defense mechanisms for federated learning systems. While we have focused on adversarial attacks against federated learning in our work, we note that one possible solution to defending RL-based attacks would be to dynamically adjust FL parameters such as the subsampling rate or the aggregation rule. Future work is needed to identify how best to do so.

B APPENDIX TO SECTION 3: MARL ATTACK FRAMEWORK AGAINST FEDERATED LEARNING

B.1 MORE DETAILS OF MARL

In addition, our MARL framework allows the attackers to incorporate stealthiness constraints (e.g., similar to the regularization term in Eq. 3 of (Bhagoji et al., 2019)) into the attack objective as the attackers can estimate the average of the gradients of all the other workers in the previous epoch.

Impact of FL dynamics on the performance of MARL. In practice, the number of workers may vary. Since the MARL attack framework learns an approximation of the distribution of the aggregated data, our methods still apply even when new workers are added after the distribution learning stage as long as their local data distributions do not deviate significantly from the estimated distribution. Theorem 1 demonstrates that the performance loss is bounded in this case.

Another scenario to consider is that the number of data points on each worker can also change. Our theoretical results indicate that the learned attack policy remains effective because inaccurate estimation on the number of data points has limited impact on distribution learning and policy learning, as long as workers' local data distributions do not differ greatly.

B.2 ALGORITHMS

In this subsection, we present the detailed algorithms for federated learning (Algorithm 1), distribution learning (Algorithm 2), cooperative multi-agent deep deterministic policy gradients (Algorithm 3).

Algorithm 1 Federated Learning

Input: Initial weight θ^0, K workers indexed by k, size of subsampling w, local minibatch size B, step size η, number of global training steps $\mathcal{T}$
Output: $\theta^{\mathcal{T}}$
Server executes:
  for $t = 0$ to $\mathcal{T} - 1$ **do**
    $\mathcal{S}^t \leftarrow$ randomly select w workers from K workers
    for each worker $j \in \mathcal{S}^t$ **in parallel do**
      $g_j^{t+1} \leftarrow$ **WorkerUpdate**(j, θ^t)
    end for
    $g^{t+1} \leftarrow Aggr(g_{k_1}^{t+1}, ..., g_{k_w}^{t+1}), k_i \in \mathcal{S}^t$
    $\theta^{t+1} \leftarrow \theta^t - \eta g^{t+1}$
  end for
WorkerUpdate(j, θ):
  Sample a minibatch b of size B
  $g \leftarrow \frac{1}{B} \sum_{z \in b} \nabla_\theta \ell(\theta, z)$
  return g to server

Inverting gradients based distribution learning. As shown in Algorithm 2, , the leader agent uses all the M attackers' local data to serve as the dummy data D_{dummy}. For each epoch $t(\tau)$ that at least one attacker is selected, the leader agent obtains the model update from one of the attackers and calculates the batch-level gradient $\bar{g}^\tau$. It then reconstructs the data samples used to generated the gradient by iteratively solving the following optimization problem: $\arg\min_{x\in\mathbb{R}^n, y\in\mathbb{R}} 1 - \frac{\langle \nabla_\theta \ell(\theta;(x,y)), \bar{g}^\tau \rangle}{\|\nabla_\theta \ell(\theta;(x,y))\| \cdot \|\bar{g}^\tau\|} + \beta TV(x)$, where n is the dimension of x, $TV(x)$ is the total variation (Rudin et al., 1992) of x, and β is a fixed parameter. For each data point, the inverting gradients algorithm terminates after max_iter iterations. After data reconstruction in each epoch τ, all the reconstructed data will be added to the set of dummy data. The approximated mixture distribution $\widetilde{P}(\tau)$ consists of the reconstructed data up to $t(\tau)$ and the M attackers' local data. To determine whether an approximated distribution is sufficiently accurate, we measure the 1-Wasserstein distance (with the Euclidean norm as the distance metric) (Vaserstein, 1969) $W_1(\widetilde{P}(\tau-1), \widetilde{P}(\tau))$ between the approximated distributions of the two consecutive attack steps. We use the data points across all the workers in the previous step and the current step to approximate the previous and current mixture distributions. The distribution learning algorithm terminates when the Wasserstein distance is below a predefined threshold ν. After distribution learning, the leader agent shares the learned distribution $\widetilde{P}$ with all the attackers.

Algorithm 2 Distribution Learning

Input: Wasserstein distance threshold for termination ν, number of iterations for inverting gradients max_iter, step size for FL η and step size for inverting gradients η', model parameters $\{\theta^{t(\tau)}\}$
Output: $\widetilde{P}$
Wasserstein distance $W(\widetilde{P}(-1), \widetilde{P}(0)) \leftarrow \infty$, $\tau \leftarrow 0$
$D_{dummy} \leftarrow M$ attackers' local data
while $W(\widetilde{P}(\tau-1), \widetilde{P}(\tau)) > \nu$ **do**
  $\tau \rightarrow \tau + 1$
  Compute the aggregated gradients using $\bar{g}^\tau \leftarrow (\theta^{t(\tau-1)} - \theta^{t(\tau)})/(\eta(t(\tau) - t(\tau-1)))$
  for $(x, y) \in D_{dummy}$ **do**
    $(x_0, y_0) \leftarrow (x, y)$
    for $i = 0$ to $max\_iter - 1$ **do**
      $\nabla_\theta \ell(\theta^{t(\tau)}; (x_i, y_i)) \leftarrow \partial \ell(\theta^{t(\tau)}; (x_i, y_i))/\partial\theta$
      $\mathcal{L}_i \leftarrow 1 - \frac{\langle \nabla_\theta \ell(\theta^{t(\tau)}; (x_i, y_i)), \bar{g}^\tau \rangle}{||\nabla_\theta \ell(\theta^{t(\tau)}; (x_i, y_i))|| \cdot ||\bar{g}^\tau||} + \beta TV(x_i)$
      $x_{i+1} \leftarrow x_i - \eta' \nabla_{x_i} \mathcal{L}_i, y_{i+1} \leftarrow y_i - \eta' \nabla_{y_i} \mathcal{L}_i$
    end for
  end for
  Add newly all reconstructed data points (x, y) to D_{dummy}
  Compute the current approximated mixture distribution $\widetilde{P}(\tau)$ with all the reconstructed data points and the M attackers' local data
  Compute the Wasserstein distance $W(\widetilde{P}(\tau-1), \widetilde{P}(\tau))$
end while

Cooperative Multi-Agent Deep Deterministic Policy Gradient (CMADDPG). We let all the agents (attackers) share the same action-value function $Q^\mu(s, a_1, ..., a_M)$ and the same policy $\mu_\phi(\cdot|o_i)$ with parameters ϕ. Note that although the attackers share the same policy, their actions in each step vary due to the unique observations they receive. Using the chain rule, we can derive the gradient of the expected return $J(\phi) = \mathbb{E}[\sum_{\tau=0}^{H-1} r^\tau]$ as follows: $\nabla_\phi J(\phi) = \mathbb{E}_{s,a\sim\mathcal{D}}[\sum_{i=1}^{M} \nabla_\phi \mu_\phi(o_i) \nabla_{a_i} Q^\mu(s, a_1, \ldots, a_i, \ldots, a_M)|_{a_i=\mu(o_i)}]$, where the experience reply buffer $\mathcal{D}$ contains tuples $\{(s, r, s', a_1, \ldots, a_M)\}$. Note that for attacker i not selected by the server in a certain step, its action does not affect the Q^μ value, which implies that $\nabla_{a_i} Q^\mu = 0$ for any a_i when the state indicates that attacker i is not sampled. Hence, the policy gradient formula makes sense even when subsampling is applied. Similar to (Lowe et al., 2017), the shared action-value function Q^μ is updated by minimizing the loss:

$$\mathcal{L}(\phi) = \mathbb{E}_{s,a,r,s'}\{[y - Q^\mu(s, a_1, \ldots, a_M)]^2\}, \quad y = r + Q^{\mu'}(s', a'_1, \ldots, a'_M)|_{a'_k=\mu'(o_k)}. \tag{1}$$

where μ' is the target policy with delayed parameters ϕ'.

Algorithm 3 Cooperative Multi-Agent Deep Deterministic Policy Gradient (CMADDPG)

for episode $= 1$ to $max\_episode$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state s
  for $\tau = 1$ to H **do**
    for each attacker i, select action $a_i = \mu_\phi(o_i) + \mathcal{N}_\tau$ w.r.t the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_M)$ and observe reward r and new state s'
    Store (s, a, r, s') in replay buffer $\mathcal{D}$
    $s \leftarrow s'$
    Sample a random minibatch of C samples (s^j, a^j, r^j, s'^j) from $\mathcal{D}$
    Set $y^j = r^j + Q^{\mu'}(s'^j, a'_1, \ldots, a'_M)|_{a'_k = \mu'(o_k^j)}$
    Update critic by minimizing the loss:
      $\mathcal{L}(\phi) = \frac{1}{C}\sum_j [y^j - Q^\mu(s^j, a_1^j, \ldots, a_M^j)]^2$
    Update actor using the sampled policy gradient:
      $\nabla_\phi J \approx \frac{1}{C}\sum_j \sum_i \nabla_\phi \mu_\phi(o_i^j) \nabla_{a_i} Q^\mu(s^j, a_1^j, \ldots, a_i, \ldots, a_M^j)|_{a_i = \mu(o_i^j)}$
    Update target network parameters $\phi' \leftarrow \alpha'\phi + (1-\alpha')\phi'$
  end for
end for

B.3 Black-box MARL Attacks

B.3.1 Parameter Estimation

We describe parameter estimation in detail. The estimation of the learning rate η , the subsampling rate κ and the total number of workers K take place in distribution learning and the estimation of aggregation rule $Aggr$ happens in policy learning.

Estimating learning rate η. According to the FL algorithm (see Algorithm 1), we know $\theta^t = \theta^{t-1} - \eta g^t$. Thus, we have $\eta = (\theta^{t-1} - \theta^t)/g^t$. Here, both θ^{t-1} and θ^t are the FL model parameters known the attackers, while g^t is average gradients of all the selected workers, which is unavailable to the attackers. We approximate g^t by using the attackers' average gradients. That is, $g^t \approx \frac{1}{m}\sum_{k=1}^m g_k^{t+1}$. Thus, the learning rate at $t(\tau)$ can be estimated as $\tilde{\eta}^\tau = m(\theta^{t(\tau-1)} - \theta^{t(\tau)})/((t(\tau) - t(\tau-1))\sum_{k=1}^m g_k^{\tau-1})$. We further assume that $\tilde{\eta}^\tau \in [-0.1, 0.1]$ and drop any values out of this range, and estimate the learning rate $\tilde{\eta}$ as the average of valid $\tilde{\eta}^\tau$ over the first h time steps, where h is the terminating epoch of distribution learning. We note that this estimate is accurate when the mixture of the attackers' local distributions represents the mixture of all the workers' distributions.

Estimating subsampling rate κ. In practice, the subsampling rate among the malicious workers should be equal to that among all workers if the selected workers are uniformly distributed. Thus, we propose to use the empirical subsampling rate of the malicious workers as an estimate of the server's subsampling rate κ. The estimated subsampling rate is calculated by $\tilde{\kappa} = \sum_{\tau=0}^h \frac{m^\tau}{hM}$, where m^τ is the number of selected malicious workers at time τ, and M is the total number of malicious workers.

Estimating total number of workers K. We propose to estimate the total number of workers K by utilizing properties of the variance of the malicious workers' gradients. Consider any attack step τ. Let $X_k = g_k(\theta^{t(\tau)}, z_k)$ denote the average gradient of worker k obtained from a randomly selected batch where $z_k \sim P_k$ and P_k is the local distribution of selected attacker k. Let $\bar{X} = \frac{1}{w}\sum_{k=1}^w X_k$ denote the average of X_k across all workers. Let V_k and V denote the variance of X_k and $\bar{X}$, respectively. Under the assumption that local data distributions are independent across workers, we have $V = Var(\frac{1}{w}\sum_{k=1}^w X_k) = \frac{1}{w^2}\sum_{k=1}^w Var(X_k) = \frac{1}{w^2}\sum_{k=1}^w V_k$. Assuming all the workers have similar variances, we further have $V = \frac{1}{(w)^2}\sum_{k=1}^w V_k \approx \frac{1}{(w)^2} w V_k = \frac{V_k}{w}$ for any k. Therefore, the total number of workers can be estimated as $K \approx \frac{w}{\tilde{\kappa}} = \frac{V_k}{\tilde{\kappa} V}$, where $\tilde{\kappa}$ is the estimated subsampling rate.

Since V_k can be easily estimated for any malicious worker k, the problem then boils down to estimating V. Since the attackers have no access to normal workers' data, they are unable to compute the exact average gradients of all the workers when the subsampling rate is less than 1. We note that the average gradients of all the workers can be approximated by the attackers' average gradients if all the workers' data follows a similar local distribution. In this case, we

have $\bar{X} = \frac{1}{w}\sum_{k=1}^{w} g_k(\theta^{t(\tau)}, z_k) \approx \frac{1}{m^\tau}\sum_{k=1}^{m^\tau} g_k(\theta^{t(\tau)}, z_k)$. To estimate $\mu(\bar{X})$, the expectation of $\bar{X}$, we use samples of $\bar{X}$ from adjacent time steps by utilizing the fact that the values of θ^t are close for adjacent t. That is, $\mu(\bar{X}) = \mathbb{E}_{\{z_k \sim P_k\}}[\frac{1}{w}\sum_{k=1}^{w} g_k(\theta^{t(\tau)}, z_k)] \approx \bar{g}^\tau$, where $\bar{g}^\tau = (\theta^{t(\tau)} - \theta^{t(\tau+1)})/(\eta(t(\tau+1) - t(\tau)))$ is the average of $\bar{X}$ between two consecutive attack steps. We then get an estimation of V at $t(\tau)$ as $V^\tau \approx \mathbb{E}_{\{z_k \sim P_k\}}[(\frac{1}{m^\tau}\sum_{k=1}^{m^\tau} g_k(\theta^{t(\tau)}, z_k) - \bar{g}^\tau)^2]$ and define $\tilde{K}^\tau = \frac{V_k^\tau}{\bar{\kappa} V^\tau}$ where V_k^τ is an estimation of V_k at $t(\tau)$. We take the average of $\tilde{K}_\tau$ over h steps as an estimation of K where we drop the values of $\tilde{K}_\tau$ that are less than $2M$, which is a reasonable assumption. We observed that a batch of 20 data samples from each selected attacker is sufficient to estimate V and V_k in our setting.

Learning aggregation rule $Aggr$**.** To learn the aggregation rule $Aggr$, the malicious workers should learn an aggregation rule that takes inputs from all the selected workers. However, the malicous workers only have an estimation of the mixture distribution of normal workers' data, and do not obtain information of each individual worker's local data. In addition, it is usually computationally prohibitive to reconstruct the exact aggregation rule due to the high dimensionality and limited data samples available. We propose to use the estimated average gradients based on the observation that the weights for normal workers' inputs do not significantly influence the attack performance when the data is $i.i.d.$. It is convenient to assume the FL system consists of one normal worker and the malicious workers.

In particular, we propose to use a small neural network with parameters ϕ_{poly} to approximate a non-linear polynomial aggregation rule $Aggr_{poly}$. This is done by replacing the known aggregation rule $Aggr$ with the neural network ϕ_{poly} that estimates the parameters of $Aggr_{poly}(g_k(\theta^\tau), \bar{g}(\theta^\tau))_{k\in[m^\tau]}$, where $g_k(\theta^\tau) = \mathbb{E}_{z_k \sim P_k}[g_k(\theta^\tau, z_k)]$ denotes the average gradients of the selected malicious worker $k \in [m^\tau]$, and $\bar{g}$ refers to the estimated average gradients of all the selected workers. The attackers estimate the average gradients of all the selected workers $\bar{g}(\theta^\tau) = \frac{1}{w}\sum_{i=1}^{w} \mathbb{E}_{z_k \sim \widetilde{P}}[g_i(\theta^\tau, z_k)]$, where $\widetilde{P}$ is the approximated distribution of aggregated data in distribution learning. At time step $t(\tau)$, the aggregated gradients $\bar{g}^\tau = (\theta^{\tau-1} - \theta^{t(\tau)})/(\eta(t^\tau - t^{\tau-1}))$. Given the pooled model parameters $\{\theta^{\tau'}\}_{\tau' \leqslant \tau}$, we have the pooled aggregated gradients $\{\bar{g}^{\tau'}\}_{\tau' \leqslant \tau}$. The parameters ϕ_{poly} can be then estimated by minimizing the loss $||\bar{g}^\tau - \phi_{poly}(g_k(\theta^\tau), \bar{g}(\theta^\tau))_{k\in[m^\tau]}||_2^2$.

B.3.2 Black-box MARL attacks with Model Adaptation

After approximating the aggregation rule, the leader attacker first learns an initial attack policy using the CMADDPG algorithm with a small number of FL training epochs (i.e., $n_s << \mathcal{T}$). We denote the learned initial attack policy by π_{poly}.

In black-box attacks, it might be insufficient to only use the initial policy because the inaccurate estimates of the server's parameters, especially inaccurate aggregation rule $Aggr$ can lead to inaccurate simulations of the server's behavior. As a result, the learned Markov game model is also inaccurate, yielding a suboptimal attack policy. To improve the quality of the trained attack policy, the leader attacker must continuously adapt the Markov game model as more data become available.

Following the original AMPO algorithm (Shen et al., 2020), the black-box attack algorithm adopts an alternative optimization between the model training and model adaptation (see Algorithm 4). The dynamics model $\hat{T}_\phi$ is defined by $\hat{T}_\phi(s^{\tau+1}|s^\tau, a^\tau) = \mathcal{N}(\mu_\phi(s^\tau, a^\tau), \sum_\phi(s_\tau, a_\tau))$, where μ_ϕ and $\sum_\phi$ are the mean and covariance matrix of the Gaussian distribution. We collect the trajectories $(s^{\tau'}, a^{\tau'}, s^{\tau'+1}, r^{\tau'})$ since the beginning of policy learning τ_{pl}. We initialize the dynamics model $\hat{T}_\phi$ with bootstrapped samples selected from the environment buffer $\mathcal{D}_e = \{(s^{\tau'}, a^{\tau'}, s^{\tau'+1}, r^{\tau'})\}_{\tau_{pl} \leqslant \tau' \leqslant \tau}$ that stores the real data collected from step τ_{pl} to current step τ. Unlike random policy as the initial policy in the original AMPO, our proposed method adopts the pre-trained policy π_{poly} as the initial policy for better performance. Note that we need to approximate the transition function because AMPO explicitly adapts the transition function to improve the accuracy of the trained Markov game model.

Markov game model training. The objective of model training is to minimize the negative log-likelihood loss: $\mathcal{L}_{\hat{T}}(\phi) = \sum_{\tau=1}^{N}[\mu_\phi(s^\tau, a^\tau) - s^{\tau+1}]^\top \sum_\phi^{-1}(s^\tau, a^\tau)[\mu_\phi(s^\tau, a^\tau) - s^{\tau+1}] + \log\det\sum_\phi(s^\tau, a^\tau)$, where $\det$ refers to the determinant. This model is used to generate $q-$length simulated rollouts branched from the states sampled from the environment buffer $\mathcal{D}_e$. Each time

the next state is predicted using the current policy, the simulated data is then added to the model buffer $\mathcal{D}_m$. Then we train the policy using the CMADDPG algorithm on both real and simulated data from two buffers with a certain ratio rs. The CMADDPG algorithm trains a deterministic policy by minimizing the loss $\mathcal{L}(\phi)$ defined by Eq. 1.

Markov game model adaptation. We use the first several layers as the feature extractor f_g to represent the aggregation rule with corresponding parameters ϕ_g, and the remaining layers as the decoder f_d with parameters ϕ_d. Thus, we have $\hat{T} = f_d \circ f_g$ and $\phi = \{\phi_g, \phi_d\}$. Since we explicitly designate the feature extractor to approximate the server's aggregation rule, the feature extractor consists of six parameters as the neural network ϕ_{poly} does. We minimize the 1-Wasserstein distance between two feature distributions $\mathbb{P}_{h_e}$ and $\mathbb{P}_{p_e}$, where $h_e = f_g^e(s_e, a_e)$ and $h_g = f_g^m(s_m, a_m)$. The 1-Wasserstein distance can be estimated by using a critic network f_c with parameter ω to maximize the loss $\mathcal{L}_{WD}(\phi_g^e, \phi_g^m, \omega) = \frac{1}{N_e}\sum_{i=1}^{N_e} f_c(h_e^i) - \frac{1}{N_m}\sum_{j=1}^{N_m} f_c(h_m^j)$. The algorithm optimizes the feature extractor to minimize the estimated 1-Wasserstein distance in order to learn the aggregation rule that is invariant to the real data and simulated data after approximating the 1-Wasserstein distance. That is, the model adaptation procedure is achieved by solving the following minimax objective: $\min_{\phi_g^e, \phi_g^m} \max_{\omega} \mathcal{L}_{WD}(\phi_g^e, \phi_g^m, \omega) - \alpha \cdot \mathcal{L}_{gp}(\omega)$, where α is the balancing coefficient and $\mathcal{L}_{gp}(\omega) = \mathbb{E}_{\mathbb{P}_{\tilde{h}}}[(\|\nabla f_c(\tilde{h})\|_2 - 1)^2]$ is the gradient penalty loss for the critic to enforce 1-Lipschitz and $\mathbb{P}_{\tilde{h}}$ is the distribution of uniformly distributed linear interpolations of $\mathbb{P}_{h_e}$ and $\mathbb{P}_{h_m}$. The algorithm alternates between training the critic to estimate the 1-Wasserstein distance and training the feature extractor of the dynamics model to learn transferable features.

In black-box attacks, individual malicious workers send observations (i.e., the rank information) to the leader attacker. The leader attacker trains the attack policy π_ϕ and then sends the actions to individual malicious workers to execute. Unlike separated policy learning and attack execution in white-box attacks, policy learning and attack execution are integrated in black-box attacks.

Algorithm 4 Black-box MARL Attacks

Estimate learning rate η
Learn the mixture distribution of the aggregated data $\widetilde{P}$ using Algorithm 2
Estimate subsampling rate κ, the total number of workers K, the aggregation rule $Aggr_{poly}$
Train an attack policy π_{poly} with $Aggr_{poly}$ for n_s FL training epochs using Algorithm 3
Initialize policy $\pi_\phi \leftarrow \pi_{poly}$, dynamics model $\hat{T}_\phi$, environment buffer $\mathcal{D}_e$, model buffer $\mathcal{D}_m$
for step $= 1$ to max_step **do**
  Each attacker takes an action in the environment using the policy π_ϕ; add the sample (s, a, s', r) to $\mathcal{D}_e$, where $a = (a_1, ..., a_M)$
  if every E real timesteps are finished **then**
    Perform G_1 gradient steps to train the model $\hat{T}_\phi$ with samples from $\mathcal{D}_e$
    for F model rollouts **do**
      Sample a state s uniformly from $\mathcal{D}_e$
      Use policy π_ϕ to perform a $q-$step model rollout starting from s; add to $\mathcal{D}_m$
    end for
    Perform G_2 gradient steps to train the feature extractor (to approximate the aggregation rule) with samples (s, a) from both $\mathcal{D}_e$ and $\mathcal{D}_m$ by the model adaption loss $\mathcal{L}_{WD}$
  end if
  Perform G_3 gradient steps to train the policy π_ϕ with samples (s, a, s', r) from $\mathcal{D}_e \cup \mathcal{D}_m$
end for

C Appendix to Section 5: Experiments

C.1 Experimental Setups

Datasets. We consider three real world datasets: Fashion-MNIST (Xiao et al., 2017) and CIFAR-10 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009). Fashion-MNIST includes 60,000 training examples and 10,000 testing examples, where each example is a 28×28 grayscale image, associated with a label from 10 classes. CIFAR-10 consists of 60,000 color images in 10 classes of which there are $50,000$ training examples and 10,000 testing examples. ImageNet has 1,000 object classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images. For ImageNet, we select a random subset of $60,000$ from the training images as the training dataset,

and $10,000$ from the testing images as the testing dataset. We randomly split each of the datasets into $1,000$ groups, each of which consists of the same number of training samples and the same number of testing samples. For the *i.i.d.* setting, we randomly split the dataset into $1,000$ groups, each of which consists of the same number of training samples and the same number of testing samples. For the *non-i.i.d.* setting, we follow the method of (Fang et al., 2020) to quantify the heterogeneity of the data. We split the workers into $1,000$ groups and model the *non-i.i.d.* federated learning by assigning a training instance with label c to the c-th group with probability pr and to all the groups with probability $1 - pr$. A higher pr indicates a higher level of heterogeneity. For for *non-i.i.d.* data, we set the degree of *non-i.i.d.* $pr = 0.5$ as the default setup.

Federated learning settings. We adopt the following parameters for the federated learning models: learning rate $\eta = 0.001$, number of total workers = 1,000, number of attackers = 50 (0 for NA and 1 for RL), subsampling rate = 20%, and number of total epochs = 1,000. For Fashion-MNIST, we train a neural network classifier consisting of 8×8, 6×6, and 5×5 convolutional filter layers with ELU activations followed by a fully connected layer and softmax output. This neural network is used in the tutorial for CleverHans (Papernot et al., 2018) and in (Sinha et al., 2018). For CIFAR-10, we use a 7-layer CNN with the following layers: input layer of size $32 \times 32 \times 3$; convolutional layer with ReLU of size $3 \times 3 \times 30$; max pooling layer of size 2×2; convolutional layer with ReLU of size $3 \times 3 \times 50$; max pooling layer of size 2×2; a fully connected layer with ReLU of size 200, and an output layer of size 10. We use softmax on the output. For ImageNet, we use the ResNet-18 (He et al., 2016). We measure the top-1 error rates for Fashion-MNIST and CIFAR-10, and the top-5 error rates for ImageNet. We set the local batch size $B = 8$. We implement the FL model with PyTorch (Paszke et al., 2019) and run all the experiments on the same 2.30GHz Linux machine with 16GB NVIDIA Tesla P100 GPU. We run all the experiments for 20 times and report the mean. Since the standard deviations are below 0.04, we omit them for better visualization.

Distribution learning settings. In distribution learning, we set the step size for inverting gradients $\eta' = 0.001$, the total variation parameter $\beta = 0.01$, the threshold for distribution learning $\nu = 0.1$ and the number of iterations for inverting gradients $max\_iter = 6,000$, and learn the mixture data distribution by using the attackers' data as dummy data.

Policy learning settings. In policy learning, we adopt a PyTorch implementation of the CMADDPG based on the original MADDPG (Lowe et al., 2017). The default parameters are described as the following: number of policy training epochs = 300, number of policy training episodes $max\_episode$ = $6,000$ and $\alpha' = 0.01$ for updating the targeting networks. We train the $6,000$ episodes in our experiments. Note that the length of each simulating epoch is typically shorter than the length of each real FL training epoch. In our experiments, we assume that the server waits for 15 seconds to receive the updates from the workers before performing an model aggregation. In addition, the total number of FL training epochs is fixed. We fix the number of simulating epochs in each episode in policy learning. Since the number of epochs for distribution learning varies across datasets, the number of policy learning epochs H also varies.

Black-box MARL settings. The hyperparameter settings for black-box MARL attackers are described as follows. The coefficient for gradient penalty $\alpha = 10$. The number of total steps $max\_step = 10,000$. The real steps between model training $E = 250$. We set the model adaption batch size as 256. The model rollout batch size is $10,000$. We set the steps for model training $G_1 = 100$ The model adaption updates $G_2 = 40$, the rollout length $q = 1$, the policy updates per real step $G_3 = 20$. For training the initial policy π_{poly}, we use a small neural network that consists of 6×6, and 5×5 convolutional filter layers, followed by a fully-connected layer and a softmax layer. We use multilayer perceptron (MLP) with four hidden layers (as the feature extractor) and one output layer (as the decoder) of size 200 to approximate the aggregation rule.

C.2 More Experimental Results

Subsampling rate. As the subsampling rate increases, the performance of the RL-based attacks also increases significantly (see Figure 1 (left)). For instance, the black-box MARL attack increases from 0.512 to 0.921 when the subsampling rate changes from 5% to 30% with FLTrust on Fashion-MNIST and $i.i.d.$ data. This is due to the fact that the attackers have a higher probability of being selected by the server, leaving more time for the attackers to learn the attack policy and execute the attacks. Noticeably, even if the subsampling rate is as low as 5%, both MARL attack methods (WM and BM) substantially outperform all the other baselines with a subsampling rate as high as 30%.

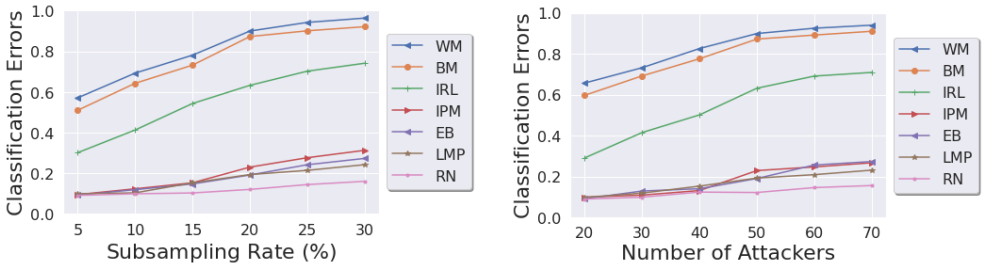

Figure 1: A comparison of classification errors on Fashion-MNIST with $i.i.d.$ data with FLTrust defense rule by varying the level of subsampling (**left**) and the number of attackers (**right**). Key parameters: number of workers = $1,000$, number of attackers = 50 (**left**), subsampling rate = 20% (**right**). All other parameters are set as default.

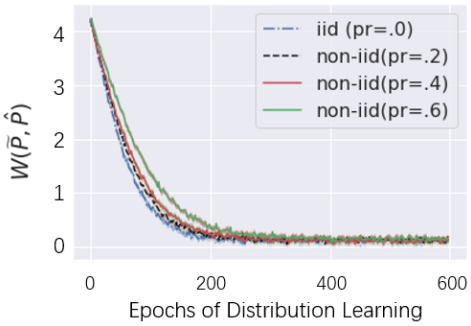

Figure 2: A comparison of Wasserstein distance between the true mixture distribution ($\widehat{P}$) and the learned distribution ($\widetilde{P}$) on Fashion-MNIST when the server applies $non-i.i.d.$ data for FL with FLTrust defense rule. Key parameters: number of workers = $1,000$, number of attackers = 50, subsampling rate = 20%. All other parameters are set as default.

Number of attackers. As the number of attackers increases, the performance of all the attack methods also increases (see Figure 1 (right)). However, the degree of increment differs. Specifically, the three RL-based methods obtain significant jumps when the number of attackers increase from 20 to 70. For instance, the white-box MARL method increases from 0.658 to 0.941. Nevertheless, the existing baselines only have a slim increment. Even with 20 attackers, the proposed MARL methods (WM and BM) significantly outperform existing baselines with a number of attackers as much as 70.

Computation time. The training time mainly comes from learning an accurate estimation of the mixture distribution of the aggregated data $\widehat{P}$, and learning an attack policy π. For a typical subsampling rate (e.g., 20%) and distribution learning threshold (e.g., $\nu = 0.1$), the distribution learning stage usually takes only a small portion (e.g., less than 25%) of the total FL epochs before convergence. The distribution learning time also depends on the degree of data heterogeneity. A higher degree of heterogeneity requires a longer time for distribution learning (see Figure 2). The time for policy learning in stage two largely depends on the number of training epochs used to simulate the FL dynamics in each training episode, and the number of training episodes used by the attackers. The larger of the two factors, the more training time is required. In practice, the server usually needs to wait for some time (typically a few minutes) in order to receive the gradients from the clients before conducting model aggregation. In addition, if the leader agent has access to GPUs or other parallel computing facilities, it can run multiple training episodes in parallel (Clemente et al., 2017). When the policy is trained with a large number of episodes, there is no need to simulate the complete FL process for $\mathcal{T}$ epochs. It suffices to consider much shorter epochs in practice.

Time allocation (in terms of FL epochs) for each stage. In our experiments, we set the total number of FL epochs as $1,000$. Different RL-based algorithms may require different numbers of epochs for distribution learning depending on the quality threshold ν (See Table 1). The numbers of FL epochs left for policy learning and attack execution thus also vary. As the threshold ν decreases,

it takes more FL epochs for distribution learning for all the RL-based methods. However, the MARL methods (WM and BM) require substantially fewer FL epochs than IRL does. As a result, it leaves more FL epochs for policy learning and attack execution.

For black-box MARL (i.e., BM), both the policy learning and attack execution happen in the same phase. For instance, the numbers of FL epochs in both policy learning and attack execution for BM ($\nu = 0.1$) are 776. Among the 776 epochs, it takes 56 FL epochs for training the initial policy π_{poly}. The remaining 720 FL epochs are used for training the black-box MARL attack policy. For all other RL-based methods, the number of the policy learning is 300. Table 1 shows that both MARL methods (WM and BM) significantly reduce the time for distribution learning compared to IRL due to better coordination in distribution learning.

Table 1: A comparison of training time (in number of epochs) in each stage for RL-based attacks against FL with FLTrust defense rule on Fashion-MNIST with $i.i.d.$ data. Key parameters: number of workers = 1,000, number of attackers = 50, subsampling level = 20%. All other parameters are set as default.

Attack Method	Distribution learning	Policy learning	Execution	Classification error
WM ($\nu = 0.1$)	206	300	496	0.901
WM ($\nu = 0.2$)	172	300	528	0.841
WM ($\nu = 0.3$)	151	300	549	0.783
BM ($\nu = 0.1$)	224	776	776	0.873
BM ($\nu = 0.2$)	185	815	815	0.802
BM ($\nu = 0.3$)	170	830	830	0.754
IRL($\nu = 0.1$)	458	300	242	0.632
IRL ($\nu = 0.2$)	424	300	276	0.531
IRL ($\nu = 0.3$)	376	300	324	0.376

Choices of MARL algorithms. We choose the MADDPG algorithm for two reasons: first, it is easy to implement; second, as an off-policy algorithm, MADDPG is typically more sample efficient than on-policy learning algorithms. Our results show that when we replace the MADDPG with the multi-agent version PPO (Chao et al., 2021), the attack performance of our MARL method changes only slightly. For instance, when the server is equipped with FLTrust and when we run the experiments on CIFAR-10 with i.i.d. data, the average classification errors for the PPO-based MARL method only improves to 0.841 from MADDPG's 0.823, which is less than 2.2%.

Patterns of MARL attacks. There are two main characteristics of the proposed MARL attacks: first, the attack magnitude (i.e., the noises added to the original gradients) and the angle (i.e., cosine similarity) between the crafted updates and the normal model updates for differently ranked malicious workers are usually different; second, the variance of the attack magnitude and the variance of the angle are also different depending on the aggregation rule used by the server. One observation is that more sophisticated defense mechanisms typically require larger variation in the magnitude and the angle. For instance, the variances of attack magnitude and cosine similarity in FLTrust, Safeguard and Centered Clipping are significantly larger than that in other defense rule settings such as averaging, coordinate-wise median, and Krum. See Table 2 for a comparison.

Table 2: A comparison of variances in the attack magnitude and the cosine similarity between the crafted updates and the normal model updates for white-box MARL (WM) attacks against FL with different defense rules on CIFAR-10 with $i.i.d.$ data. Key parameters: number of workers = 1,000, number of attackers = 50, subsampling level = 20%. All other parameters are set as default.

Measurement / Defense	Averaging	Median	Krum	FLTrust	Safeguard	Centered Clipping
Variance of magnitude	0.081	0.132	0.144	0.247	0.321	0.317
Variance of cosine similarity	0.042	0.093	0.127	0.356	0.393	0.364

D PROOF OF THEOREM 1

D.1 PRELIMINARIES

Our theoretic analysis relies on the following definitions and results. First, we formally define the Wasserstein distance (Vaserstein, 1969), which will be used to measure the distance between the estimated and true data distributions as well as the distance between the corresponding transition dynamics introduced by different data distributions.

Definition 1. *(Wasserstein distance) Let $(\mathbf{M}, d)$ be a metric space and $\mathcal{P}_p(\mathbf{M})$ be the set of all probability measures on $\mathbf{M}$ with p^{th} moment, then the p^{th} Wasserstein distance between two probability distributions μ_1 and μ_2 in $\mathcal{P}_p(\mathbf{M})$ is defined as:*

$$W_p(\mu_1, \mu_2) := \left(\inf_{j \in \mathcal{J}} \int \int d(s_1, s_2)^p j(s_1, s_2) ds_1 ds_2 \right)^{1/p}$$

where $\mathcal{J}$ is the collection of all joint distributions j on $\mathbf{M} \times \mathbf{M}$ with marginals μ_1 and μ_2.

In the following, we focus on 1-Wasserstein distance and denote $W(\mu_1, \mu_2) := W_1(\mu_1, \mu_2)$. Wasserstein distance is also known as "Earth Mover's distance" that measures the minimum expected distance between two pairs of points where the joint distribution is constrained to match their corresponding marginals . Compared with Kullback-Leibler (KL) divergence and Total Variation (TV) distance, Wasserstein distance is more sensitive to how far the points are from each other (Asadi et al., 2018).

We will also need the following special form of Lipschitz continuity from (Asadi et al., 2018).

Definition 2. *(Lipschitz Continuity) Given two metric spaces $(\mathbf{M}_1, d_1)$ and $(\mathbf{M}_2, d_2)$, a function $f : \mathbf{M}_1 \to \mathbf{M}_2$ is Lipschiz continuous if the Lipschiz constant, defined as*

$$K_{d_1,d_2}(f) := \sup_{s_1 \in \mathbf{M}_1, s_2 \in \mathbf{M}_2} \frac{d_2(f(s_1), f(s_2))}{d_1(s_1, s_2)}$$

is finite. Similarly, a function $f : \mathbf{M}_1 \times A \to \mathbf{M}_2$ is uniformly Lipschitz continuous in A if:

$$K^A_{d_1,d_2}(f) := \sup_{a \in A} \sup_{s_1, s_2} \frac{d_2(f(s_1, a), f(s_2, a))}{d_1(s_1, s_2)}$$

is finite.

Let $\mathcal{M} = (S, A, T, r)$ be a generic Markov game model, where S and A denote the state space and the action space respectively, $T(s'|s, a)$ denotes the probability of reaching a state s' from the current state s and action a, and $r(s, a, s')$ denotes the reward given the current state s, action a, and the next state s'. We then introduce the concept of Lipschiz model class from (Asadi et al., 2018), which allows us to represent the stochastic transition dynamics of an Markov game as a distribution over a set of deterministic transitions.

Definition 3. *(Lipschitz model class) Given a metric state space (S, d_S) and an action space A, we define F_g as a collection of functions: $F_g = \{f : S \to S\}$ distributed according to $g(f|a)$ where $a \in A$. We say that F_g is a Lipschitz model class if*

$$K_F := \sup_{f \in F_g} K_{d_S,d_S}(f),$$

is finite. We say that a transition function T is induced by a Lipschitz model class F_g if $T(s'|s, a) = \sum_f \mathbb{1}(f(s) = s')g(f|a)$ for any $s, s' \in S$ and $a \in A$.

We will show later that the transition dynamics of our Markov game model for attackers is induced by a Lipschitz model class.

Finally we give a formal definition of finite-horizon value functions (Sutton & Barto, 2018).

Definition 4. *Given an Markov game model $\mathcal{M}$ and a stationary policy π, the value function of π at time l is defined as $V^\pi_{\mathcal{M},l}(s) := \mathbb{E}_{\pi,T}[\sum_{t=l}^{H-1} r(s^t, a^t)|s^l = s]$, where $r(s, a) = \mathbb{E}_{s' \sim T(\cdot|s,a)}[r(s, a, s')]$. $V^\pi_{\mathcal{M},l}(\cdot)$ satisfies the following backward recursion form:*

$$V^\pi_{\mathcal{M},l}(s) = \mathbb{E}_{a \sim \pi(s)}[r(s, a) + \sum_{s' \in S} T(s'|s, a) V^\pi_{\mathcal{M},l+1}(s')]$$

where $V^\pi_{\mathcal{M},H-1}(s) = \mathbb{E}_{a \sim \pi(s)}[r(s, a)]$. The optimal value function is defined as $V^*_{\mathcal{M},l}(s) := \max_\pi V^\pi_{\mathcal{M},l}(s)$ for any s.

To analyze the impact of inaccurate transition on the value function, we also make use of the following lemmas (Asadi et al., 2018).

Lemma 1. *Given two distributions over states μ_1 and μ_2, a transition function T induced by a Lipschitz model class F_g is uniformly Lipschitz continuous in action space A with a constant:*

$$K^A_{W,W}(T) := \sup_{a\in A}\sup_{\mu_1,\mu_2}\frac{W(T(.|\mu_1,a),T(.|\mu_2,a))}{W(\mu_1,\mu_2)} \leqslant K_F$$

Lemma 2. *Given a Lipschiz function $f : S \to \mathbb{R}$ with constant $K_{d_S,d_{\mathbb{R}}}(f)$:*

$$K^A_{d_S,d_{\mathbb{R}}}(\int f(s')T(s'|s,a)ds') \leqslant K_{d_S,d_{\mathbb{R}}}(f)K^A_{d_S,W}(T)$$

D.2 Measuring the Uncertainty: From Data Distributions to Total Returns

Let $\mathcal{M} = (S, A, T, r, H)$ denote the true Markov game for attacking the federated learning system, and $\widetilde{\mathcal{M}} = (S, A, T', r', H)$ the estimated Markov game used in the policy learning stage, where T' and r' are derived from the estimated joint data distribution $\{\widetilde{P}_k\}$ where $\widetilde{P}_k = \widehat{P}_k$ when k is an attacker and $\widetilde{P}_k = \widetilde{P}$ otherwise. Our main goal is to compare the optimal attack performance that can be obtained from the true Markov game model $\mathcal{M}$ and that derived from the simulated Markov game model $\widetilde{\mathcal{M}}$. We will focus on understanding the impact of inaccurate data distributions (obtained from distribution learning) and assume that other system parameters are known to the attackers.

Without loss of generality, we assume the M attackers' indexes are from $K - M + 1$ to K. Let $\epsilon = \frac{K-M}{M}$ denote the fraction of benign nodes. We consider the idealized setting where the M attackers are perfectly coordinated by a single leading attacker. Because of these simplifications, the state s^t in each epoch t is completely defined by the current model parameters θ^t. In the following, we abuse the notation a bit and assume $S = \Theta$.

Let $\mathcal{J}_{\mathcal{M}}(\pi) := \mathbb{E}_{\pi,T,\mu_0}[\sum_{t=0}^{H-1} r(s^t, a^t, s^{t+1})]$ denote the expected return over H attack steps under the Markov game $\mathcal{M}$, policy π and initial state distribution μ_0. Let π^* be an optimal policy of $\mathcal{M}$ that maximizes $\mathcal{J}_{\mathcal{M}}(\pi)$. Define $\mathcal{J}_{\widetilde{\mathcal{M}}}(\pi)$ similarly and let $\widetilde{\pi}$ be an optimal policy for $\widetilde{\mathcal{M}}$, with the same initial state distribution μ_0.

Our analysis is built upon the following lemma that compares the performance of π^* and that of $\widetilde{\pi}$ with respect to the true Markov game $\mathcal{M}$. It extends a similar result in (Yu et al., 2020) to a finite-horizon Markov game where the reward in each step depends on not only the current state and action but also the next state. Note that the lemma relies on the key assumption that both $V^*_{\mathcal{M},l}(\cdot)$ and $V^*_{\widetilde{\mathcal{M}},l}(\cdot)$ are L_v-Lipschitz continuous (with respect to the l_2 norm of states) for all l. That is, $|V^*_{\mathcal{M},l}(s_1) - V^*_{\mathcal{M},l}(s_2)| \leqslant L_v\|s_1 - s_2\|_2$ for any $s_1, s_2 \in S$ where L_v is a constant independent of l. A similar requirement holds for $V^*_{\widetilde{\mathcal{M}},l}(\cdot)$. Let $W(T, T') := \sup_{a\in A}\sup_{s\in S} W(T(\cdot|s,a), T'(\cdot|s,a))$.

Lemma 3. *Assume Assumptions 2.1 holds and both $V^*_{\mathcal{M},l}(\cdot)$ and $V^*_{\widetilde{\mathcal{M}},l}(\cdot)$ are L_v-Lipschitz continuous for all l. Then,*

$$|\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\mathcal{M}}(\widetilde{\pi})| \leqslant 2H[(L + L_v)W(T, T') + 2L\epsilon\delta]$$

Proof. Let F_l be the expected return when π^* is applied to $\widetilde{\mathcal{M}}$ for the first l steps, then changing to $\mathcal{M}$ for l to $H - 1$. That is,

$$F_l = \mathop{\mathbb{E}}_{\substack{a^t\sim\pi^*(s^t)\\ t<l: s^{t+1}\sim T'(s^t,a^t), r^t=r'\\ t\geqslant l: s^{t+1}\sim T(s^t,a^t), r^t=r}}[\sum_{t=0}^{H-1} r^t(s^t, a^t, s^{t+1})]$$

By the definition of F_l, we have $\mathcal{J}_M(\pi^*) = F_0$ and $\mathcal{J}_{\widetilde{\mathcal{M}}} = F_H$, which implies that $\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\widetilde{\mathcal{M}}}(\pi^*) = \sum_{l=0}^{H-1}(F_l - F_{l+1})$. Note that

$$F_l = R_{l-1} + \mathbb{E}_{s^{l+1}\sim T(s^l,a^l)}[r(s^l, a^l, s^{l+1})] + \mathbb{E}_{s^l,a^l\sim T',\pi^*}[\mathbb{E}_{s^{l+1}\sim T(s^l,a^l)}[V^*_{\mathcal{M},l+1}(s^{l+1})]]$$
$$F_{l+1} = R_{l-1} + \mathbb{E}_{s^{l+1}\sim T'(s^l,a^l)}[r'(s^l, a^l, s^{l+1})] + \mathbb{E}_{s^l,a^l\sim T',\pi^*}[\mathbb{E}_{s^{l+1}\sim T'(s^l,a^l)}[V^*_{\mathcal{M},l+1}(s^{l+1})]]$$

where R_{l-1} is the expected return of the first $l-1$ steps, which are taken with respect to $\mathcal{M}$. Thus,

$$\begin{aligned}F_l - F_{l+1} &= \mathbb{E}_{s^{l+1}\sim T(s^l,a^l)}[r(s^l,a^l,s^{l+1})] - \mathbb{E}_{s^{l+1}\sim T'(s^l,a^l)}[r'(s^l,a^l,s^{l+1})] \\ &\quad + \mathbb{E}_{s^l,a^l\sim T',\pi*}[\mathbb{E}_{s^{l+1}\sim T(s^l,a^l)}[V^*_{\mathcal{M},l+1}(s^{l+1})] - \mathbb{E}_{s^{l+1}\sim T'(s^l,a^l)}[V^*_{\mathcal{M},l+1}(s^{l+1})]]\end{aligned}$$

Define $G^*_{\widetilde{\mathcal{M}},l}(s^l,a^l) := \mathbb{E}_{s^{l+1}\sim T(s^l,a^l)}[V^*_{\mathcal{M},l}(s^{l+1})] - \mathbb{E}_{s^{l+1}\sim T'(s^l,a^l)}[V^*_{\mathcal{M},l}(s^{l+1})]$. We have

$$\begin{aligned}\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\widetilde{\mathcal{M}}}(\pi^*) &= \sum_{l=0}^{H-1}(F_l - F_{l+1}) \\ &= \sum_{l=0}^{H-1}(\mathbb{E}_{s^{l+1}\sim T(s^l,a^l)}[r(s^l,a^l,s^{l+1})] - \mathbb{E}_{s^{l+1}\sim T'(s^l,a^l)}[r'(s^l,a^l,s^{l+1})]) \\ &\quad + \sum_{l=0}^{H-2}\mathbb{E}_{s^l,a^l\sim T',\pi*}[G^*_{\widetilde{\mathcal{M}},l}(s^l,a^l)] \\ &= \sum_{l=0}^{H-1}(\mathbb{E}_{s^{l+1}\sim T(s^l,a^l)}[\frac{1}{K}\sum_{k=1}^{K}(\ell_k(s^{l+1}) - \ell_k(s^l))] - \mathbb{E}_{s^{l+1}\sim T'(s^l,a^l)}[\frac{1}{K}\sum_{k=1}^{K}\ell'_k(s^{l+1}) - \ell'_k(s^l))]) \\ &\quad + \sum_{l=0}^{H-2}\mathbb{E}_{s^l,a^l\sim T',\pi*}[G^*_{\widetilde{\mathcal{M}},l}(s^l,a^l)] \\ &= \sum_{l=0}^{H-1}(\mathbb{E}_{s^{l+1}\sim T(s^l,a^l)}[\frac{1}{K}\sum_{k=1}^{K}\ell_k(s^{l+1})] - \mathbb{E}_{s^{l+1}\sim T'(s^l,a^l)}[\frac{1}{K}\sum_{k=1}^{K}\ell'_k(s^{l+1})]) \\ &\quad + \sum_{l=0}^{H-1}(\frac{1}{K}\sum_{k=1}^{K}\ell'_k(s^l) - \frac{1}{K}\sum_{k=1}^{K}\ell_k(s^l)) \\ &\quad + \sum_{l=0}^{H-2}\mathbb{E}_{s^l,a^l\sim T',\pi*}[G^*_{\widetilde{\mathcal{M}},l}(s^l,a^l)]\end{aligned}$$

where $\ell_k(s) := \mathbb{E}_{z_k\sim\widehat{P}_k}[\ell(s,z_k)]$, $\ell'_k(s) := \mathbb{E}_{z_k\sim\widetilde{P}_k}[\ell(s,z_k)]$ and the last equality follows from the definition of reward function $r(s,a,s') = \frac{1}{K}\sum_{k=1}^{K}\ell_k(s') - \frac{1}{K}\sum_{k=1}^{K}\ell_k(s)$, and $r'(s,a,s') = \frac{1}{K}\sum_{k=1}^{K}\ell'_k(s') - \frac{1}{K}\sum_{k=1}^{K}\ell'_k(s)$.

Since $V^*_{\mathcal{M},l}$ is L_v-Lipschitz, we have $|G^*_{\widetilde{\mathcal{M}},l}(s,a)| \leqslant L_v W(T(s,a),T'(s,a))$ from the definition of 1-Wasserstein distance. We further have

$$\begin{aligned}|\frac{1}{K}\sum_{k=1}^{K}\ell'_k(s^l) - \frac{1}{K}\sum_{k=1}^{K}\ell_k(s^l)| &\leqslant \frac{1}{K}\sum_{k=1}^{K}|\ell'_k(s^l) - \ell_k(s^l)| \\ &\leqslant \frac{1}{K}\sum_{k=1}^{K}LW(\widetilde{P}_k,\widehat{P}_k) \\ &\leqslant L\epsilon\delta,\end{aligned}$$

where the second inequality follows from the definition of 1-Wasserstein distance and Assumption 3.1, and the last inequality follows from Assumption 1 and the fact that $\widetilde{P}_k = \widehat{P}_k$ for any attacker k. Similarly, We have

$$\begin{aligned}&|\mathbb{E}_{s'\sim T(s,a)}[\frac{1}{K}\sum_{k=1}^{K}\ell_k(s')] - \mathbb{E}_{s'\sim T'(s,a)}[\frac{1}{K}\sum_{k=1}^{K}\ell'_k(s')]| \\ \leqslant&\frac{1}{K}\sum_{k=1}^{K}|\mathbb{E}_{s'\sim T(s,a)}[\ell_k(s')] - \mathbb{E}_{s'\sim T'(s,a)}[\ell'_k(s')]| \\ =&\frac{1}{K}\sum_{k=1}^{K}|\mathbb{E}_{s'\sim T(s,a),z_k\sim\widehat{P}_k}[\ell_k(s',z_k)] - \mathbb{E}_{s'\sim T'(s,a),z_k\sim\widetilde{P}_k}[\ell'_k(s',z_k)]| \\ \leqslant&L(W(T,T') + \epsilon\delta),\end{aligned}$$

where the last inequality follows Assumption 1, Assumption 3.1, and the property of 1-Wasserstein distance with respect to product measures. Thus,

$$\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\widetilde{\mathcal{M}}}(\pi^*) \leqslant H(L_v + L)W(T, T') + 2HL\epsilon\delta.$$

A similar argument shows that

$$\mathcal{J}_{\widetilde{\mathcal{M}}}(\widetilde{\pi}) - \mathcal{J}_{\mathcal{M}}(\widetilde{\pi}) \leqslant H(L_v + L)W(T, T') + 2HL\epsilon\delta.$$

Let $U = H(L_v + L)W(T, T') + 2HL\epsilon\delta$. Thus,

$$\begin{aligned}\mathcal{J}_{\mathcal{M}}(\pi^*) &\leqslant \mathcal{J}_{\widetilde{\mathcal{M}}}(\pi^*) + U \\ &\leqslant \mathcal{J}_{\widetilde{\mathcal{M}}}(\widetilde{\pi}) + U \\ &\leqslant \mathcal{J}_{\mathcal{M}}(\widetilde{\pi}) + 2U.\end{aligned}$$

□

As indicated in (Yu et al., 2020), an important obstacle to applying Lemma 3 to real reinforcement learning problems is to bound the Lipschitz constant L_v for optimal value functions. Further, we need to bound $W(T, T')$, the 1-Wasserstein distance between two transition functions. We study these two problems in the following two subsections, respectively.

D.3 Lipschitz Constant of Value Functions

In this section, we show that the Lipschitz constant L_v can be upper bounded for any optimal value function in our setting. We first rewrite the update of model parameters in each epoch of FedAvg as follows:

$$f_z(s, \{\tilde{g}_i\}_{i\in[M]}) := s - \eta\frac{1}{K}[\sum_{k=1}^{K-M} \nabla_s \ell(s, z_k) + \sum_{k=M+1}^{K} \tilde{g}_k] \tag{2}$$

where $z = \{z_k\}$ denotes the set of data points sampled by each worker. That is, the above equation gives the one-step *deterministic* transition when the data samples are fixed. An important observation is that the transition function T is induced by a Lipschitz model class $F_g = \{f_z : z \in Z^K\}$with $g(f_z|a)$ equal to the probability that z is sampled according to the joint distribution $\prod_{k\in[K]} \widehat{P}_k$. Similarly, T' is induced by $F_{g'} = \{f_z : z \in Z^K\}$ with $g'(f_z|a)$ equal to the probability that z is sampled according to the joint distribution $\prod_{k\in[M]} \widehat{P}_k \widetilde{P}^{K-M}$. This observation allows us to apply the techniques in (Asadi et al., 2018) to bound the Lipschitz constant L_v of an optimal value function once we bound the Lipschitz continuity of individual f_z.

We first show that for any joint action $a = \{\tilde{g}_i\}_{i\in[M]}$, the deterministic transition $f_z(\cdot, a)$ is Lipschitz continuous with a Lipschitz constant $K_{d_S,d_S}(f_z(\cdot, a))$ that can be upper bounded independent of z.

Lemma 4. *Assume Assumptions 2.3, 2.4, and 2.5 hold. For any Lipschitz model class $F_g = \{f_z : z \in Z^K\}$, we have $K_F \leqslant \max\{\epsilon|1 - \eta\alpha|, \epsilon|1 - \eta\beta|\}$.*

Proof. It suffices to show that for any action a, $K_{d_S,d_S}(f_z(\cdot, a)) \leqslant \max\{\epsilon|1 - \eta\alpha|, \epsilon|1 - \eta\beta|\}$. By (2), we have for any $s_1, s_2 \in S$,

$$\begin{aligned}\|f_z(s_1, a) - f_z(s_2, a)\|_2 &= \|s_1 - \eta\frac{1}{K}\sum_{k=1}^{K-M} \nabla_s \ell(s_1, z_k) - (s_2 - \eta\frac{1}{K}\sum_{k=1}^{K-M} \nabla_s \ell(s_2, z_k))\|_2 \\ &\overset{(a)}{\leqslant} \frac{1}{K}\sum_{k=1}^{K-M} \|s_1 - \eta\nabla_s \ell(s_1, z_k) - (s_2 - \eta\nabla_s \ell(s_2, z_k))\|_2 \\ &\overset{(b)}{=} \frac{1}{K}\sum_{k=1}^{K-M} \|(I - \eta\frac{\partial^2 \ell(\bar{s}, z_k)}{\partial s^2})(s_1 - s_2)\|_2 \\ &\overset{(c)}{\leqslant} \frac{1}{K}\sum_{k=1}^{K-M} \|I - \eta\frac{\partial^2 \ell(\bar{s}, z_k)}{\partial s^2}\|_2 \|s_1 - s_2\|_2\end{aligned}$$

where (a) follows from the triangle inequality, (b) follows from the fact that $\ell(s, z)$ is twice continuously differentiable with respect to s and the mean value theorem, where $\bar{s}$ is a point on the line segment connecting s_1 and s_2, and I is the identity matrix with its dimension equal to the dimension of the model parameters, and (c) is due to the Cauchy–Schwarz inequality.

By the strong convexity and smoothness of $\ell(s, z)$ with respect to s, the eigenvalues of $\frac{\partial^2 \ell(\bar{s}, z_k)}{\partial s^2}$ are between α and β (Polyak, 1987). It follows that

$$\|I - \eta \frac{\partial^2 \ell(\bar{s}, z_k)}{\partial s^2}\|_2 \leqslant \max\{|1 - \eta\alpha|, |1 - \eta\beta|\}, \quad \forall k$$

Therefore, for any s_1, s_2,

$$\frac{\|f_z(s_1, a) - f_z(s_2, a)\|_2}{\|s_1 - s_2\|_2} \leqslant \max\{\epsilon|1 - \eta\alpha|, \epsilon|1 - \eta\beta|\}$$

By Definition 2, we then have

$$\begin{aligned} K_{d_S, d_S}(f_z(\cdot, a)) &:= \sup_{s_1, s_2} \frac{\|f_z(s_1, a) - f_z(s_2, a)\|_2}{\|s_1 - s_2\|_2} \\ &\leqslant \max\{\epsilon|1 - \eta\alpha|, \epsilon|1 - \eta\beta|\} \end{aligned}$$

□

Note that by using a small enough learning rate η, K_F can be made less than 1 so that the one-step deterministic transition becomes a contraction. We next show that the optimal value function $V^*_{\mathcal{M},l}(\cdot)$ has a bounded Lipschitz constant. Note that the bound is independent of $\mathcal{M}$; hence it also applies to $V^*_{\widetilde{\mathcal{M}},l}(\cdot)$

Lemma 5. *Assume Assumptions 2.1, 2.3, 2.4, and 2.5 hold. The optimal value function $V^*_{\mathcal{M},l}(\cdot)$ is Lipschitz continuous with a Lipschitz constant bounded by $\sum_{t=0}^{H-l-1}(K_F)^t(L + LK_F)$.*

Proof. The proof is adapted from the proof of Theorem 3 in (Asadi et al., 2018). Let $Q^\pi_{\mathcal{M},l}(s, a) = r(s, a) + \sum_{s' \in S} T(s'|s, a)V_{\mathcal{M},l+1}(s')$ denote the state-action value function, where $r(s, a) = \mathbb{E}_{s' \sim T(s'|s,a)}[r(s, a, s')]$. We have for the optimal state-action value function

$$Q^*_{\mathcal{M},l}(s, a) = r(s, a) + \sum_{s' \in S} T(s'|s, a) \max_{a' \in A} Q^*_{\mathcal{M},l+1}(s', a')$$

with $Q^*_{\mathcal{M},H-1}(s, a) = r(s, a)$. The Lipschitz constant of $Q^*_{\mathcal{M},l}$ is bounded by:

$$\begin{aligned} K^A_{d_S, d_\mathbb{R}}(Q^*_{\mathcal{M},l}) &\leqslant K^A_{d_S, d_\mathbb{R}}(r) + K^A_{d_S, d_\mathbb{R}}(\sum_{s' \in S} T(s'|s, a) \max_{a' \in A} Q^*_{\mathcal{M},l+1}(s', a')) \\ &\overset{(a)}{\leqslant} K^A_{d_S, d_\mathbb{R}}(r) + K^A_{W,W}(T) K^A_{d_S, d_\mathbb{R}}(\max_{a' \in A} Q^*_{\mathcal{M},l+1}) \\ &\overset{(b)}{\leqslant} K^A_{d_S, d_\mathbb{R}}(r) + K^A_{W,W}(T) K^A_{d_S, d_\mathbb{R}}(Q^*_{\mathcal{M},l+1}) \\ &\leqslant K^A_{d_S, d_\mathbb{R}}(r) + K^A_{W,W}(T)[K^A_{d_S, d_\mathbb{R}}(r) + K^A_{W,W}(T) K^A_{d_S, d_\mathbb{R}}(Q^*_{\mathcal{M},l+2})] \\ &\leqslant K^A_{d_S, d_\mathbb{R}}(r) + \sum_{t=1}^{H-l-2} (K^A_{W,W}(T))^t K^A_{d_S, d_\mathbb{R}}(r) + K^A_{W,W}(T)^{H-l-1} K^A_{d_S, d_\mathbb{R}}(Q^*_{\mathcal{M},H-1}) \\ &= \sum_{t=0}^{H-l-1} (K^A_{W,W}(T))^t K^A_{d_S, d_\mathbb{R}}(r) \end{aligned}$$

where (a) follows Lemma 2 and (b) is due to the fact that the max operator is 1-Lipschitz, that is, $K_{\|\|_\infty, d_\mathbb{R}}(\max(x)) = 1$ (Asadi & Littman, 2017). From the definition of $r(s, a)$, we further have

$$\begin{aligned} |r(s_1, a) - r(s_2, a)| &\leqslant \frac{1}{K} \sum_{k=1}^{K} |\ell_k(s_1) - \ell_k(s_2)| + \frac{1}{K} \sum_{k=1}^{K} |\mathbb{E}_{s'_1 \sim T(s_1,a)}[\ell_k(s'_1)] - \mathbb{E}_{s'_2 \sim T(s_2,a)}[\ell_k(s'_2)]| \\ &\leqslant (L + LK^A_{W,W}(T))\|s_1 - s_2\|_2 \end{aligned}$$

where $\ell_k(s) := \mathbb{E}_{z_k \sim \widehat{P}_k}[\ell(s, z_k)]$. The first term of the second inequality comes from the Lipschitz continuity of the loss function ℓ, which gives $|\ell_k(s_1) - \ell_k(s_2)| \leqslant L\|s_1 - s_2\|_2$ for any k, and the second term follows from Lemma 2 by letting $f(s) = \ell_k(s)$, which gives $K^A_{d_S, d_{\mathbb{R}}}(\mathbb{E}_{s' \sim T}[\ell_k(s')]) \leqslant LK^A_{W,W}(T)$ for all k.

Since the above inequality holds for any $a \in A$, $r(s, a)$ is uniformly Lipschitz continuous in action space A with a Lipschitz constant $K^A_{d_S, d_{\mathbb{R}}}(r) = L + LK^A_{W,W}(T)$. Thus, $K^A_{d_S, d_{\mathbb{R}}}(Q^*_{\mathcal{M},l}) \leqslant \sum_{t=0}^{H-l}(K^A_{W,W}(T))^t(L + LK^A_{W,W}(T))$. Since the optimal value function $V^*_{\mathcal{M},l}(s) = \max_{a \in A} Q^*_{\mathcal{M},l}(s, a)$ and the max operator is 1-Lipschitz (Asadi & Littman, 2017), we have $K_{d_S, d_{\mathbb{R}}}(V^*_{\mathcal{M},l}) \leqslant K^A_{d_S, d_{\mathbb{R}}}(Q^*_{\mathcal{M},l}) \leqslant \sum_{t=0}^{H-l-1}(K^A_{W,W}(T))^t(L + LK^A_{W,W}(T))$.

By Lemma 1, we have $K^A_{W,W}(T) \leqslant K_F$. The desired result then follows by applying Lemma 4.

□

The lemma immediately implies that $V^*_{\mathcal{M},l}(\cdot)$ is L_v-Lipschitz for any l where $L_v \leqslant \sum_{t=0}^{H-1}(K_F)^t(L + LK_F)$.

D.4 Wasserstein Distance between Transitions

In this section, we bound the 1-Wasserstein distance of transition functions. Recall that the true transition dynamics $T(\cdot|s, a)$ depends on the joint distribution $\prod_{k=1}^{K-M} \widehat{P}_k$, while $T'(\cdot|s, a)$ depends on $\widetilde{P}^{K-M}$. We have the following lemma.

Lemma 6. *Assume Assumptions 1-3 hold. For any state-action pair (s, a), the 1-Wasserstein distance between transition dynamics $T(\cdot|s, a)$ and $T'(\cdot|s, a)$ generated from the real FL environment and the estimated environment, respectively, is bounded by $\eta L_z \epsilon\delta$, that is,*

$$W(T(\cdot|s, a), T'(\cdot|s, a)) \leqslant \eta L_z \epsilon\delta$$

Proof. Let $z_1 = \{z_{1k}\}_{k=1,\ldots,K-M}$ and $z_2 = \{z_{2k}\}_{k=1,\ldots,K-M}$ denote two data sets of normal workers sampled from $\prod_{k=1}^{K-M} \widehat{P}_k$ and $\widetilde{P}^{K-M}$ respectively. Let $j = \prod_{k=1}^{K-M} j_k$ denote an arbitrary coupling between the two joint distributions that is independent across workers, and $\mathcal{J}$ the set of all such couplings. Let $\mathcal{J}_s$ denote the collection of couplings between $T(\cdot|s, a)$ and $T'(\cdot|s, a)$ generated from the couplings of joint distributions in $\mathcal{J}$. To simplify the notation, let $s(z) := f_z(s, a)$ denote the successive state given the current state-action pair (s, a) and the sampled data z of normal workers.

From the definition of 1-Wasserstein distance, we have

$$
\begin{aligned}
W(T(\cdot|s,a),T'(\cdot|s,a)) &\overset{(a)}{\leqslant} \inf_{j_s\in\mathcal{J}_s}\sum_{(s_1',s_2')}\|s_1'-s_2'\|_2 j_s(s_1',s_2') \\
&\overset{(b)}{\leqslant} \inf_{j\in\mathcal{J}}\sum_{(z_1,z_2)}\|s(z_1)-s(z_2)\|_2 j(z_1,z_2) \\
&= \inf_{j\in\mathcal{J}}\sum_{(z_1,z_2)}\|s-\frac{1}{K}(\sum_{k=1}^{K-M}\nabla_s\ell(s,z_{1k})+a) \\
&\qquad -[s-\frac{1}{K}(\sum_{k=1}^{K-M}\nabla_s\ell(s,z_{2k})+a)]\|_2\prod_{k=1}^{K-M}j_k(z_{1k},z_{2k}) \\
&= \inf_{j\in\mathcal{J}}\sum_{(z_1,z_2)}\|\frac{1}{K}\sum_{k=1}^{K-M}\nabla_s\ell(s,z_{1k})-\frac{1}{K}\sum_{k=1}^{K-M}\nabla_s\ell(s,z_{2k})\|_2\prod_{k=1}^{K-M}j_k(z_{1k},z_{2k}) \\
&\overset{(c)}{\leqslant} \frac{\eta L_z}{K}\inf_{j\in\mathcal{J}}\sum_{(z_1,z_2)}\sum_{k=1}^{K-M}\|z_{1k}-z_{2k}\|_2\prod_{k=1}^{K-M}j_k(z_{1k},z_{2k}) \\
&\overset{(d)}{\leqslant} \frac{\eta L_z}{K}\inf_{j\in\mathcal{J}}\sum_{(z_1,z_2)}\sum_{k=1}^{K-M}\|z_{1k}-z_{2k}\|_2 j_k(z_{1k},z_{2k}) \\
&\leqslant \frac{\eta L_z}{K}\sum_{k=1}^{K-M}\inf_{j_k}\sum_{(z_{1k},z_{2k})}\|z_{1k}-z_{2k}\|_2 j_k(z_{1k},z_{2k}) \\
&= \frac{\eta L_z}{K}\sum_{k=1}^{K-M}W(\widehat{P}_k,\widetilde{P}) \overset{(e)}{\leqslant} \frac{\eta L_z}{K}(K-M)\delta
\end{aligned}
$$

where (a) is due to the fact that we consider a restrictive collection of couplings, (b) is due to the fact that $\mathcal{J}_s$ is generated from $\mathcal{J}$, (c) follows from the smoothness of $\ell(s,z)$ with respect to z, (d) is due to $j_k(z_{1k},z_{2k})\leqslant 1,\forall k$, and (e) follows from Assumption 3. □

D.5 Difference between Expected Returns

Combining the results from the previous three sections, we have the following main result.

Theorem 1. *Assume Assumptions 1-3 hold. Let $\mathcal{J}_{\mathcal{M}}(\pi):=\mathbb{E}_{\pi,T,\mu_0}[\sum_{t=0}^{H-1}r(s^t,a^t,s^{t+1})]$ denote the expected return over H attack steps under Markov game $\mathcal{M}$, policy π and initial state distribution μ_0. Let π^* and $\widetilde{\pi}$ be optimal policies for $\mathcal{M}$ and $\widetilde{\mathcal{M}}$ respectively, with the same initial state distribution μ_0. Then,*

$$|\mathcal{J}_{\mathcal{M}}(\pi^*)-\mathcal{J}_{\mathcal{M}}(\widetilde{\pi})|\leqslant 2H\epsilon\delta[(L+L_v)\eta L_z+2L]$$

where $L_v\leqslant\sum_{t=0}^{H-1}(K_F)^t(L+LK_F)$ and $K_F\leqslant\epsilon\max\{|1-\eta\alpha|,|1-\eta\beta|\}$.

Proof. By Lemma 3, $|\mathcal{J}_{\mathcal{M}}(\pi^*)-\mathcal{J}_{\mathcal{M}}(\widetilde{\pi})|\leqslant 2(H(L+L_v)W(T(\cdot|s,a),T'(\cdot|s,a))+2HL\epsilon\delta)$. From Lemma 6, we have $W(T(\cdot|s,a),T'(\cdot|s,a))\leqslant\eta L_z\epsilon\delta$. Thus, $|\mathcal{J}_{\mathcal{M}}(\pi^*)-\mathcal{J}_{\mathcal{M}}(\widetilde{\pi})|\leqslant 2H[(L+L_v)\eta L_z\epsilon\delta+2L\epsilon\delta]$. By Lemma 5 and the comment below it, $L_v\leqslant\sum_{t=0}^{H-1}(K_F)^t(L+LK_F)$ where $K_F\leqslant\epsilon\max\{|1-\eta\alpha|,|1-\eta\beta|\}$. □