

POWER TO THE CLIENTS: FEDERATED LEARNING IN A DICTATORSHIP SETTING

Mohammadsajad Alipour, Mohammad Mohammadi Amiri

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
{alipom, mamiri}@rpi.edu

ABSTRACT

Federated learning (FL) has emerged as a promising paradigm for decentralized model training, enabling multiple clients to collaboratively learn a shared model without exchanging their local data. However, the decentralized nature of FL also introduces vulnerabilities, as malicious clients can compromise or manipulate the training process. In this work, we introduce **dictator clients**—a novel, well-defined, and analytically tractable class of malicious participants capable of entirely erasing the contributions of all other clients from the server model, while preserving their own. We propose concrete attack strategies that empower such clients and systematically analyze their effects on the learning process. Furthermore, we explore complex scenarios involving multiple dictator clients, including cases where they collaborate, act independently, or form an alliance in order to ultimately betray one another. For each of these settings, we provide a theoretical analysis of their impact on the global model’s convergence. Our theoretical algorithms and findings about the complex scenarios including multiple dictator clients are further supported by empirical evaluations on both computer vision and natural language processing benchmarks.

1 INTRODUCTION

Federated learning (FL) (McMahan et al., 2017) is a distributed learning paradigm in which model training is performed collaboratively by a set of clients. In centralized FL, a global server broadcasts the current model to all clients, each of which updates the model using its local dataset and sends back the resulting gradients to the server. The server then aggregates these gradients to update the global model. This approach accelerates training by distributing computation across multiple machines, while also enhancing data privacy since clients share only gradients, not their raw data. FL is especially well-suited for privacy-sensitive applications, such as training on confidential medical records across hospitals.

Despite its advantages, FL remains vulnerable to malicious behavior by the participating clients. *Byzantine clients* are adversarial participants that disrupt the training process by sending arbitrary or manipulated updates to the central server (Lamport et al., 2019)(Blanchard et al., 2017). The presence of such adversaries can significantly degrade model performance, making Byzantine robustness a critical area of study (Li et al., 2019; Wu et al., 2020; Shejwalkar & Houmansadr, 2021; Guerraoui et al., 2018; Xie et al., 2018; Xie, 2022). Moreover, several studies have demonstrated the possibility of backdoor attacks in FL via collusion attacks where multiple malicious clients coordinate their actions to inject hidden triggers into the global model in FL (Liu et al., 2024; Ranjan et al., 2022; Xiao et al., 2022; Bagdasaryan et al., 2020). These clients may exchange information and strategically craft updates that steer the aggregated model toward a compromised state.

However, the majority of existing literature primarily focuses on defending against Byzantine clients, while comparatively little attention has been given to characterizing specific and well-defined behaviors of Byzantine clients that have a different specific goal—especially in exploring diverse scenarios involving their presence within the system. In FL, a malicious client may aim to impose the statistical properties or specific patterns of its own dataset onto the global model. Such a client

effectively attempts to *dictate* the final model by aligning it more closely with its local data distribution. This behavior may serve various objectives, such as improving performance on a target task, biasing global model’s decisions toward a desired objective, embedding backdoors, or degrading the model’s generalization on other clients’ data. By exploiting vulnerabilities in the model aggregation process, especially when contributions are blindly averaged or insufficiently audited, a malicious client can steer the training dynamics to serve its own objectives, ultimately dominating the global model’s behavior.

In this work, we introduce a novel and formally defined class of Byzantine clients in FL, characterized by precise assumptions about their knowledge of the system and limitations. In contrast to prior studies, which often assumed omniscient or overly powerful adversaries, we consider malicious clients with only minimal communication capabilities among themselves. These clients lack visibility into the internal structure of the global model and have no information about the data or updates of benign clients. By clearly bounding their capabilities, our framework offers a more realistic and fine-grained understanding of adversarial behavior in practical FL environments.

The goal of these malicious clients is to preserve their own influence on the final global model while entirely eliminating the contributions of all other participants—as if the benign clients had never been involved in the training process. We refer to such independent malicious clients as *dictator* clients due to their unilateral domination of the model. When multiple such clients coordinate via their limited communication link to jointly dominate training, we refer to them as *collaborative dictator* clients. We show that these clients do not require any privileged access to the server or any external metadata—making their attack strategies particularly concerning from a security perspective.

To demonstrate the feasibility of this threat, we develop a series of algorithms that enable malicious clients to achieve their goals within the defined constraints. Our theoretical findings are further supported by empirical results, which validate the effectiveness of the proposed attack strategies. Beyond isolated attacks, we also investigate complex and previously underexamined dynamics that arise among malicious clients themselves. For example, we examine scenarios in which all participants in the system act as dictators, as well as cases where collaborative dictator clients can betray one another within their own partnership. These scenarios reveal internal conflicts among adversaries and broaden the understanding of multi-agent adversarial behavior in FL.

2 RELATED WORK

The distributed nature of FL, combined with the server’s limited visibility into local training processes, makes it vulnerable to various security threats posed by malicious or compromised clients (Zhang et al., 2023b). In this section, we review existing literature across three major category of attacks—Byzantine attacks, backdoor attacks, and collusion attacks.

BYZANTINE ATTACKS

Byzantine attacks pose a fundamental threat in distributed systems including FL, where a subset of clients, known as *Byzantine clients*, arbitrarily deviate from the prescribed protocol by submitting malicious or anomalous updates to the central server (Lamport et al., 2019). The goals of such attacks typically include degrading the global model’s performance or preventing convergence (Blanchard et al., 2017). Attack strategies vary in complexity, ranging from simple approaches such as random noise injection or submitting zero gradients to more sophisticated methods like sign-flipping (Samy & Girdzijauskas, 2023; Shen et al., 2025). Advanced attacks are often crafted to evade specific defenses, making them challenging to detect and mitigate (Shejwalkar & Houmansadr, 2021; Baruch et al., 2019).

BACKDOOR ATTACKS

Backdoor attacks (also known as Trojan attacks) are a more insidious threat in FL where attackers aim to embed hidden malicious behavior into the global model (Gu et al., 2017; Li et al., 2022). An attacker, typically controlling one or more clients, manipulates their local dataset or model updates to create a “backdoor trigger”—a specific pattern or feature (e.g., a small patch in an image, a specific

phrase in text). The compromised global model performs normally on clean inputs but exhibits attacker-chosen behavior, such as misclassification, when the trigger is present. These attacks can be implemented through various strategies, including **data poisoning**, where labels are manipulated for samples containing the trigger, and **model poisoning**, where malicious updates are directly crafted to influence model behavior (Bagdasaryan et al., 2020; Xie et al., 2020). Triggers may be static and predefined (Bagdasaryan et al., 2020) or dynamically generated using optimization techniques to make them more subtle and difficult to detect (Zhang et al., 2023a). Comprehensive surveys on backdoor attacks and defenses in FL can be found in Nguyen et al. (2024).

COLLUSION ATTACKS

Collusion attacks occur when multiple malicious clients coordinate their actions to enhance the effectiveness of the attacks or bypass defenses designed for independent attackers. Colluding attackers can amplify the impact of Byzantine or backdoor attacks. For example, multiple Byzantine clients might coordinate their updates to overwhelm Byzantine-resilient aggregation rules that assume the number of attackers are limited (Xie et al., 2020). Similarly, colluding clients can implement distributed backdoor attacks, where each attacker contributes a part of the malicious update, making individual contributions appear benign while collectively embedding a backdoor into the global model (Lyu et al., 2023). More advanced and specific collusion strategies include *alternating attacks* and *stealthy collusion*. In alternating (on-off) attacks, malicious clients alternate between benign and malicious behavior to build reputation or evade history-based detection (Lewis et al., 2023). In stealthy collusion attacks, attackers coordinate to make their cumulative malicious impact significant while keeping individual updates close to benign ones to evade detection (Lyu et al., 2025). Such attacks aim for sparsity and stealthiness.

While prior research has primarily focused on degrading model utility or embedding backdoors, our work introduces and formalizes a new adversarial paradigm: *dictator clients*—malicious participants whose goal is not to harm performance but to fully preserve their own contribution to the global model while completely erasing the influence of other clients. Unlike traditional Byzantine or backdoor attacks, dictator clients aim to *bias* the learning outcome toward their local objectives without necessarily compromising overall model accuracy. Moreover, we investigate nuanced interaction dynamics among multiple dictator clients, including collaboration, conflict, and strategic deception. To the best of our knowledge, this is the first systematic exploration of such influence-preserving and interaction-aware attacks, revealing a novel and underexplored threat model in FL.

3 PROBLEM FORMULATION AND PRELIMINARIES

We consider a centralized FL setting in which, during each communication round, a central server broadcasts the current model weights to all clients. Each client then performs stochastic gradient descent on the loss function on its local dataset to compute an update. These local updates are sent back to the server, which aggregates them—most commonly through simple averaging—and applies a global gradient descent step scaled by a predefined learning rate. To enable a more precise formulation and analysis of the attacks, we assume that the server aggregates updates from all clients in every round—an assumption that commonly holds in cross-silo FL settings (Huang et al., 2024). We defer to future work the exploration of FL variants that either allow partial client participation or permit clients to perform several local updates before aggregation.

Let θ_t denote the global model weights maintained by the server at iteration t , and let $\mathcal{N} = \{1, 2, \dots, N\}$ represent the set of N participating clients. For each client $n \in \mathcal{N}$, let $\nabla \mathcal{L}_n(\theta_t)$ denote the gradient of its local loss function with respect to the current model θ_t . The server updates the global model at each round after collecting the gradients from all clients as follows:

$$\theta_{t+1} = \theta_t - \eta \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_t), \quad (1)$$

where $\eta > 0$ denotes the server-side learning rate. The global model is initialized as θ_0 at the server and distributed to all clients at the beginning of training.

We further define a hypothetical baseline scenario where only a single client $m \in \mathcal{N}$ participates in the learning process. Let $\hat{\theta}_t^m$ denote the model weights at iteration t in this single-client scenario.

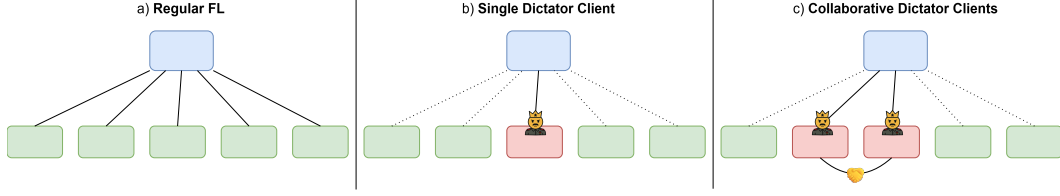


Figure 1: Regular FL compared to scenarios where one dictator client or collaborative dictator clients try to remove other clients from the training procedure.

The corresponding update rule simplifies to:

$$\hat{\theta}_{t+1}^m = \hat{\theta}_t^m - \eta \nabla \mathcal{L}_m(\hat{\theta}_t^m), \quad (2)$$

with initialization $\hat{\theta}_0^m = \theta_0$. We further generalize this formulation to a subset of clients. Let $\mathcal{P} \subset \mathcal{N}$ denote a subset of P clients, where $1 < P < N$. We define $\hat{\theta}_t^{\mathcal{P}}$ as the model weights at iteration t when only clients in $\mathcal{P}$ participate in training. The update rule for this partial participation scenario is given by:

$$\hat{\theta}_{t+1}^{\mathcal{P}} = \hat{\theta}_t^{\mathcal{P}} - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_t^{\mathcal{P}}), \quad (3)$$

with initialization $\hat{\theta}_0^{\mathcal{P}} = \theta_0$. Next, we introduce scenarios involving dictator clients in FL, including both single-dictator and multi-dictator cases. We describe how these clients modify their local updates to achieve their objectives. Specifically, a single dictator client aims to steer the global model’s updates and convergence to follow Eq. 2, while a group of coordinated dictator clients (collaborative dictators) seeks to enforce convergence toward Eq. 3, effectively overriding the standard FL update rule in Eq. 1.

4 DICTATOR CLIENT SCENARIOS

In this section, we propose algorithms that enable clients to become dictators—retaining their own contributions to the global model while eliminating those of others. We begin with the case of a single dictator client in Section 4.1 and then extend to scenarios involving multiple collaborating dictator clients in Section 4.2. Figure 1 illustrates different dictator client scenarios compared with standard FL.

4.1 SINGLE DICTATOR CLIENT

In this section, we demonstrate how a single dictator client can craft its updates to entirely nullify the contributions of all other clients while preserving its own influence on the global model. We assume that the dictator client knows only the server’s learning rate and requires no additional information. Notably, as shown in Appendix C, even this assumption can be relaxed, as the learning rate can be numerically estimated after a single iteration. Suppose client $m \in \mathcal{N}$ such that $1 \leq m \leq N$ is the designated dictator client and only knows server’s learning rate η . At iteration 0, the server broadcasts the initial model θ_0 to all clients, which each use to compute their local gradients. Upon receiving these gradients, the server updates the global model as $\theta_1 = \theta_0 - \eta \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0)$. In the next iteration, the server broadcasts θ_1 to all clients. Each client except client m , computes and sends their gradient with respect to θ_1 . Meanwhile, client m retains a local copy of the initial server model θ_0 from the previous iteration. Using this, it computes a hypothetical model update, denoted by $\hat{\theta}_1^m$, which represents the model that would have resulted if only client m ’s gradient had been used in the first iteration. This is computed as:

$$\hat{\theta}_1^m = \theta_0 - \eta \nabla \mathcal{L}_m(\theta_0). \quad (4)$$

The dictator client m sends a carefully crafted update M_1 instead of its actual gradient $\nabla \mathcal{L}_m(\theta_1)$ to delete the contribution of all other clients from the previous iteration and preserve only its own contribution. This manipulated update is defined as $M_1 = \nabla \mathcal{L}_m(\hat{\theta}_1^m) - \left(\frac{\theta_0 - \theta_1}{\eta} - \nabla \mathcal{L}_m(\theta_0) \right)$. Here,

the term $\frac{\theta_0 - \theta_1}{\eta}$ reconstructs the aggregate gradient used by the server in the first round, allowing client m to effectively cancel out the influence of all other clients while steering the update toward its own objective. We now analyze the updated global model θ_2 after the server aggregates all client updates in the second iteration:

$$\begin{aligned}\theta_2 &= \theta_1 - \eta \left(M_1 + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_1) \right) \\ &= \theta_1 - \eta (\nabla \mathcal{L}_m(\hat{\theta}_1^m) - (\frac{\theta_0 - \theta_1}{\eta} - \nabla \mathcal{L}_m(\theta_0)) + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_1)) \\ &= \theta_0 - \eta \nabla \mathcal{L}_m(\theta_0) - \eta (\nabla \mathcal{L}_m(\hat{\theta}_1^m) + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_1)).\end{aligned}$$

Now, substituting from Eq. 4, we can express θ_2 as:

$$\theta_2 = \hat{\theta}_1^m - \eta \left(\nabla \mathcal{L}_m(\hat{\theta}_1^m) + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_1) \right).$$

This demonstrates that by sending the carefully crafted update M_1 , client m effectively nullifies the contributions of all other clients from the previous iteration while retaining its own gradient contribution. In doing so, the server's model state is steered toward the single-client trajectory $\hat{\theta}_1^m$ instead of the standard FL update. To generalize this strategy for any round t , client m maintains a local model $\hat{\theta}_t^m$, which is updated independently as:

$$\hat{\theta}_t^m = \hat{\theta}_{t-1}^m - \eta \nabla \mathcal{L}_m(\hat{\theta}_{t-1}^m). \quad (5)$$

We define M_t as the update that the dictator client m sends to the server at iteration t :

$$M_t = \nabla \mathcal{L}_m(\hat{\theta}_t^m) - \left(\frac{\hat{\theta}_{t-1}^m - \theta_t}{\eta} - \nabla \mathcal{L}_m(\hat{\theta}_{t-1}^m) \right). \quad (6)$$

Now, we analyze the server's model update at iteration $t + 1$ after aggregating all client updates:

$$\begin{aligned}\theta_{t+1} &= \theta_t - \eta \left(M_t + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_t) \right) \\ &= \theta_t - \eta (\nabla \mathcal{L}_m(\hat{\theta}_t^m) - (\frac{\hat{\theta}_{t-1}^m - \theta_t}{\eta} - \nabla \mathcal{L}_m(\hat{\theta}_{t-1}^m)) + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_t)) \\ &= \hat{\theta}_{t-1}^m - \eta \nabla \mathcal{L}_m(\hat{\theta}_{t-1}^m) - \eta (\nabla \mathcal{L}_m(\hat{\theta}_t^m) + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_t)).\end{aligned}$$

Using Eq. 5, it follows that $\theta_{t+1} = \hat{\theta}_t^m - \eta \left(\nabla \mathcal{L}_m(\hat{\theta}_t^m) + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_t) \right)$.

After T rounds of training, the final model weights θ^* will be:

$$\theta^* = \hat{\theta}_T^m - \eta (\nabla \mathcal{L}_m(\hat{\theta}_T^m) + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_T)) \quad (7)$$

$$\begin{aligned}&= \hat{\theta}_T^m - \eta \nabla \mathcal{L}_m(\hat{\theta}_T^m) - \eta \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_T) \\ &= \hat{\theta}_{T+1}^m - \eta \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_T) \approx \hat{\theta}_{T+1}^m.\end{aligned} \quad (8)$$

This final expression shows that the dictator client drives the model toward its own trajectory $\hat{\theta}_{T+1}^m$, effectively overriding the influence of other clients up to a residual term. As shown in Eq. 8, the exact final weights under our method are given by $\theta^* = \hat{\theta}_{T+1}^m - \eta \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_T)$, where $\hat{\theta}_{T+1}^m$ represents the weights for the final iteration if only the dictator client m had participated in training, as if no other client existed. The residual term $\eta \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_T)$ captures the contributions from the other clients in the final iteration. In practice, this term is negligible, as it stems from a single round of updates and has minimal impact on the final model—especially when the model produced by the dictator client is robust to such perturbations. Our empirical results in Section 6 further support the insignificance of this residual term on the dictator client's objective. Algorithm 1 outlines the complete procedure for a client to act as a dictator.

Algorithm 1 Single dictator client m

```

1: Require: Initialized weights  $\theta_0$ , learning rate  $\eta$ 
2: for iteration  $t = 0$  to  $T$  do
3:   if  $t = 0$  then
4:     Send  $M_0 = \nabla \mathcal{L}_m(\theta_0)$  as update
5:     Create a local copy of  $\theta_0$  as  $\hat{\theta}_0^m = \theta_0$ 
6:     Update local model:  $\hat{\theta}_1^m = \theta_0 - \eta \nabla \mathcal{L}_m(\theta_0)$ 
7:   else
8:      $M_t = \nabla \mathcal{L}_m(\hat{\theta}_t^m) - \left( \frac{\hat{\theta}_{t-1}^m - \theta_t}{\eta} - \nabla \mathcal{L}_m(\hat{\theta}_{t-1}^m) \right)$ 
9:     Send  $M_t$  as update
10:    Update local model:  $\hat{\theta}_{t+1}^m = \hat{\theta}_t^m - \eta \nabla \mathcal{L}_m(\hat{\theta}_t^m)$ 
11:   end if
12: end for

```

Algorithm 2 Collaborative Dictator clients $k \in \mathcal{P}$

```

1: Require: Initialized weights  $\theta_0$ , learning rate  $\eta$ , Communication link between  $P$  collaborative dictator clients
2: for iteration  $t = 0$  to  $T$  do
3:   if  $t = 0$  then
4:     Send  $M_0^k = \nabla \mathcal{L}_k(\theta_0)$  as update
5:     Share  $\nabla \mathcal{L}_k(\theta_0)$  with other dictator partners
6:     Create a local copy of  $\theta_0$  as  $\hat{\theta}_0^P = \theta_0$ 
7:     Update local model:  $\hat{\theta}_1^P = \theta_0 - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\theta_0)$ 
8:   else
9:      $M_t^k = \nabla \mathcal{L}_k(\hat{\theta}_t^P) - \left( \frac{\hat{\theta}_{t-1}^P - \theta_t}{P\eta} - \nabla \mathcal{L}_k(\hat{\theta}_{t-1}^P) \right)$ 
10:    Send  $M_t^k$  as update
11:    Share  $\nabla \mathcal{L}_k(\hat{\theta}_t^P)$  with other dictator partners
12:    Update local model:  $\hat{\theta}_{t+1}^P = \hat{\theta}_t^P - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_t^P)$ 
13:   end if
14: end for

```

4.2 COLLABORATIVE DICTATOR CLIENTS

In this section, we extend the single dictator client scenario to a group of P dictator clients acting in coordination. As illustrated in Figure 1(c), these clients collaborate to suppress the influence of all others while preserving their own contributions, relying only on inter-client communication. As discussed in Appendix C, they do not require prior knowledge of the server’s learning rate—it can be accurately estimated after a single training round. Let $\mathcal{P} \subset \mathcal{N}$ denote the set of P collaborating dictator clients, where $1 < P < N$, capable of communicating with each other. These clients coordinate their updates according to Algorithm 2 so that the global model evolves as if only they had participated in training. Each client in $\mathcal{P}$ maintains a synchronized local model, denoted as $\hat{\theta}_t^P$, representing the model state at iteration t under their exclusive contributions from the start. At each round, every dictator client $k \in \mathcal{P}$ submits the following crafted update to the server, effectively nullifying the impact of the remaining $N - P$ clients in $\mathcal{N} \setminus \mathcal{P}$:

$$M_t^k = \nabla \mathcal{L}_k(\hat{\theta}_t^P) - \left(\frac{\hat{\theta}_{t-1}^P - \theta_t}{P\eta} - \nabla \mathcal{L}_k(\hat{\theta}_{t-1}^P) \right), \forall k \in \mathcal{P}.$$

Afterwards, the clients exchange gradients to jointly compute the next local model state, $\hat{\theta}_{t+1}^P$. As long as all P clients in $\mathcal{P}$ follow this protocol and continues to share gradients for updating the collective local model, the global model will converge as if only the clients in $\mathcal{P}$ had trained it. A formal proof of this outcome is provided in Appendix D. Next, we turn our attention to more intricate interactions that emerge in FL systems with the presence of such dictator clients.

5 COMPETITION AND COLLUSION AMONG DICTATOR CLIENTS

In this section, we explore the nuanced interactions that can arise among dictator clients in FL systems. We begin by examining a competitive setting where every participating client independently aims to become the sole dictator and dominate the global model—an extreme yet insightful scenario that is discussed in Section 5.1. Furthermore, in Section 5.2, we explore a more collaborative dynamic, where multiple dictator clients form alliances. We investigate whether such cooperation is inherently stable or if, ultimately, some clients can strategically betray their collaborators to gain a greater influence over the model.

5.1 MUTUAL DOMINATION: WHEN EVERY CLIENT SEEKS CONTROL

Here, we explore the scenario where all clients independently act as dictators, each attempting to retain only its own contribution while nullifying the influence of others. In other words, each client follows the update strategy outlined in Algorithm 1. In practice, such behavior leads to a catastrophic failure of learning, with the global model failing to converge and the loss growing exponentially. We analyze the underlying reason behind this phenomenon in what follows. At iteration 0, the server sends the initialized weights θ_0 to all clients. Each client then computes its local gradient, and the server aggregates these to update the global model as $\theta_1 = \theta_0 - \eta \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0)$. In the next iteration, the server broadcasts θ_1 to all clients. Now, each client attempts to simulate what the global model would have been if it alone had contributed to the update. For each client $n \in \mathcal{N}$, we define $\hat{\theta}_1^n$ as the hypothetical global model after iteration 0 only if client n had participated. Using this, each client computes its malicious update M_1^n , as defined in Section 4.1 as $M_1^n = \nabla \mathcal{L}_n(\hat{\theta}_1^n) - \left(\frac{\theta_0 - \theta_1}{\eta} - \nabla \mathcal{L}_n(\theta_0) \right)$. Now, we analyze the updated global model θ_2 after the server aggregates the updates from all clients in the second iteration:

$$\begin{aligned} \theta_2 &= \theta_1 - \eta \sum_{n=1}^N M_1^n = \theta_1 - \eta \left(\sum_{n=1}^N \nabla \mathcal{L}_n(\hat{\theta}_1^n) - \left(\frac{\theta_0 - \theta_1}{\eta} - \nabla \mathcal{L}_n(\theta_0) \right) \right) \\ &= \theta_1 - \eta \left(\sum_{n=1}^N \nabla \mathcal{L}_n(\hat{\theta}_1^n) - \frac{N(\theta_0 - \theta_1)}{\eta} + \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0) \right) \\ &= \theta_1 - \eta \left(\sum_{n=1}^N \nabla \mathcal{L}_n(\hat{\theta}_1^n) - (N-1) \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0) \right) \end{aligned} \quad (9)$$

$$\begin{aligned} &= \theta_1 + \eta(N-1) \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0) - \eta \sum_{n=1}^N \nabla \mathcal{L}_n(\hat{\theta}_1^n) \\ &= \theta_0 - \eta \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0) + \eta(N-1) \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0) - \eta \sum_{n=1}^N \nabla \mathcal{L}_n(\hat{\theta}_1^n) \\ &= \theta_0 + \eta(N-2) \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0) - \eta \sum_{n=1}^N \nabla \mathcal{L}_n(\hat{\theta}_1^n). \end{aligned} \quad (10)$$

Since $N - 2 > 0$ assuming that we have more than 2 clients in the system, and the learning rate η is a positive real number, it follows that $\eta(N - 2) > 0$. Consequently, from Eq. 10, it follows that when all clients act as independent dictators and send the defined malicious update, the resulting model update effectively moves in the *opposite* direction of intended gradient. In other words, the updating procedure resembles *gradient ascent* instead of *gradient descent*, and thereby increasing the loss rather than minimizing it. This behavior causes the model to “unlearn” the progress made in previous iteration. Therefore, when every client behaves as an independent dictator, the global model fails to learn meaningful representations and make no effective progress. It unlearns the knowledge acquired in the previous iteration. Therefore, in the scenario where every client is an independent dictator, the global model will learn almost nothing. Our empirical results, presented in Section 6.3, confirms this breakdown in learning in practice.

5.2 BETRAYAL IN COLLABORATION: STRATEGIC CHEATING AMONG DICTATOR CLIENTS

Here, we show that even *collaborative dictators*—those collaborating to erase other participants’ contributions—may ultimately betray one another. For simplicity, we focus on a setting where the set of collaborative dictator clients is $\mathcal{P} = \{1, 2\}$. As illustrated in Figure 2, we consider the case where dictator client 1, decides to cheat its partner, client 2, after a specific iteration E . While both

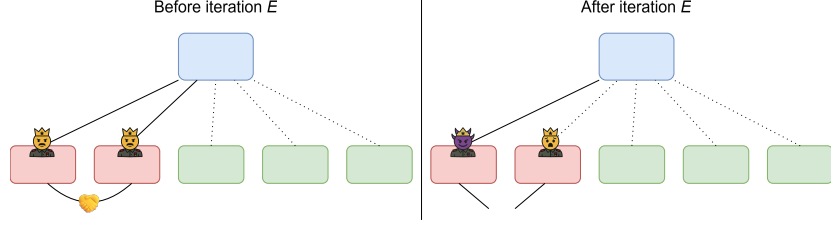


Figure 2: Client 1 and 2 collaborate as dictators until iteration E , when client 1 betrays.

clients initially cooperate using Algorithm 2 to jointly eliminate the influence of all other clients, we introduce Algorithm 3 (in Appendix B), which enables client 1 to unilaterally eliminate client 2’s contribution as well, effectively taking full control of the model.

Prior to iteration E , client 1 shares its gradients with client 2, contributing jointly to a local model $\hat{\theta}_t^P$. However, simultaneously, client 1 secretly maintains a private model $\hat{\theta}_t^1$, which simulates the evolution of the global model if only client 1 participated in training. At each iteration, client 1 computes a correction term $\Delta_t = \nabla \mathcal{L}_1(\hat{\theta}_t^m) - (\nabla \mathcal{L}_1(\hat{\theta}_t^P) + \nabla \mathcal{L}_2(\hat{\theta}_t^P))$, which captures the discrepancy between acting alone and collaborating. These differences are accumulated into a cheating offset, denoted as `Cheating_Update`. At iteration E , client 1 sends this accumulated update to the server in place of the expected collaborative update. This forces the server to jump to a state equivalent to one where if only client 1 had participated throughout the training process—effectively eliminating the contribution of client 2, despite their prior collaboration, as well as the benign clients’ influence. A formal proof of this strategy is provided in Appendix E; our empirical results in Section 6.4 confirm the effectiveness of this betrayal strategy in practice.

6 EXPERIMENTS

In this section, we empirically evaluate the effectiveness of our proposed attack algorithms across both computer vision and natural language processing (NLP) tasks. For our main experiments, we focus on image classification using the MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky et al., 2009) datasets with a simple convolutional neural network (CNN) as the global model. To maintain consistency with our theoretical framework, all experiments are conducted using stochastic gradient descent (SGD) as the optimizer. We simulate a FL environment with five clients, each assigned a disjoint and non-overlapping subset of the training data to create a highly not independent and identically distributed (non-IID) setting. Specifically, the training set is partitioned such that client 1 receives samples with labels 0 and 1, client 2 with labels 2 and 3, and so on, ensuring that each client maintains only two unique classes. Additional results for NLP tasks are provided in Appendix F.

6.1 SINGLE DICTATOR CLIENT

Table 1 reports the resulting classification accuracies across all clients’ datasets. We begin by evaluating the scenario in which a single client attempts to dominate the global model, following the attack strategy defined in Algorithm 1. As shown, the global model entirely fails to learn from the data of non-dictator clients, achieving a striking 0.00% accuracy on their datasets. In contrast, the model maintains high accuracy on the dictator client’s local dataset, confirming that the attack successfully isolates and preserves only the dictator’s contribution. These results empirically validate the feasibility and effectiveness of the proposed single-client dictatorship attack algorithm described in Section 4.1. Furthermore, Figure 3 illustrates this effect by showing the global model’s loss on each client’s dataset under two settings: regular FL and the case where client 3 becomes dictator. In regular FL, losses decrease across all clients, whereas under dictatorship by client 3, only the loss corresponding to client 3’s dataset is minimized, while losses for all other clients worsen over time. This confirms that the dictator client successfully minimizes its own local loss while significantly impeding the global model’s ability to learn from the data of the remaining clients.

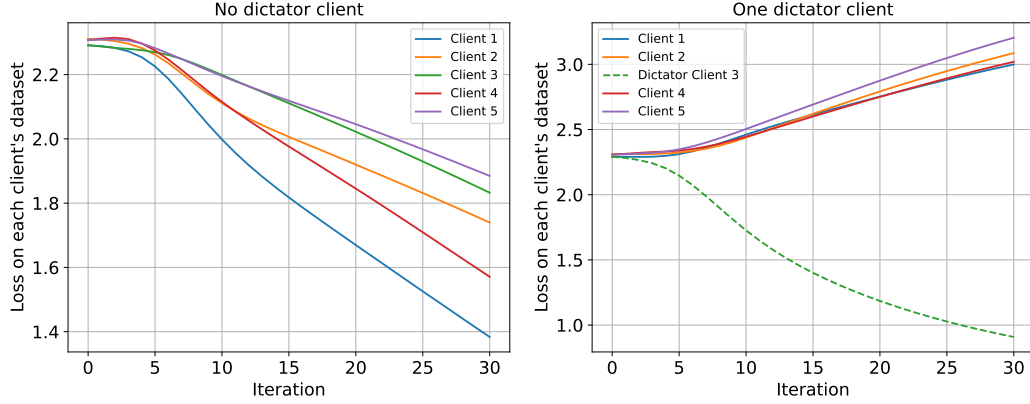


Figure 3: Loss function on each client’s dataset, comparing scenarios with no dictator clients and with one dictator client where in this figure client 3 is the dictator client.

Method	MNIST					CIFAR-10				
	[0,1]	[2,3]	[4,5]	[6,7]	[8,9]	[0,1]	[2,3]	[4,5]	[6,7]	[8,9]
Regular FL	96.18	79.25	66.84	88.12	66.38	39.04	12.51	31.07	23.74	52.59
Dictator client: 1	99.63	0.00	0.00	0.00	0.00	73.65	0.00	0.00	0.00	0.00
Dictator client: 2	0.00	93.92	0.00	0.00	0.00	0.00	65.19	0.00	0.00	0.00
Dictator client: 3	0.00	0.00	97.43	0.00	0.00	0.00	0.00	66.51	0.00	0.00
Dictator client: 4	0.00	0.00	0.00	98.91	0.00	0.00	0.00	0.00	73.98	0.00
Dictator client: 5	0.00	0.00	0.00	0.00	94.42	0.00	0.00	0.00	0.00	77.06

Table 1: Performance of the global model on each local dataset for MNIST and CIFAR-10 datasets and the single dictator client scenario.

6.2 COLLABORATIVE DICTATOR CLIENTS

We next examine the impact of coordinated attacks involving multiple dictator clients. In this setting, two or three clients jointly follow the attack strategy, described in Algorithm 2, aiming to eliminate the influence of all other participants. Table 2 and Figure 4 summarize the outcomes. The results demonstrate that the collaborating dictator clients succeed in entirely erasing the influence of the benign clients, leading the global model to achieve 0.00% accuracy on their data. Simultaneously, the global model maintains high accuracy on the data held by the collaborative dictators, indicating that it has effectively converged to a model tailored solely to their objectives. These findings further reinforce the practicality and scalability of our proposed attack strategy in multi-attacker scenarios. The coordinated behavior among the dictator clients allows them to dominate the training process, ensuring that the global model exclusively reflects their data distributions while ignoring the contributions of the remaining benign participants. The success of this attack highlights the vulnerability of FL even when malicious clients are in minority, provided they act in collaboration.

6.3 MUTUAL DOMINATION

We now consider the extreme scenario where every client behaves as an independent dictator, each executing Algorithm 1 to preserve only its own contribution while nullifying the effects of all others. As established theoretically in Section 5.1, this adversarial configuration results in mutually destructive behavior, where no single client’s update can effectively influence the global model without being canceled out by others, resulting in a destructive equilibrium where no useful learning can occur. The empirical results, shown in Figure 5, strongly corroborate this. The global model fails to make progress on any client’s data; instead of converging, the loss increases rapidly and consistently across all datasets. This behavior aligns with the theoretical finding that the aggregated updates approximate a form of gradient ascent, undoing prior learning and leading to model divergence. This experiment underscores a key insight: when all clients prioritize their own influence at the expense

Method	MNIST					CIFAR-10				
	[0,1]	[2,3]	[4,5]	[6,7]	[8,9]	[0,1]	[2,3]	[4,5]	[6,7]	[8,9]
Regular FL	96.18	79.25	66.84	88.12	66.38	39.04	12.51	31.07	23.74	52.59
Dictator clients: 2,3	0.00	88.19	87.80	0.00	0.00	0.00	35.08	43.17	0.00	0.00
Dictator clients: 2,3,4	0.00	84.87	80.22	94.19	0.00	0.00	18.38	40.02	46.05	0.00

Table 2: Performance of the global model on each local dataset for MNIST and CIFAR-10 datasets and the collaborative dictator clients scenario.

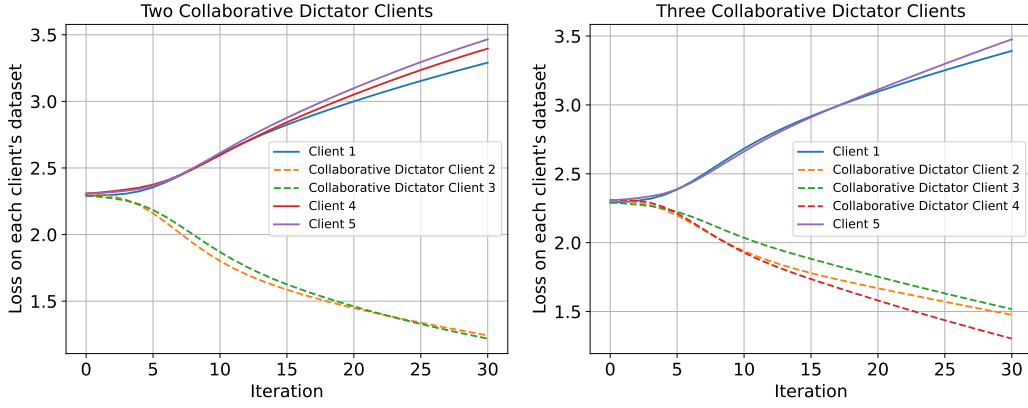


Figure 4: Loss function on each client’s dataset, when two clients become collaborative dictators (left) and three clients become collaborative dictators (right).

of others, the entire system collapses. FL becomes ineffective, highlighting the need for defenses against not only isolated attackers but also adversarial groups.

6.4 BETRAYAL IN COLLABORATION

In this experiment, we evaluate a scenario in which two clients, client 1 and client 2, initially act as collaborative dictators. While both begin by coordinating via Algorithm 2, client 1 eventually deviates and follows the betrayal strategy outlined in Algorithm 3 (discussed in Section 5.2). This setup allows client 1 to secretly prepare for a unilateral takeover of the model. As shown in Figure 6, at iteration 10—the predetermined betrayal point—the global model abruptly loses performance on client 2’s dataset, while having even lower loss on client 1’s data. This result confirms that client 1 successfully erases not only the contributions of the benign clients, but also those of its former collaborator, client 2. These findings empirically validate that a malicious client can strategically cooperate to gain trust, only to later betray its partners and assert full control over the global model. This highlights a critical vulnerability in FL; even collaborative adversaries can be exploited by more sophisticated attackers acting within their own group.

7 PRACTICAL IMPLICATIONS

Our methods show that a single or a group of dictator clients, can manipulate the FL process so that the global model converges toward their local data distribution. This creates a “dictator client” effect, where the global model no longer represents the collective data of all participants, but instead becomes biased toward a particular client or group. Such bias can have serious consequences in real-world applications. For example, in healthcare, a global model biased toward data from one hospital or demographic group may make less accurate or unsafe predictions for underrepresented populations. In recommendation systems, it could prioritize the preferences of a few users over the majority, reinforcing algorithmic unfairness. This manipulation shifts the model’s decision boundaries, leading to skewed or inequitable outcomes and reducing trust in the system. Moreover, another motivation for such an attack arises in reward-driven learning environments, where clients are incentivized based on their contributions—such as the impact of their data on improving the global model.

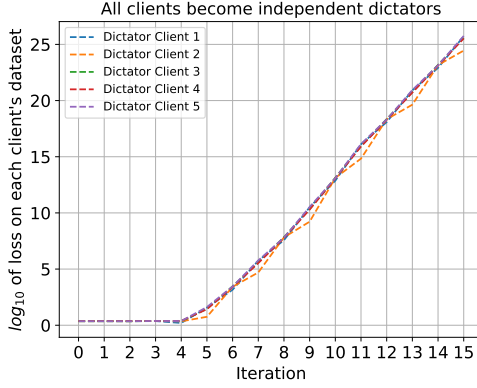


Figure 5: Loss functions for mutual domination scenario

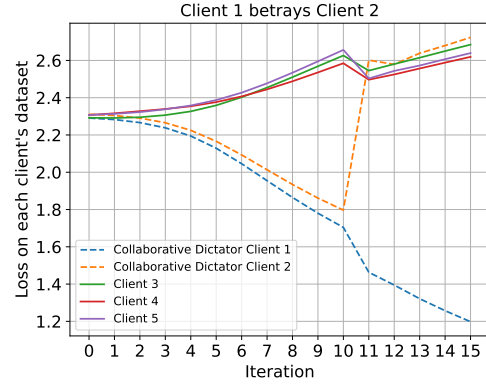


Figure 6: Loss functions for betrayal in collaboration scenario

A dictator client could exploit this by amplifying its influence while suppressing the contributions of other participants, thus increasing its perceived value and securing a larger share of the reward. Our work highlights how easily such influence can be exerted, especially in non-IID settings.

8 LIMITATIONS

Despite its conceptual novelty, the current form of the attacks may be vulnerable to detection. In particular, the gradient updates produced by a malicious client—especially in the single-dictator setting—can deviate from those of honest participants and may be identified by common server-side defenses such as anomaly detection or norm-based filtering. As a result, while the attack effectively exposes a weakness in the aggregation process, it may lack the stealth required for practical deployment in real-world systems without further refinement or adaptation. Nevertheless, this work should be viewed as a starting point toward a broader line of research. It raises important questions about the boundaries between personalization, dictator behavior, and adversarial manipulation in federated systems. In addition to exposing a new class of client-driven attacks, our work also explores the dynamics of collaboration and strategic behavior among clients, including scenarios where multiple clients collude or compete to influence the global model. These insights contribute to a deeper understanding of how clients may cheat, cooperate, or exploit the system for individual gain—offering a foundation for analyzing real-world risks in decentralized learning environments. Future work could build on this by developing more nuanced attack strategies that are harder to detect, incorporating techniques such as norm-constrained updates, or optimization-based stealth objectives.

9 CONCLUSION

In this work, we introduced a new perspective on Byzantine behavior in FL by formalizing the concept of **dictator clients**, malicious partners who seek to preserve their own influence while erasing that of others. We proposed attack algorithms for both individual and collaborative dictators and demonstrated their effectiveness through both theoretical analysis and empirical validation. Our results show that a single dictator can fully dominate the global model, and groups of collaborative dictators can entirely suppress the contributions of benign clients. However, this cooperation is inherently unstable: we also show that even within a coalition, a dictator can betray its partners to gain sole control. In the extreme case where every client behaves as an independent dictator, the global model fails to learn altogether, confirming the destructive consequences of uncoordinated selfish behavior.

REFERENCES

- Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2938–2948. PMLR, 26–28 Aug 2020. URL <https://proceedings.mlr.press/v108/bagdasaryan20a.html>.
- Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/ec1c59141046cd1866bbbcdfb6ae31d4-Paper.pdf.
- Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf.
- Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.
- Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International conference on machine learning*, pp. 3521–3530. PMLR, 2018.
- Chao Huang, Ming Tang, Qian Ma, Jianwei Huang, and Xin Liu. Promoting collaboration in cross-silo federated learning: Challenges and opportunities. *IEEE Communications Magazine*, 62(4): 82–88, 2024. doi: 10.1109/MCOM.005.2300467.
- Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pp. 203–226. 2019.
- 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.
- Cody Lewis, Vijay Varadharajan, and Nasimul Noman. Attacks against federated learning defense systems and their mitigation. *Journal of Machine Learning Research*, 24(30):1–50, 2023. URL <http://jmlr.org/papers/v24/22-0014.html>.
- Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1544–1551, 2019.
- Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE transactions on neural networks and learning systems*, 35(1):5–22, 2022.
- Tao Liu, Wu Yang, Chen Xu, Jiguang Lv, Huanran Wang, Yuhang Zhang, Shuchun Xu, and Dapeng Man. Act in collusion: A persistent distributed multi-target backdoor in federated learning. *arXiv preprint arXiv:2411.03926*, 2024.
- Xiaoting Lyu, Yufei Han, Wei Wang, Jingkai Liu, Bin Wang, Jiqiang Liu, and Xiangliang Zhang. Poisoning with cerberus: Stealthy and colluded backdoor attack against federated learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):9020–9028, Jun. 2023. doi: 10.1609/aaai.v37i7.26083. URL <https://ojs.aaai.org/index.php/AAAI/article/view/26083>.
- Xiaoting Lyu, Yufei Han, Wei Wang, Jingkai Liu, Bin Wang, Kai Chen, Yidong Li, Jiqiang Liu, and Xiangliang Zhang. Cobra: Collusive backdoor attacks with optimized trigger to federated learning. *IEEE Transactions on Dependable and Secure Computing*, 22(2):1506–1518, 2025. doi: 10.1109/TDSC.2024.3445637.

- Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.
- Thuy Dung Nguyen, Tuan Nguyen, Phi Le Nguyen, Hieu H. Pham, Khoa D. Doan, and Kok-Seng Wong. Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions. *Engineering Applications of Artificial Intelligence*, 127:107166, 2024. ISSN 0952-1976. doi: <https://doi.org/10.1016/j.engappai.2023.107166>. URL <https://www.sciencedirect.com/science/article/pii/S0952197623013507>.
- Priyesh Ranjan, Ashish Gupta, Federico Coro, and Sajal K Das. Securing federated learning against overwhelming collusive attackers. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pp. 1448–1453. IEEE, 2022.
- Ahmed E. Samy and Šarūnas Girdzijauskas. Mitigating sybil attacks in federated learning. In *Information Security Practice and Experience: 18th International Conference, ISPEC 2023, Copenhagen, Denmark, August 24–25, 2023, Proceedings*, pp. 36–51, Berlin, Heidelberg, 2023. Springer-Verlag. ISBN 978-981-99-7031-5. doi: 10.1007/978-981-99-7032-2_3. URL https://doi.org/10.1007/978-981-99-7032-2_3.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
- Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.
- Wei Shen, Wenke Huang, Guancheng Wan, and Mang Ye. Label-free backdoor attacks in vertical federated learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(19): 20389–20397, Apr. 2025. doi: 10.1609/aaai.v39i19.34246. URL <https://ojs.aaai.org/index.php/AAAI/article/view/34246>.
- Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68:4583–4596, 2020.
- Xiong Xiao, Zhuo Tang, Chuanying Li, Bin Xiao, and Kenli Li. Sca: Sybil-based collusion attacks of iiot data poisoning in federated learning. *IEEE Transactions on Industrial Informatics*, 19(3): 2608–2618, 2022.
- Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rkgyS0VFvr>.
- Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018.
- Wanyun Xie. A game-theoretical framework for byzantine-robust federated learning, 2022.
- Hangfan Zhang, Jinyuan Jia, Jinghui Chen, Lu Lin, and Dinghao Wu. A3fl: Adversarially adaptive backdoor attacks to federated learning. *Advances in neural information processing systems*, 36: 61213–61233, 2023a.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.
- Yifei Zhang, Dun Zeng, Jinglong Luo, Zenglin Xu, and Irwin King. A survey of trustworthy federated learning with perspectives on security, robustness and privacy. In *Companion Proceedings of the ACM Web Conference 2023, WWW '23 Companion*, pp. 1167–1176, New York, NY, USA, 2023b. Association for Computing Machinery. ISBN 9781450394192. doi: 10.1145/3543873.3587681. URL <https://doi.org/10.1145/3543873.3587681>.

A APPENDIX

B ALGORITHM 3: CHEATER CLIENT

Algorithm 3 Cheater collaborative dictator client 1

```

1: Require: Initialized weights  $\theta_0$ , learning rate  $\eta$ , Communication link with its partner client 2
   that is going to be cheated by client 1.  $\mathcal{P} = \{1, 2\}$  and  $P = 2$ . The desired cheating iteration is
    $E$ .
2: for iteration  $t = 0$  to  $T$  do
3:   if  $t = 0$  then
4:     Cheating_Update = 0
5:     Send  $M_0^1 = \nabla \mathcal{L}_1(\theta_0)$  as update
6:     Share  $\nabla \mathcal{L}_1(\theta_0)$  with other dictator partners
7:     Create a local copy of  $\theta_0$  as  $\hat{\theta}_0^{\mathcal{P}} = \theta_0$ 
8:     Update local model:  $\hat{\theta}_1^{\mathcal{P}} = \theta_0 - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\theta_0)$ 
9:     Create a secret copy of  $\theta_0$  as  $\hat{\theta}_0^1 = \theta_0$ 
10:    Update secret model:  $\hat{\theta}_1^1 = \theta_0 - \eta \nabla \mathcal{L}_1(\theta_0)$ 
11:    else if  $t < E$  then
12:       $M_t^1 = \nabla \mathcal{L}_t(\hat{\theta}_t^{\mathcal{P}}) - \left( \frac{\hat{\theta}_{t-1}^{\mathcal{P}} - \theta_t}{P\eta} - \nabla \mathcal{L}_t(\hat{\theta}_{t-1}^{\mathcal{P}}) \right)$ 
13:      Send  $M_t^1$  as update
14:      Share  $\nabla \mathcal{L}_1(\hat{\theta}_t^{\mathcal{P}})$  with other dictator partners
15:      Update local model:  $\hat{\theta}_{t+1}^{\mathcal{P}} = \hat{\theta}_t^{\mathcal{P}} - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_t^{\mathcal{P}})$ 
16:      Update secret model:  $\hat{\theta}_{t+1}^1 = \hat{\theta}_t^1 - \eta \nabla \mathcal{L}_1(\hat{\theta}_t^1)$ 
17:       $\Delta_t = \nabla \mathcal{L}_1(\hat{\theta}_t^1) - (\nabla \mathcal{L}_1(\hat{\theta}_t^{\mathcal{P}}) + \nabla \mathcal{L}_2(\hat{\theta}_t^{\mathcal{P}}))$ 
18:      Cheating_Update = Cheating_Update +  $\Delta_t$ 
19:    else
20:      Cheat client 2 by sending Cheating_Update as the update to the server
21:    end if
22:  end for

```

C WHAT IF THE DICTATOR CLIENT DOES NOT HAVE THE LEARNING RATE?

In this section, we show that even if dictator clients did not know the learning rate η , they could still approximate it after only one iteration.

Suppose we are at iteration t . Since gradient updates aren't usually too large, the dictator client m sends a very large number B as its update. Thus, the weight θ_{t+1} would be calculated by the server as the following:

$$\theta_{t+1} = \theta_t - \eta \left(B + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_t) \right). \quad (11)$$

At the next iteration $t + 1$, server sends θ_{t+1} to all clients. The dictator client m could approximate the learning rate η via the following equation:

$$\hat{\eta} = \frac{\theta_t - \theta_{t+1}}{B} = \frac{\eta \left(B + \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_t) \right)}{B} = \eta + \frac{\eta \sum_{n=1, n \neq m}^N \nabla \mathcal{L}_n(\theta_t)}{B} \approx \eta.$$

Moreover, now that client m has gained the learning rate, it can undo the previous bad contribution B and continue preserving its normal contribution while deleting other clients' contribution.

D PROOF OF ALGORITHM 2

At iteration 0 the server sends the initialized weight θ_0 to all the clients. Then, clients send their gradients to server. So θ_1 will be calculated as:

$$\theta_1 = \theta_0 - \eta \sum_{n=1}^N \nabla \mathcal{L}_n(\theta_0). \quad (12)$$

In the next iteration, server sends θ_1 to all the clients. Every client except the P dictator clients sends their gradient with respect to θ_1 . However, the dictator clients calculate $\hat{\theta}_1^{\mathcal{P}}$ as what would be the weight after the update in iteration 0 if only they contributed to that. In order to do that, they send their gradients with respect to θ_0 for each other. Afterwards, they can calculate $\hat{\theta}_1^{\mathcal{P}}$ via the following equation:

$$\hat{\theta}_1^{\mathcal{P}} = \theta_0 - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\theta_0). \quad (13)$$

Then, each dictator client $k \in \mathcal{P}$ sends the following update M_1^k instead of $\nabla \mathcal{L}_k(\theta_1)$ to the server in order to delete the contribution of other clients in the previous iteration and only preserve their own contribution:

$$M_1^k = \nabla \mathcal{L}_k(\hat{\theta}_1^{\mathcal{P}}) - \left(\frac{\theta_0 - \theta_1}{P\eta} - \nabla \mathcal{L}_k(\theta_0) \right) \quad (14)$$

Now, we analyze what would be the weight θ_2 after server receives the updates from clients and updates the weights:

$$\begin{aligned} \theta_2 &= \theta_1 - \eta \left(\sum_{k \in \mathcal{P}} M_1^k + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_1) \right) \\ &= \theta_1 - \eta \left(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_1^{\mathcal{P}}) - \left(\frac{\theta_0 - \theta_1}{\eta} - \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\theta_0) \right) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_1) \right) \\ &= \theta_0 - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\theta_0) - \eta \left(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_1^{\mathcal{P}}) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_1) \right). \end{aligned}$$

Using Eq. 13, we can write θ_2 as the following:

$$\theta_2 = \hat{\theta}_1^{\mathcal{P}} - \eta \left(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_1^{\mathcal{P}}) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_1) \right). \quad (15)$$

As a result, the collaborative dictator clients successfully deleted the contribution of other clients in the previous iteration while preserving their own contribution just by sending M_1^k as their update for each dictator client $k \in \mathcal{P}$.

We now generalize our method to any iteration t . The collaborative dictators calculate $\hat{\theta}_t^{\mathcal{P}}$ via the following equation:

$$\hat{\theta}_t^{\mathcal{P}} = \hat{\theta}_{t-1}^{\mathcal{P}} - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_{t-1}^{\mathcal{P}}). \quad (16)$$

We define the update M_t^k as the update that each dictator client $k \in \mathcal{P}$ sends at iteration t as the following:

$$M_t^k = \nabla \mathcal{L}_k(\hat{\theta}_t^{\mathcal{P}}) - \left(\frac{\hat{\theta}_{t-1}^{\mathcal{P}} - \theta_t}{P\eta} - \nabla \mathcal{L}_k(\hat{\theta}_{t-1}^{\mathcal{P}}) \right). \quad (17)$$

Now, we analyze what would be the weight θ_{t+1} after server updates the weights:

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta \left(\sum_{k \in \mathcal{P}} M_t^k + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_t) \right) \\ &= \theta_t - \eta \left(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_t^{\mathcal{P}}) - \left(\frac{\hat{\theta}_{t-1}^{\mathcal{P}} - \theta_t}{\eta} - \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_{t-1}^{\mathcal{P}}) \right) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_t) \right) \\ &= \hat{\theta}_{t-1}^{\mathcal{P}} - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_{t-1}^{\mathcal{P}}) - \eta \left(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_t^{\mathcal{P}}) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_t) \right). \end{aligned}$$

Using Eq. 16 we can write θ_{t+1} as the following:

$$\theta_{t+1} = \hat{\theta}_t^P - \eta \left(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_t^P) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_t) \right). \quad (18)$$

After T rounds of training, the final model weights θ^* will be:

$$\begin{aligned} \theta^* &= \hat{\theta}_T^P - \eta \left(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_T^P) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_T) \right) \\ &= \hat{\theta}_T^P - \eta \sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_T^P) - \eta \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_T) \\ &= \hat{\theta}_{T+1}^P - \eta \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_T) \approx \hat{\theta}_{T+1}^P. \end{aligned} \quad (19)$$

As it can be seen in Eq. 19, the exact final weights with our method would be $\hat{\theta}_{T+1}^P - \eta \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_T)$ where $\hat{\theta}_{T+1}^P$ would represent the weights for the final iteration if only the P collaborative dictator clients were contributing to the system during the training procedure and like the other clients never existed. Again, the term $\eta \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_t)$ is negligible since it is the updates only for one iteration and can not affect the final model too much, especially if the model achieved by the collaborative dictators is robust to such perturbations. Moreover, because of the nature of FL, the dictator clients are always one step behind and can not cancel this residual term. However, one could come up with more sophisticated methods in order to approximate or predict this residual term by observing the gradients through the training process.

E PROOF OF ALGORITHM 3

Before iteration E , the global model evolves exactly as if both client 1 and client 2 had followed Algorithm 2. Hence, the global weights at iteration $E - 1$ is updated as follows:

$$\theta_E = \hat{\theta}_{E-1}^P - \eta \left(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_{E-1}^P) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_{E-1}) \right). \quad (20)$$

However, at iteration E , client 1 sends the Cheating_Update which by iteration E has become the following:

$$\text{Cheating_Update} = \sum_{i=1}^{E-1} \nabla \mathcal{L}_1(\hat{\theta}_i^1) - \sum_{i=1}^{E-1} (\nabla \mathcal{L}_1(\hat{\theta}_i^P) + \nabla \mathcal{L}_2(\hat{\theta}_i^P)).$$

We also know that we can write $\hat{\theta}_{E-1}^P$ and $\hat{\theta}_{E-1}^1$ as the following:

$$\hat{\theta}_{E-1}^P = \theta_0 - \eta \sum_{i=1}^{E-1} (\nabla \mathcal{L}_1(\hat{\theta}_i^P) + \nabla \mathcal{L}_2(\hat{\theta}_i^P)), \quad (21)$$

$$\hat{\theta}_{E-1}^1 = \theta_0 - \eta \sum_{i=1}^{E-1} \nabla \mathcal{L}_1(\hat{\theta}_i^1). \quad (22)$$

So when server receives all the updates from all clients, the resulting model will be:

$$\begin{aligned} \theta_{E+1} &= \theta_E - \eta (\text{Cheating_Update} + M_t^2 + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_E)) \\ &= \hat{\theta}_{E-1}^P - \eta (\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_{E-1}^P) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_{E-1})) \\ &\quad - \eta (\sum_{i=1}^{E-1} \nabla \mathcal{L}_1(\hat{\theta}_i^1) - \sum_{i=1}^{E-1} (\nabla \mathcal{L}_1(\hat{\theta}_i^P) + \nabla \mathcal{L}_2(\hat{\theta}_i^P)) + M_t^2 + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_E)). \end{aligned}$$

Using Eq. 21 we will have:

$$\begin{aligned} \theta_{E+1} &= \theta_0 - \eta \sum_{i=1}^{E-1} \nabla \mathcal{L}_1(\hat{\theta}_i^1) - \eta (\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_{E-1}^P) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_{E-1})) \\ &\quad - \eta (M_t^2 + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_E)). \end{aligned}$$

Finally, from Eq. 22 we have:

$$\begin{aligned}\theta_{E+1} = & \hat{\theta}_{E-1}^1 - \eta(\sum_{k \in \mathcal{P}} \nabla \mathcal{L}_k(\hat{\theta}_{E-1}^{\mathcal{P}}) + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_{E-1})) \\ & - \eta(M_t^2 + \sum_{n=1, n \notin \mathcal{P}}^N \nabla \mathcal{L}_n(\theta_E)).\end{aligned}$$

Hence, client 1 has successfully replaced $\hat{\theta}_{E-1}^{\mathcal{P}}$ with $\hat{\theta}_{E-1}^1$ and cheated client 2.

F EXPERIMENTS FOR NLP

For NLP experiments, we used the `distilbert-base-uncased` Sanh et al. (2019) model for text classification as the global model and selected the AG news dataset Zhang et al. (2015) which has 4 different labels. Hence, we considered a FL with four clients for this case where each client has samples of only one label.

F.1 SINGLE DICTATOR CLIENT

Figure 7 demonstrates the loss function of global model when there is no dictator client and when client 1 becomes a dictator. Table 3 demonstrates accuracy of the global model when each client becomes dictator. We can see that each client has successfully dominated the training and led the global model to learn only that client’s dataset.

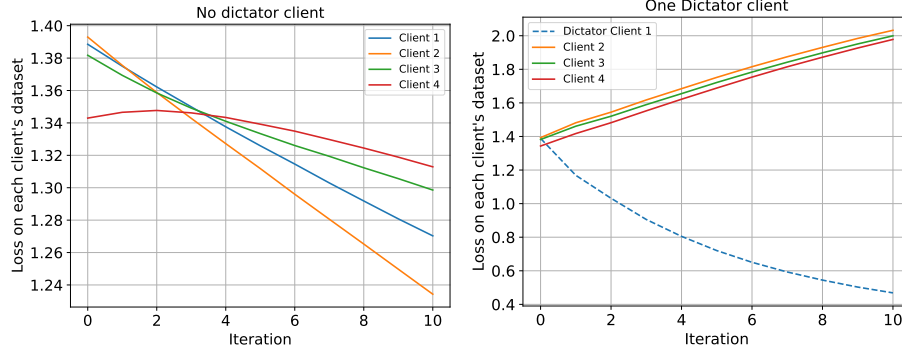


Figure 7: Loss function on each client’s dataset, comparing scenarios with no dictator clients and with one dictator client where in this figure client 2 is the dictator client.

Method	[0]	[1]	[2]	[3]
Regular FL	85.42	93.37	76.21	72.42
Dictator client: 1	100.00	0.00	0.00	0.00
Dictator client: 2	0.00	100.00	0.00	0.00
Dictator client: 3	0.00	0.00	100.00	0.00
Dictator client: 4	0.00	0.00	0.00	100.00

Table 3: Performance of the global model on each local dataset for AG news dataset and the single dictator client scenario.

F.2 COLLABORATIVE DICTATOR CLIENTS

Figure 8 demonstrates the loss function of global model when two or three clients become collaborative dictators. Table 4 demonstrates accuracy of the global model for these cases. We can see that the collaborative dictators successfully dominated the training and led the global model to learn only their dataset.

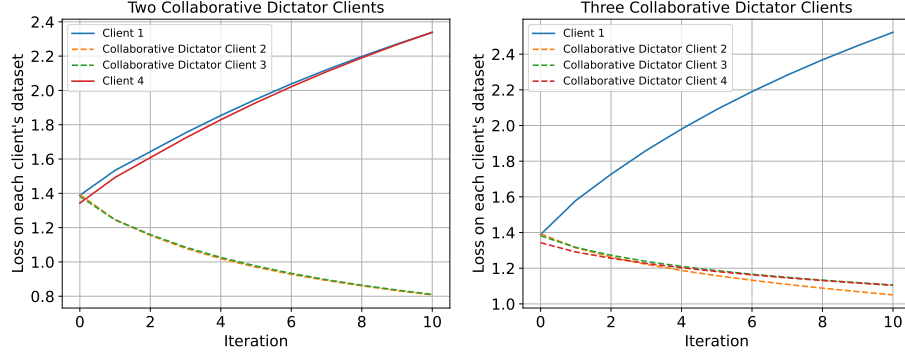


Figure 8: Loss function on each client’s dataset, when two clients become collaborative dictators (left) and three clients become collaborative dictators (right)

Method	[0]	[1]	[2]	[3]
Regular FL	85.42	93.37	76.21	72.42
Dictator clients: 2,3	0.00	96.89	97.47	0.00
Dictator clients: 2,3,4	0.00	96.00	75.47	85.32

Table 4: Performance of the global model on each local dataset for AG news dataset and the collaborative dictator clients scenario.

G MAIN RESULTS WITH 1-SIGMA ERROR BARS

In this section, we present extended versions of our main results with additional statistical details. All experiments were repeated using 5 different random seeds to account for variability. Tables 5 and 6 provide expanded versions of Table 1, reporting the mean and the corresponding 1-sigma error bars. Similarly, Tables 7 and 8 offer detailed results corresponding to Table 2.

Method	[0,1]	[2,3]	[4,5]	[6,7]	[8,9]
Regular FL	96.18 ±0.85	79.25 ±5.91	66.84 ±8.34	88.12 ±3.65	66.38 ±12.18
Dictator client: 1	99.63 ±0.11	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
Dictator client: 2	0.00 ±0.00	93.92 ±1.64	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
Dictator client: 3	0.00 ±0.00	0.00 ±0.00	97.43 ±0.99	0.00 ±0.00	0.00 ±0.00
Dictator client: 4	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	98.91 ±0.68	0.00 ±0.00
Dictator client: 5	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	94.42 ±0.48

Table 5: Performance of the global model on each local dataset for MNIST and the single dictator client scenario.

Method	[0,1]	[2,3]	[4,5]	[6,7]	[8,9]
Regular FL	39.04 ±3.85	12.51 ±5.82	31.07 ±2.30	23.74 ±4.53	52.59 ±1.59
Dictator client: 1	73.65 ±11.99	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
Dictator client: 2	0.00 ±0.00	65.19 ±8.65	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
Dictator client: 3	0.00 ±0.00	0.00 ±0.00	66.51 ±11.90	0.00 ±0.00	0.00 ±0.00
Dictator client: 4	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	73.98 ±4.66	0.00 ±0.00
Dictator client: 5	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	77.06 ±4.79

Table 6: Performance of the global model on each local dataset for CIFAR10 and the single dictator client scenario.

Method	[0,1]	[2,3]	[4,5]	[6,7]	[8,9]
Regular FL	96.18 ± 0.85	79.25 ± 5.91	66.84 ± 8.34	88.12 ± 3.65	66.38 ± 12.18
Dictator clients: 2,3	0.00 ± 0.00	88.19 ± 4.15	87.80 ± 4.18	0.00 ± 0.00	0.00 ± 0.00
Dictator clients: 2,3,4	0.00 ± 0.00	84.87 ± 2.98	80.22 ± 6.43	94.19 ± 2.13	0.00 ± 0.00

Table 7: Performance of the global model on each local dataset for MNIST and the collaborative dictator clients scenario.

Method	[0,1]	[2,3]	[4,5]	[6,7]	[8,9]
Regular FL	39.04 ± 3.85	12.51 ± 5.82	31.07 ± 2.30	23.74 ± 4.53	52.59 ± 1.59
Dictator clients: 2,3	0.00 ± 0.00	35.08 ± 18.92	43.17 ± 17.73	0.00 ± 0.00	0.00 ± 0.00
Dictator clients: 2,3,4	0.00 ± 0.00	18.38 ± 10.77	40.02 ± 8.37	46.05 ± 5.85	0.00 ± 0.00

Table 8: Performance of the global model on each local dataset for CIFAR10 and the collaborative dictator clients scenario.