




Article

# ARMOR: Differential Model Distribution for Adversarially Robust Federated Learning

Yanting Zhang <sup>1,2</sup>, Jianwei Liu <sup>3</sup>, Zhenyu Guan <sup>3</sup>, Bihe Zhao <sup>3</sup>, Xianglun Leng <sup>4</sup> and Song Bian <sup>3,\*</sup><sup>1</sup> School of Electronic and Information Engineering, Beihang University, Beijing 100191, China<sup>2</sup> ShenYuan Honors College, Beihang University, Beijing 100191, China<sup>3</sup> School of Cyber Science and Technology, Beihang University, Beijing 100191, China<sup>4</sup> PowerTensors.AI, Shanghai 200031, China

\* Correspondence: sbian@buaa.edu.cn

**Abstract:** In this work, we formalize the concept of *differential model robustness* (DMR), a new property for ensuring model security in federated learning (FL) systems. For most conventional FL frameworks, all clients receive the same global model. If there exists a Byzantine client who maliciously generates adversarial samples against the global model, the attack will be immediately transferred to all other benign clients. To address the attack transferability concern and improve the DMR of FL systems, we propose the notion of *differential model distribution* (DMD) where the server distributes different models to different clients. As a concrete instantiation of DMD, we propose the ARMOR framework utilizing differential adversarial training to prevent a corrupted client from launching white-box adversarial attack against other clients, for the local model received by the corrupted client is different from that of benign clients. Through extensive experiments, we demonstrate that ARMOR can significantly reduce both the attack success rate (ASR) and average adversarial transfer rate (AATR) across different FL settings. For instance, for a 35-client FL system, the ASR and AATR can be reduced by as much as 85% and 80% over the MNIST dataset.

**Keywords:** federated learning; model robustness; adversarial training; differential model distribution; Byzantine robustness



**Citation:** Zhang, Y.; Liu, J.; Guan, Z.; Zhao, B.; Leng, X.; Bian, S. ARMOR: Differential Model Distribution for Adversarially Robust Federated Learning. *Electronics* **2023**, *12*, 842. <https://doi.org/10.3390/electronics12040842>

Academic Editor: Aryya Gangopadhyay

Received: 5 January 2023

Revised: 1 February 2023

Accepted: 3 February 2023

Published: 7 February 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Federated learning (FL) became one of the most active areas of research in large-scale and trustworthy machine learning [1,2]. The main goal of FL is to enable distributed learning across data domains while protecting personal or institutional privacy, which is essential in financial [3] and medical [4] applications to develop large-scale joint learning platforms. More recently, the versatile framework of FL is proven beneficial in many other applications as well, notably in the area of distributed learning over Internet-of-things devices, such as the joint training of autonomous driving systems [5,6].

Following the popularity, we see two main lines of research within the realm of FL: one that improves the utility (e.g., prediction accuracy) of FL [7,8], and the other that studies the security and privacy of FL [9,10]. In particular, a plethora of attack and defense techniques were proposed for FL, which help sketched the overall security and privacy properties of FL frameworks. We note that, the malicious party in an attack against an FL framework can either be the server [11–13] or clients potentially controlled by third-party adversaries [14,15]. At the same time, various countermeasures were proposed against both server-side attacks [9,10,16] and client-side (or third-party adversarial) attacks [17–19].

Amongst the different offense schemes, we focus on the study of adversarial attacks in the presence of a Byzantine failure, as such attacks are serious threats to client-side security. It has been shown that FL is vulnerable towards traditional Byzantine attacks [14,20]. However, we make the observation that, to the best of our knowledge, no existing works study the countermeasures against adversarial attacks launched by Byzantine clients inside

the FL systems. Therefore, in this work, we propose a new notion of *differential model robustness* for FL. We point out the fact that, in most FL protocol designs, the server distributes the same global model to each and every client. When a Byzantine client becomes malicious against the other clients, the Byzantine client immediately gains full knowledge on the exact model architecture and parameters of all other clients, translating to a significant attack advantage on the adversary side.

Upon the above observations, we ask the simple question: can we distribute *different* models to clients, such that, while each client can still utilize its model for benign inferences, successful attacks against one client model do not transfer to other models? To answer this important research question, we propose new definitions and differential model distribution (DMD) techniques for FL systems. We also propose a concrete construction of our DMD technique called ARMOR. The name ARMOR is taken from **A**dversariially **R**obust differential **M**odel dist**R**ibution. The main contributions of this work can be summarized as follows.

- **Differential Model Robustness:** To the best of our knowledge, we are the first to formalize the notion of differential model robustness (DMR) under the FL context. Roughly speaking, the goal of DMR is to attain the same level of utility while keeping the client models as different as possible against adversarial attacks.
- **Differential Model Distribution:** We explore how can DMR be realized in concrete FL protocols based on neural networks (NNs). Specifically, we develop the differential model distribution technique, which distributes different NN models using differential adversarial training.
- **Thorough Experiments and Ablation Studies:** We provided detailed ablation studies and thorough experiments to study the utility and robustness of client models in our ARMOR framework. Through experiments, we show that, by carefully designing the DMD, the ASR and AATR can be reduced by as much as 85% and 80% respectively, at an accuracy cost of only 8% over the MNIST dataset for a 35-client FL system.

## 2. Background

### 2.1. Notation

In this work, we use  $\mathcal{D}$ ,  $\mathcal{P}$  and  $\mathcal{S}$  to denote datasets, and we use  $|\mathcal{D}|$  to depict the size of the dataset  $\mathcal{D}$ .  $d$  is a sample in dataset  $\mathcal{D}$ ,  $[R]$  is short for set  $\{1, 2, \dots, R\}$ , and  $\mathbb{E}$  means the expectation of a sequence.  $\mathcal{C}$  represents the set of clients. In terms of FL parameters, we consider an FL system with one server and a total of  $K$  clients, within which one client is malicious while other  $K - 1$  clients are benign.

### 2.2. Federated Learning

The notion of FL is first proposed by McMahan et al. [21]. Algorithm 1 shows the well-known Federated Averaging (FedAvg) [21] protocol. Here we give a detailed interpretation. First, on line 2, the server assigns each client with the same initialized  $w_0$ . In each communication round  $t$ , the server chooses a set of clients  $\mathcal{C}_t$ . Each client  $k \in \mathcal{C}_t$  in possession of a local dataset  $\mathcal{P}_k$  follows Algorithm 2 to train a local NN model  $w^k$ , and uploads local model to the server. Next, on line 10, the server aggregates the uploaded client models of all clients in  $\mathcal{C}_t$  in the  $t$ -th communication round, to produce the global model  $w_t$  for next communication round. The protocol then repeats, where the server re-distributes the model  $w_t$  to clients for the next epoch of local training.

**Algorithm 1:** Federated Averaging.

---

**Input:**  $K$  clients indexed by  $k$ , the ratio of clients to be selected for aggregation  $c$ .  
**Output:** The aggregated model  $w_R$  after  $R$  communication rounds.

```

1 # Server executes:
2 Assign each client with the same initialized  $w_0$ 
3 for  $t \in [R]$  communication round do
4    $m \leftarrow \max(c \cdot K, 1)$ 
5    $\mathcal{C}_t \leftarrow$  (random set of  $m$  clients)
6   for each client  $k \in \mathcal{C}_t$  in parallel do
7      $w_t^k \leftarrow \text{ClientUpdate}(k, w_{t-1})$ 
8   end
9   Update the global parameters as
10   $w_t \leftarrow \sum_{k=1}^K p_i w_t^k$ 
11 end

```

---

**Algorithm 2:** ClientUpdate( $k, w$ ).

---

**Input:** The client ID  $k$ , the global model  $w$ , the local dataset  $\mathcal{D}_k$ , the minibatch size  $B$ , the number of local epochs  $E$ , and the learning rate  $\kappa$   
**Output:** The locally trained model  $w^k$ .

```

1 # Client executes:
2  $\mathcal{B} \leftarrow$  (split  $\mathcal{D}_k$  into batches of size  $B$ )
3 for each local epoch  $i \in [E]$  do
4   for batch  $b \in \mathcal{B}$  do
5      $w \leftarrow w - \kappa \nabla \ell(w; b)$ 
6   end
7 end
8  $w^k = w$ 
9 return  $w^k$  to server

```

---

We note that most (if not all) traditional FL protocols distribute the same global model to each and every client. Consequently, Byzantine attack has become one of the most powerful attacks against FL systems.

### 2.3. Adversarial Attack

The notion of adversarial attack is first proposed by Goodfellow et al. [22], and is proven powerful against centralized learning mechanisms [23]. When the adversary is able to obtain full access to the victim model, the adversarial attack is known as white-box attack [24], which is the case for traditional FL systems in the presence of a corrupted client. Upon receiving the victim model  $f$ , the adversary begins to generate adversarial samples. The goal of the attack is to find some  $\delta$  such that  $f(x + \delta) \neq f(x)$ . Here, the optimized  $x + \delta$  is referred to as the adversarial samples.

### 2.4. Adversarial Training

Adversarial training [22] seeks to train deep neural networks that are robust against adversarial samples by leveraging robustness optimization. For each data point  $x \in \mathcal{D}$  and its label  $y$ , adversarial training introduces a set of perturbations  $\delta \in \mathcal{S}$ . Let  $\mathcal{L}$  denote the loss function (e.g., the Cross-Entropy loss). The objective function of adversarial training is as follows:

$$\min_f \mathbb{E}_{(x,y) \in \mathcal{D}} \left[ \max_{\delta \in \mathcal{S}} \mathcal{L}(f, x + \delta, y) \right]. \quad (1)$$

For the inner maximization of the saddle point formulation in Equation (1), the projected gradient descent (PGD) method [25] is usually applied. PGD adopts an iterative approach to generating the optimized adversarial sample  $x + \delta$  from some clean sample  $x$ . In the

iteration of PGD method, for each step  $t$ , it essentially executes projected gradient descent on the negative loss function

$$x^t = \Pi_{\mathcal{X}+\mathcal{S}}(x^{t-1} + \delta_{\text{step}} \text{sign}(\Delta_x \mathcal{L}(f, x, y))). \quad (2)$$

### 3. Related Works

#### 3.1. Federated Adversarial Training

Adversarial training is a typical method to enhance model robustness [22,26–31]. Recently, many works explored the application of adversarial training in FL. Adversarial training is originally developed primarily for IID data, and it is still challenging to be carried out in non-IID FL settings. Zizzo et al. [32] took the first step towards federated adversarial training (FAT), and evaluated the feasibility of practical FAT in realistic scenarios. The main objective of FAT is to utilize adversarial training to solve the robustness and accuracy challenges faced by FL systems when applied in real-world tasks.

#### 3.2. Byzantine-Robust Federated Learning

A stream of works have considered Byzantine-robust federated learning. Generally speaking, FL clients exchange knowledge through the aid of a trusted server. Under this setting, any client can be an adversary who wants to damage or poison the federated model. i.e., a Byzantine failure can occur within the FL system when clients become malicious. Byzantine-type attacks against FL systems include untargeted poisoning attacks [14,15,17,33] and targeted poisoning attacks [20].

To deal with Byzantine failures, several techniques are proposed to perform robust aggregation of user updates [34–36] or to detect and eliminate malicious user updates by similarity-based metrics [17–19,37]. Zizzo et al. [32] analyzed federated adversarial training in the presence of Byzantine clients, and concluded that it is still an open problem if both Byzantine-robustness and adversarial-robustness can co-exist within an FL system. Wang et al. [38] gave the theoretical proof that the existence of an adversarial sample implies the existence of a backdoor attack.

#### 3.3. Research Gaps and Our Goal

Existing works on federated adversarial training try to train one globally robust model through the cooperation of all clients. If we adopt this traditional adversarial training procedure, the robustness of the global model will be indeed enhanced. Nonetheless, in the presence of a Byzantine failure (i.e., corrupted clients), successful adversarial samples generated on one client through powerful white-box attacks can still be perfectly transferred to other client models, rendering the entire system vulnerable. Therefore, we point out that none of the above works follow our definition of DMR, and are independent from our work. In defining DMR and instantiating an effective DMD technique, our main objective is to differentiate client models, so as to resist adversarial attack in the presence of a Byzantine failure.

In regard of Byzantine-robustness, we share the similar goal with Byzantine-robust federated learning, i.e., to develop Byzantine-robust federated learning frameworks. However, our key insight is that, if we distribute different models to different clients, we can significantly increase the difficulty of the corrupted clients to launch attacks. The challenge is how to maintain the overall model utility while carrying out the differentiating procedure.

### 4. Methodology

In this section, we first define a set of notions to assist us in formalizing the notion of DMR under FL. Then, we discuss the threat model we consider in this work. Next, we propose a concrete DMD method to improve the DMR of client models and introduce the ARMOR framework. Finally, we analyze the robustness for FL under DMD.

### 4.1. Definition

To formulate the differential robustness of our framework, we present two important metrics: *Attack Success Rate* (ASR) and *Average Adversarial Transfer Rate* (AATR). While ASR is a conventional metric to measure the attacking capability of an adversary, AATR is a novel metric that evaluates the transferability of adversarial samples within a multi-client FL system.

The definitions for ASR vary in different works [24,39–42]. In this work, we define ASR as follows. For any clean dataset  $\mathcal{D}$ ,  $\mathcal{D}_{adv}$  is the set of adversarial samples generated from  $\mathcal{D}$  by some adversary  $\mathcal{A}$ , i.e., for any  $d_i^A \in \mathcal{D}_{adv}$ , there exists some  $\delta_i$  such that  $d_i^A = d_i + \delta_i$ , where  $d_i \in \mathcal{D}$  and  $|\delta_i| \leq \delta_{max}$ . Under this setting, we define ASR as follows.

**Definition 1** (Attack Success Rate). *The adversary  $\mathcal{A}$  chooses a clean dataset  $\mathcal{D}$ , and generates the adversarial dataset  $\mathcal{D}_{adv}$ . For some victim model  $X$ , we denote the set of all  $d_i \in \mathcal{D}$  that  $X$  predicts correctly by  $\mathcal{D}^X$ , and the set of all corresponding  $d_i^A$  by  $\mathcal{D}_{adv}^X$ . For each  $d_i^A \in \mathcal{D}_{adv}^X$ , if  $X$  gives incorrect prediction on  $d_i^A$ , we call  $d_i^A$  a successful adversarial sample. Denote the set of all successful adversarial samples  $d_i^A$  by  $\mathcal{S}_{adv}^X$ , and the set of all corresponding  $d_i$  by  $\mathcal{S}^X$ . We say that the attack success rate of the adversary  $\mathcal{A}$  on the victim model  $X$  is  $ASR_{\mathcal{D}}^{A,X} = |\mathcal{S}^X| / |\mathcal{D}^X|$ .*

ASR describes how effective is the attack launched by the adversary against a single victim model. In an FL scheme where clients receive different global models, however, we need a new metric to describing the transferability of adversarial samples across clients. We consider a  $K$ -client FL system with one malicious client the adversary  $\mathcal{A}$  and  $K - 1$  benign clients. Under this setting, we define AATR as follows.

**Definition 2** (Average Adversarial Transfer Rate). *The adversary  $\mathcal{A}$  first chooses a clean dataset  $\mathcal{D}$ , and generate the adversarial dataset  $\mathcal{D}_{adv}$  from  $\mathcal{D}$ . For each pair of  $d_i \in \mathcal{D}$  and  $d_i^A \in \mathcal{D}_{adv}$ , assume that a total number of  $\ell_i$  benign clients give correct predictions on  $d_i$  but incorrect predictions on the corresponding  $d_i^A$ . Then, we say that the adversarial transfer rate of  $d_i^A$  is  $ATR_{d_i^A} = \ell_i / (K - 1)$ . The average adversarial transfer rate is defined as  $AATR_{\mathcal{D}_{adv}}^A = \mathbb{E}_{d_i^A \in \mathcal{D}_{adv}} (ATR_{d_i^A})$ .*

Note that in differentially robust FL, the malicious client has no knowledge of the exact parameters of other benign client models. As a result,  $\mathcal{A}$  cannot decide which adversarial samples generated can be more powerful over the others. Thus,  $\mathcal{A}$  can only use adversarial samples that are correctly classified on its own model to launch adversarial attacks against the benign clients.

With AATR in hand, we are ready to formalize the notion of *Differential Model Robustness* (DMR) in FL setting. We point out that there are two (somewhat conflicting) goals that the server wishes to achieve here. On the one hand, the server wants the client models to stay close to the global model when predicting on the clean samples. On the other hand, client models should respond as different as possible against adversarial samples. Formally, we express the above idea of DMR as follows.

**Definition 3** (Differential Model Robustness). *Let  $\mathbb{I}$  be the indicator function such that  $\mathbb{I}(F(x)) = 1$  if  $F(x)$  is the correct classification of sample  $x$ . Given an FL system with one malicious client,  $K - 1$  benign clients, and some generalization dataset  $\mathcal{G}$ , let the global model be some function  $Y$ , and the differentiated client models be functions  $X_j$  for  $j \in [K - 1]$ . We say that the system achieves  $(\rho, \epsilon, \delta_{max})$ -DMR if the following inequalities hold:*

$$\min_{j \in [K-1]} \frac{\sum_{g_i \in \mathcal{G}} \mathbb{I}(X_j(g_i))}{|\mathcal{G}|} \geq \frac{\sum_{g_i \in \mathcal{G}} \mathbb{I}(Y(g_i))}{|\mathcal{G}|} - \rho, \tag{3}$$

$$AATR_{\mathcal{D}_{adv}}^A \leq \epsilon. \tag{4}$$

Here, when  $\mathcal{A}$  generates  $d_i^A = d_i + \delta_i$  from  $d_i$ , we restrict the amount of perturbations that the adversary  $\mathcal{A}$  can add to the clean data samples by  $|\delta_i| \leq \delta_{\max}$ .

In Equation (3), the left hand side is the minimum accuracy of the differentiated client models, and the right hand side is the accuracy of the global model with an acceptable accuracy deterioration of  $\rho$ . In other words, Equation (3) expresses our utility demand inside the notion of DMR, where client models should respond similarly to the global model over clean samples. Meanwhile, Equation (4) gives the maximum level of AATR, which describes the robustness towards adversarial samples across different clients, i.e., how client models respond differently against adversarial samples.

#### 4.2. Threat Model

We assume that the server and all clients participating in the the learning procedure of the original FL system are honest, which means that the server aggregates local models in the way it is supposed to, and each client honestly trains the local model using its own private dataset (we assume that any malicious local model update in the learning procedure can be detected or terminated by the server, since there exists various effective robust aggregation method [17–19,35–37]). However, after the global model being distributed to all clients, there exists one client that is malicious or compromised by a third-party adversary. As mentioned, we call this corrupted client the adversary  $\mathcal{A}$ . In this work, the attack is outside of the pristine FL learning procedure, and is targeted at the model which will be deployed by each client to its practical applications.

**Adversary’s capability.** We consider a  $K$ -client federated learning system with one client is malicious or compromised by a third-party adversary, and other  $K - 1$  clients are benign. The adversary is assumed to have the following attack capabilities.

1. To access the local training data of the compromised client, but have no knowledge of the private training datasets of other benign clients.
2. To launch white-box attacks at its will, as any client participating in the training process of FL has direct access to its own local model parameters and the global model parameters.

**Adversary’s goal.** Different from a centralized learning scheme, the goal of an adversarial attack under a distributed learning setting is to find some perturbation  $\delta$  such that other clients, as many as possible, classify the adversarial sample  $d_{\text{adv}} = d + \delta$  incorrectly. Formally, let  $\mathbb{I}_{\text{adv}}$  be the indicator function where  $\mathbb{I}_{\text{adv}}^{F,d}(y) = 1$  if  $y \neq F(d)$  and 0 otherwise, and  $X_j$  be the differentiated model received by client  $j$ , the above idea can be expressed as the optimization over

$$\arg \max_{\delta} \sum_{j=1}^L \mathbb{I}_{\text{adv}}^{X_j,d} (X_j(d + \delta)), \tag{5}$$

and that  $|\delta| \leq \delta_{\max}$  to restrict the perturbation. However, we note that in practical FL systems, the number of benign clients can be very large, and according to Theorem 1, we can hardly achieve Equation (5) when there are enough number of benign clients whose models are properly differentiated. Therefore, we also define an empirical goal for the adversary in terms of the AATR simply as

$$\arg \max_{\mathcal{D}_{\text{adv}}} \text{AATR}_{\mathcal{D}_{\text{adv}}}^A. \tag{6}$$

#### 4.3. Intuitions and Framework Overview

In this section, we give the intuition and overview of our approach to enhancing DMR of FL systems. Then we propose the ARMOR framework as a concrete construction of our differential model distribution.

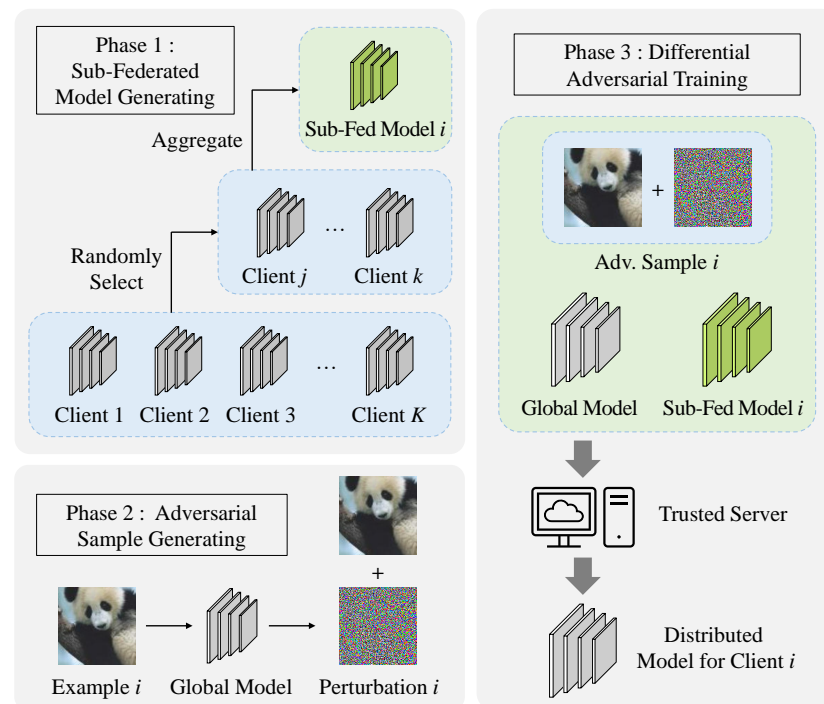
**Targeted Problem:** As described in Section 4.2, we consider adversarial attacks launched by Byzantine clients inside the FL systems. A Byzantine client has direct



access to the global model and can construct effective adversarial samples efficiently. In traditional FL protocols, the server distributes the same global model to each client. Consequently, adversarial samples constructed by the Byzantine client can easily attack all of other benign clients. We aim at preventing such Byzantine adversarial attacks from generalizing inside the FL system.

**General Solution:** Our main insight is that model differentiation can reduce the transferability of adversarial samples among clients in FL systems. However, if we conduct trivial model differentiation, such as adding noise in the way similar to differential privacy, a satisfactory level of differential model robustness will be accompanied by high levels of utility deteriorations. We manage to attain the same level of utility over normal inputs while keeping the client models as different as possible against adversarial inputs. We discover that combining with suitable differentiating operations, adversarial training can be useful to produce differentially robust models.

As is illustrated in Figure 1, the ARMOR framework can be roughly divided into three parts: sub-federated model generating phase, adversarial sample generating phase and differential adversarial training phase. Here we give the intuitional explanation and general description of the three phases.



**Figure 1.** The general working flow of the ARMOR framework. ARMOR consists of three phases: sub-federated model generating, adversarial sample generating and differential adversarial training. The first two phases produce two types of differentiation while the last phase completes the differentiation fusion.

### Phase 1. Sub-Federated Model Generating:

- **Intuition:** In traditional FL systems, there is only one aggregated model known as global model (or federated model). If we develop our differential client models from the same global model, the differentiation can be too weak to powerful Byzantine clients. We try to find out how to decide the directions in which we differentiate the global model.
- **Solution:** The last round of aggregation is shown in Algorithm 3. From Line 12 to 15, after getting all client model updates, for each client, the server randomly aggregates a set of client models into a *sub-federated model*. System manager can decide the number

of clients included in one sub-federated model by adjusting the proportion parameter to achieve satisfactory model utility. That is, for a total of  $K$  clients, the server will generate  $K$  different sub-federated models for preparation of directing and regulating the subsequent differential adversary training phase.

---

**Algorithm 3:** Differential Model Distribution.

---

**Input:**  $K$  clients indexed by  $k$ ,  $c$  the proportion of clients participating in each round,  $B$  the local mini-batch size,  $E$  the number of local epochs,  $E_{adv}$  the number of adversarial training epochs,  $\kappa$  the learning rate,  $R$  the number of communication rounds,  $\mathcal{D}$  the public dataset chosen by the server,  $\eta$  the proportion of sub-federated model, and  $\lambda$  the differentiation factor of sub-federated model.

**Output:** A group of differentially robust models  $w_k^t (k \in [K])$  after  $R$  communication rounds.

```

1 # Server executes: The regular FL protocol
2 Assign each client with the same initialized  $w_0$ 
3 for  $t \in [R]$  communication round do
4      $m \leftarrow \max(c \cdot K, 1)$ 
5      $C_t \leftarrow$  (an  $m$ -size random subset of client index  $[R]$ )
6     for each Client  $k \in C_t$  in parallel do
7          $w_k^t \leftarrow$  ClientUpdate( $k, w^{t-1}$ )
8     end
9     Update the global parameters as  $w^t \leftarrow \sum_{k \in C_t} (p_k w_k^t)$ 
10 end
11 # Step one: Generate sub-federated models
12 for each Client  $k \in [K]$  do
13      $\mathcal{W}_k \leftarrow$  (a random set of  $\eta K$  local model parameters  $w_i$ )
14      $w_k^{sub} \leftarrow \sum_{w_i \in \mathcal{W}_k} (p_i w_i)$ 
15 end
16 # Step two: Adversarial training
17 for each client  $k \in C_t$  do
18      $\mathcal{D}_k \leftarrow$  (an  $N$ -size random subset of public dataset  $\mathcal{D}$ )
19      $\tilde{w}_k = w^R$ 
20     for each epoch  $i \in [E_{adv}]$  do
21          $\mathcal{D}_{adv,k} \leftarrow$  PGD( $\mathcal{D}_k, \tilde{w}_k$ )
22          $\tilde{w}_k \leftarrow$  DiffTrain( $\tilde{w}_k, w_k^{sub}, \lambda, \mathcal{D}_{adv,k}$ )
23     end
24     Distribute  $\tilde{w}_k$  to Client  $k$ 
25 end
26 DiffTrain( $w, w_k^{sub}, \lambda, \mathcal{D}_{adv,k}$ ):
27  $\mathcal{B} \leftarrow$  (split  $\mathcal{D}_{adv,k}$  into batches of size  $B$ )
28 for batch  $b \in \mathcal{B}$  do
29      $w \leftarrow w - \kappa \nabla \mathcal{L}(w, w_k^{sub}; b)$ 
30 end
31 return  $w$ 

```

---

**Phase 2. Adversarial Samples Generating:**

- **Intuition:** In centralized adversary training, the server generates adversarial samples through adversarial attack methods such as FGSM attack [22] or PGD attack [25]. If we simply follow the same paradigm and utilize the whole public dataset to train the global model, we will come back to the problem of Byzantine clients again. As pointed out in Section 3.3, it is dangerous for all clients to hold the same global model in the model deployment phase. We need to generate different adversarial samples for each client.



- **Solution:** As shown in Algorithm 3, from Line 17 to 21, after aggregating client models in the last round into a final global model, the server further generates adversarial samples based on the final global model. For each client, the server chooses a different set of samples from its public dataset, and uses different randomness to generate adversarial samples from the chosen sample set. That is, for a total of  $K$  clients, the server will generate  $K$  different sets of adversarial samples.

**Phase 3. Differential Adversary Training:**

- **Intuition:** Now, we need to find an efficient way to conduct the differentiation while retaining the model accuracy. We are faced with two challenges. First, how to decide the metric of model distance (or model similarity)? A suitable metric is extremely important as it will directly influence our differential adversary training directions. Second, how to quantitatively produce different levels of differentiation? As model utility and DMR is a trade-off, a higher level of differentiation will lead to stronger DMR but weaker model utility. We should be able to adjust the level of differentiation to achieve a balance between utility and DMR.
- **Solution:** Utilizing Phase 1 and Phase 2, the server allocates each client a sub-federated model and a set of adversarial samples. When conducting differential adversarial training, we choose cosine similarity as the criterion to measure model distance. We use the cosine similarity between the output vectors of global model and sub-federated model to construct a similarity loss. We combine the similarity loss with the regular cross-entropy loss during adversarial training to accomplish our goal of differentiation.

**4.4. Key Algorithms In ARMOR**

As discussed in Section 4.3, in the ARMOR framework, our differential adversarial training consists of two types of differentiation followed by a differentiation fusion step. The detailed algorithm is given in Algorithm 3. Here, we give a line-by-line interpretation of Algorithm 3.

We consider an FL system with  $K$  clients. From Line 1 to 10, ARMOR follows the traditional FL protocol. We choose the standard Federated Averaging (FedAvg) [21] protocol as the basic FL algorithm. On Line 2, the server samples an initialized global model  $w_0$  and assigns each client with the same initialized  $w_0$ . We assume the total number of global communication rounds to be  $R$ . For each round  $t \in [R]$ , the server first computes the number of clients participating in this training round as  $m \leftarrow \max(c \cdot K, 1)$ . Then the server randomly samples an  $m$ -size random subset of client index  $[R]$ . We denote the collection of these  $m$  clients by  $C_t$ . Each client  $k$  in  $C_t$  executes ClientUpdate to train the current global model  $w^{t-1}$  using its own local dataset, and outputs local model update  $w_k^t$  (The description of ClientUpdate is shown in Algorithm 2). When all clients in  $C_t$  complete their local training and return the local model update, the server updates the global parameters as  $w^t \leftarrow \sum_{k \in C_t} (p_k w_k^t)$ .

Then, Lines 11 to 25 describe what we refer to as the *differential adversarial training* technique, which is proven effective in enhancing the DMR while retaining a high level of utility of each client model. Differential adversarial training can be divided into two steps. In the first step, i.e., Line 11 to 15, the server generates sub-federated models for clients. The server first decides a factor  $\eta$ , which denotes the proportion of clients included in one sub-federated model. For each client  $k \in [K]$ , the server randomly samples a set of  $\eta K$  local model parameters  $w_i$ , and aggregates them into sub-federated model for client  $k$  as  $w_k^{sub} \leftarrow \sum_{w_i \in \mathcal{W}_k} (p_i w_i)$ . In the second step, i.e., Line 16 to 25, the server conducts adversarial training. We assume the total number of adversarial training epochs to be  $E_{adv}$ . For each client  $k \in [K]$ , the server randomly samples an  $N$ -size subset of public dataset  $\mathcal{D}$ . In each adversarial training epoch  $i \in [E_{adv}]$ , the server uses PGD method to generate adversarial samples  $\mathcal{D}_{adv,k}$ , and executes DiffTrain to finish the training procedure. The DiffTrain algorithm is described in Line 26 to 30, which is run by the server to produce differentiated models based on the global model, the chosen sub-federated model and the

chosen adversarial samples. Finally, for each client  $k \in [K]$ , the server distributes  $\tilde{w}_k$  to client  $k$ .

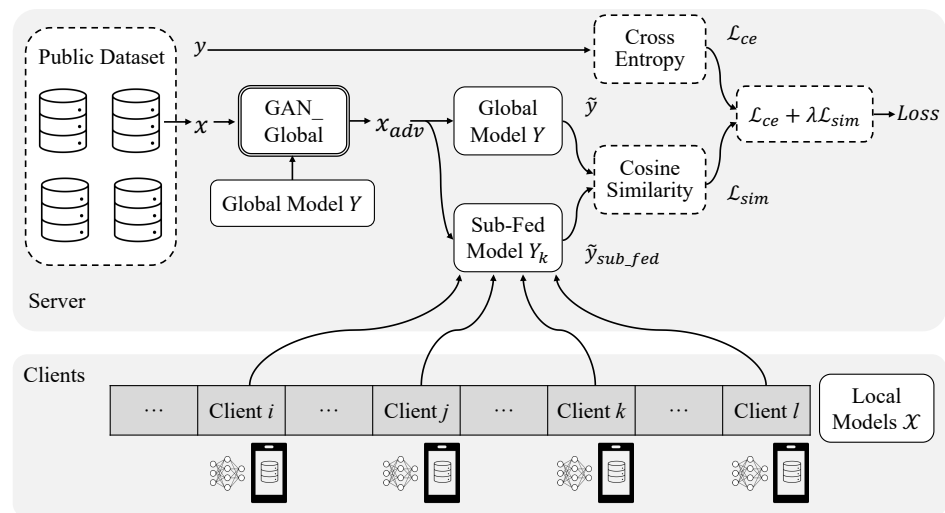
The formal descriptions of the three key components in the ARMOR framework are given as follows.

1. **Sub-Federated Model Based Model Differentiation:** At the last round of aggregation, the server gets the set of all local models  $\mathcal{X} = \{X_1, X_2, \dots, X_K\}$  from  $K$  clients, and aggregate these local models into a global federated model  $Y = \sum_{X_i \in \mathcal{X}} (p_i X_i)$  (With some abuse of notations, we use  $Y = \sum_{X_i \in \mathcal{X}} (p_i X_i)$  to denote the server’s operation of aggregating several local models  $X_i$  with model parameter  $w_i$  to generate the global model  $Y$  with model parameter  $w$ , i.e.,  $w = \sum_{i=1}^{|\mathcal{X}|} p_i w_i$ . In this work, we have  $p_i = 1/|\mathcal{X}_i|, i \in [|\mathcal{X}|]$ ). For each client  $k$ , the server randomly chooses  $\lceil \eta K \rceil$  local models from the set  $\mathcal{X}$  to form a subset  $\mathcal{X}_k$ , and aggregate the  $\lceil \eta K \rceil$  local models in  $\mathcal{X}_k$  into a sub-federated model  $Y_k = \sum_{X_i \in \mathcal{X}_k} (p_i X_i)$ . We denote the set of all sub-federated models by  $\mathcal{Y}_{\text{sub}} = \{Y_1, Y_2, \dots, Y_{\lceil \eta K \rceil}\}$ .
2. **Adversarial Samples Based Model Differentiation:** In ARMOR, the server generates  $K$  different sets of adversarial samples based on the global model. For each client  $k$ , the server chooses a set of samples  $\mathcal{D}_k$  from its public training dataset  $\mathcal{D}$ , and adopts PGD method [25] to generate a set of adversarial samples  $\mathcal{D}_{\text{adv},k}$  in preparation for adversarial training. In this step, each adversarial dataset  $\mathcal{D}_{\text{adv},k}$  for  $k \in [K]$  contains a different flavor of robustness, which will be introduced to the global model in the following adversarial training phase.
3. **Differential Adversary Training:** Combining the above two steps, the server associates each Client  $k$  with a sub-federated model  $Y_k$  and a set of different adversarial samples  $\mathcal{D}_{\text{adv},k}$ . Figure 2 illustrates the detailed relationships between models and losses in our training process. The server executes DiffTrain in Algorithm 3 to make each client find its way from  $Y$  towards the direction between  $Y_k$  and the robustness introduced by  $\mathcal{D}_{\text{adv},k}$ . Next, for DiffTrain, our goal is to produce differentiated model  $Y_k$  based on the global model  $Y$  (we note that directly using  $Y_k$  as the  $k$ -th client model results in degraded accuracy performance). Here, we choose the cosine distance as the criterion to measure the similarity between the global model and the sub-federated models. Given input samples  $\mathcal{D}_{\text{adv},k}$ , we compute the cosine embedding loss of the output of global model  $Y$  and the corresponding sub-federated model  $Y_k$ . Let  $Y$  and  $Y_k$  be the model functions whose outputs are the probability vectors over the class labels. We define the similarity loss for sample  $d_i^A$  as

$$\ell_{\text{sim},i}(Y, Y_k) = 1 - \frac{Y(d_i^A) \cdot Y_k(d_i^A)^T}{\|Y(d_i^A)\|_2 \times \|Y_k(d_i^A)\|_2}, \tag{7}$$

where  $\cdot$  is the inner product between vectors,  $\|\cdot\|_2$  depicts the  $L_2$ -norm of a vector, and  $\times$  denotes integer multiplication. When measuring similarity, we define the target labels  $T = \{t_1, t_2, \dots, t_{|\mathcal{D}_{\text{adv},k}|}\}$  for the cosine similarity loss, where each  $t_i$  follows the Bernoulli distribution  $P(t_i)$  where  $\Pr[t_i = 1] = p$ . Then, we use  $T$  to select a portion of  $p$  samples with label 1 to participate in similarity measurement. Now, we have our total cosine similarity loss

$$\mathcal{L}_{\text{sim}}(Y, Y_k) = \sum_{d_i^A \in \mathcal{D}_{\text{adv},k}} \frac{t_i \times \ell_{\text{sim},i}(Y, Y_k)}{|\mathcal{D}_{\text{adv},k}|}. \tag{8}$$



**Figure 2.** The server conducts differential adversarial training with the aid of a public dataset. The loss function consists of two parts: one is the cross entropy loss of the regular adversarial training, and the other is the cosine similarity loss between the global model and the sub-federated model.

In addition to the similarity measurement, we follow the method of [22] to train on a mixture of clean and adversarial examples. We compute the regular cross-entropy loss  $\mathcal{L}_{ce}$  between input  $\mathcal{D}_{adv,k}$  and the label set of the corresponding clean dataset  $\mathcal{D}_k$  as

$$\mathcal{L}_{ce}(Y) = \sum_{d_i^A \in \mathcal{D}_{adv,k}} \left[ \ell_{ce,i}(Y, d_i) + \ell_{ce,i}(Y, d_i^A) \right] \times \frac{1}{|\mathcal{D}_{adv,k}|}. \tag{9}$$

Combining the two losses, we have our final objective loss function as

$$\mathcal{L} = \mathcal{L}_{ce}(Y) + \lambda \mathcal{L}_{sim}(Y, Y_k), \tag{10}$$

where  $\lambda$  is the controlling factor to decide how strong our differential models will be differentiated in the directions of the randomly generated sub-federated models.

#### 4.5. Robustness Analysis

Before delving into the theory, we first note that there exists adversarial perturbations which are effective against any classifier [43–45]. As a result, when only a small number (e.g., one) client is benign in an FL system, the derivations in [43] indicate that there is an upper limit on the adversarial robustness of any classifier.

Consequently, we seek the following alternative. From a high level of view, the differential model distribution technique, i.e., the differential adversarial training method we proposed above, can be seen as conducting stochastic perturbations to the global model while retaining an acceptable level of accuracy deterioration. Due to the randomness introduced by stochastic functions during the training procedure, we can take these perturbations as independent random variables added to the global model. Subsequently, the following theorem can be established.

**Theorem 1.** We consider an  $K$ -client FL system with one malicious client and  $K - 1$  benign clients. Each client receives a linear classifier  $F_i$  for  $i \in [K]$ . Assume that there exists some DMD mechanism in the FL system such that for any two benign client models  $F_i(x) = w_i x$  and  $F_j(x) = w_j x$  where  $i \neq j$ , it holds that  $w_i = w + \beta_i$  and  $w_j = w + \beta_j$  where  $\beta_i$  and  $\beta_j$  are independent random

variables satisfying  $\beta_i, \beta_j \geq \beta_{\min}$  for all  $i, j \in [L]$ . Then, for any adversarial sample  $d_i^A = d_i + \delta_i$  with restriction of  $|\delta_i| \leq \delta_{\max}$ , we have that

$$\Pr[\text{ATR}_{d_i^A} \geq \theta] \leq (1 - \gamma)^{\theta(K-1)} \tag{11}$$

for some real numbers  $\theta, \gamma \in [0, 1]$ .

**Proof.** We can prove the theorem via a simple probabilistic argument. Without loss of generality, we assume that the malicious client is Client 0. First, observe that for any successful adversarial sample  $d^A = d + \delta$  on the corrupted client, it holds that

$$\text{sign}(w_0(d + \delta)) \neq \text{sign}(w_0(d)). \tag{12}$$

Expanding the terms, we get

$$\text{sign}(w_0d + w_0\delta) \neq \text{sign}(w_0d), \text{ which means that} \tag{13}$$

$$\text{sign}(wd + \beta_0d + w\delta + \beta_0\delta) \neq \text{sign}(wd + \beta_0d). \tag{14}$$

For the adversarial sample  $d^A$  to transfer to the  $j$ -th client, we need to achieve a similar goal, i.e.,

$$\text{sign}(wd + \beta_jd + w\delta + \beta_j\delta) \neq \text{sign}(wd + \beta_jd). \tag{15}$$

Since both Client 0 and Client  $j$  correctly classify  $d$  (which is a necessary condition for any adversarial attack to be meaningful), we know that

$$\text{sign}(wd + \beta_jd) = \text{sign}(wd + \beta_0d). \tag{16}$$

Then, we know that the objective of transferring the adversarial sample from Client 0 to the Client  $j$  can be formulated as

$$\text{sign}(wd + \beta_jd + w\delta + \beta_j\delta) = \text{sign}(w_0(d + \delta)). \tag{17}$$

We point out that  $w_j$  can be formulated as a “differentiated” version of  $w_0$ , i.e.,

$$w_j = w_0 + \beta_{0j}, \tag{18}$$

where  $\beta_{0j} = -\beta_0 + \beta_j$ . As a result, the LHS for Equation (17) becomes

$$\text{sign}(w_0(d + \delta) + \beta_{0j}(d + \delta)) = \text{sign}(w_0(d + \delta)). \tag{19}$$

Here, we can consider the term  $\beta_{0j}(d + \delta)$  as an additive noise to the classification result  $w_0(d + \delta)$ , which is a label that is the opposite of  $w_0d$ . Now, to satisfy the objective of Equation (17), we basically need the noise of  $\beta_{0j}(d + \delta)$  to be small enough such that the addition of this term does not cause the classification result to cross the decision boundary (i.e., flip the sign). Unfortunately, from the results in [43], we know that the robustness of any classifier is bounded from above. Let  $\xi$  denote that the probability of classifying the result to 1, then the probability of 0 becomes  $1 - \xi$ , and the fraction of adversarial samples that does not work on  $w_j$  becomes

$$\Pr(\mathcal{R}(x) \leq \eta) \geq 1 - \sqrt{\frac{\pi}{2}} e^{-\omega^{-1}(\eta)^2/2}, \tag{20}$$

where  $\mathcal{R}(x) = \min_{\delta} \|\delta\|$  such that  $w_0(x + \delta) \neq w_0(x)$  (i.e., the robustness of the input sample  $x$ ),  $\eta$  is the robustness threshold,  $\omega$  is the modulus of continuity, Simialr to [43], when we take  $\omega^{-1}(\eta) = \eta/\mathcal{L}$  where  $\mathcal{L}$  is the Lipschitz constant, we have that

$$\Pr(\mathcal{R}(x) \leq \eta) \geq 1 - \sqrt{\frac{\pi}{2}} e^{-(\eta/\mathcal{L})^2/2}. \tag{21}$$

Replacing  $x$  with the adversarial sample  $d + \delta$ , we see that the stability of transferring the adversarial sample can be seen as its robustness, and this robustness is bounded from above by some factor  $\eta$ . Hence, when we have a model-wise perturbation that is larger than the robustness of the adversarial sample, i.e.,

$$\beta_{0j}(d + \delta) \geq \eta, \tag{22}$$

the  $j$ -th model will produce a “mis-classified” adversarial sample with non-negligible probability, which is exactly the probability of the  $j$ -th model to produce a correct prediction on the adversarial sample in the binary classification case. Since  $\beta_{0j}$  can be adjusted according to  $\beta_{\max}$ , we are guaranteed that

$$\Pr[\text{sign}(w_0(d + \delta) + \beta_{0j}(d + \delta)) = \text{sign}(w_0(d + \delta))] \tag{23}$$

$$= \Pr[\text{sign}(w_0(d + \delta) + \eta) = \text{sign}(w_0(d + \delta))] \tag{24}$$

$$= 1 - \Pr[\mathcal{R}(x) \leq \eta] \tag{25}$$

$$\leq 1 - \Pr[\mathcal{R}(x) \leq \beta_{0j}(d + \delta)]. \tag{26}$$

Let  $\gamma = \Pr[\mathcal{R}(x) \leq \beta_{0j}(d + \delta)]$ , we then know that the probability that any adversarial sample  $d + \delta$  can transfer to the  $j$ -th client model is  $1 - \gamma$ . Since all pairs of  $\beta_{0j}$  for  $j \in [K - 1]$  is mutually independent, the probability that an adversarial sample  $d^A$  simultaneously transfers to  $\ell$  benign clients is  $(1 - \gamma)^\ell$ . Then, we have that for any adversarial sample  $d^A$ ,

$$\Pr[\text{All } \ell \text{ clients are compromised}] = (1 - \gamma)^\ell. \tag{27}$$

As defined in Definition 2, since  $\ell$  is the number of clients that are compromised,  $\text{ATR}_{d^A} = \ell/(K - 1)$ , we have

$$\Pr[\text{ATR}_{d^A} \geq \theta] = \Pr[\ell/(K - 1) \geq \theta] \leq (1 - \gamma)^{\theta(K-1)}, \tag{28}$$

and the theorem follows.  $\square$

We note that Equation (28) goes to a probability of 0 as the total number of benign clients  $K - 1$  goes to infinity. That is to say that, for any adversarial sample  $d^A$  and any attack transfer rate  $\theta$  that is non-zero, the probability of an adversary achieving this ATR goes to zero when the number of benign clients grow.

## 5. Experiment Results

### 5.1. Experiment Flow and Setup

To validate the effectiveness of our DMD method, we perform experiments in FL settings of different client numbers, and in each setting we balanced the partition of the whole training dataset among all clients in a non-i.i.d. manner. We follow the DMD technique described in Algorithm 3 to conduct our experiment. In this experiment, we take nine different FL settings of clients number  $K = 10, 15, 20, 25, 30, 35, 40, 45, 50$  and set  $c = 1$  in FedAvg algorithm.

**Physical Specifications:** We conduct our experiments on Linux platform with NVIDIA A100 SXM4 with a GPU memory of 40GB. The platform is equipped with a driver of version 470.57.02 and CUDA of version 11.4.

**Datasets:** We empirically evaluate the ARMOR framework on two datasets: MNIST [46] and CIFAR-10 [47]. To simulate the heterogeneous data distributions, we make non-i.i.d. partitions of the datasets, which is a similar partition method as [21].

- (1) Non-IID MNIST: The MNIST dataset contains 60,000 training images and 10,000 testing images of 10 classes. Each sample is a  $28 \times 28$  size gray-level image of a handwritten digit. We first sort the training dataset by digit label, divide it into  $3K$  shards of size  $60,000/\lceil K \rceil$ , and assign each client 3 shards.
- (2) Non-IID CIFAR-10: The CIFAR-10 dataset contains 50,000 training images and 10,000 test images of 10 classes. Each sample is a  $32 \times 32$  size tiny color image. We first sort the training dataset by class label, divide it into  $4K$  shards of size  $50,000/\lceil K \rceil$ , and assign each client 4 shards.

**Model:** For the MNIST dataset, we use a CNN model with two  $5 \times 5$  convolution layers (the first with 4 channels, the second with 10 channels, each followed with  $2 \times 2$  max pooling), a fully connected layer with 100 units, an ReLU activation, and a final output layer. For the CIFAR-10 dataset, we use the VGG-16 model [48].

**Hyperparameters:** For both datasets, we first train with the federated averaging algorithm. In each communication round, we let all clients to participate in the training (i.e.,  $c = 1$ ), where the client model is trained by one epoch using the local datasets. On the server side, the model update from each client is weighted uniformly (since we assume that each client has the same number of training samples). For MNIST and CIFAR-10, we set the number of communication round  $R$  to 50 and 500, the learning rate  $\kappa$  to 0.07 and 0.05, and the client batch size to 10 and 64, respectively.

When applying the PGD attack for adversarial training, we need to decide the upper bound on the gradient steps in the  $\ell_\infty$ -norm. For MNIST, the server uses 1000 public images to generate adversarial samples for training, each adversarial sample is constructed with  $\delta_{\max} = 0.2$  (also denoted as  $\epsilon$  in many works) as the max perturbation range by a step size of  $\delta_{\text{step}} = 0.01$  for 40 iterations, while for CIFAR-10, we choose 1000 public images with  $\delta_{\max} = 0.03$  by a step size of  $\delta_{\text{step}} = 0.008$  for 20 iterations. Note that we have  $K$  clients in this FL setting, so the server generates  $K$  different sets of adversarial samples based on the global model.

## 5.2. Main Results

In this section, we show the results of applying ARMOR over the MNIST and CIFAR-10 datasets.

### 5.2.1. Results on MNIST

Table 1 illustrates the results of the average model accuracy over all clients, and average ASR and AATR of adversarial samples on benign clients with a varying number of FL clients. Before delving into the results, we first explain the subtle relationship between Acc and  $\text{Acc}_{\mathcal{D}}$ . Recall that our goal is to make the client models stay close to the global model when predicting the clean samples while responding as differently as possible against adversarial samples. Here, Acc is the accuracy on randomly selected testing samples, while  $\text{Acc}_{\mathcal{D}}$  is the accuracy on clean dataset  $\mathcal{D}$ . Note that when launching attacks,  $\mathcal{A}$  only chooses successful adversarial samples (i.e., misclassified by the client model of  $\mathcal{A}$ ) whose clean samples can be correctly classified using its own client model. In other words, for the client model of  $\mathcal{A}$ ,  $\text{ASR}_{\mathcal{D}}^{\mathcal{A}} = 100\%$ . Therefore, the difference between Acc and  $\text{Acc}_{\mathcal{D}}$  indicates whether correctly predicted samples in  $\mathcal{D}$  is different from the datasets of all other clients. To this end, Table 1 confirms that  $\text{Acc}_{\mathcal{D}}$  is almost identical to Acc. The conclusion here is that, the adversary  $\mathcal{A}$  cannot tell which sample will be more susceptible to the other models possessed by the benign clients with high confidence.

Comparing the accuracy and AATR of models with and without our differential adversarial training, we point out that model utility and DMR is indeed a trade-off. For example, for client number  $K = 35$  in Table 1, after deploying our DMD technique, we

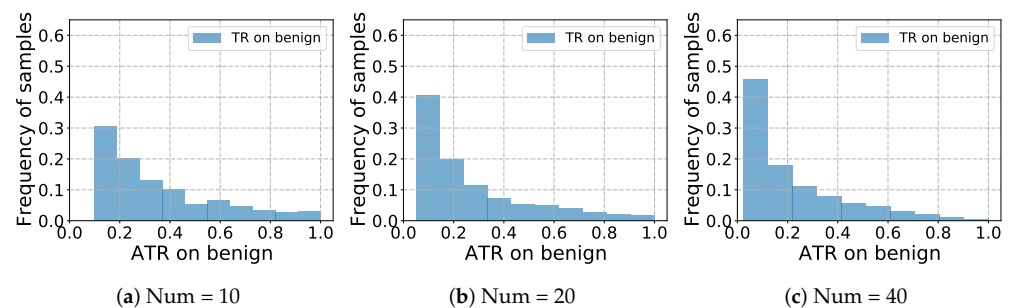


are able to reduce the ASR on benign clients from 100% to **15.21%** and AATR from 100% to **20.45%** while maintaining the overall client model accuracy of 90.35%. At the same time, as the number of FL clients increases, AATR in Table 1 exhibits a smooth decrease, as predicted in Section 4.5.

**Table 1.** Results of FL with/without DMD under different settings for MNIST.

Client Num	DMD	Acc (%)	Acc <sub>D</sub> (%)	ASR (%)	AATR (%)
10	×	98.33	98.33	100.00	100.00
	✓	90.12	84.66	16.76	28.82
15	×	98.33	98.33	100.00	100.00
	✓	90.18	86.33	14.82	23.64
20	×	98.38	98.38	100.00	100.00
	✓	90.21	84.83	15.99	23.38
25	×	98.38	98.38	100.00	100.00
	✓	90.33	85.32	15.43	21.99
30	×	97.95	97.95	100.00	100.00
	✓	90.09	85.20	16.93	22.67
35	×	98.38	98.38	100.00	100.00
	✓	90.35	85.43	15.21	20.45
40	×	98.38	98.38	100.00	100.00
	✓	89.11	84.41	14.57	19.47
45	×	98.38	98.38	100.00	100.00
	✓	88.01	83.20	13.95	18.44
50	×	98.38	98.38	100.00	100.00
	✓	87.07	81.89	14.60	18.88

We present the frequency of the average transfer rate of each adversarial sample generated by adversary  $\mathcal{A}$  in Figure 3. Note that without our DMD method, the ATR of every adversarial sample is 100%, and consequently, the AATR is also 100%. Here we have the following two main remarks.



**Figure 3.** The probability distribution of the transfer rate of all adversarial samples generated by the adversary  $\mathcal{A}$ . The horizontal axis is the transfer rate of a single adversarial sample, and the vertical axis is the frequency of samples at that transfer rate.

**Remark 1.** The empirical observations agree with Theorem 1. Due to this special property, apart from our differential adversarial training method, we can develop more DMD techniques to step further towards the DMR goal in FL and propose various trustworthy FL protocols under the presence of Byzantine failure.

**Remark 2.** The DMR improvement due to the proposed DMD technique increases when the client number  $K$  of FL increases. Intuitively, as the number of FL clients increases from 10 to 30, the distribution histograms are pushed toward the zero-ATR side, and the shape becomes more squeezed and narrow. This trend shows that as the client number increases, the sub-federated models become

more different from each other, which makes the differentially distributed client models more robust against attacks from the malicious FL client.

By applying differential adversarial training to the global model, we are able to control the attack transferability within an acceptable range. For example, we consider the FL setting of  $K = 40$  clients in Figure 3c. With our differential adversarial training method, we obtain an AATR of 19.47%. With this level of AATR, we can make sure that 45% of the adversarial samples generated by the malicious client have  $ATR \leq 10\%$ , while 75% samples have that  $ATR \leq 30\%$ . In particular, almost no adversarial samples can achieve an ATR of 100%.

### 5.2.2. Results on CIFAR-10

Similar to the analyses for MNIST, as is illustrated in Table 2, ARMOR can reduce ASR and AATR of benign clients over the CIFAR-10 dataset as well. For example, for  $K = 35$ , if we apply the DMD method, the ASR is reduced from 100% to 26.09% and the AATR can be reduced from 100% to 35.07%. Results on both datasets confirm that ARMOR is effective in reducing the vulnerability of FL client models against Byzantine-style adversarial attacks.

**Table 2.** Results of FL with/without DMD under different settings for CIFAR-10.

$K$	DMD	Acc (%)	Acc $_D$ (%)	ASR (%)	AATR (%)
10	×	73.27	73.27	100.00	100.00
	✓	51.58	65.53	24.70	41.75
15	×	74.48	74.48	100.00	100.00
	✓	54.66	69.27	27.59	40.32
20	×	73.53	73.53	100.00	100.00
	✓	56.19	70.17	28.96	39.57
25	×	71.71	71.71	100.00	100.00
	✓	52.94	68.18	27.91	38.36
30	×	70.61	70.61	100.00	100.00
	✓	52.67	67.83	27.49	36.79
35	×	72.28	72.28	100.00	100.00
	✓	51.88	66.72	26.09	35.07
40	×	72.27	72.27	100.00	100.00
	✓	53.02	68.14	28.75	37.18
45	×	70.21	70.21	100.00	100.00
	✓	51.65	66.94	25.53	33.71
50	×	70.07	70.07	100.00	100.00
	✓	51.70	67.98	26.71	34.54

Comparing Table 2 with Table 1, we can find that accuracy deterioration in Table 2 is much higher than that in Table 1. If we keep accuracy deterioration in Table 2 around the same level of Table 1, the reduction of ASR and AATR on CIFAR-10 will be less than that on MNIST. As a potential explanation, existing works [25,27,49] show that, under a centralized learning setting, various adversarial training methods are generally less effective on models trained on CIFAR-10 than models trained on MNIST. For example, in centralized settings [25], PGD adversarial training achieves a robustness of at most 45.80% (ASR = 54.20%) in CIFAR-10 while achieving a much better robustness of 89.30% (ASR = 10.70%) in MNIST.

### 5.3. Ablation Study

**Differentiation methods:** In Table 3, we perform the ablation studies on our framework over the MNIST dataset to confirm that both of the sub-federated model-based and

adversarial sample-based differentiation techniques are crucial in improving the DMR of FL. We note that, in the case where we directly perform adversarial sample based differentiation without sub-federated model generation, the objective function is reduced to the cross-entropy loss

$$\mathcal{L}_{ce}(Y) = \sum_{d_i^A \in \mathcal{D}_{adv,k}} \left[ \ell_{ce,i}(Y, d_i) + \ell_{ce,i}(Y, d_i^A) \right] \times \frac{1}{|\mathcal{D}_{adv,k}|}. \tag{29}$$

**Table 3.** Ablation study of the two different DMD steps on MNIST.  $\mathcal{L}_{ce}$  denotes the case where we only use adversarial sample-based model differentiation, and  $\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$  specifies the case of combining sub-federated model-based model differentiation with adversarial samples based model differentiation.

K	DMD	Acc (%)	Acc <sub>D</sub> (%)	ASR (%)	AATR (%)
10	$\mathcal{L}_{ce}$	95.82	95.17	49.06	56.04
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	94.65	94.02	31.64	40.89
15	$\mathcal{L}_{ce}$	95.84	94.73	49.47	54.96
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	93.83	92.61	24.57	32.27
20	$\mathcal{L}_{ce}$	95.68	95.18	50.50	54.93
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	92.66	89.20	20.15	27.00
25	$\mathcal{L}_{ce}$	95.99	94.71	50.28	54.43
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	91.20	88.35	20.38	26.54
30	$\mathcal{L}_{ce}$	95.69	94.87	50.27	54.05
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	91.93	88.81	20.13	25.78
35	$\mathcal{L}_{ce}$	95.79	94.77	49.41	53.03
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	91.85	87.94	18.60	23.95
40	$\mathcal{L}_{ce}$	95.62	94.52	48.49	52.08
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	91.54	87.57	18.59	23.55
45	$\mathcal{L}_{ce}$	95.54	94.45	49.79	53.14
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	91.87	87.94	20.21	24.95
50	$\mathcal{L}_{ce}$	95.56	94.82	49.10	52.29
	$\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$	91.35	87.31	18.72	23.17

Comparing the trends of ASR and AATR as the DMD method and client number  $K$  changes, we have two main observations here:

- First, the adversarial samples based model differentiation does have a positive influence on reducing ASR and AATR of benign clients. Nevertheless, the reduction is limited. However, when combined with the sub-federated model-based model differentiation, both ASR and AATR of benign clients are reduced significantly. For example, when  $K = 50$ , if we only apply  $\mathcal{L}_{ce}$  based DMD, the AATR is reduced from 100% to 52.29%. If we further combine  $\mathcal{L}_{ce}$  with  $\mathcal{L}_{sim}$ , the AATR is further reduced to 23.17%, which demonstrates that the key in enhancing DMR is the combination of sub-federated model generation and differential adversarial training.
- Second, the DMR improvement increases as the client number  $K$  of FL increases. Table 4 illustrates that ASR and AATR decrease as  $K$  increases. For example, when applying  $\mathcal{L}_{ce} + \lambda\mathcal{L}_{sim}$ , the AATR is 40.89% for 10 clients, 26.54% for 25 clients, and 23.17% for 50 clients. This is reasonable because as the client number increases, the diversity of sub-federated model is enlarged. As sub-federated models become more different from each other, the differentially distributed client models become more robust against attacks from the malicious client, resulting in additional DMR improvements.

**Table 4.** Ablation study of different DMD parameters on MNIST.  $\lambda$  is the differentiation factor of the sub-federated model.  $\eta$  is the proportion of the sub-federated model.  $p$  is the probability of Bernoulli distribution.

$K$	$\lambda$	$\eta$	$p$	Acc (%)	Acc $_{\mathcal{D}}$ (%)	ASR (%)	AATR (%)
10	500	0.25	0.10	94.65	94.02	31.64	40.89
	600	0.25	0.10	88.13	84.54	<b>15.29</b>	<b>27.66</b>
	600	0.35	0.10	89.13	84.04	16.38	28.32
	300	0.25	0.20	92.49	90.63	21.67	32.30
	350	0.25	0.20	90.12	84.66	16.76	28.82
15	500	0.25	0.10	93.83	92.61	24.57	32.27
	600	0.25	0.10	90.18	86.33	14.82	23.64
	600	0.35	0.10	87.66	83.45	14.85	23.90
	300	0.25	0.20	91.75	88.09	17.25	25.97
	350	0.25	0.20	87.37	83.24	<b>14.08</b>	<b>23.05</b>
20	500	0.25	0.10	92.66	89.20	20.15	27.00
	600	0.25	0.10	90.86	85.59	17.06	24.50
	600	0.35	0.10	90.21	84.83	15.99	23.38
	300	0.25	0.20	92.78	89.24	19.43	26.34
	350	0.25	0.20	89.14	83.63	<b>14.68</b>	<b>21.75</b>
25	500	0.25	0.10	91.20	88.35	20.38	26.54
	600	0.25	0.10	86.85	82.71	14.82	21.16
	600	0.35	0.10	88.04	82.57	14.35	20.67
	300	0.25	0.20	90.33	85.32	15.43	21.99
	350	0.25	0.20	87.98	82.82	<b>14.02</b>	<b>20.57</b>
30	500	0.25	0.10	91.93	88.81	20.13	25.78
	600	0.25	0.10	88.53	82.72	14.12	19.98
	600	0.35	0.10	87.89	82.41	15.97	21.80
	300	0.25	0.20	90.09	85.20	16.93	22.67
	350	0.25	0.20	89.26	83.52	<b>13.59</b>	<b>19.43</b>
35	500	0.25	0.10	91.85	87.94	18.60	23.95
	600	0.25	0.10	89.06	84.06	15.74	21.04
	600	0.35	0.10	88.58	84.02	16.48	21.62
	300	0.25	0.20	90.10	85.44	16.71	21.94
	350	0.25	0.20	90.35	85.43	<b>15.21</b>	<b>20.45</b>
40	500	0.25	0.10	91.54	87.57	18.59	23.55
	600	0.25	0.10	89.11	84.41	14.57	19.47
	600	0.35	0.10	87.65	82.68	14.92	20.05
	300	0.25	0.20	89.56	85.08	17.60	22.63
	350	0.25	0.20	87.06	81.11	<b>14.34</b>	<b>19.37</b>

Table 4. Cont.

$K$	$\lambda$	$\eta$	$p$	Acc (%)	Acc <sub>D</sub> (%)	ASR (%)	AATR (%)
45	500	0.25	0.10	91.87	87.94	20.21	24.95
	600	0.25	0.10	88.01	83.20	<b>13.95</b>	<b>18.44</b>
	600	0.35	0.10	87.57	83.01	15.13	19.72
	300	0.25	0.20	89.07	84.32	15.42	19.99
	350	0.25	0.20	87.15	82.24	14.01	18.70
50	500	0.25	0.10	91.35	87.31	18.72	23.17
	600	0.25	0.10	88.13	83.22	16.02	20.52
	600	0.35	0.10	88.69	83.47	15.46	19.91
	300	0.25	0.20	89.26	84.44	16.51	20.84
	350	0.25	0.20	87.07	81.89	<b>14.60</b>	<b>18.88</b>

**The impact of DMD parameters:** As is shown in Table 4, we choose five different combinations of the proportion of sub-federated model  $\eta$ , the differentiation factor of sub-federated model  $\lambda$  and the probability  $p$  of Bernoulli distribution: (1)  $\lambda = 500, \eta = 0.25, p = 0.10$ , (2)  $\lambda = 600, \eta = 0.25, p = 0.10$ , (3)  $\lambda = 600, \eta = 0.35, p = 0.10$ , (4)  $\lambda = 300, \eta = 0.25, p = 0.20$ , (5)  $\lambda = 350, \eta = 0.25, p = 0.20$ . When the differentiation factor  $\lambda$  increases, clients' models tend to produce more mispredictions. Hence, the server needs to carefully adjust  $\lambda$  to retain a practical level of utility while improving the robustness for the differentiated client models. Comparing the results of different DMD parameters settings, we have two main observations here:

- First, the DMR of FL client models is strengthened as the differentiation factor  $\lambda$  increases. We fix  $\eta = 0.25$  and  $p = 0.10$ , then set  $\lambda = 500$  and  $\lambda = 600$ , respectively. Similarly, we fix  $\eta = 0.25$  and  $p = 0.20$ , then set  $\lambda = 300$  and  $\lambda = 350$ , respectively. We find that for  $p = 0.10$  (resp.,  $p = 0.20$ ), DMD with  $\lambda = 600$  (resp.,  $\lambda = 350$ ) constantly leads to lower ASR and AATR than DMD with  $\lambda = 500$  (resp.,  $\lambda = 300$ ), which validates the positive effect of the sub-federated model differentiation.
- Second, we observe that as the proportion of sub-federated model  $\eta$  increases from  $1/K$ , the overall accuracy of model also increases. However, as long as the sub-federated model is of enough utility, further increasing  $\eta$  does not help much. We fix  $\lambda = 600$  and  $p = 0.10$ , then set  $\eta = 0.25$  and  $\eta = 0.35$ , respectively. We find that slight change in  $\eta$  does not lead to much differences in ASR and AATR. However, choosing only one single local model as the sub-federated model (i.e.,  $\eta = 1/K$ ) leads to significant deterioration of performance.

From the ablation study results, we conclude that the combination of sub-federated model based differentiation and adversarial sample based differentiation is effective in reducing both the ASR and AATR while maintaining the overall accuracy of the benign client models.

## 6. Discussion

We aim at ensuring the safety of FL systems when Byzantine failure occurs in real-life applications. Such failure is very likely to occur in real-world FL systems, and can be life-threatening when critical inference devices are attacked. For example, in the joint training of autonomous driving systems, each autonomous vehicle is deployed with an NN model under an FL setting. If there exists a malicious client in this system, the malicious client can easily generate adversarial samples to attack all other vehicles, potentially causing serious accidents. Consequently, all other vehicles in the system using the same model will collectively produce wrong predictions on these adversarial traffic signs, potentially

causing serious accidents. ARMOR is one of the first works to explore defense mechanisms against such attacks, and can be essential in developing safe and trustworthy FL protocols.

## 7. Conclusions

In this work, we study the differential robustness of NN models against adversarial attacks in FL systems in the presence of Byzantine failures. By providing with clients carefully differentiated NN models, the main objective of the proposed ARMOR framework is to reduce the risks of corrupted FL clients launching white-box adversarial attacks against benign clients. By carefully designing the experiments and ablation studies under various FL settings, we show that techniques proposed in the ARMOR framework are indeed effective in reducing both the ASR and AATR of adversarial samples generated by the corrupted clients.

**Author Contributions:** Conceptualization, Y.Z., J.L., Z.G., X.L. and S.B.; methodology, Y.Z., S.B. and X.L.; software, Y.Z.; investigation, B.Z.; writing—original draft preparation, Y.Z. and B.Z.; writing—review and editing, Y.Z., S.B., J.L., Z.G., X.L. and B.Z.; supervision, J.L., Z.G., X.L. and S.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the National Key R&D Program of China under Grants 2021YFB2700200, in part by the National Natural Science Foundation of China under Grant 62202028, U21B2021, 61972018, and 61932014. This work was supported in part by PowerTensors.AI.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** A proof-of-concept implementation of our technique is available from <https://github.com/ARMOR-FL/ARMOR> (accessed on 19 December 2022). The data used to support the findings of this study are included within the article.

**Acknowledgments:** Our deepest gratitude goes to the anonymous reviewers for their careful work.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Bonawitz, K.A.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, B.; et al. Towards Federated Learning at Scale: System Design. In Proceedings of the Machine Learning and Systems 1 (MLSys 2019), Stanford, CA, USA, 31 March–2 April 2019.
2. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [\[CrossRef\]](#)
3. Long, G.; Tan, Y.; Jiang, J.; Zhang, C. Federated Learning for Open Banking. In *Federated Learning—Privacy and Incentive*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2020; Volume 12500, pp. 240–254.
4. Guo, P.; Wang, P.; Zhou, J.; Jiang, S.; Patel, V.M. Multi-Institutional Collaborations for Improving Deep Learning-Based Magnetic Resonance Image Reconstruction Using Federated Learning. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) 2021, Nashville, TN, USA, 20–25 June 2021; pp. 2423–2432.
5. Du, Z.; Wu, C.; Yoshinaga, T.; Yau, K.A.; Ji, Y.; Li, J. Federated Learning for Vehicular Internet of Things: Recent Advances and Open Issues. *IEEE Open J. Comput. Soc.* **2020**, *1*, 45–61. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Pokhrel, S.R.; Choi, J. Federated Learning With Blockchain for Autonomous Vehicles: Analysis and Design Challenges. *IEEE Trans. Commun.* **2020**, *68*, 4734–4746. [\[CrossRef\]](#)
7. Li, Q.; He, B.; Song, D. Model-Contrastive Federated Learning. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) 2021, Nashville, TN, USA, 20–25 June 2021; pp. 10713–10722.
8. Lai, F.; Zhu, X.; Madhyastha, H.V.; Chowdhury, M. Oort: Efficient Federated Learning via Guided Participant Selection. In Proceedings of the Operating Systems Design and Implementation (OSDI) 2021, Virtual, 14–16 July 2021; pp. 19–35.
9. Zhang, C.; Li, S.; Xia, J.; Wang, W.; Yan, F.; Liu, Y. BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. In Proceedings of the USENIX Security 2020, San Diego, CA, USA, 12–14 August 2020; pp. 493–506.
10. Wei, K.; Li, J.; Ding, M.; Ma, C.; Yang, H.H.; Farokhi, F.; Jin, S.; Quek, T.Q.S.; Poor, H.V. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3454–3469. [\[CrossRef\]](#)
11. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks Against Machine Learning Models. In Proceedings of the SP 2017, San Jose, CA, USA, 22–26 May 2017; pp. 3–18.



12. Nasr, M.; Shokri, R.; Houmansadr, A. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In Proceedings of the SP 2019, San Francisco, CA, USA, 19–23 May 2019; pp. 739–753.
13. Zhang, Y.; Jia, R.; Pei, H.; Wang, W.; Li, B.; Song, D. The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) 2020, Seattle, WA, USA, 13–19 June 2020; pp. 250–258.
14. Fang, M.; Cao, X.; Jia, J.; Gong, N.Z. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In Proceedings of the USENIX Security 2020, San Diego, CA, USA, 12–14 August 2020; pp. 1605–1622.
15. Shejwalkar, V.; Houmansadr, A. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning. In Proceedings of the Network and Distributed System Security Symposium (NDSS) 2021, Virtual, 21–25 February 2021.
16. Kido, H.; Yanagisawa, Y.; Satoh, T. Protection of Location Privacy using Dummies for Location-based Services. In Proceedings of the International Conference on Data Engineering (ICDE) 2005, Tokyo, Japan, 3–4 April 2005; p. 1248.
17. Blanchard, P.; Mhamdi, E.M.E.; Guerraoui, R.; Stainer, J. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In Proceedings of the NeurIPS 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 119–129.
18. Yin, D.; Chen, Y.; Ramchandran, K.; Bartlett, P.L. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In Proceedings of the International Conference on Machine Learning (ICML) 2018, Stockholm, Sweden, 10–15 July 2018; pp. 5636–5645.
19. Pillutla, K.; Kakade, S.M.; Harchaoui, Z. Robust aggregation for federated learning. *IEEE Trans. Signal Process.* **2022**, *70*, 1142–1154. [[CrossRef](#)]
20. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How To Backdoor Federated Learning. In Proceedings of the AISTATS 2020, Palermo, Italy, 26–28 August 2020; Volume 108, pp. 2938–2948.
21. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the AISTATS 2017, Fort Lauderdale, FL, USA, 20–22 April 2017; Volume 54, pp. 1273–1282.
22. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. In Proceedings of the International Conference on Learning Representations (ICLR) 2015, San Diego, CA, USA, 7–9 May 2015.
23. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.J.; Fergus, R. Intriguing properties of neural networks. In Proceedings of the International Conference on Learning Representations (ICLR) 2014, Banff, AB, Canada, 14–16 April 2014.
24. Ru, B.; Cobb, A.D.; Blaas, A.; Gal, Y. BayesOpt Adversarial Attack. In Proceedings of the International Conference on Learning Representations (ICLR) 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
25. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. In Proceedings of the ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
26. Miyato, T.; Dai, A.M.; Goodfellow, I.J. Adversarial Training Methods for Semi-Supervised Text Classification. In Proceedings of the International Conference on Learning Representations (ICLR) 2017, Toulon, France, 24–26 April 2017.
27. Shafahi, A.; Najibi, M.; Ghiasi, A.; Xu, Z.; Dickerson, J.P.; Studer, C.; Davis, L.S.; Taylor, G.; Goldstein, T. Adversarial training for free! In Proceedings of the NeurIPS 2019, Vancouver, CA, Canada, 8–14 December 2019; pp. 3353–3364.
28. Zhang, D.; Zhang, T.; Lu, Y.; Zhu, Z.; Dong, B. You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle. In Proceedings of the NeurIPS 2019, Vancouver, CA, Canada, 8–14 December 2019; pp. 227–238.
29. Zhu, C.; Cheng, Y.; Gan, Z.; Sun, S.; Goldstein, T.; Liu, J. FreeLB: Enhanced Adversarial Training for Natural Language Understanding. In Proceedings of the International Conference on Learning Representations (ICLR) 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
30. Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; Zhao, T. SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization. In Proceedings of the ACL 2020, Virtual, 5–10 July 2020; pp. 2177–2190.
31. Qin, C.; Martens, J.; Gowal, S.; Krishnan, D.; Dvijotham, K.; Fawzi, A.; De, S.; Stanforth, R.; Kohli, P. Adversarial Robustness through Local Linearization. In Proceedings of the NeurIPS 2019, Vancouver, CA, Canada, 8–14 December 2019; pp. 13824–13833.
32. Zizzo, G.; Rawat, A.; Sinn, M.; Buesser, B. FAT: Federated Adversarial Training. *arXiv* **2020**, arXiv:2012.01791.
33. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S.B. Analyzing Federated Learning through an Adversarial Lens. In Proceedings of the ICML 2019, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 634–643.
34. Li, L.; Xu, W.; Chen, T.; Giannakis, G.B.; Ling, Q. RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets. In Proceedings of the AAAI 2019, Honolulu, HI, USA, 27 January–1 February 2019; pp. 1544–1551.
35. Kerkouche, R.; Ács, G.; Castelluccia, C. Federated Learning in Adversarial Settings. *arXiv* **2020**, arXiv:2010.07808.
36. Fu, S.; Xie, C.; Li, B.; Chen, Q. Attack-Resistant Federated Learning with Residual-based Reweighting. *arXiv* **2019**, arXiv:1912.11464.
37. Chen, Y.; Su, L.; Xu, J. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *Proc. ACM Meas. Anal. Comput. Syst.* **2017**, *1*, 44:1–44:25. [[CrossRef](#)]
38. Wang, H.; Sreenivasan, K.; Rajput, S.; Vishwakarma, H.; Agarwal, S.; Sohn, J.; Lee, K.; Papailiopoulos, D.S. Attack of the Tails: Yes, You Really Can Backdoor Federated Learning. In Proceedings of the NeurIPS 2020, Virtual, 6–12 December 2020.
39. Zhou, M.; Wu, J.; Liu, Y.; Liu, S.; Zhu, C. DaST: Data-Free Substitute Training for Adversarial Attacks. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) 2020, Seattle, WA, USA, 13–19 June 2020; pp. 231–240.

40. Wang, W.; Yin, B.; Yao, T.; Zhang, L.; Fu, Y.; Ding, S.; Li, J.; Huang, F.; Xue, X. Delving into Data: Effectively Substitute Training for Black-box Attack. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) 2021, Nashville, TN, USA, 20–25 June 2021; pp. 4761–4770.
41. Ma, C.; Chen, L.; Yong, J. Simulating Unknown Target Models for Query-Efficient Black-Box Attacks. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) 2021, Nashville, TN, USA, 20–25 June 2021; pp. 11835–11844.
42. Li, X.; Li, J.; Chen, Y.; Ye, S.; He, Y.; Wang, S.; Su, H.; Xue, H. QAIR: Practical Query-Efficient Black-Box Attacks for Image Retrieval. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) 2021, Nashville, TN, USA, 20–25 June 2021; pp. 3330–3339.
43. Fawzi, A.; Fawzi, H.; Fawzi, O. Adversarial vulnerability for any classifier. In Proceedings of the NeurIPS 2018, Montreal, QC, Canada, 3–8 December 2018; pp. 1186–1195.
44. Tramèr, F.; Papernot, N.; Goodfellow, I.J.; Boneh, D.; McDaniel, P.D. The Space of Transferable Adversarial Examples. *arXiv* **2017**, arXiv:1704.03453.
45. Fawzi, A.; Fawzi, O.; Frossard, P. Analysis of classifiers' robustness to adversarial perturbations. *Mach. Learn.* **2018**, *107*, 481–508. [[CrossRef](#)]
46. LeCun, Y.; Cortes, C.; Burges, C.J.C. The MNIST Database of Handwritten Digits. 1998. Available online: <http://yann.lecun.com/exdb/mnist/> (accessed on 20 January 2022).
47. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; University of Toronto: Toronto, ON, Canada, 2009.
48. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations (ICLR) 2015, San Diego, CA, USA, 7–9 May 2015.
49. Wong, E.; Rice, L.; Kolter, J.Z. Fast is better than free: Revisiting adversarial training. In Proceedings of the International Conference on Learning Representations (ICLR) 2020, Addis Ababa, Ethiopia, 26–30 April 2020.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.