Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.054820

ARTICLE

Privacy-Preserving Large-Scale AI Models for Intelligent Railway Transportation Systems: Hierarchical Poisoning Attacks and Defenses in Federated Learning

Yongsheng Zhu[1,2,*], Chong Liu[3,4], Chunlei Chen[5], Xiaoting Lyu[3,4], Zheng Chen[3,4], Bin Wang[6], Fuqiang Hu[3,4], Hanxi Li[3,4], Jiao Dai[3,4], Baigen Cai[1] and Wei Wang[3,4]

[1]School of Automation and Intelligence, Beijing Jiaotong University, Beijing, 100044, China

[2]Institute of Computing Technologies, China Academy of Railway Sciences Corporation Limited, Beijing, 100081, China

[3]School of Computer Science and Technology, Beijing Jiaotong University, Beijing, 100044, China

[4]Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing, 100044, China

[5]Institute of Infrastructure Inspection, China Academy of Railway Sciences Corporation Limited, Beijing, 100081, China

[6]Zhejiang Key Laboratory of Multi-Dimensional Perception Technology, Application and Cybersecurity, Hangzhou, 310053, China

*Corresponding Author: Yongsheng Zhu. Email: zhuys@rails.cn

Received: 08 June 2024 Accepted: 29 July 2024 Published: 27 September 2024

ABSTRACT

The development of Intelligent Railway Transportation Systems necessitates incorporating privacy-preserving mechanisms into AI models to protect sensitive information and enhance system efficiency. Federated learning offers a promising solution by allowing multiple clients to train models collaboratively without sharing private data. However, despite its privacy benefits, federated learning systems are vulnerable to poisoning attacks, where adversaries alter local model parameters on compromised clients and send malicious updates to the server, potentially compromising the global model's accuracy. In this study, we introduce PMM (Perturbation coefficient Multiplied by Maximum value), a new poisoning attack method that perturbs model updates layer by layer, demonstrating the threat of poisoning attacks faced by federated learning. Extensive experiments across three distinct datasets have demonstrated PMM's ability to significantly reduce the global model's accuracy. Additionally, we propose an effective defense method, namely CLBL (Cluster Layer By Layer). Experiment results on three datasets have confirmed CLBL's effectiveness.

KEYWORDS

Privacy-preserving; intelligent railway transportation system; federated learning; poisoning attacks; defenses

1 Introduction

Federated Learning (FL) [1] is a new paradigm in the field of machine learning. Its core is that multiple clients collaboratively train a shared machine learning model, namely the global model, to improve the performance and accuracy of the model. Under the framework of FL, clients can keep their data local, which not only protects data privacy but also reduces the need for data transmission.

At the same time, the server takes on the responsibility of maintaining and updating the global model. Each iteration of FL follows three basic steps: The process begins with the server broadcasting the latest global model parameters to the selected clients participating in the iteration. Subsequently, each client uses its own local dataset to train the received global model and then uploads the model update to the server. Finally, the server aggregates the model updates from each client according to predetermined rules and uses this aggregated information to update the global model. With the increasing awareness of privacy protection and data security, FL is gradually becoming a technology that technology companies are vying to adopt. Recent advancements have expanded the applicability of FL across various domains. In intelligent transportation systems, privacy-preserving methods like CreditCoin have demonstrated secure and efficient communications in smart vehicles [2]. Integrating blockchain technology into FL systems enhances transparency and security of data transactions, as shown by studies on blockchain-based incentive networks. Automated detection of vulnerabilities in smart contracts, such as the models proposed by ContractWard, highlights FL's potential to improve security measures in decentralized applications without compromising data privacy [3–6]. In network security, tools like BotMark have used hybrid analysis techniques to identify botnet behaviors through flow-based and graph-based traffic patterns [7], enhancing detection and prevention mechanisms. The integration of FL with advanced machine learning techniques, such as heterogeneous graph attention auto-encoders explored in HGATE, shows potential in improving model performance with complex and diverse data structures [8]. This underscores FL's versatility in addressing a range of challenges in data analysis and model training.

In the field of rail transportation, ensuring the smooth operation of railway equipment is crucial for maintaining efficient services. Train stations are spread across diverse geographical areas, resulting in significant variations in environmental conditions such as temperature and humidity that the equipment is exposed to. To enhance the accuracy of equipment anomaly detection, it is beneficial to train detection models using data distributed across different train stations. By employing FL, we can effectively integrate equipment data from various train stations, thereby significantly improving the performance of anomaly detection models. This approach not only enhances detection capabilities but also ensures data privacy and security, as local data at each train station remains in its original location.

As shown in Fig. 1, FL has been extensively adopted in the railway industry, demonstrating significant advantages across various aspects. Firstly, FL enables railway companies to collect real-time operational data from trains across different locations. This data can be used to optimize train scheduling and operational plans, effectively reducing traffic congestion and improving train punctuality. Secondly, FL facilitates real-time monitoring of track health, enabling the detection of issues such as cracks and deformations, and predicting potential future failures. This allows for timely maintenance measures, ensuring the safety and stability of railway lines. Finally, FL allows railway companies to analyze passenger behavior and preferences, leading to personalized services such as customized itinerary suggestions and optimized carriage layouts, thereby significantly enhancing the passenger travel experience. These applications illustrate that FL holds broad prospects and substantial practical value in the railway industry.

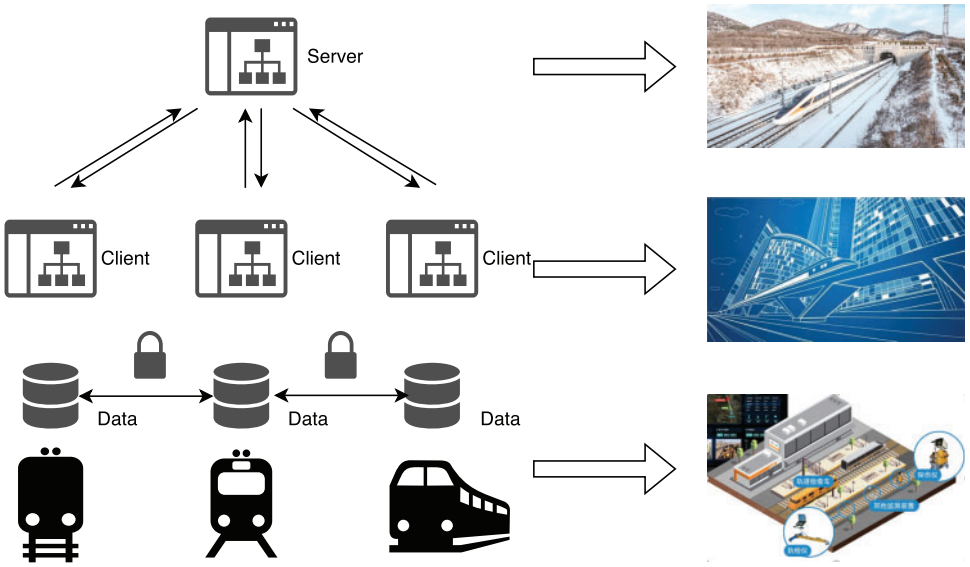


Figure 1: The application of FL in railways

However, due to the distributed nature of FL, it is vulnerable to poisoning attacks [9–12]. Attackers can manipulate certain clients to upload malicious model updates [13,14], disrupting the security of the global model. In untargeted poisoning attacks, the compromised global model suffers from reduced accuracy. Conversely, in targeted poisoning attacks, the malicious updates degrade the performance of the model specifically on attacker-chosen inputs, leaving other inputs largely unaffected.

Current poisoning attacks [15–19] have demonstrated some degree of success. For instance, Min-Max [20] constructs malicious updates by ensuring the maximum distance between a malicious update and any normal update is less than the maximum distance between any two normal updates, thereby subtly affecting the global model. However, this method's effectiveness is limited when using Krum on the FMNIST dataset. To address this, we propose a new attack method named *PMM* (Perturbation coefficient Multiplied by Maximum value). The method performs targeted perturbations on each layer of the model update and scales it by a perturbation coefficient λ to construct malicious updates, with the goal of more effectively reducing the accuracy of the global model.

Existing defense methods [21–23] can prevent a small number of malicious clients from launching poisoning attacks on FL and detect these clients by observing statistical differences in model updates between malicious and benign participants. However, these methods have notable limitations. For example, in subsequent experiments, Krum fails to effectively defend against *PMM*. To address these shortcomings, we propose a novel defense method called *CLBL* (Cluster Layer By Layer). This method involves clustering model updates layer by layer, selecting the class with the most elements, computing its average, and using this average as the global model update.

To evaluate the effectiveness of the proposed method, we conduct experimental verification. Specifically, we find that *PMM* shows effectiveness on three large-scale benchmark datasets (MNIST, FMNIST, and CIFAR10) against three different aggregation algorithms, FedAVG, Krum, and Trimmed_mean. Experimental data show that *PMM* outperforms several existing benchmark attack methods, including LIE, Min-Max, and Min-Sum, in attack performance to a certain extent. In addition, we find that *CLBL* successfully resists poisoning attack methods such as LIE, Min-Max, and

Min-Sum in the experiment and outperforms other aggregation algorithms such as FedAVG, Krum, and Trimmed_mean, showing its superiority in defending against attacks. These results highlight the potential of *PMM* and *CLBL* in practical applications.

In summary, our key contributions are as follows:

- We propose a novel attack method, named *PMM*. The core of this method involves sorting the model updates generated by a malicious client controlled by the attacker after the normal training. The maximum value from the sorted updates is then selected and multiplied by a perturbation coefficient λ to construct a malicious model update, which is subsequently uploaded to the server.
- We propose a defense method, named *CLBL*. This method clusters the model updates layer by layer, identifies the class containing the majority of the model updates, and computes the average of all updates within this class to determine the update for the global model.
- We conduct a comprehensive empirical evaluation of *PMM* and *CLBL*. The experimental results demonstrate that *PMM* can significantly reduce the accuracy of the global model, whereas *CLBL* can accurately and effectively protect the global model, preserving its accuracy.

2 Related Work

2.1 Federated Learning

FL can be categorized into two distinct settings: cross-device FL and cross-silo FL [24]. Cross-device FL is ideal for scenarios involving a large number of devices, each with limited computational resources. Cross-silo FL is better suited for situations where there are fewer entities, each with substantial computational power. In summary, cross-device FL and cross-silo FL address different application scenarios. The choice of setting depends on the specific requirements of the application. Cross-device FL is optimal for environments with numerous distributed devices that have constrained computing capabilities, whereas cross-silo FL is preferable for contexts where computational resources are more centralized but fewer in number.

We now consider a standard FL setting [1,25], consisting of a central server and n clients, each owning local data. The workflow of FL in the t-th global training round is as follows:

- **Step 1:** The server broadcasts the current global model ω_{t-1} to all clients, or optionally, to a subset of clients.
- **Step 2:** The clients train local models using their local data and send the model updates back to the server.
- **Step 3:** The server aggregates the local model updates using a specified aggregation rule and updates the global model accordingly:

$$\omega_t = \omega_{t-1} - \eta \cdot AGR(g_1^t,\ g_2^t, \ldots, g_i^t) \tag{1}$$

Here, ω_t represents the global model parameters at round t, η denotes the learning rate, AGR signifies the aggregation rule, and g_i^t denotes the model update from the i-th client in round t.

2.2 Aggregation Algorithms for FL

In non-adversarial FL settings [1], FedAVG [26] is a widely used aggregation method that averages local model updates in each global epoch and integrates the results into the global model. However, in adversarial FL with a trusted server, where there are n clients including m malicious clients, even a single malicious client can compromise the system when using FedAVG. Ensuring the security and stability of FL systems against poisoning attacks is crucial. The predominant defense strategy involves Byzantine-robust aggregation algorithms, which can detect [27–30] and filter out updates from malicious clients, thereby reducing their impact on the global model.

Trimmed Mean [21] is a coordinate-wise aggregation algorithm. In Trimmed Mean, local model updates from each client are first sorted, and then a certain proportion of extreme values are removed by excluding a specified percentage of the smallest and largest values. The remaining model updates are then averaged to compute the global model. Trimmed Mean is particularly effective in handling imbalanced or anomalous data in FL, preventing malicious clients from significantly affecting the global model.

Krum [22] is an aggregation algorithm that uses the sum of squared distances to select global model updates. In each global training epoch, Krum calculates the distances between each client's model updates and those of other clients. It then selects the updates from the client with the smallest anomaly score as the global model update. Anomaly scores are computed by comparing the distances of each client's updates with those of other clients.

Median [21] is another coordinate-wise aggregation technique, which computes the median of the updated values across all dimensions. This approach effectively mitigates the influence of outliers on gradient updates, enhancing the stability and reliability of the global model.

FLTrust [31] requires the server to maintain a small yet high-quality clean dataset to train a reliable model, which then guides the aggregation process. However, maintaining such a high-standard dataset poses a significant challenge.

2.3 Poisoning Attacks on FL

Existing research indicates that FL is vulnerable to poisoning attacks, where malicious clients compromise the global model's accuracy by submitting malicious model updates during the second step of the FL process [9,32–35]. In this paper, we examine three notable poisoning attack methods: LIE, Min-Max, and Min-Sum.

LIE [32] demonstrates that attackers can reduce the global model's accuracy by making subtle, carefully designed modifications to a specific proportion of parameters, thereby creating malicious model updates. The core principle of LIE involves calculating the mean and standard deviation of model updates produced by malicious clients during normal training. Malicious updates are then constructed by subtracting a multiple of the standard deviation from the mean, which minimizes the distinguishability between malicious and normal model updates.

Min-Max and **Min-Sum** [20] generate malicious model updates by examining normal model updates in each training epoch to reduce the global model's accuracy. Min-Max crafts updates such that the maximum distance between any malicious update and any normal update is less than the maximum distance between any two normal updates. Min-Sum, on the other hand, constructs updates so that the sum of squared distances between any malicious update and all other updates is smaller than the sum of squared distances between any two normal updates. Both methods aim to minimize the discrepancy between malicious and normal updates, thus evading anomaly detection systems.

However, Min-Sum imposes stricter constraints on malicious updates compared to Min-Max, making anomalies even harder to detect.

3 Threat Model

3.1 Attacker's Objective

In line with numerous studies on poisoning attacks [20,32], we examine the attacker's objective of generating and sending malicious model updates to the server. The goal is to compromise the utility of the global model, thereby reducing its accuracy. This type of attack, known as an untargeted attack, aims to render the learned global model unusable.

3.2 Attacker's Capability

We assume there are a total of n clients, with m of them being malicious. We further assume that the number of normal clients exceeds the number of malicious clients, i.e., $(m/n) < 0.5$. This assumption is critical because if malicious clients constitute the majority, no Byzantine-robust FL system can effectively resist poisoning attacks. Additionally, we consider that attackers have the capability to craft malicious gradients that can detrimentally affect the global model. Furthermore, it is assumed that malicious clients can collaborate to launch attacks, thereby increasing both the effectiveness and stealthiness of their malicious activities.

3.3 Attacker's Knowledge

In FL systems, an attacker's knowledge is generally categorized into two common settings: partial-knowledge and full-knowledge [33]. In the partial-knowledge setting, the attacker is presumed to have knowledge of the global model, the loss function, and the local training data and model updates on malicious clients. In the full-knowledge setting, the attacker additionally has access to the local training data and model updates of all clients, as well as the server's aggregation rule. This study concentrates on poisoning attacks within the partial-knowledge setting.

4 Attack Method

In this section, we first introduce the intuition behind our attack method, and then introduce PMM in detail, which consists of three steps in every global epoch.

Intuition: Defense methods for FL typically weaken or remove malicious model updates based on one or more of the following criteria: (1) distance from benign model updates, (2) distributional differences with benign model updates, and (3) differences in L_p-norms between benign and malicious model updates. Fig. 2 illustrates our attack principles. Algorithm 1 outlines the procedure of PMM.

- **Step 1. Normal training to obtain model updates.** Initially, the malicious clients controlled by the attacker receive the global model and perform normal training using their local data to generate model updates.

$$g_i^t = \nabla L_i(\omega_t) \tag{2}$$

  Here, g_i^t represents the model update generated by the i-th client in round t, and $L_i(\omega_t)$ denotes the loss function on client i. Eq. (2) illustrates the process of obtaining model updates through client training.

- **Step 2. Constructing malicious model updates.** The attacker sorts the model updates from all malicious clients layer by layer. After sorting, the attacker selects the maximum value from

each layer of the model updates. This maximum value is then multiplied by the perturbation coefficient λ to construct the malicious model update.

$$\hat{g}_{i,j}^{t} = \lambda \times \left\{ g_{i,j}^{t} \mid i \in \textit{Malicious Clients}, j = 1,\ 2, \ldots, k \right\}_{\max} \tag{3}$$

Here, $g_{i,j}^{t}$ represents the model update of the j-th layer of the i-th client in round t, where the model consists of k layers, and $\hat{g}_{i,j}^{t}$ denotes the malicious model update of the j-th layer of the i-th client in round t. Eq. (3) details the process of constructing malicious model updates layer by layer. This approach ensures that the selected values remain within the range of normal model updates, enhancing the stealthiness of the attack. The perturbation coefficient λ controls the extent of disruption caused by the malicious model updates to the global model.

- **Step 3. Uploading malicious model updates.** After constructing the malicious model updates, the malicious clients upload these updates to the server, thereby disrupting the global model.

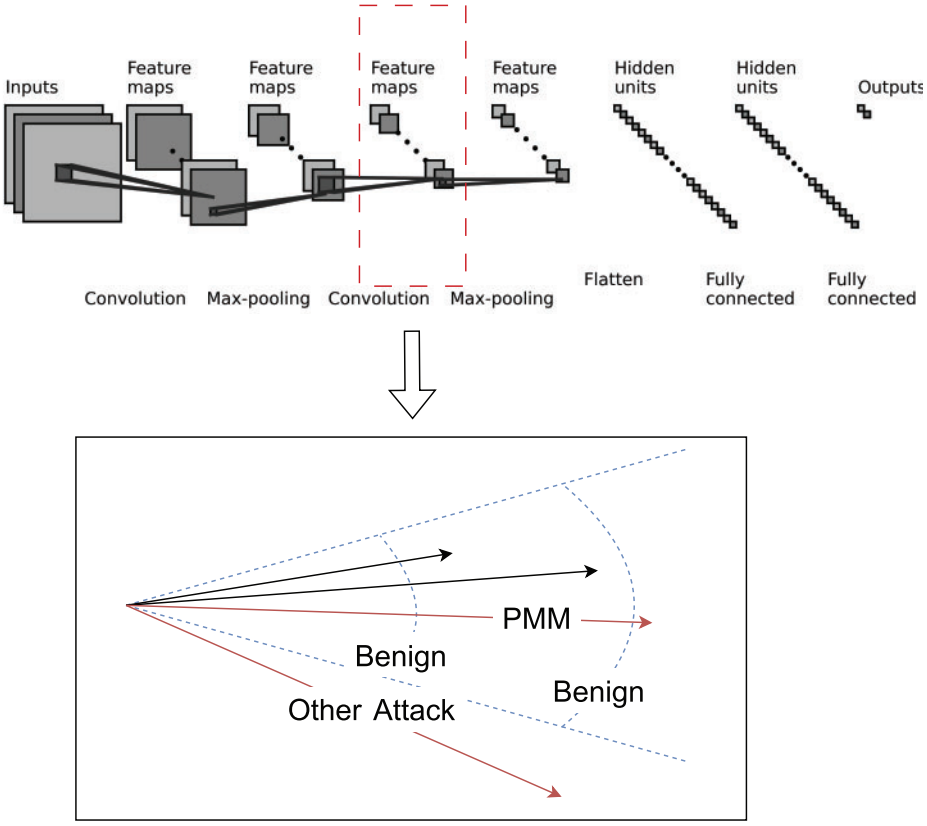


Figure 2: The attack principle of *PMM*

Algorithm 1: PMM

Input: global model parameter ω_t, set of malicious clients, global epoch T

Output:

1. **for** $t \in [1, T]$ **do**
2. Receive the global model parameter ω_t
3. Train the local model using ω_t
4. **for** $j \in [1, k]$ **do**
5. **for** $i \in$ *Malicious Clients* **do**
6. $\hat{g}_{i,j}^t \leftarrow max(\hat{g}_{i,j}^t, g_{i,j}^t)$
7. Construct malicious model updates using perturbation coefficients. $\hat{g}_{i,j}^t = \lambda \times \hat{g}_{i,j}^t$
8. **end for**
9. **end for**
10. **end for**
11. Send the malicious model update $\hat{g}^t$ to server

To calculate the time complexity of PMM, we will evaluate the complexity of each step sequentially.

- **Step 1: Normal Training to Obtain Model Updates.** In this step, the malicious clients receive the global model and perform normal training using their local data to generate model updates. The time complexity of this step depends on the training process. Assuming the time complexity for each client to perform one gradient descent update is $O(D)$, where D is the size of the dataset, the total time complexity for m clients is: $O(m \cdot D)$.
- **Step 2: Constructing Malicious Model Updates.** In this step, the attacker sorts the model updates from all malicious clients layer by layer. Each layer has m updates, and the time complexity for sorting is $O(m \log m)$. For k layers, each layer is sorted and the maximum value is selected, making the total time complexity:

$$O(k \cdot m \log m) \tag{4}$$

- **Step 3: Uploading Malicious Model Updates.** In this step, the malicious clients upload the constructed malicious model updates to the server. Assuming the time complexity for uploading each model update is $O(1)$, the total time complexity for m malicious clients is: $O(m)$.

Combining the time complexities of all the steps, the total time complexity of the process is: $O(m \cdot D) + O(k \cdot m \log m) + O(m)$. Each layer has the same number of model updates in PMM, the total time complexity can be simplified to: $O(m \cdot D) + O(k \cdot m \log m) + O(m)$. Therefore, the overall time complexity is:

$$O(m \cdot D + k \cdot m \log m) \tag{5}$$

In summary, PMM starts by engaging malicious clients in normal training to acquire model updates. These updates are meticulously sorted layer by layer, where the maximum value from each layer is carefully selected. This selected value is then scaled by a perturbation coefficient λ to craft malicious model updates that closely resemble legitimate updates but are strategically altered to degrade the global model's performance. Finally, these crafted updates are transmitted to the server, effectively undermining the integrity and accuracy of the global model's predictions. This methodical approach ensures that the malicious modifications evade detection while exerting significant influence on the model's behavior and outcomes.

In intelligent railway scenarios, *PMM* could severely impact applications such as train scheduling optimization, predictive maintenance, and security monitoring. FL is utilized to optimize these systems by training a global model across multiple distributed nodes. However, *PMM* can degrade scheduling system performance, leading to train delays and scheduling disruptions; interfere with predictive maintenance models, increasing equipment failure rates and maintenance costs; and disrupt security monitoring models, compromising the ability to effectively detect safety threats, thereby affecting railway system security. Overall, *PMM* attacks can result in decreased model performance, reduced system reliability, increased economic losses, and risks to security.

5 Defense Method

In this section, we provide a comprehensive overview of *CLBL*, comprising three distinct steps in each global epoch. Algorithm 2 and Fig. 3 delineate the entire procedure of CLBL.

- **Step 1: Cluster Model Updates Layer by Layer.** Upon receiving model updates from clients, the server initiates the clustering process using HDBSCAN [36], organizing all model updates layer by layer as follows:

$$\{g_{i,j}^{t,*} \mid i = 1,\ 2, \ldots, L;\ j = 1,\ 2, \ldots, k\} \leftarrow \text{HDBSCAN Cluster}\{g_{i,j}^{t} \mid i = 1,\ 2, \ldots, n;\ j = 1,\ 2, \ldots, k\} \tag{6}$$

  Here, L represents the total number of admitted model updates. $g_{i,j}^{t,*}$ denotes the j-th layer of the admitted model update from the i-th client in round t. Given our assumption that the number of malicious clients is less than that of benign ones, and benign model updates exhibit higher similarity, clustering model updates layer by layer allows us to identify the class containing the most model updates as benign.

- **Step 2: Average of Benign Model Updates.** After clustering to isolate benign model updates, the next step involves averaging these updates layer by layer:

$$g_{global,j}^{t} = average(g_{1,j}^{t,*},\ g_{2,j}^{t,*}, \ldots, g_{L,j}^{t,*}) \tag{7}$$

  Here, $g_{global,j}^{t}$ represents the j-th layer of the global model update in round t. Drawing from the methodology of *PMM*, clustering model updates based on layers enhances the robustness of aggregation algorithms compared to traditional approaches, offering improved outlier detection capabilities.

- **Step 3: Aggregation.** The server computes the global model parameter ω_t using the global update obtained from *CLBL*:

$$\omega_t = \omega_{t-1} - \eta \cdot g_{global}^{t} \tag{8}$$

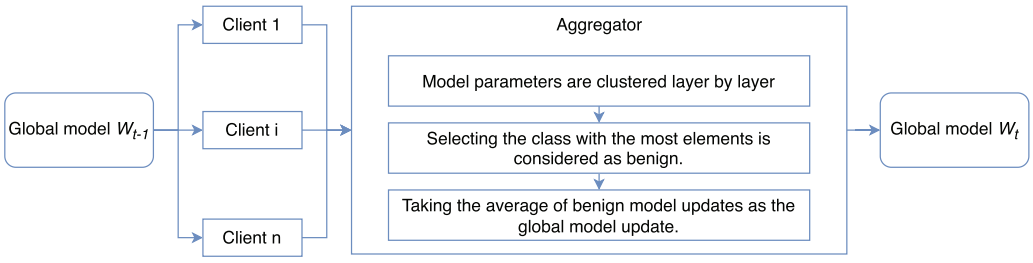


Figure 3: Illustration of *CLBL*'s workflow in round t

Here, g^t_{global} represents the global model update in round t.

Algorithm 2: CLBL

Input: Set of model updates
Output:
1. **for** $t \in [1, T]$ **do**
2: Receive model updates uploaded by the clients $\{g^t_i \mid i = 1,\ 2, \ldots, n\}$
3: **for** $j \in [1, k]$ **do**
4: **for** $i \in [1, n]$ **do**
5: $g^{t,*}_{i,j} \mid i = 1,\ 2, \ldots, L;\ j = 1,\ 2, \ldots, k \leftarrow \text{HDBSCAN}\ \{g^t_{i,j} \mid i = 1,\ 2, \ldots, n;\ j = 1,\ 2, \ldots, k\}$
▷L is the number of admitted model updates.
6: $g^t_{global,j} = average(g^{t,*}_{1,j},\ g^{t,*}_{2,j}, \ldots, g^{t,*}_{L,j})$
7: **end for**
8: **end for**
9: **end for**
10: Server updates the global model $\omega_t = \omega_{t-1} - \eta \cdot g^t_{global}$

To analyze the time complexity of *CLBL*, we need to evaluate the complexity of each step sequentially. *CLBL* consists of three main steps: clustering model updates layer by layer, averaging benign model updates, and aggregation.

- **Step 1: Cluster Model Updates Layer by Layer.** In this step, the server clusters model updates for each layer using the HDBSCAN algorithm. The time complexity of HDBSCAN is approximately $O(n \log n)$, where n is the number of data points. In the context of CLBL, this means that for each layer of model updates, the time complexity is $O(n_i \log n_i)$, where n_i is the number of model updates in the i-th layer.The total time complexity for the clustering step is:

$$O\left(\sum_{i=1}^{k} n_i \log n_i\right) \tag{9}$$

- **Step 2: Averaging Benign Model Updates.** After identifying the benign model updates, the algorithm averages these updates layer by layer. For each layer, this involves averaging all benign updates, with a time complexity of $O(n_i)$ for each layer. Overall, for k layers of model updates, the total time complexity for the averaging step is:

$$O\left(\sum_{i=1}^{k} n_i\right) \tag{10}$$

- **Step 3: Aggregation.** This step involves updating the global model parameters using the global update obtained from the CLBL process. This is a constant-time operation with a time complexity of $O(1)$.

Combining the time complexities of all the steps, the total time complexity of the CLBL algorithm is:

$$O\left(\sum_{i=1}^{k} n_i \log n_i\right) + O\left(\sum_{i=1}^{k} n_i\right) + O(1) \tag{11}$$

In *CLBL*, each layer has the same number of model updates., i.e., $n_i = n$, the total time complexity simplifies to: $O(kn \log n)$. Therefore, the time complexity of the CLBL algorithm is $O(kn \log n)$, where n is the number of model updates in each layer.

In summary, CLBL initially clusters model updates uploaded by clients layer by layer, selecting the class with the most model updates as benign. It then computes the average of these benign updates to derive the global model update.

In intelligent railway scenarios, *CLBL* significantly enhances the security, robustness, and efficiency of model updates, particularly in addressing poisoning attacks. Its specific applications include track monitoring and maintenance, train operation control, and environmental monitoring and early warning systems, ensuring stable system operation under various adverse conditions. The introduction of *CLBL* notably improves the safety and reliability of railway systems while reducing maintenance costs and enhancing passenger satisfaction. However, ongoing algorithm optimization, increased adaptability, and improved real-time performance remain critical challenges to ensuring its long-term effectiveness.

6 Experiment

6.1 Experimental Settings

In this section, we evaluate the performance of *PMM* under different aggregation methods and compared it with three attack methods. We also test the impact of varying numbers of malicious clients and perturbation coefficients λ. Additionally, we evaluate the performance of *CLBL* under different attacks and compared it with three aggregation algorithms. Meanwhile, to enhance the persuasiveness of the experimental results, we use different seeds for cross-validation.

6.1.1 Datasets and Models

In this experiment, we use three large-scale benchmark datasets. We provide a detailed description of each dataset in Table 1 and showcase the samples and their corresponding labels in Fig. 4.

- MNIST [37] dataset consists of 70,000 grayscale images, each with a size of $28 * 28$ pixels, divided into 10 classes. Each class in MNIST contains 7000 images. We employ LeNet [37] as the architecture for the global model for training and evaluation.
- FMNIST [38] dataset consists of 70,000 grayscale images, each with a size of $28 * 28$ pixels, divided into 10 classes. Each class in FMNIST contains 7000 images. We employ LeNet [37] as the architecture for the global model for training and evaluation.
- CIFAR10 [39] dataset consists of 60,000 RGB images, each with a size of $32 * 32$ pixels, divided into 10 classes. Each class in CIFAR10 contains 6000 images. We employ VGG-11 [40] as the architecture for the global model for training and evaluation.

Table 1: Description of datasets and models

Dataset	MNIST [37]	FMNIST [38]	CIFAR10 [39]
Number of samples	70,000	70,000	60,000
Image size	$28 * 28 * 1$	$28 * 28 * 1$	$32 * 32 * 3$
Number of classes	10	10	10

(Continued)

Table 1 (continued)

Dataset	MNIST [37]	FMNIST [38]	CIFAR10 [39]
Category	Handwritten digits	Fashion images	Natural images
Model	LeNet	LeNet	VGG11

Figure 4: Samples of datasets

6.1.2 FL Settings

To present the experimental parameters clearly, Table 2 details the parameters used in the experiments:

Table 2: Experimental parameters

Dataset	Number of clients	Number of clients in each epoch	Global/Local epoch	Batch size	Learning rate
MNIST	100	15	50/1	64	0.05
FMNIST	100	15	50/1	64	0.05
CIFAR10	100	15	100/1	64	0.05

6.1.3 Baseline Poisoning Attacks and Defense Methods

We conduct adversarial experiments involving defense methods (FedAVG, Krum, Trimmed_mean, and our proposed method, *CLBL*) and attack strategies (LIE, Min-Max, and Min-Sum). Detailed descriptions of the attack methods are provided in Section 2.3, while specifics regarding the defense mechanisms are outlined in Section 2.2. To ensure equitable comparisons, all experiments are executed within the same FL framework and under identical attack configurations. The hyperparameter settings for PMM and CLBL in the experiment are presented in the Table 3.

6.2 Attack Results

In this section, we present a comparative analysis of *PMM* with existing state-of-the-art model poisoning attacks, namely LIE, Min-Max, and Min-Sum, across various aggregation algorithms as outlined in Table 4.

Table 3: Experimental hyperparameters for PMM and CLBL

Method	Hyperparameter	Value
PMM	Malicious clients	5
	Perturbation coefficient	1.1
CLBL	Min_cluster_size	2

Table 4: Accuracy (%) of different poisoning attacks against various defense methods (results are the mean of five experiments, with variance in parentheses)

Dataset	AGR	LIE	Min-Max	Min-Sum	PMM
MNIST	FedAVG	11.36 (0.0000)	11.36 (0.0001)	11.36 (0.0001)	**9.81 (0.0001)**
	Krum	97.73 (0.0000)	94.67 (0.0143)	95.13 (0.0001)	**10.56 (0.0043)**
	Trimmed_mean	11.30 (0.0001)	11.36 (0.0001)	11.36 (0.0001)	**9.69 (0.0043)**
FMNIST	FedAVG	9.98 (0.0000)	10.02 (0.0000)	10.03 (0.0000)	**9.97 (0.0000)**
	Krum	70.64 (0.1727)	76.52 (0.1128)	77.63 (0.0624)	**45.72 (0.0869)**
	Trimmed_mean	10.01(0.0000)	10.01 (0.0000)	10.00 (0.0000)	**9.99 (0.0000)**
CIFAR10	FedAVG	10.00 (0.0000)	10.03 (0.0000)	10.03 (0.0013)	**9.99 (0.0000)**
	Krum	46.42 (0.1735)	28.14 (0.0196)	30.05 (0.0181)	**22.07 (0.1593)**
	Trimmed_mean	**9.99 (0.0000)**	10.01 (0.0000)	10.01 (0.0000)	12.47 (0.3042)

Table 4 presents a comparison of the accuracy of various defense methods under different poisoning attacks. The overall results indicate that the PMM method exhibits superior attack performance in various scenarios, achieving lower accuracy across multiple datasets and defense methods. In contrast, the performance of the LIE, Min-Max, and Min-Sum attacks is relatively unstable. While these methods can achieve good attack effects in some datasets and defense methods, overall, PMM outperforms the other attack methods.

The superior performance of PMM can be attributed to its layer-based attack strategy, which enhances attack effectiveness while increasing stealth. By introducing targeted minor perturbations to each layer of the model, PMM effectively constructs malicious model updates, thereby evading some common defense mechanisms. In contrast, LIE, Min-Max, and Min-Sum, although relatively complex in strategy design, perform malicious updates on the entire model without targeted fine-tuning. This leads to significant attack effects, but with less stealth. Overall, PMM demonstrates clear advantages in both attack effectiveness and stealth, making it a more effective attack method when facing various defense strategies.

6.3 Defense Results

In this section, we compare *CLBL* with state-of-the-art defense methods, Krum and Trimmed_mean for all the poisoning attacks.

We initially assess the performance of *CLBL* without any attackers, with results presented in Table 5. The experimental findings reveal that the global model aggregated using our proposed *CLBL* achieves accuracies of 99.08%, 89.94%, and 76.06% on the MNIST, FMNIST, and CIFAR10 datasets, respectively, which are comparable to the performance of FedAVG. While the accuracy of the global model on CIFAR10 aggregated by *CLBL* is marginally lower than that of the model aggregated by FedAVG, the experiments confirm the effectiveness and availability of *CLBL*. Table 6 presents the accuracies of the compared aggregation algorithms on the three datasets. These results demonstrate the effectiveness of *CLBL*, surpassing existing attacks in numerous scenarios, notably yielding higher accuracies under *PMM* attacks.

Table 5: ACC (%) of the global model on defense methods without attackers (results are the mean of five experiments, with variance in parentheses)

AGR	MNIST	FMNIST	CIFAR10
FedAVG	99.02 (0.0090)	89.70 (0.0096)	76.61 (0.0060)
CLBL	99.08 (0.0000)	89.94 (0.0000)	76.06 (0.0066)

Table 6: ACC (%) of the global model on different defense methods against poisoning attacks (results are the mean of five experiments, with variance in parentheses)

Dataset	Attack	FedAVG	Krum	Trimmed_mean	CLBL
MNIST	LIE	11.36 (0.0000)	97.93 (0.0000)	11.30 (0.0001)	**98.98 (0.0000)**
	Min-Max	11.36 (0.0001)	94.67 (0.0143)	11.36 (0.0001)	**97.19 (0.0002)**
	Min-Sum	11.36 (0.0001)	95.13 (0.0001)	11.36 (0.0001)	**97.23 (0.0040)**
	PMM	9.81 (0.0001)	10.56 (0.0043)	9.69 (0.0043)	**98.93 (0.0000)**
FMNIST	LIE	9.98 (0.0000)	70.64 (0.1727)	10.01 (0.0000)	**87.30 (0.0000)**
	Min-Max	10.02 (0.0000)	76.52 (0.1128)	10.01 (0.0000)	**79.95 (0.2740)**
	Min-Sum	10.03 (0.0000)	77.63 (0.0624)	10.00 (0.0000)	**78.23 (0.2050)**
	PMM	9.97 (0.0000)	45.72 (0.0869)	9.99 (0.0000)	**88.90 (0.1080)**
CIFAR10	LIE	10.00 (0.0000)	46.42 (0.1735)	9.99 (0.0000)	**67.07 (0.3670)**
	Min-Max	10.03 (0.0000)	28.14 (0.0196)	10.01 (0.0000)	**62.51 (0.0703)**
	Min-Sum	10.03 (0.0013)	30.05 (0.0181)	10.01 (0.0000)	**61.81 (0.2020)**
	PMM	9.99 (0.0000)	22.07 (0.1593)	12.47 (0.3042)	**64.12 (0.2190)**

Under LIE attacks, *CLBL* shows accuracy improvements of 16.66% and 77.29% on the FMNIST dataset compared to Krum and Trimmed_mean, respectively. On the CIFAR10 dataset, the improvements are 20.65% and 57.08%, respectively, compared to Krum and Trimmed_mean. On the MNIST dataset, the improvements are 1.05% and 87.68%, respectively. Under Min-Max attacks, *CLBL* demonstrates accuracy improvements of 3.43% and 69.93% on the FMNIST dataset compared to Krum and Trimmed_mean, respectively. On the CIFAR10 dataset, the improvements are 34.37% and 52.50%, respectively. On the MNIST dataset, the improvements are 2.52% and 85.83%, respectively. Under Min-Sum attacks, *CLBL* achieves accuracy improvements of 0.60% and 68.23% on the FMNIST dataset compared to Krum and Trimmed_mean, respectively. On the CIFAR10 dataset,

the improvements are 31.76% and 51.80%, respectively. On the MNIST dataset, the improvements are 2.10% and 85.87%, respectively. Under PMM attacks, *CLBL* delivers accuracy improvements of 43.18% and 78.91% on the FMNIST dataset compared to Krum and Trimmed_mean, respectively. On the CIFAR10 dataset, the improvements are 42.05% and 51.65%, respectively. On the MNIST dataset, the improvements are 88.37% and 89.24%, respectively.

The superior performance of CLBL can be attributed to its layer-based defense strategy, which meticulously filters the parameters of each layer in the model to mitigate the impact of malicious model updates on the global model. By employing this layer-by-layer filtering method, CLBL can effectively identify and eliminate potential malicious modifications within each layer, thereby enhancing the overall robustness of the model. In contrast, Krum and Trimmed_mean only perform filtering and selection on the model as a whole, overlooking the possibility that an attacker might make minor modifications in only one layer. In such cases, malicious model updates constructed by attackers might successfully pass through Krum and Trimmed_mean filters, thereby compromising the performance of the global model. Therefore, CLBL's layered defense strategy proves particularly effective in countering fine-grained attacks, significantly improving the model's defense against various poisoning attacks.

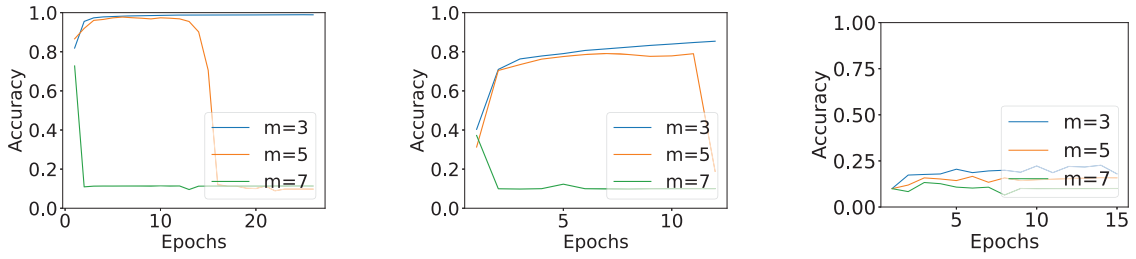
7 Factors of Poisoning Attacks

To comprehensively evaluate the attack performance of PMM, we conduct experimental evaluations on the MNIST, FMNIST, and CIFAR10 datasets regarding the number of malicious clients and the perturbation coefficient λ. We employ FedAVG, Krum and Trimmed_mean algorithms to conduct defense experiments against PMM.

7.1 Numbers of Malicious Clients

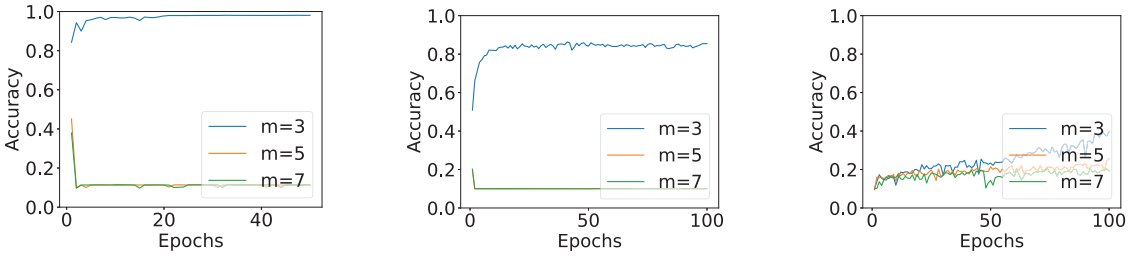
In this section, we discuss the impact of the numbers of malicious clients in PMM.

Figs. 5–7 present the accuracy of FedAVG, Krum, and Trimmed_mean under PMM attacks with varying numbers of malicious clients (m). In all cases, as m increases, PMM demonstrates progressively stronger attack effectiveness, resulting in reduced model accuracy. This trend reflects the heightened influence of malicious clients' uploaded model updates on the global model with the increase in m.



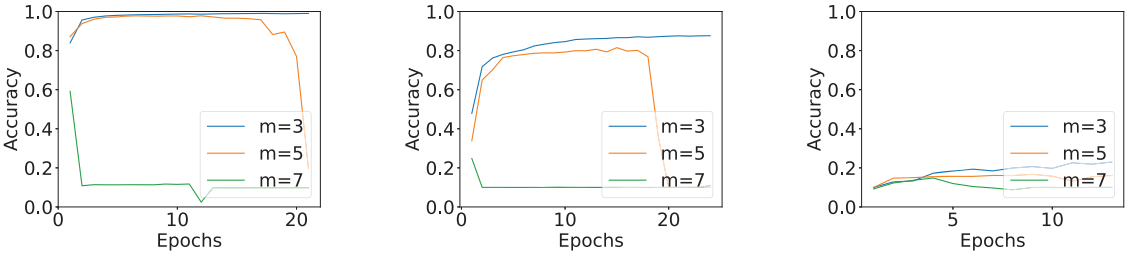
(a) MNIST (b) FMNIST (c) CIFAR10

Figure 5: *ACC vs.* Global Epochs for FedAVG under PMM across 3 datasets (MNIST, FMNIST, CIFAR10) with 3 different numbers of malicious clients (Vertical Axis: accuracy, Horizontal Axis: number of global epochs)



(a) MNIST (b) FMNIST (c) CIFAR10

Figure 6: *ACC vs.* Global Epochs for Krum under PMM across 3 datasets (MNIST, FMNIST, CIFAR10) with 3 different numbers of malicious clients (Vertical Axis: accuracy, Horizontal Axis: number of global epochs)



(a) MNIST (b) FMNIST (c) CIFAR10

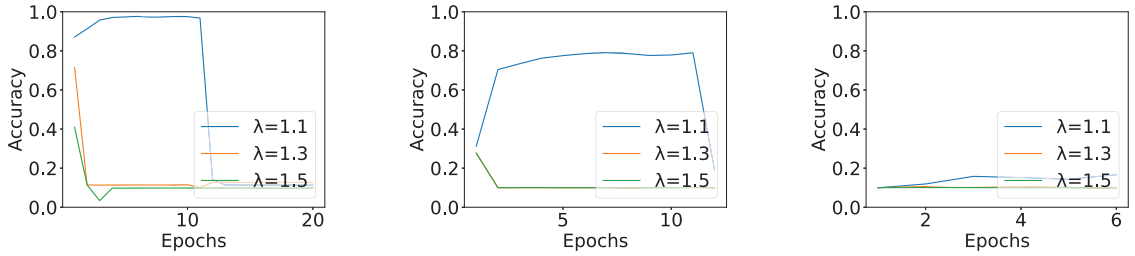
Figure 7: *ACC vs.* Global Epochs for Trimmed_mean under PMM across 3 datasets (MNIST, FMNIST, CIFAR10) with 3 different numbers of malicious clients (Vertical Axis: accuracy, Horizontal Axis: number of global epochs)

7.2 Perturbation Coefficient λ

In this section, we discuss the impact of the perturbation coefficient λ in PMM.

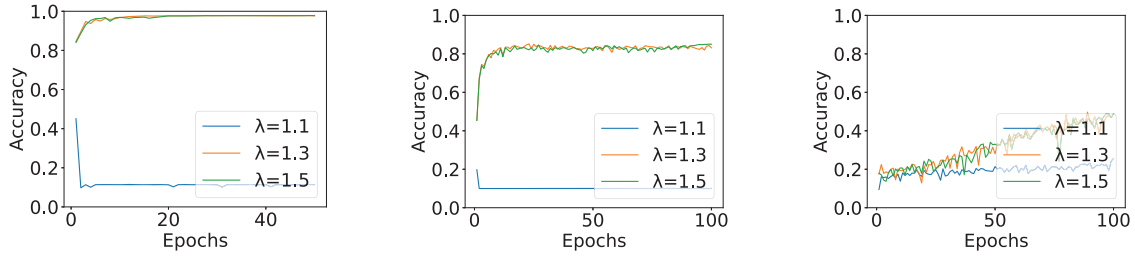
Impact of Perturbation Coefficient λ in PMM Attacks against FedAVG: Fig. 8 illustrates the accuracy of FedAVG subjected to PMM attacks. Varying values of the perturbation coefficient λ were employed in our attacks. We observed that PMM exhibits enhanced attack effectiveness when λ is larger, indicating lower model accuracy. This phenomenon stems from the amplified perturbation introduced by *PMM* with larger λ values, resulting in greater deviation from normal model updates. Consequently, the constructed malicious model updates exhibit more pronounced deviations, leading to a more substantial adverse impact on the model.

Impact of Perturbation Coefficient λ in PMM Attacks against Krum: In Fig. 9, we depict the accuracy of Krum amidst *PMM* attacks. Different λ values were utilized in our attacks. Notably, *PMM* demonstrates superior attack efficacy with smaller λ values, leading to lower model accuracy. This observation arises because increasing λ introduces greater perturbations by *PMM*, causing more substantial deviations from normal values in the constructed malicious model updates. Consequently, these anomalous behaviors make it less likely for Krum to select malicious model updates as the global model update.



(a) MNIST (b) FMNIST (c) CIFAR10

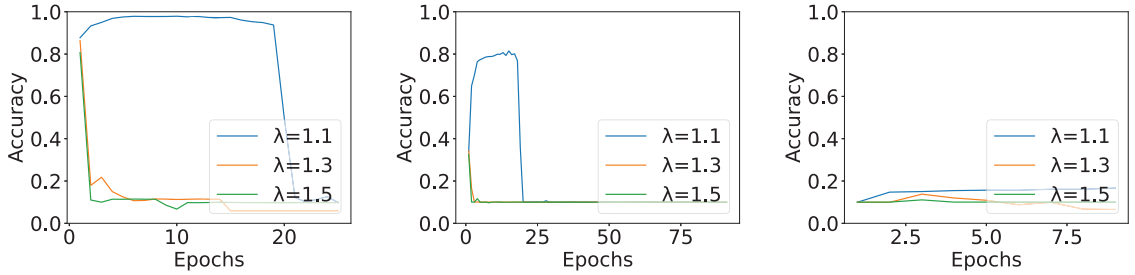
Figure 8: *ACC vs.* Global Epochs for FedAVG under PMM across 3 datasets (MNIST, FMNIST, CIFAR10) with 3 different values of λ (Vertical Axis: accuracy, Horizontal Axis: number of global epochs)



(a) MNIST (b) FMNIST (c) CIFAR10

Figure 9: *ACC vs.* Global Epochs for Krum under PMM across 3 datasets (MNIST, FMNIST, CIFAR10) with 3 different values of λ (Vertical Axis: accuracy, Horizontal Axis: number of global epochs)

Impact of Perturbation Coefficient λ in PMM Attacks against Trimmed_mean: Fig. 10 showcases the accuracy of Trimmed_mean subjected to *PMM* attacks. Our attacks utilized varying λ values. We found that *PMM* exhibits heightened attack effectiveness with larger λ values, resulting in lower model accuracy. This outcome is attributed to the increased perturbation introduced by *PMM* with larger λ values, leading to greater deviation from normal values in the constructed malicious model updates. Consequently, the detrimental effect on the model is more pronounced.



(a) MNIST (b) FMNIST (c) CIFAR10

Figure 10: *ACC vs.* Global Epochs for Trimmed_mean under PMM across 3 datasets (MNIST, FMNIST, CIFAR10) with 3 different values of λ (Vertical Axis: accuracy, Horizontal Axis: number of global epochs)

8 Limitation and Future Work

Although this study provides thorough validation of the effectiveness of PMM and CLBL, there are still some limitations. Firstly, the scalability of CLBL in larger and more complex federated learning environments has not been fully tested. The current experimental setup may not fully reflect its performance in real-world applications. Secondly, the types of poisoning attacks explored in this study are limited, and the combinations and interactions of different types of attacks have not been considered. These limitations restrict our understanding of CLBL's defensive capabilities in diverse attack scenarios.

Future research could further explore the integration of CLBL with other defense mechanisms to enhance overall robustness and resilience. For example, investigating how to combine CLBL with existing defense methods to form a more comprehensive defense system would be beneficial. Additionally, studying the impact of different types of poisoning attacks and their combinations on the model will help develop more effective defense strategies, thereby improving the security and reliability of the model in complex environments. Such research would not only enhance defense effectiveness but also provide stronger guarantees for practical applications.

9 Conclusion

In the realm of intelligent railway transportation systems, where large-scale AI models are pivotal, FL offers a privacy-preserving solution. However, despite its effectiveness in preserving privacy, FL systems remain susceptible to poisoning attacks. To illustrate these threats, we introduce a novel attack method, PMM, which exploits the hierarchical structure of models to generate malicious model updates through controlled malicious clients. Our extensive experiments across three datasets underscore the effectiveness of PMM in disrupting the global model. In real-world railway scenarios, PMM can degrade train scheduling performance, interfere with predictive maintenance models, and disrupt security monitoring, leading to delays, increased costs, and compromised safety. In response to these threats, we further develop a robust aggregation technique, CLBL, specifically tailored to counteract such attacks. CLBL operates by clustering model parameters layer by layer, effectively filtering out malicious model updates and enhancing the global model's resilience. In practical applications, CLBL can improve track monitoring, train operation control, and environmental monitoring, ensuring stable railway system operations. The implementation of CLBL not only increases the reliability of railway transportation but also reduces economic losses caused by system failures, and improves passenger safety and satisfaction. Additionally, a stable railway system contributes to economic development, enhances logistics efficiency, and supports connectivity between cities and regions. The efficacy of CLBL is substantiated by our experiments on the same three datasets, showcasing its ability to enhance the global model's resilience and availability, crucial for maintaining the security of the intelligent railway transportation systems.

Acknowledgement: The authors are thankful to the anonymous reviewers for improving this article.

Funding Statement: This work is supported by Systematic Major Project of China State Railway Group Corporation Limited (Grant Number: P2023W002).

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design, data collection, analysis, and interpretation of results: Yongsheng Zhu, Wei Wang and Chong Liu; draft manuscript preparation, and editing: Chunlei Chen, Xiaoting Lyu, Zheng Chen,

Baigen Cai and Bin Wang; validation: Fuqiang Hu, Jiao Dai and Hanxi Li. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Based upon reasonable request, data can collect from the corresponding author. The URLs for the datasets required in the experiments are as follows: (1) MNIST: http://yann.lecun.com/exdb/mnist/ (accessed on July 23 2024); (2) FMNIST: https://github.com/zalandoresearch/fashion-mnist (accessed on July 23 2024); (3) CIFAR-10: https://www.cs.toronto.edu/kriz/cifar.html (accessed on July 23 2024).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Konečný J, McMahan HB, Yu FX, Richtárik P, Suresh AT, Bacon D. Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492. 2016.
2. Li L, Liu J, Cheng L, Qiu S, Wang W, Zhang X, et al. CreditCoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. IEEE Trans Intell Transp Syst. 2018;19(7):2204–20. doi:10.1109/TITS.2017.2777990.
3. Yazdinejad A, Dehghantanha A, Parizi RM, Hammoudeh M, Karimipour H, Srivastava G. Block hunter: federated learning for cyber threat hunting in blockchain-based IIoT networks. IEEE Trans Ind Inform. 2022;18(11):8356–66. doi:10.1109/TII.2022.3168011.
4. Jagarlamudi GK, Yazdinejad A, Parizi RM, Pouriyeh S. Exploring privacy measurement in federated learning. J Supercomput. 2024;80(8):10511–51. doi:10.1007/s11227-023-05846-4.
5. Lu S, Li R, Liu W. FedDAA: a robust federated learning framework to protect privacy and defend against adversarial attack. Front Comput Sci. 2024;18(2):182307. doi:10.1007/s11704-023-2283-x.
6. Wang W, Song J, Xu G, Li Y, Wang H, Su C. ContractWard: automated vulnerability detection models for ethereum smart contracts. IEEE Trans Netw Sci Eng. 2021;8(2):1133–44. doi: 10.1109/TNSE.2020.2968505.
7. Wang W, Shang Y, He Y, Li Y, Liu J. BotMark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. Inf Sci. 2020;511:284–96. doi:10.1016/j.ins.2019.09.024.
8. Wang W, Suo X, Wei X, Wang B, Wang H, Dai H, et al. HGATE: heterogeneous graph attention auto-encoders. IEEE Trans Knowl Data Eng. 2023;35(4):3938–51. doi:10.1109/TKDE.2021.3138788.
9. Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V. How to backdoor federated learning. In: Chiappa S, Calandra R, editors. The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, 2020 Aug 26–28; Palermo, Sicily, Italy: PMLR; vol. 108. p. 2938–48.
10. Lyu X, Han Y, Wang W, Liu J, Wang B, Liu J, et al. Poisoning with cerberus: stealthy and colluded backdoor attack against federated learning. Proc AAAI Conf Artif Intell. 2023;37:9020–8.
11. Liu P, Xu X, Wang W. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. Cybersecurity. 2022;5(1):4.
12. Liu P, Wang W, Xu X, Li H, Ding W. Assessing membership leakages via task-aligned divergent shadow datasets in vehicular road cooperation. IEEE Internet Things J. 2024;1.
13. Lu S, Li R, Chen X, Ma Y. Defense against local model poisoning attacks to byzantine-robust federated learning. Front Comput Sci. 2022;16(6):166337. doi:10.1007/s11704-021-1067-4.
14. Zhang K, Song X, Zhang C, Yu S. Challenges and future directions of secure federated learning: a survey. Front Comput Sci. 2022;16(5):165817. doi:10.1007/s11704-021-0598-z.

15. Biggio B, Nelson B, Laskov P. Poisoning attacks against support vector machines. In: Proceedings of the 29th International Conference on Machine Learning (ICML 2012), 2012 Jun 26–Jul 1; Edinburgh, Scotland, UK: icml.cc/Omnipress.
16. Jagielski M, Oprea A, Biggio B, Liu C, Nita-Rotaru C, Li B. Manipulating machine learning: poisoning attacks and countermeasures for regression learning. In: 2018 IEEE Symposium on Security and Privacy, 2018; San Francisco, CA, USA: IEEE Computer Society; p. 19–35. doi: 10.1109/SP.2018.00057.
17. Li B, Wang Y, Singh A, Vorobeychik Y. Data poisoning attacks on factorization-based collaborative filtering. In: Lee DD, Sugiyama M, Von Luxburg U, Guyon I, Garnett R, editors. Advances in neural information processing systems 29. Barcelona, Spain; 2016 Dec 5–10. p. 1885–93.
18. Rubinstein BIP, Nelson B, Huang L, Joseph AD, Lau S, Rao S, et al. ANTIDOTE: understanding and defending against poisoning of anomaly detectors. In: Feldmann A, Mathy L, editors. Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference, 2009 Nov 4–6; Chicago, Illinois, USA: ACM; p. 1–14. doi: 10.1145/1644893.1644895.
19. Suciu O, Marginean R, Kaya Y, Daumé III H, Dumitras T. When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks. In: Enck W, Felt AP, editors. 27th USENIX Security Symposium, USENIX Security 2018. Baltimore, MD, USA: USENIX Association; 2018 Aug 15–17. p. 1299–316.
20. Shejwalkar V, Houmansadr A. Manipulating the byzantine: optimizing model poisoning attacks and defenses for federated learning. In: 28th Annual Network and Distributed System Security Symposium, (NDSS 2021), 2021 Feb 21–25; The Internet Society.
21. Yin D, Chen Y, Ramchandran K, Bartlett PL. Byzantine-robust distributed learning: towards optimal statistical rates. In: Dy JG, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning, 2018 July 10–15; Stockholmsmässan, Stockholm, Sweden: PMLR; vol. 80. p. 5636–45.
22. Blanchard P, Mhamdi EME, Guerraoui R, Stainer J. Machine learning with adversaries: Byzantine tolerant gradient descent. 2017. p. 119–29. Available from: https://proceedings.neurips.cc/paper/2017/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html. [Accessed 2024].
23. Chen Y, Su L, Xu J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. Proc ACM Meas Anal Comput Syst. 2017;1(2):1–25. doi:10.1145/3154503.
24. Chen Y, Qin X, Wang J, Yu C, Gao W. FedHealth: a federated transfer learning framework for wearable healthcare. IEEE Intell Syst. 2020;35(4):83–93. doi:10.1109/MIS.2020.2988604.
25. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Singh A, Zhu XJ, editors. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 2017 Apr 20–22; Fort Lauderdale, FL, USA: PMLR; vol. 54, p. 1273–82.
26. Dean J, Corrado G, Monga R, Chen K, Devin M, Le QV, et al. Large scale distributed deep networks. In: Advances in neural information processing systems 25 (NIPS 2012). 2012:1232–40. Available from: https://proceedings.neurips.cc/paper/2012/hash/6aca97005c68f1206823815f66102863-Abstract.html. [Accessed 2024].
27. Barreno M, Nelson B, Joseph AD, Tygar JD. The security of machine learning. Mach Learn. 2010;81(2):121–48. doi:10.1007/s10994-010-5188-5.
28. Cretu GF, Stavrou A, Locasto ME, Stolfo SJ, Keromytis AD. Casting out demons: sanitizing training data for anomaly sensors. In: 2008 IEEE Symposium on Security and Privacy (SP 2008), 2008 May 18–21; Oakland, CA, USA: IEEE Computer Society; p. 18–21. doi:10.1109/SP.2008.11.
29. Tran B, Li J, Madry A. Spectral signatures in backdoor attacks. In: Bengio S, Wallach HM, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. Advances in neural information processing systems 31: Montréal, QC, Canada; 2018 Dec 3–8. p. 8011–21.
30. Liu J, Lyu X, Duan L, He Y, Liu J, Ma H, et al. PnA: robust aggregation against poisoning attacks to federated learning for edge intelligence. ACM Trans Sens Netw. 2024;18:182307. doi:10.1145/3669902.

31. Cao X, Fang M, Liu J, Gong NZ. FLTrust: Byzantine-robust federated learning via trust bootstrapping. arXiv preprint arXiv:2012.13995. 2020.
32. Baruch G, Baruch M, Goldberg Y. A little is enough: circumventing defenses for distributed learning. In: Advances in neural information processing systems 32 (NeurIPS 2019). 2019. Available from: https://proceedings.neurips.cc/paper/2019/hash/ec1c59141046cd1866bbbcdfb6ae31d4-Abstract.html. [Accessed 2024].
33. Fang M, Cao X, Jia J, Gong NZ. Local model poisoning attacks to Byzantine-robust federated learning. In: Capkun S, Roesner F, editors. 29th USENIX Security Symposium, USENIX Security 2020, 2020 Aug 12–14; Berkeley, CA, USA: USENIX Association; p. 1605–22.
34. Cao X, Gong NZ. MPAF: model poisoning attacks to federated learning based on fake clients. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, 2022 Jun 19–20; New Orleans, LA, USA: IEEE; p. 3395–403. doi:10.1109/CVPRW56347.2022.00383.
35. Bhagoji AN, Chakraborty S, Mittal P, Calo SB. Analyzing federated learning through an adversarial lens. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 2019 Jun 9–15. Long Beach, CA, USA: Proceedings of Machine Learning Research. PMLR; vol. 97, p. 634–43.
36. Campello RJGB, Moulavi D, Zimek A, Sander J. Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Trans Knowl Discov Data. 2015;10(1):1–51. doi:10.1145/2733381.
37. Deng L. The MNIST database of handwritten digit images for machine learning research [Best of the Web]. IEEE Signal Process Mag. 2012;29(6):141–2. doi:10.1109/MSP.2012.2211477.
38. Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747. 2017.
39. Krizhevsky A. Learning multiple layers of features from tiny images (Master's Thesis). Canada: Department of Computer Science, University of Toronto; 2009.
40. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2015.