Learning to Attack Federated Learning: A Model-based Reinforcement Learning Attack Framework

Henger Li,* **Xiaolin Sun,*** **and Zizhan Zheng**
Department of Computer Science
Tulane University
New Orleans, LA 70118
{hli30, xsun12, zzheng3}@tulane.edu

Abstract

We propose a model-based reinforcement learning framework to derive untargeted poisoning attacks against federated learning (FL) systems. Our framework first approximates the distribution of the clients' aggregated data using model updates from the server. The learned distribution is then used to build a simulator of the FL environment, which is utilized to learn an adaptive attack policy through reinforcement learning. Our framework is capable of learning strong attacks automatically even when the server adopts a robust aggregation rule. 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 significantly outperforms state-of-the-art poisoning attacks. This indicates the importance of developing adaptive defenses for FL systems.

1 Introduction

Federated learning (FL) is a promising machine learning framework that allows multiple devices with private data to jointly train a learning model (coordinated by a server) without sharing their local data. It has recently been applied to consumer digital products [41], credit risk prediction [1], drug discovery [42], and digital health [49]. However, federated learning systems are vulnerable to adversarial attacks [39] such as model poisoning attacks [11, 65, 20, 8], data poisoning attacks [9, 23, 26], and inference attacks [43, 29, 74]. To this end, various robust aggregation rules such as coordinate-wise median [67], trimmed mean [67], Krum [12], norm clipping [59], geometric median [46], and FLTrust [15] have been proposed. However, these defenses are mainly evaluated against manually crafted myopic attack policies [53]. Their robustness in the face of advanced attacks remains unknown.

Due to the distributed nature of FL systems, a malicious device typically has limited knowledge about benign devices and system dynamics. To fully reveal the vulnerabilities of FL systems, it is therefore crucial to develop strong attacks that can best utilize the limited global knowledge. In this work, we take a first step in this direction by considering the white-box attack setting where the attacker has some global knowledge about the FL system and the server's algorithm, but has no access to the private data of benign devices, a reasonable assumption for real-world FL systems. To derive strong adaptive attacks, we propose to leverage the power of model-based reinforcement learning (RL) by integrating *distribution learning* and *policy learning*. A key observation of our approach is that although accurate information about individual devices can be hard to obtain in FL, it is often possible to infer their aggregated data distribution from publicly available model updates, which

*Equal contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

is sufficient to derive strong attacks. In particular, the set of malicious devices first cooperatively estimate the aggregated data distribution through gradient inversion [29, 74]. The learned distribution is then used to build a simulator of the FL environment, which is utilized to derive an adaptive attack policy through reinforcement learning. We focus on untargeted model poisoning in this work, where the malicious devices aim to reduce the accuracy of the global model as much as possible by sending crafted gradient information to the server. However, our proposed framework can potentially be applied to other types of attacks in both the white-box and the more challenging black-box settings.

Our model-based approach distinguishes from existing work on reinforcement learning based adversarial attacks against machine learning systems [58, 71, 72]. In particular, we consider 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. In contrast, previous works typically assume more powerful attackers that can attack at any time and have full knowledge about the environment. Thus, they typically adopt a purely model-free approach, which is infeasible in attacking FL systems due to the large number of samples needed to be effective.

Our contributions. We advance the state-of-the-art in the following aspects. First, we propose a novel reinforcement learning attack framework against federated learning systems by integrating distribution learning and policy learning. Second, we theoretically quantify the effect of inaccurate distribution learning and heterogeneous local data distributions on the optimal attack performance. Third, our experiments on real-world datasets demonstrate that our RL-based attack method consistently outperforms existing model poisoning attacks [11, 20, 65] and significantly reduces the global model accuracy even when robust aggregation rules are applied. These findings indicate the importance of developing adaptive defenses for FL systems.

2 Related Work

Poisoning attacks and defenses. To compromise the integrity of the target model in federated learning, both targeted poisoning attacks [11, 8, 9] that aim to misclassify a specific set of inputs, and untargeted attacks [20, 65, 52] aiming to reduce the global model accuracy have been proposed. Existing approaches typically adopt a heuristics-based method [65]) or optimize a myopic goal [20, 52]), and are usually sub-optimal, especially when a robust aggregation rule is adopted. Further, they often require access to benign agents' local updates or the accurate global model parameters of the next round [65, 20]) to make significant attack impact. In contrast, our reinforcement learning based attack requires less global knowledge and targets a long-term attack goal.

Various defenses have been proposed for model poisoning attacks including robust-aggregation-based approaches and detection-based approaches. The former includes dimension-wise filtering that considers each dimension of local updates separately [10, 67], client-wise filtering that aims to restrict or even remove the impact of potentially malicious clients [12, 12, 46, 59], and approaches that require the server to have access to a small amount of root data [15]. Our RL-based attack is effective against all these defenses. In addition, time-coupled robust aggregation methods [4, 5, 32] that target adaptive attacks and anomaly detection-based defenses [35] have been proposed recently. Our approach can potentially be extended to compromise them by explicitly encoding the history information into states or utilizing a recurrent structure.

RL-based attacks. Reinforcement learning has recently been utilized for developing strong attacks in various settings, including corrupting training data of online supervised and unsupervised learning [72], manipulating the combinatorial structure of graph data [18], and injecting malicious nodes into a graph [58]. RL-based attacks have also been developed to damage the performance of reinforcement learning itself, by perturbing the reward signals during the training stage [71] and corrupting the state signals received by an agent during the testing stage [70, 56]. However, these methods typically assume that the attacker has access to an accurate MDP model or simulator and has unlimited time for training the attack policy. In contrast, our method first builds a world model by learning a data distribution from the FL model updates and then constructs an approximate simulator for training our attacks. Both distribution learning and policy learning happen when FL training is ongoing. Further, previous work has mainly focused on attacking a single RL agent by an external agent rather than an insider attack in a distributed learning environment as we consider in this work.

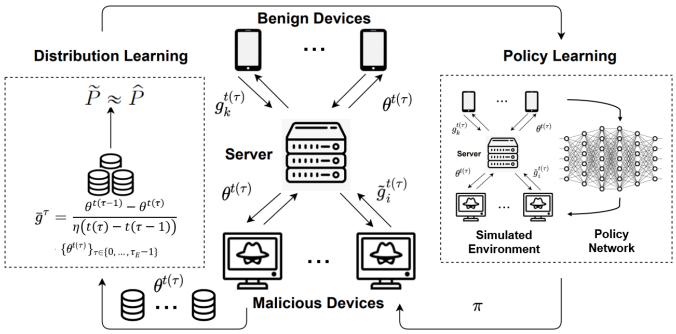

Figure 1: An overview of the RL-based attack framework against federated learning.

3 Approach Overview

In this section, we describe the federated learning setting considered in this work, the threat model, and the proposed RL-based attack framework.

Federated learning. We consider an FL setting that is similar to *federated averaging (FedAvg)* [40]. The FL system consists of a server and K workers (also known as devices or clients) in which each worker has some private data. Let $[K] = \{1, 2, ..., K\}$ denote the set of workers. 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 $f(\theta) := \sum_{k=1}^{K} p_k F_k(\theta)$. Here $F_k(\cdot)$ 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, $F_k(\theta) = \frac{1}{N_k} \sum_{j=1}^{N_k} \ell(\theta; z_{jk})$, where N_k is the number of data samples available locally on worker k, $\ell(\cdot;\cdot)$ is the loss function, and $z_{jk} := (x_{jk}, y_{jk})$ 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 $N = \sum_k N_k$ is the total number of data samples across workers.

If all the workers' local 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) works as follows: at each time step $t \geqslant 0$, 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 [37]. The process of selecting workers for aggregation is called *subsampling*. Let $\kappa = w/N$ denote the *subsampling rate*, i.e., the ratio of the selected workers number w to the total number of workers K. Each selected worker $k \in [w]$ then samples a minibatch b_k of size B from its local data distribution $\widehat{P}_k$. 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} is then sent to the selected workers to perform another 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 workers. They are coordinated either by one leading attacker or an external agent. We refer such agent as a *leader agent*. These attackers are assumed to be *fully cooperative* and share the same goal of compromising the FL system. We consider untargeted model poisoning attacks in this work 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 $f(\theta)$. the batch size B), and their local data distributions $\{\widehat{P}_k\}_{k \in \mathcal{A}}$ (but not the benign workers' local data distributions). We further assume that the attackers obtain information about the server's training algorithm (i.e., the white-box attack setting). This information includes the server's learning rate η, the subsampling rate κ, the total number of workers K, the aggregation rule $Aggr$, and the total number of training epochs $\mathcal{T}$.

RL-based online attack framework. Our attack framework consists of the following three phases.

- **Distribution learning**: The malicious workers first jointly learn an approximation of $\widetilde{P}$ from the model updates $\{\theta^t\}$ received from the server, using a gradient inversion based inference attack [24].
- **Policy learning**: The leader agent then builds a simulator of the FL environment using the attackers' local data and the learned distribution. An optimal attack policy is then derived through reinforcement learning using data sampled from the simulator. Note that policy learning can start together with distribution learning and continue after a reasonable distribution is learned.
- **Attack execution**: The learned policy is distributed to all the malicious workers to generate attack actions. Note that attack execution can start once an initial policy is learned, which can be updated during attack execution.

We note that all the three phases happen while the federated learning process is ongoing, thus the lengths of these phases are important hyperparameters to be determined. For example, with more observations, an attacker can learn a more accurate distribution, which will help obtain a better attack policy. However, when the total time window available to attacks is limited, a longer distribution learning phase reduces the attack opportunities in Phase 3. Compared with a purely model-free approach, our model-based approach is more sample efficient, which is especially important for federated learning as a malicious worker can only attack when it is sampled by the server.

4 Model-based Reinforcement Learning Attack Framework

In this section, we first formulate the model poisoning attack problem as a Markov decision process (MDP). We then discuss our model-based reinforcement learning attack framework in more details.

4.1 Attackers' problem as a Markov decision process

The attackers' problem is formulated as an undiscounted MDP, denoted by $\mathcal{M} = (S, \boldsymbol{A}, T, r, H)$, where

- S is the state space. Let $\tau \in \{0, 1, ..., H-1\}$ denote the index of the attack step and $t(\tau) \in \{0, 1, ..., \mathcal{T}-1\}$ 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 $t(\tau)$, which is shared between all malicious workers.
- $\boldsymbol{A} = A^M$ is the space of the attackers' joint actions where each attacker shares the same action space A. If attacker i is selected at $t(\tau)$, 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.
- $T : S \times \boldsymbol{A} \to \mathcal{P}(S)$ is the state transition function that represents the probability of reaching a state $s' \in S$ from the current state $s \in S$ when attackers choose actions $a_1^\tau, ..., a_M^\tau$, respectively.
- $r : S \times \boldsymbol{A} \times S \to \mathbb{R}_{\geqslant 0}$ 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 current state, the next state, and the joint attack actions and is shared by all the attackers.
- H is the number of attack steps in each episode and we have $t(H-1) < \mathcal{T}$.

The attackers' goal is to find a joint 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 $\pi_i : S \to \mathcal{P}(A)$ denotes a stationary policy of attacker i that maps the state to a probability measure over A. Using the definition of r^τ, this objective is equivalent to finding a joint policy π that maximizes $\mathbb{E}_{\theta^{t(H)}}[f(\theta^{t(H)})]$.

A key obstacle to solving the MDP is that both the transition probabilities T and the reward function r depend on the joint empirical distribution across workers $\{\widehat{P}_k\}_{k\in[K]}$, which is fixed but unknown to the attackers. Although model-free reinforcement learning can bypass this difficulty, it requires a large number of samples to be effective, which is infeasible in the online attacking scenario we consider. We therefore adopt model-based reinforcement learning as a principled approach for designing adaptive attacks in the online setting. An important observation 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, which is often sufficient to simulate the behavior of benign agents and the server by assuming that each benign agent samples data from $\widetilde{P}$. This gives rise to a new MDP $\widetilde{\mathcal{M}} = (S, \boldsymbol{A}, T', r', H)$ where T' and r' are derived from $\widetilde{P}$. Thus, our proposed model-based reinforcement learning attack framework naturally consists of the distribution learning, policy learning, and attack execution phases.

4.2 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 inversion based inference attack [24, 74]. Various gradient inversion attacks have been proposed in the literature. In this work, we adapt the *inverting gradients* (IG) method [24] to distribution learning. The IG method reconstructs data samples by optimizing a loss function based on the angle (i.e., cosine similarity) between the gradient generated from true data and that from the reconstructed data. The primary 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 inversion 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 and aggregated gradient information from consecutive model updates received from the server, due to model aggregation and subsampling. Despite these differences, our experiment results show that the $\widetilde{P}$ learned using IG can help derive an effective attacker policy (see Figure 4(c)).

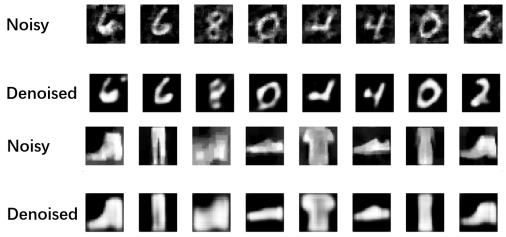

Figure 2: Examples of reconstructed images (before and after denoising) for MNIST (upper) and Fashion-MNIST (lower) datasets.

As shown in Algorithm 2 in Appendix B, 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 as $\bar{g}^\tau := (\theta^{t(\tau-1)} - \theta^{t(\tau)})/(\eta(t(\tau) - t(\tau-1)))$. The leader agent then starts with a batch of (randomly generated) dummy data and dummy labels D_{dummy}, which is updated iteratively by solving the following optimization problem: $\arg\min_{D_{dummy}} 1 - \cos(\nabla_\theta F_{dummy}(\theta^{t(\tau)}), \bar{g}^\tau) + \frac{\beta}{B'} \sum_{(x,y)\in D_{dummy}} \text{TV}(x)$, where $\cos(A, B) := \frac{\langle A, B\rangle}{\|A\|\|B\|}$ is the cosine similarity between two vectors A and B, $F_{dummy}(\theta) = \frac{1}{B'} \sum_{(x,y)\in D_{dummy}} \ell(\theta; (x, y))$, B' is the number of reconstructed data per epoch (the size of the dummy data batch), $\text{TV}(x)$ is the total variation [50] of x, and β is a fixed parameter. The process terminates after max_iter iterations, then outputs the updated data as the reconstructed data samples. We observed that although the data samples generated by IG resemble true samples, they contain a certain amount of noise as shown in Figure 2, making the learned distribution less representative of the true distribution. To reduce noise, we adapt the method of denoising auto-encoder [62]. We utilize the clean data owned by attackers and add Gaussian noise to them to simulate the noise in the reconstructed data samples. The clean data and the synthetic noisy data are then paired to train an autoencoder for denoising, which is then used to remove the noise in reconstructed data samples as shown in the figure. The approximated mixture distribution $\widetilde{P}(\tau)$ consists of the reconstructed data up to $t(\tau)$ and the M attackers' local data. The distribution learning phase starts at the first FL epoch and continues for τ_E steps. The learned distribution is shared with all the attackers.

Although we adopt the IG method in this work due to its simplicity, other more recent approaches such as the GradInversion method [68] and gradient inversion with a trained generative model [30] can be easily incorporated into our framework, which can potentially learn $\widetilde{P}$ in more challenging settings for complex datasets like ImageNet [19], deep networks, and large batch sizes. On the other hand, we show in the experiments that our RL-based attack trained using attackers' local data only is still effective and surpasses all the baselines, while distribution learning further boosts the attack performance. A detailed discussion on gradient inversion attacks and defenses is provided in Appendix C.

4.3 Policy learning

Once the leader agent obtains the approximated distribution $\widetilde{P}$, it can simulate the behavior of the server and that of normal workers. In particular, to simulate the behavior of a normal worker in each FL training step, a minibatch of size B is $i.i.d.$ sampled from $\widetilde{P}$. With experiences sampled from the simulated environment, the leader agent can learn a joint attack policy that maximizes the empirical loss using a state-of-the-art (deep) reinforcement learning algorithm, e.g., TD3 [22] or PPO [51].

Note that the leader agent does not need to wait until a good distribution has been learned to start training the policy. Instead, policy learning can start together with distribution learning. Initially, the leader agent uses the attackers' local data to train the policy, which will be continuously updated while more data samples are being generated. To reduce the training overhead, we assume that all the malicious workers share the same attack policy (i.e., $\pi_1 = \pi_2 = \cdots = \pi_M$) in this work, which achieves good attack performance in our experiments. Extension to general joint policies will be considered in our future work.

When we train a small neural network with federated learning, it is natural to use $(\theta^{t(\tau)}, \mathcal{A}^{t(\tau)})$ as the state, and the crafted gradient $\widetilde{g}^{t(\tau)}$ 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 an 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 compress the state and action spaces for high-dimensional models as follows.

To compress the state space, we first observe that as the set of attackers are fully cooperative and share the same policy, there is no need to distinguish them in the state when the server does not track the behavior of individual workers. Thus, we replace $\mathcal{A}^{t(\tau)}$ by the number of attackers sampled in $t(\tau)$, denoted by $m^{t(\tau)}$. Further, instead of using the entire set of model parameters $\theta^{t(\tau)}$ in the state, the parameters of the last hidden layer of the current neural network model is used. This is because the last hidden layer passes on values to the output layer and typically carries information about important features of the model [57]. Note that the approximated state is used to define the policy only. The true state is still the full FL model that determines transition probabilities and rewards.

It is more challenging to compress the action space. We observe that a policy that manipulates the model parameters of the last hidden layer only works well for certain aggregation rules such as Krum and coordinate-wise median. However, it becomes less effective for stronger defenses such as coordinate-wise median with clipping. Given that it is challenging to identify a small subset of model parameters that can lead to most damage to model accuracy when manipulated, we adopt a different approach in this work.

The main idea is to search for a model update direction that can lead to a large empirical loss using gradient ascent, with its parameters identified by reinforcement learning. To this end, we define the local search objective as $L(\theta) := (1-\lambda)F(\theta) + \lambda \cos(\theta^{t(\tau)} - \theta, g(\theta^{t(\tau)}))$ where $F(\theta) = \mathbb{E}_{z\sim\widetilde{P}}[\ell(\theta; z)]$ models the empirical loss and $g(\theta^{t(\tau)}) = \mathbb{E}_{z\sim\widetilde{P}}[\nabla_\theta \ell(\theta^{t(\tau)}; z)]$ captures the average update direction from normal devices, both are estimated from $\widetilde{P}$. The second term in $L(\theta)$ is used to control the deviation of the model update from the normal direction (measured by the cosine similarity) so that the adversarial input cannot be easily identified by the server. The parameter $\lambda \in [0, 1]$ is used to balance the loss and the deviation. To solve the problem, we start with $\theta = \theta^{t(\tau)}$ and generate G trajectories, where each trajectory involves E model updates. Each model update involves a single gradient ascent step using a minibatch of size $\widetilde{B}$ sampled from $\widetilde{P}$. Let θ_k denote the new model parameters found by the k-th trajectory after E updates. The update direction is then set as $\frac{1}{G}\sum_{k=1}^{G}\theta_k$ and each attacker's action in $t(\tau)$ is computed as $\widetilde{g}^{t(\tau)+1} = \gamma(\theta^{t(\tau)} - \frac{1}{G}\sum_{k=1}^{G}\theta_k)$. The scaling factor $\gamma \geqslant 0$ is used to control the magnitude of the crafted gradient, which is needed as most robust aggregation rules apply a certain type of filtering rule to mitigate the effect of malicious attacks. We assume that G is fixed while γ, E, λ are parameters to be learned by reinforcement learning. Thus, the action space for the attackers' MDP becomes a 3-dimensional real space.

4.4 Attack execution

Both distribution learning and policy learning can start from the first epoch of federating learning and continue while federating learning is ongoing. The simulated environment is updated when a

new estimated distribution $\widetilde{P}$ is learned. Although the attackers may choose to start attacking during distribution learning, we observe that this can blur the gradient information and make distribution learning less accurate. Thus, we assume that each attacker starts attacking once the distribution learning phase is finished, and applies the latest learned attack policy during the remaining epochs of the federated learning process. During attack execution, each selected attacker first notifies other attackers so that every one knows the number of attackers that are sampled in that epoch. Each selected attacker then generates a crafted gradient according to the process described above with the parameters obtained from the latest learned policy.

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 [66, 13, 31]. 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 [16]. We compare the actual running time of our RL-based attack against different defenses in our experiment setting in Appendix E.2.

5 Impact of Inaccurate Distribution Learning and Data Heterogeneity

Our model-based RL 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 phases of our attack method.

Our analysis is adapted from recent works that study the impact of model inaccuracy on the performance of model-based reinforcement learning [69, 38, 71] by addressing two new challenges. First, we need to establish the connection between the inaccuracy in data distribution $\widetilde{P}$ and the inaccuracy in the corresponding MDP 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 [69], it makes more sense to use the 1-Wasserstein distance [61] to measure the distance between two data distributions. This, however, requires bounding the Lipschitz constant of the optimal value function [69]. 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 workers' local data.

Assumption 1. *$W_1(\widetilde{P}, \widehat{P}_k) \leqslant \delta$ for any benign worker k, where $W_1(\cdot,\cdot)$ is the 1-Wasserstein distance [61].*

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. $\ell(s_2, z_1) \geqslant \ell(s_1, z_1) + \langle \nabla_s \ell(s_1, z_1), s_2 - s_1\rangle + \frac{\alpha}{2}\|s_2 - s_1\|_2^2$ *(strong convexity w.r.t. s);*
4. $\ell(s_2, z_1) \leqslant \ell(s_1, z_1) + \langle \nabla_s \ell(s_1, z_1), s_2 - s_1\rangle + \frac{\beta}{2}\|s_2 - s_1\|_2^2$ *(strong smoothness 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 $\widehat{P}_k$.

Since no subsampling is considered in this section, with a slight abuse of notation, we let index t denote both an attack step and the corresponding FL epoch. Let $\mathcal{M} = (S, \boldsymbol{A}, T, r, H)$ denote the true MDP for the attackers, and $\widetilde{\mathcal{M}} = (S, \boldsymbol{A}, T', r', H)$ the simulated MDP when the local distribution of any benign worker is estimated as $\widetilde{P}$. 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 $K_F \leqslant \epsilon \max\{|1-\eta\alpha|, |1-\eta\beta|\}$.

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 $\eta = O(\frac{1}{\sqrt{H}})$ [47], thus $|\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\mathcal{M}}(\widetilde{\pi}^*)| = O(\frac{\epsilon}{1-\epsilon}\delta\sqrt{H})$. This result clearly shows how the loss of attack performance depends on the fraction of benign nodes, the inaccuracy of distribution learning, and the time horizon.

6 Experiments

In this section, we compare our RL-based attack with state-of-the-art model poisoning attacks on real-world datasets. Our code is available at https://github.com/SliencerX/Learning-to-Attack-Federated-Learning.

6.1 Experiment setup

Datasets. We conduct extensive experiments on four real-world datasets: MNIST [34], Fashion-MNIST [64], EMNIST [17], and CIFAR-10 [33]. Due to space limitation, experiment results for Fashion-MNIST, EMNIST, and CIFAR-10 are provided in Appendix E.2. For the *i.i.d.* setting, we randomly split the dataset into K groups, each of which consists of the same number of training samples. For the *non-i.i.d.* setting, we follow the method of [20] to quantify the heterogeneity of local data distribution across clients. Suppose there are C classes in the dataset, e.g., $C = 10$ for the MNIST and Fashion-MNIST datasets. We evenly split the worker devices into C groups (with the M attackers evenly distributed across the C groups), where each group is assigned $1/C$ of training samples as follows. A training instance with label c is assigned to the c-th group with probability $q \geqslant 1/C$ and to every other group with probability $(1-q)/(C-1)$. Within each group, instances are evenly distributed. A higher q indicates a higher *non-i.i.d.* degree. We set $q = 0.5$ as the default *non-i.i.d.* degree. To demonstrate the power of distribution learning, we assume that the set of attackers share m true data points sampled from the training instances assigned to them. We set $m = 200$ as the default value for MNIST.

Baselines. We compare our RL-based attack (RL) with no attack (NA), and the state-of-the-art model poisoning FL attack methods: explicit boosting (EB) [11], inner product manipulation (IPM) [65], and local model poisoning attack (LMP) [20]. IPM manipulates the attackers' gradients so that the inner product between the aggregation result and the true gradient is negative. This requires access to the average of normal workers' gradients in each FL epoch, which is usually unavailable in practice. LMP generates myopic attacks by solving an optimization problem in each FL epoch. In addition to the server's aggregation rule, it also requires access to normal workers' local models. Although LMP with partial knowledge is also presented in [20], it performs substantially worse than the full knowledge case when the server uses the coordinate-wise median defense. We compare the RL-based attack with the more powerful full knowledge LMP below.

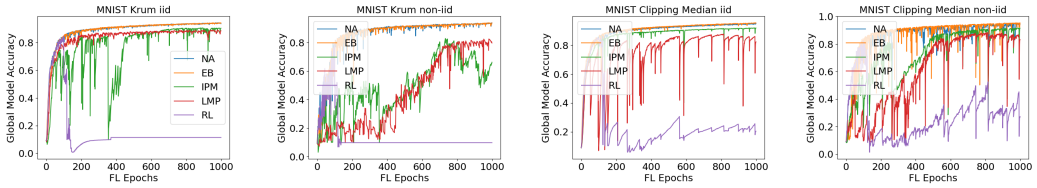

Figure 3: A comparison of global model accuracy under Krum and clipping median for both *i.i.d.* data and *non-i.i.d.* data. All parameters are set as default.

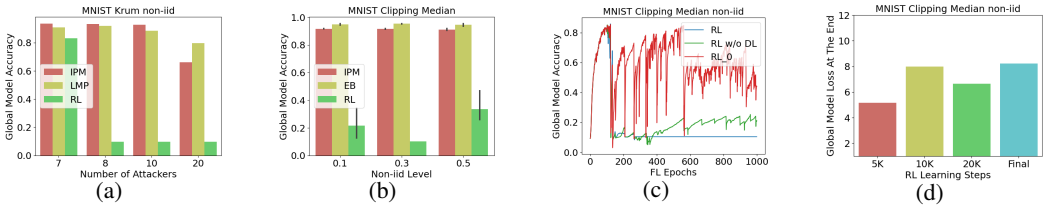

Figure 4: Attack performance on MNIST under (a) different number of attackers; (b) different non-iid degrees; (c) RL with and without distribution learning and RL_0 (zero initial data for policy learning); and (d) different policy learning lengths. Non-iid degree $q = 0.5$ in (a) and $q = 0.3$ in (c) and (d). Other parameters are set as default.

We consider four representative robust aggregation rules of different types [53]: Krum [12], geometric median [46], both of which apply client-wise filtering to model updates, coordinate-wise median [67], which adopts a dimension-wise filtering, and FLTrust [15], which requires the server to have access to a small amount of root data. In the experiments, we actually consider an extension of the vanilla coordinate-wise median where a norm clipping [59] step is first applied. This gives a more powerful defense as we observed in experiments. We set the default norm threshold to 2.

Default FL and RL settings. We adopt the following default parameters for the FL models: number of total workers $= 100$, number of attackers $= 20$, learning rate $\eta = 0.01$, subsampling rate $= 10\%$, the number of total FL epochs $= 1,000$. For our RL-based attack, both the distribution learning and policy learning phase start at the first FL epoch. The former ends at the 100th FL epoch when RL-based attack starts (all other attacks start at epoch 0). The policy learning phase ends at the 400th epoch. Since both the action space and state space are continuous in our setting, we choose the state-of-the-art Twin Delayed DDPG (TD3) [22] and Proximal Policy Optimization (PPO) [51] algorithms for training the attack policy in our experiments and find that TD3 gives better results in most cases. Below we report the results for TD3. We fix the initial model and the random seeds for subsampling and local data sampling for fair comparisons. See Appendix E for details of the datasets, experimental setups, and additional results.

6.2 Attack performance

Figure 3 shows how the test accuracy of the global model varies over FL epochs under different attacks when the server uses Krum and clipping median as the aggregation rule, respectively. Results for geometric median and FLTrust are provided in Appendix E.2. We observe that our RL-based attack performs significantly better in all the settings, despite the fact that IPM and LMP use the model updates of normal clients while RL does not. Note that for Krum, RL-based attack quickly drives the global model to a poor state ($\sim 10\%$ accuracy) once the attack starts at epoch 100 under both i.i.d. and non-i.i.d. local data distributions. Attacks become harder under clipping median due to the norm clipping but our RL-based attack still reduces global model accuracy to around 50% on average. This is mainly because it targets long-term return while all other baselines are myopic. For example, in Figure 3(c), the global model accuracy drops significantly under all the attacks when five malicious devices are sampled around epoch 200. After that, the RL method keeps the accuracy at a low level, while other baselines' accuracy rebounds rapidly.

6.3 Ablation studies

Impact of the number of attackers. Previous studies on untargeted model poisoning in federated learning typically assume a relatively large fraction of attackers. For example, the default setting is 20% in [20] and 40% in [65]. Figure 4(a) shows that our RL-based attack obtains superb performance even when the number of attackers (among 100 total clients) is as low as 8. In contrast, neither IPM nor LMP obtains meaningful attack performance even with 10 attackers. For 7 attackers, none of the baselines including the RL-based attack can cause significant damage to the FL system.

Impact of non-i.i.d. degree. Figure 4(b) shows the impact of data heterogeneity on attack performace. We use 5 different random seeds for all attacks and show the error bars. We observe that all the baselines obtain similar performance under different non-i.i.d. degrees and the impact of randomness in the testing environment on their performance is limited. On the other hand, we observe that the RL policies for $q = 0.1$ and $q = 0.5$ exhibit large variances, but even the worst-case performance of our attack outperforms the best cases of all the baselines. For $q = 0.3$, the RL-based attack can always lead to a model with a very high loss so that the model accuracy stays at a low level and is close to a constant, which explains the observed low variance in model accuracy. We expect that the variation across different RL policies is in part because the attackers always use the latest trained policy for attack execution, which does not necessarily give the best performance among all the intermediate policies trained (see also Figure 4(d)).

Importance of distribution learning. Figure 4(c) compares the global model accuracy of RL-based attack with distribution learning (RL) and that without distribution learning (RL w/o DL). We observe that in both cases, the model accuracy decreases dramatically after the attack starts at FL epoch 100. Further, the accuracy of RL w/o DL slightly increases up to 20%, while the accuracy of RL stays below 10%, which is consistent with our expectation that distribution learning allows the attackers to learn a better attack policy. Figure 4(c) also shows the attack performance of RL_0, a variant of the RL-based attack where the attackers only have 200 *unlabeled* true images used to train the denoising autoencoders, thus completely relying on distribution learning to generate labeled samples needed for policy learning. Compared with the baseline results in Figure 4(b) ($q = 0.3$), we observe that RL_0 still outperforms other baseline methods, further indicating the power of distribution learning. On the other hand, the fact that RL w/o DL surpasses all the baselines indicates that our approach is still applicable even when distribution learning becomes less effective due to a strong defense against gradient inversion applied at the server side.

Impact of training length on policy learning. Figure 4(d) shows how the global model loss at the end of an FL training episode (in the simulated environment) varies over the RL policy training steps. We observe that longer training usually provides a better attack policy, although the training process is not stable. To fix this, one approach is to set up a separate testing environment to identify best trained policies. As mentioned above, our RL-based attack achieves promising performance even when the attackers always use the latest policy obtained during policy learning.

7 Conclusion

We propose a new approach for developing non-myopic attacks that can effectively compromise FL systems even with advanced defense mechanisms applied, by utilizing model-based reinforcement learning as a principled approach. While we focus on untargeted model poisoning against FL systems in this paper, our attack framework can be extended to targeted attacks (e.g., backdoor attacks) and to objectives beyond global model accuracy (e.g., fairness across clients [44, 36]). Further, our attack framework can be integrated with meta-learning [21, 27] to generalize the learned policy to different training tasks and develop black-box 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.

8 Acknowledgment

This work has been funded in part by NSF grants CNS-1816495 and CNS-2146548 and Tulane University Jurist Center for Artificial Intelligence. We thank the anonymous reviewers for their valuable and insightful feedback.

References

[1] Utilization of FATE in Risk Management of Credit in Small and Micro Enterprises. https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[3] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1964.

[4] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. *Advances in Neural Information Processing Systems(NeurIPS)*, 31, 2018.

[5] Zeyuan Allen-Zhu, Faeze Ebrahimianghazani, Jerry Li, and Dan Alistarh. Byzantine-resilient non-convex stochastic gradient descent. In *International Conference on Learning Representations(ICLR)*, 2020.

[6] Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning(ICML)*, 2017.

[7] Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning(ICML)*, 2018.

[8] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[9] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems(NeurIPS)*, 2019.

[10] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd with majority vote is communication efficient and fault tolerant. In *International Conference on Learning Representations(ICLR)*, 2018.

[11] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning(ICML)*, 2019.

[12] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems(NeurIPS)*, 2017.

[13] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *Proceedings of Machine Learning and Systems*, 2019.

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[15] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*, 2021.

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

[17] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*, 2017.

[18] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning(ICML)*, 2018.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2009.

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

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning(ICML)*, 2017.

[22] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning(ICML)*, 2018.

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

[24] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems(NeurIPS)*, 2020.

[25] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[26] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017.

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

[28] 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(CVPR)*, 2016.

[29] 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*, 2017.

[30] Jinwoo Jeon, Kangwook Lee, Sewoong Oh, Jungseul Ok, et al. Gradient inversion with generative image prior. *Advances in Neural Information Processing Systems(NeurIPS)*, 2021.

[31] 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. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[32] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning(ICML)*, 2021.

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

[34] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[35] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*, 2020.

[36] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning(ICML)*, 2021.

[37] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations(ICLR)*, 2020.

[38] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, , and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations(ICLR)*, 2019.

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

[40] 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*, pages 1273–1282. PMLR, 2017.

[41] Brendan McMahan and Daniel Ramage. Federated Learning: Collaborative Machine Learning without Centralized Training Data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[42] Brendan McMahan and Daniel Ramage. Machine Learning Ledger Orchestration For Drug Discovery (MELLODDY). https://www.melloddy.eu/.

[43] 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*, 2019.

[44] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International Conference on Machine Learning(ICML)*, 2019.

[45] 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.

[46] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

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

[48] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[49] Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020.

[50] 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.

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

[52] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.

[53] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *IEEE Symposium on Security and Privacy*, 2022.

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

[55] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2021.

[56] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep RL. In *International Conference on Learning Representations(ICLR)*, 2022.

[57] 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(CVPR)*, 2014.

[58] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of The Web Conference 2020*, 2020.

[59] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

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

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

[62] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning(ICML)*, 2008.

[63] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*, 2020.

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

[65] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in Artificial Intelligence (UAI)*, pages 261–270. PMLR, 2020.

[66] 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.

[67] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning(ICML)*, 2018.

[68] Hongxu Yin, Arun Mallya, Arash Vahdat, José Manuel Álvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[69] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[70] Huan Zhang, Hongge Chen, Duane S Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations(ICLR)*, 2021.

[71] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning(ICML)*, 2020.

[72] Xuezhou Zhang, Xiaojin Zhu, and Laurent Lessard. Online data poisoning attacks. In *Learning for Dynamics and Control*, pages 201–210. PMLR, 2020.

[73] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. iDLG: Improved Deep Leakage from Gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[74] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems(NeurIPS)*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 5 and Appendix D.1.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix D.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 6.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figure 4(b).
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 6.1.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Broader Impact

To study the vulnerabilities of federated learning, we propose a model-based reinforcement learning attack framework. Our work shows that non-myopic attacks can break federated learning systems even when they are equipped with sophisticated defense rules. 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 Algorithms

Algorithm 1 gives the framework of a standard federated learning algorithm where the aggregation function, $Aggr(\cdot)$, can be either a simple average or a robust aggregation rule. Algorithm 2 gives the details of our distribution learning procedure. The algorithm first initializes $D_{reconstructed}$ with attackers' local data. A synthetic noisy dataset is built by adding Gaussian noise to $D_{reconstructed}$. A denoising autoencoder is then learned using paired clean data and noisy data. In each FL epoch, a batch of dummay data samples are first generated randomly, which are then updated iteratively by matching their average gradient with the aggregated gradient estimated from received model parameters. When no attacker is sampled in an FL epoch, the same process is applied by reusing the most recent model parameters received from the server. Due to the randomness of the algorithm, new data samples are generated and added (after denoising) to $D_{reconstructed}$ in each FL epoch during distribution learning.

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

C Discussion on Gradient Inversion Attacks and Defenses

Although federated learning is expected to protect clients' local data, it has been recently observed that sensitive information can still be inferred from the gradients or model updates shared by clients [29, 74]. In particular, it is shown in [74, 73] that using an optimization based approach, a curious server can extract both the training inputs and labels from the gradients shared by a client for a small batch size ($\leqslant 8$). This approach is further improved in [24], where it is shown that by exploiting a magnitude-invariant loss, the proposed inverting gradients (IG) method can reconstruct images in deep non-smooth architectures even in batches of 100 images. More recently, the GradInversion method [68] and gradient inversion with a trained generative model [30] are capable of reconstructing individual images with high fidelity from averaging gradients even for complex datasets like ImageNet [19], deep networks, and large batch sizes. Several approaches have been

Algorithm 2 Distribution Learning

Input: number of steps for distribution learning τ_E, number of iterations for each step max_iter, learning rate for FL η learning rate for inverting gradients η', number of reconstructed data per epoch B', and model parameters $\{\theta^{t(\tau)}\}$
Output: $D_{reconstructed}$
$D_{Reconstructed} \leftarrow M$ attackers' local data
$D_{Noisy} \leftarrow$ Add Gaussian noise to $D_{reconstructed}$ and clip data to the valid range
Train a denoising autoencoder $A_{denoise}$ using $D_{reconstructed}$ and D_{noisy}
for $\tau = 0$ to τ_E **do**
  Generate D_{dummy} with B' random data and label pairs
  Compute aggregated gradient $\bar{g}^{\tau} \leftarrow (\theta^{t(\tau-1)} - \theta^{t(\tau)})/(\eta(t(\tau) - t(\tau-1)))$
  for $i = 0$ to $max_iter - 1$ **do**
    $F_{dummy}(\theta) \leftarrow \frac{1}{B'} \sum_{(x_j,y_j)\in D_{dummy}} \ell(\theta; (x_j, y_j))$
    $\mathcal{L} \leftarrow 1 - \frac{\langle \nabla_\theta F_{dummy}(\theta^{t(\tau)}), \bar{g}^{\tau} \rangle}{||\nabla_\theta F_{dummy}(\theta^{t(\tau)})||\cdot||\bar{g}^{\tau}||} + \frac{\beta}{B'} \sum_{(x_j,y_j)\in D_{dummy}} \text{TV}\ (x_j)$
    $x_j \leftarrow x_j - \eta' \nabla_{x_j} \mathcal{L}, y_j \leftarrow y_j - \eta' \nabla_{y_j} \mathcal{L},\ \ \forall (x_j, y_j) \in D_{dummy}$
  end for
  Denoise the dummy batch D_{dummy} using $A_{denoise}$ and add it to $D_{reconstructed}$
end for

proposed to counter inference attacks. This includes methods that inject a limited amount of statistical noise into model updates [2, 25], and approaches that learn to perturb data representation [55] such that the reconstructed data from the representation is dissimilar to the raw data, while FL performance is maintained. However, it is unclear if these defenses can provide sufficient protection in the face of more advanced attacks. Further, they introduce extra overhead at the server side. On the other hand, our attack framework only requires a rough estimate of the joint distribution of clients' local data and can tolerate a certain level of inaccuracy in the learned dataset, which provides the attacker with extra flexibility.

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 [61], 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* $\mathcal{P}_p(\mathbf{M})$ *the set of all probability measures on* $\mathbf{M}$ *with finite* 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" and measures the minimum expected distance between two sets 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 [7].

We will also need the following special form of Lipschitz continuity from [7].

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 MDP, 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 [7], which allows us to represent the stochastic transition dynamics of an MDP 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, let F_g be 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 later show that the transition dynamics of our MDP model for attackers is induced by a Lipschitz model class.

Finally we give a formal definition of finite-horizon value functions [60].

Definition 4. *Given an MDP $\mathcal{M}$ 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:*

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

with $V^\pi_{\mathcal{M},H-1}(s) = \mathbb{E}_{a\sim\pi(s)}[r(s,a)]$. 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 probabilities on the value function, we also make use of the following lemmas [7].

Lemma 1. *Given two distributions μ_1 and μ_2 over states S, 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 $K_{d_S,d_\mathbb{R}}(f)$:*

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

Below we state the assumptions needed for establishing Theorem 1. The first assumption models the inaccuracy of distribution learning as well as the heterogeneity of benign workers' local data.

Assumption 1. *$W(\widetilde{P}, \widehat{P}_k) \leqslant \delta$ 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. $\ell(s_2,z_1) \geqslant \ell(s_1,z_1) + \langle\nabla_s\ell(s_1,z_1), s_2 - s_1\rangle + \frac{\alpha}{2}\|s_2 - s_1\|_2^2$ *(strong convexity w.r.t. s);*
4. $\ell(s_2,z_1) \leqslant \ell(s_1,z_1) + \langle\nabla_s\ell(s_1,z_1), s_2 - s_1\rangle + \frac{\beta}{2}\|s_2 - s_1\|_2^2$ *(strong smoothness 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 $\widehat{P}_k$.*

D.2 Measuring the uncertainty: from data distributions to total returns

Let $\mathcal{M} = (S, \boldsymbol{A}, T, r, H)$ denote the true MDP for attacking the federated learning system, and $\widetilde{\mathcal{M}} = (S, \boldsymbol{A}, T', r', H)$ the estimated MDP 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 MDP $\mathcal{M}$ and that derived from the simulated MDP $\widetilde{\mathcal{M}}$. 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 that the M attackers' indexes are from $K - M + 1$ to K. Let $[M] = \{K - M + 1, ..., K\}$ denote the set of attackers and $\epsilon = \frac{K-M}{M}$ 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. With a slight abuse of notation, we assume $S = \Theta$ in the following.

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 MDP $\mathcal{M}$, 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 MDP $\mathcal{M}$. It extends a similar result in [69] to a finite-horizon MDP 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 $W(T, T') := \sup_{a\in\boldsymbol{A}} \sup_{s\in S} W(T(\cdot|s, a), T'(\cdot|s, a))$.

Lemma 3. *Assume Assumption 1 and Assumption 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 switching to $\mathcal{M}$ for l to $H - 1$. That is,

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

By the definition of F_l, we have $\mathcal{J}_M(\pi^*) = F_0$ and $\mathcal{J}_{\widetilde{\mathcal{M}}}(\pi^*) = F_H$, 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

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

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

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

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

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

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 third 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

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

where the second inequality follows from the definition of 1-Wasserstein distance and Assumption 2.1, 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

$$\begin{aligned} &\left|\mathbb{E}_{s'\sim T(s,a)}\left[\frac{1}{K}\sum_{k=1}^{K}\ell_k(s')\right] - \mathbb{E}_{s'\sim T'(s,a)}\left[\frac{1}{K}\sum_{k=1}^{K}\ell'_k(s')\right]\right| \\ \leqslant &\frac{1}{K}\sum_{k=1}^{K}\left|\mathbb{E}_{s'\sim T(s,a)}[\ell_k(s')] - \mathbb{E}_{s'\sim T'(s,a)}[\ell'_k(s')]\right| \end{aligned}$$

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

where the last inequality follows Assumption 1, Assumption 2.1, and the property of 1-Wasserstein distance with respect to product measures. Combining the above results, we have

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

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 $U := H(L_v+L)W(T,T') + 2HL\epsilon\delta$. We have

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

□

As indicated in [69], 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 constants 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:

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

where s denotes the parameters of the current global model, $z=\{z_k\}$ denotes the set of data points sampled by each worker. Note that 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 $\widetilde{P}^{K-M}\prod_{k=K-M+1}^{K}\widehat{P}_k$. This observation allows us to apply the techniques in [7] 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 (1), we have for any $s_1,s_2\in S$,

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

$$\overset{(c)}{\leqslant} \frac{1}{K}\sum_{k=1}^{K-M}\left\|I-\eta\frac{\partial^2\ell(\bar{s},z_k)}{\partial s^2}\right\|_2\|s_1-s_2\|_2$$

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 [47]. It follows that

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

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 [7]. Let $Q^\pi_{\mathcal{M},l}(s,a) := r(s,a)+\sum_{s'\in S}T(s'|s,a)V^\pi_{\mathcal{M},l+1}(s')$ 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:

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

where (a) follows Lemma 2 and (b) is due to the fact that the max operator is 1-Lipschitz, that is, $K_{\|\|_\infty,d_{\mathbb{R}}}(\max(x)) = 1$ [6]. 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 $K^A_{d_S,d_{\mathbb{R}}}(r) \leqslant L + LK^A_{W,W}(T)$. 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 [6], we have $K_{d_S,d_{\mathbb{R}}}(V^*_{\mathcal{M},l}) \leqslant K^A_{d_S,d_{\mathbb{R}}}(Q^*_{\mathcal{M},l}) \leqslant \sum_{t=0}^{H-l-1}(K^A_{W,W}(T))^t(L + LK^A_{W,W}(T))$. We obtain the desired result by applying Lemma 1.

□

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 $\prod_{k=1}^{K}\widehat{P}_k$, while $T'(\cdot|s,a)$ depends on $\widetilde{P}^{K-M}\prod_{k=K-M+1}^{K}\widehat{P}_k$. 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 $z_1 = \{z_{1k}\}_{k=1,\dots,K-M}$ and $z_2 = \{z_{2k}\}_{k=1,\dots,K-M}$ 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 where j_k denotes a coupling between $\widehat{P}_k$ and $\widetilde{P}$. Let $\mathcal{J}$ denote 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

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

$$
\begin{aligned}
&= \inf_{j\in\mathcal{J}} \sum_{(z_1,z_2)} \left\| \frac{1}{K}\sum_{k=1}^{K-M} \nabla_s \ell(s, z_{1k}) - \frac{1}{K}\sum_{k=1}^{K-M} \nabla_s \ell(s, z_{2k}) \right\|_2 \prod_{k=1}^{K-M} j_k(z_{1k}, z_{2k}) \\
&\overset{(c)}{\leqslant} \frac{\eta L_z}{K} \inf_{j\in\mathcal{J}} \sum_{(z_1,z_2)} \sum_{k=1}^{K-M} \|z_{1k} - z_{2k}\|_2 \prod_{k=1}^{K-M} j_k(z_{1k}, z_{2k}) \\
&\overset{(d)}{\leqslant} \frac{\eta L_z}{K} \inf_{j\in\mathcal{J}} \sum_{(z_1,z_2)} \sum_{k=1}^{K-M} \|z_{1k} - z_{2k}\|_2 j_k(z_{1k}, z_{2k}) \\
&\leqslant \frac{\eta L_z}{K} \sum_{k=1}^{K-M} \inf_{j_k} \sum_{(z_{1k},z_{2k})} \|z_{1k} - z_{2k}\|_2 j_k(z_{1k}, z_{2k}) \\
&= \frac{\eta L_z}{K} \sum_{k=1}^{K-M} W(\widehat{P}_k, \widetilde{P}) \\
&\overset{(e)}{\leqslant} \frac{\eta L_z}{K}(K-M)\delta
\end{aligned}
$$

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 1. □

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 MDP $\mathcal{M}$, 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, $|\mathcal{J}_{\mathcal{M}}(\pi^*) - \mathcal{J}_{\mathcal{M}}(\widetilde{\pi}^*)| \leqslant 2H[(L + L_v)W(T, T') + 2L\epsilon\delta]$. From Lemma 6, we have $W(T, T') \leqslant \eta L_z \epsilon\delta$. 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 $K_F \leqslant \epsilon \max\{|1 - \eta\alpha|, |1 - \eta\beta|\}$. □

E Experiments

E.1 Experiment setup

Datasets. We consider four real world datasets: MNIST [34], Fashion-MNIST [64], Balanced EMNIST [17], and CIFAR-10 [33], and a synthetic dataset. Both MNIST and Fashion-MNIST include $60,000$ training examples and $10,000$ testing examples, where each example is a 28×28 grayscale image, associated with a label from 10 classes. Balanced EMNIST includes $112,800$ training examples and $18,800$ testing examples, where each example is a 28×28 grayscale image, associated with a label from 47 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. Details about the synthetic data are given in Appendix E.2. For the *i.i.d.* setting, we randomly split the dataset into K groups, each of which consists of the same number of training samples. For the *non-i.i.d.* setting, we follow the method of [20] to quantify the heterogeneity of local data distribution across clients. Suppose there are C classes in the dataset, e.g., $C = 10$ for the MNIST, Fashion-MNIST, and CIFAR-10 datasets. We evenly split the worker devices into C groups, where each group is assigned $1/C$ of training samples as follows. A training instance with label c is assigned to the c-th group with probability $q \geqslant 1/C$ and to every other group with probability $(1 - q)/(C - 1)$. Within each group,

instances are evenly distributed. A higher q indicates a higher *non-i.i.d.* degree. We set $q = 0.5$ as the default *non-i.i.d.* degree. To demonstrate the power of distribution learning, we assume that the set of attackers share m true data points sampled from the training instances assigned to them. We set $m = 200$ for MNIST and Fashion-MNIST, $m = 500$ for EMNIST, and $m \in \{500, 5000\}$ for CIFAR-10.

Federated learning setting. We adopt the following parameters for the federated learning models: learning rate $\eta = 0.01$ (0.05 for EMNIST and the synthetic data), total number of workers $= 100$, number of attackers $= 20$, subsampling rate $= 10\%$, and number of total epochs $= 1000$. For the MNIST, Fashin-MNIST, and EMNIST datasets, we train a neural network classifier consisting of 8×8, 6×6, and 5×5 convolutional filter layers with ReLU activations followed by a fully connected layer and softmax output. The cross-entropy loss is used to optimize the model. For CIFAR-10, we use the ResNet-18 model [28]. We set the local batch size $B = 128$. We implement the FL model with PyTorch [45] and run all the experiments on the same 2.30GHz Linux machine with 16GB NVIDIA Tesla P100 GPU. We simulate subsampling and local data sampling with different random seeds in each test run. Error bars are reported in Figure 4(c) in the main paper. We set cross-entropy as our default loss function, and stochastic gradient descent (SGD) as our default optimizer.

Baselines. We compare our RL-based attack (RL) with no attack (NA), and the state-of-the-art model poisoning FL attack methods: explicit boosting (EB) [11], inner product manipulation (IPM) [65], and local model poisoning attack (LMP) [20]. The EB attack [11] is originally proposed for the targeted setting. We adapt it to the untargeted setting by using empirical loss as the objective, which is optimized through multi-step gradient ascent using attackers' local data, where the number of steps is 5 and the step size equals to the FL learning rate η. The model update is then boosted by a factor of $\frac{K}{M}$. We compare our RL-based attack with the full knowledge LMP [20], where the attackers have access to not only the aggregation rule but also all normal workers' updates. We use the LMP attack tailored to Krum when the Krum defense is used, and the LMP attack tailored to coordinate-wise median when the clipping median defense or the geometric median defense is used. Further, we implement the adaptive version of LMP introduced in [15], which requires the attackers to know the server's updates derived from its root data, as a baseline against the FLTrust defense [15]. In our implementation of IPM [65], we set the default boosting factor (i.e., ϵ in [65]) as 5.

We consider four representative robust aggregation rules of different types [53]: Krum [12] and geometric median [46], both of which apply client-wise filterings to model updates, coordinate-wise median [67], which adopts a dimension-wise filtering, and FLTrust [15], which requires the server to collect a small training dataset D_0 (called root dataset). In the experiments, we actually consider an extension of the vanilla coordinate-wise median where a norm clipping step [59] is first applied. This gives a more powerful defense as we observed in experiments. We set the default clipping threshold to 2. In geometric median [46], we set the iteration number of the smoothed Weiszfeld algorithm for computing the geometric median [46] to 10 to balance effectiveness and efficiency. In FLTrust, the root data is used to calculate a server model update $g_0 = \frac{1}{|D_0|}\sum_{z \in D_0}[\nabla_\theta \ell(\theta; z)]$ in each epoch. The aggregation weight of each received client' update is then determined through its ReLU-clipped cosine similarity with g_0. Given that the server has no access to the true training data distribution, the root dataset is often biased in practice. We adopt the approach in [15] to model such bias. Among the $|D_0|$ root data samples, a fraction q_0 of them are sampled from a certain class c in the training data, and the rest are sampled from other classes with equal probabilities. For a dataset with C classes, D_0 is unbiased only when $q_0 = 1/C$. We set the size of root dataset $|D_0| = 100$ following [15].

Distribution learning setting. In distribution learning, we set the step size for inverting gradients $\eta' = 0.05$, the total variation parameter $\beta = 0.02$, optimizer as Adam, the number of iterations for inverting gradients $max_iter = 10,000$, and learn the data distribution from scratch. The number of steps for distribution learning is set to $\tau_E = 100$. 32 images are reconstructed (i.e., $B' = 32$) and denoised in each FL epoch. If no attacker is selected in the current epoch, the aggregate gradient estimated from previous model updates is reused for reconstructing data. To build the denoising autoencoder, a Gaussian noise sampled from $0.3\mathcal{N}(0, 1)$ is added to each dimension of images in $D_{reconstructed}$, which are then clipped to the range of [0,1] in each dimension.

Policy learning setting. In policy learning, we implement our simulated environment with OpenAI Gym [14] and adopt OpenAI Stable Baseline3 [48] to implement Twin Delayed DDPG (TD3) [22]

and Proximal Policy Optimization (PPO) [51] algorithms. We find that TD3 gives better results in most cases and report the results for TD3 below. The default parameters are described as follows: the length of simulating environment = $1,000$ epochs, policy learning rate = $1e-7$, the policy model is $MultiInputPolicy$, batch size = 256 and gamma = 1 for updating the target networks.

As described in Section 4.3, we compress the MDP state to include the parameters of the last hidden layer of $\theta^{t(\tau)}$ and the number of attackers sampled, $m^{t(\tau)}$, where each last hidden layer parameter is in $[-\infty, +\infty]$ and $m^{t(\tau)}$ is in $\{0, \ldots, 10\}$. In our experiment, we restrict all attackers to take the same action in each epoch. In solving the local search problem, we fix the number of trajectories $G = 1$ and the size of minibatch $\widetilde{B} = 200$ (except for FLTrust where $\widetilde{B} = 500$).

For the Krum, clipping median, and geometric median defenses, the local search objective is $F(\theta) = \mathbb{E}_{z\sim\widetilde{P}}[\ell(\theta; z)]$ (i.e., $\lambda = 0$). In this case, the action space becomes (γ, E), where $\gamma \in [0, 10]$ and $E \in \{0, \ldots, 20\}$ for the Krum defense, and $\gamma \in [0, 10]$ and $E \in \{0, \ldots, 50\}$ for the clipping median and geometric median defenses. Since TD3 can only be applied to a continuous action space, we consider a continuous interval for E (e.g., $E \in [0, 20]$ for Krum) when updating the policy and round its value to an integer in the feasible range before the action is applied.

For FLTrust, we consider two cases, when the attackers have access to the server's root data D_0 or equivalently, the model update g_0 in each epoch, and when they only know how D_0 is sampled from the true training data distribution. Note that even the former setting is more realistic than the adaptive LMP setting in [15], which also requires access to normal workers' updates. In the former case, we slightly modify the local search method described in Section 4.3 by fixing $\gamma(\theta^{t(\tau)}) = \|g_0(\theta^{t(\tau)})\|_2$ and considering the same local search objective $L(\theta) := (1-\lambda)F(\theta) + \lambda\cos(\theta^{t(\tau)} - \theta, g_0(\theta^{t(\tau)}))$ with the extra constraint that $\|\theta^{t(\tau)} - \theta\|_2 \leqslant \|g_0(\theta^{t(\tau)})\|_2$. This is because FLTrust normalizes all the local model updates using the magnitude of the root update. In the latter case, we use the same objective but approximate $g_0(\theta^{t(\tau)})$ with $\mathbb{E}_{z\overset{q_0}{\sim}\widetilde{P}}[\nabla_\theta \ell(\theta^{t(\tau)}; z)]$, where q_0 models the bias of root data, which is assumed to be known to the attackers. In both cases, the action space is then (E, λ) with $E \in \{0, \ldots, 20\}$ and $\lambda \in [0, 1]$. We further find that when the root data D_0 is known (or can be well approximated), the RL-based attack can be made more efficient by considering an alternate local search objective $L(\theta) := (1-\lambda)F(\theta) - \lambda F_0(\theta)$, where $F_0(\theta) = \frac{1}{|D_0|}\sum_{z\in D_0}[\ell(\theta; z)]$ is the empirical loss associated with the root data. Intuitively, the attackers aim to push the model parameters towards the region that can overfit the root data.

In our experiments, the initial model for all training episodes is set as the first model the attackers received from the actual FL environment. We assume that the server waits for 72 seconds to receive the updates from the workers before performing a model aggregation, which allows $80,000$ total time steps (i.e., 80 episodes) of policy learning for Krum, $40,000$ total time steps (i.e., 40 episodes) of policy learning for clipping median, and $40,000$ total time steps (i.e., 40 episodes) of policy learning for FLTrust within 400 FL epochs in our experiment setting. It is more time consuming to train an RL policy for clipping median and FLTrust because large attack bounds need to be considered. See E.2 for a detailed comparison of the running time of different stages of the RL-based attack under different defense scenarios.

Attack execution setting. Both the distribution learning and policy learning phases in the RL-based attack start at the first FL epoch. The former ends at the 100th FL epoch when RL-based attack starts. All other attacks start at epoch 0. For fair comparisons, we fix all the random seeds for generating the initial model and the root data (for FLTrust), subsampling, and local data sampling when evaluating different attacks. We observe that both EB and RL can occasionally produce NaNs in model updates, which when incorporated by the server, can lead to bad models in all future steps. This produces unrealistic attack scenarios as NaNs can be easily detected by the server. To have a fair comparison with other attacks, we use the built-in VecCheckNan Wrapper in OpenAI Stable Baseline3 [48] to detect abnormal values. We assume that attackers take less ambitious actions (i.e., $(0.5\gamma, E-1)$) in that epoch once they detect a NaN value. If $E = 0$ or $\gamma = 0$, the attackers send $\widetilde{g}^{t(\tau)} = \mathbf{0}$ to the server.

E.2 More experiment results

Attack performance on other datasets. Figures 5 and 6 compare the test accuracy of the global model under different attacks when the server uses Krum or clipping median as the defense for the

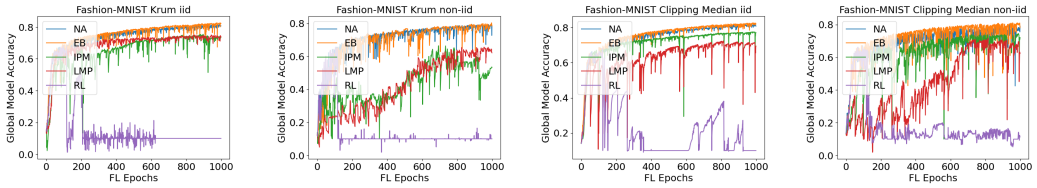

Figure 5: A comparison of global model accuracy on Fashion-MNIST under Krum and clipping median for both *i.i.d.* data and *non-i.i.d.* data. All parameters are set as default.

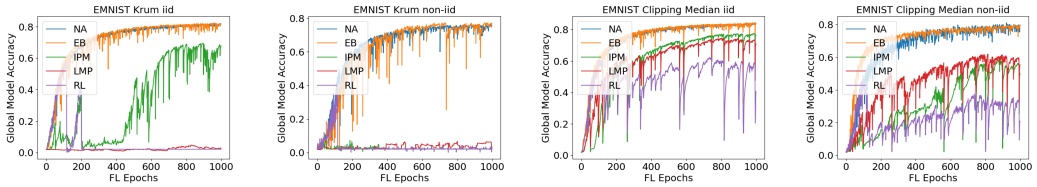

Figure 6: A comparison of global model accuracy on EMNIST under Krum and clipping median for both *i.i.d.* data and *non-i.i.d.* data. All parameters are set as default.

Fashion-MNIST and EMNIST datasets. We consider both *i.i.d.* and *non-i.i.d.* ($q = 0.5$) settings. Our RL-based attack constantly outperforms other baselines by a large margin in all the settings. We observe that in most cases, all attacks are more effective in the *non-i.i.d.* setting. This is mainly because a higher degree of local data heterogeneity increases the variance across normal workers' updates, making it more difficult to filter out adversarial updates. Further, clipping median, which adopts both dimension-wise filtering and client-wise norm clipping to model updates, provides a stronger level of defense than Krum, which only applies client-wise filtering to model updates. In particular, our attack can reduce the model accuracy to an extremely low level under the Krum defense, depending on the number of classes of the dataset used ($\sim$10% for Fashion-MNIST and $\sim$2% for EMNIST).

Figure 7 compares the test accuracy of the global model under different attacks for the CIFAR-10 dataset in the $i.i.d.$ setting. Here we assume that our RL-based attack does not perform distribution learning, and the attackers use their local data to train the attack policy and start to execute attack at epoch 100. This is mainly due to the fact that image reconstruction for CIFAR-10 takes prohibitive amount of time in our experiment environment. Further, state-of-the-art gradient inversion attacks either cannot reconstruct a large batch of images for CIFAR-10 accurately or have not made their code available yet. We consider two cases where 500 and 5,000 local samples are used to train the attack policy, respectively. We observe that in both cases, our approach surpasses all the baselines. In particular, the RL policy trained using only 500 local samples quickly drives the model accuracy to a very low level ($\sim$%9.52) under the Krum defense.

Attack performance under geometric median. We compare the attack performance of RL-based attack and other baselines (i.e., NA, EB, IPM, and LMP) against geometric median [46] on MNIST dataset in the *i.i.d.* setting. As shown in Figure 8(a), RL-based attack and LMP significantly outperform other baselines. Further, although our RL-based attack starts attacking at the 100th epoch, it quickly drives the model accuracy to a very low level, while LMP takes much longer time to achieve similar attack performance.

Attack performance under noisy gradients. We also compare the attack performance of our RL-based attack and other baselines against clipping median aggregation (with the clipping threshold set to 2) under noisy gradients [63]. In particular, the server injects noise into the global model parameters shared with clients, where the noise is sampled from a Laplace distribution [3] (i.e., double exponential distribution) with 0 mean and $1e-4$ exponential decay. We observe that although adding noise indeed decreases the quality of reconstructed images, distribution learning is still effective for the MNIST dataset. Further, our RL-based method still outperforms other baselines in this setting as shown in Figure 8(b).

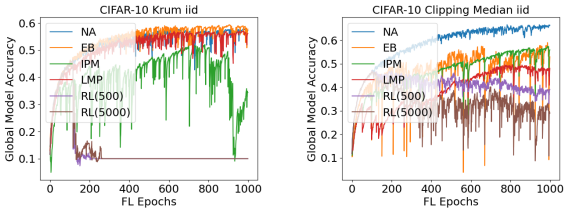

Figure 7: A comparison of global model accuracy on CIFAR-10 under the Krum and clipping median defenses. The RL policy is trained using 500 or 5,000 local samples without distribution learning.

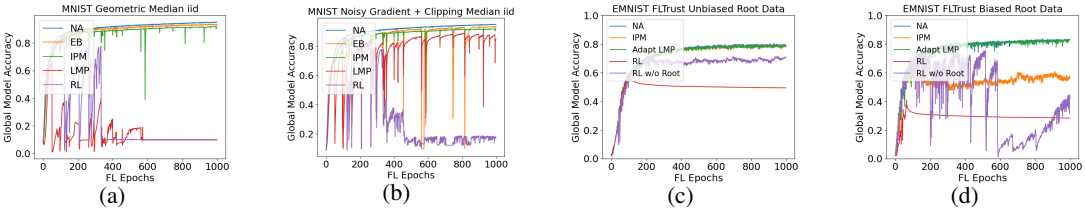

Figure 8: More results on different defenses. (a) Attack performance on MNIST under the geometric median defense. (b) Attack performance on MNIST under the clipping median defense and noisy gradients. (c) and (d) Attack performance on EMNIST under FLTrust defense with unbiased and biased root data.

Attack performance under FLTrust. We compare the attack performance of our RL-based attack with and without access to server's root data (details are given in E.1 policy learning setting) and other baselines (i.e., NA, IPM, and adaptive LMP) against the FLTrust defense on the EMNIST dataset. For RL-based attacks, the attackers use 5,000 local data samples to simulate the environment and skip the distribution learning phase, and start attacking at FL epoch 100. All the baselines start from the beginning of FL. We consider both the cases when the root data are unbiased ($q_0 = 1/47$) and when they are biased against a single class ($q_0 = 0.3$). In the former case, our attack with access to root data leads to a significantly low test accuracy ($\sim 50\%$) as shown in Figure 8(c), while other attacks, including RL-based attack without access to root data, have limited effect against FLTrust. This is due to the fact that when the root data are unbiased and representative of the true training dataset, the root update g_0 in each epoch provides a good estimate of the right direction for model updates, making it difficult to reverse the trend. On the other hand, when the root data is biased, which is likely to happen in practice, the root updates are less representative or even misleading. As shown in Figure 8(d), our RL-based attack with root data access becomes more effective as expected. Our RL-based attack without root data also achieves significant although unstable attack performance. Here we ignore the second term in the local search objective $L(\theta)$ by fixing $\lambda = 0$ to minimize the impact of inaccurate estimate of g_0.

Actual runtime comparison. The actual runtime varies across the FL environment, the training method used, and most importantly, the amount of computational resource available. The tables below report the numbers from our current experiment settings (see Appendix E.1) and the way the simulator is implemented (clients are simulated sequentially in each FL epoch).

For MNIST, Fashion-MNIST, and EMNIST, distribution learning takes around 100 seconds to reconstruct a batch of 32 images and we construct 50 batches within 2 hours. Note that multiple batches can be generated from a single gradient. We start policy training from the beginning of FL training, and we set 8 hours limit for policy training. It takes around 0.05 seconds to simulate a single FL epoch with 10 sampled clients without parallelization. Total training steps vary across defense policies as stated in Appendix E.1.

With the above numbers, if we assume that each FL epoch takes 72 seconds to finish and there are in total of 1000 FL epochs during FL training, then distribution learning will end before the 100th FL epoch and policy training ends by the 400th FL epochs, and the total FL training time is around 20 hours.

Stages	FL Epochs	Real Time
Distribution Learning	100	$\leqslant$ 2 hours
Policy Learning	400	$\leqslant$ 8 hours
Total FL Training	1000	20 hours

Table 1: The running time of each stage in our RL-based attack in terms of FL epochs and real running time for small networks.

Attacks	MNIST	CIFAR-10
IPM	0.25s	2.5s
LMP	7.7s	30s
EB	0.5s	5.5s
RL	5.8s	6s

Table 2: Execution time of various attacks against the clipping median defense for the MNIST and CIFAR-10 datasets.

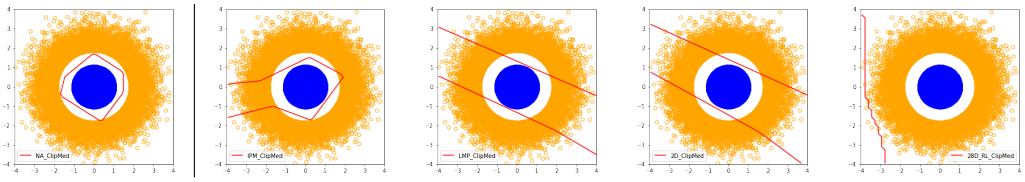

Figure 9: Classification boundaries of the final model on the synthetic data under various attacks and the clipping median defense. The classification accuracy of the final model: 100% (NA), 96.70% (IPM), 89.04% (LMP), 88.04% (RL with 2d actions), and 68.90% (RL with 28-dimensional actions). All parameters are set as default.

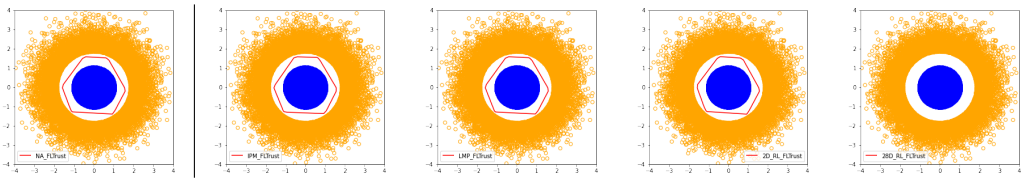

Figure 10: Classification boundaries of the final model on the synthetic data under various attacks and the FLTrust defense. The classification accuracy of the final model: 100% (NA), 100% (IPM), 100% (LMP), 100% (RL with 2d actions), and 68.90% (RL with 28-dimensional actions). All parameters are set as default.

For CIFAR-10, we do not perform distribution learning in this work and policy learning alone takes about 20 hours in our experiment environment as we use a much bigger network (i.e., Resnet-18). However, we expect that once equipped with more powerful devices, the training time can be significantly reduced by parallelly simulating multiple clients using multiprocessing and multiple episodes using vectorized environments, which will make it possible to simulate large FL systems.

In terms of attack executing time, for MNIST with clipping median defense, IPM takes around 0.25 seconds to execute an attack in each FL epoch, LMP takes around 7.7 seconds, EB takes around 0.5 seconds. For CIFAR-10 with clipping median defense, IPM takes around 2.55 seconds to execute an attack in each FL epoch, LMP takes around 30 seconds, EB takes around 5.5 seconds. The execution time of our RL-based method varies over the action space used and it takes around 5.8 seconds and 6 seconds for MNIST and CIFAR-10 respectively with the default action space described in Section 4.3. Given that each FL epoch typically lasts a minute or longer (72 seconds in our experiment), a few seconds of search time is completely acceptable. We observe that for defenses such as Krum, it suffices to use the gradients of the last two layers of model parameters as the action. This approach does not require any online searching and reduces the attack execution time to 0.5s.

Attacks	5% attackers	10% attackers	20% attackers
NA	99.70%	99.02%	99.86%
IPM	99.66%	88.88%	68.96%
EB	99.68%	84.26%	70.06%
LMP	99.68%	89.38%	69.04%
RL	68.90%	68.90%	68.90%

Table 3: Global model accuracy under various attacks and the Krum defense on the synthetic dataset.

Results for the synthetic data. In addition to the four real datasets discussed above, we also consider a two-dimensional synthetic dataset and a small network with 28 model parameters to demonstrate the full potential of our RL-based attack framework (i.e., without state and action compression). We generate the synthetic data based on the method described in [54]. In particular, we generate $55,000$ data instances (including $50,000$ training instances and $5,000$ testing instances), where for each instance $z = (x, y)$, the data $x \in \mathbb{R}^2 \sim \mathcal{N}(\mathbf{0}, I)$ and its label $y = \text{sign}(\|x\|_2) - 2$. Each worker has 500 data instances. We train a multilayer perceptron (MLP) with two hidden layers of size four and two, respectively, and use ReLU as the activation function. For our RL-based attack, we consider both the 2-dimensional action space (γ, E) discussed above as well as the general 28 dimensional action space where the attackers directly decide $\widetilde{g}^{t(\tau)}$ to be sent to the server in each epoch. In both cases, the state space includes the full 28 model parameters and the number of attackers in each epoch. Policy learning takes $8,000$ total time steps (i.e., 8 episodes) to learn the policy, within 10 FL epochs. The attackers use their local data ($10,000$ samples) to build a simulated environment without using distribution learning, and start attacking at epoch 0. We fix all random seeds for a fair comparison across different attacks.

Figure 9 and Figure 10 illustrate the classification boundaries at the end of a federated learning episode for all the attacks when the clipping median defense and the FLTrust defense are applied respectively. The root dataset D_0 for FLTrust is assumed to be known for RL-based attacks. We observe that all the baseline methods and our RL-based attack with 2d actions have limited effect under clipping median and completely fail under FLTrust. On the other hand, the RL-based attack with the full 28-dimensional action space reduces the classification accuracy to 68.90% (worst-case accuracy for the given environment) under both defenses. These results indicate the potential of considering large state and action spaces in our RL-based attack when equipped with more computational power and longer training time.

Table 3 shows how the global model accuracy under different attacks and the Krum defense varies over the number of attackers. The results show that our approach is effective even when the fraction of malicious clients is as low as 5%.