ClieND: Client-Side Neuron-Level Detection against Poisoning Attacks on Cross-Silo Federated Learning

Mengyao Ma
The University of Queensland
CSIRO's Data61
Brisbane, Australia

Shuofeng Liu
The University of Queensland
Brisbane, Australia

Viet Vo
Swinburne University of Technology
Melbourne, Australia

Minghong Fang
University of Louisville
Kentucky, USA

Surya Nepal
CSIRO's Data61
Sydney, Australia

Guangdong Bai
City University of Hong Kong
Kowloon Tong, Hong Kong SAR

Abstract

Poisoning attacks have been shown to pose significant threats to federated learning (FL), including both untargeted and targeted attacks. To mitigate such threats, the majority of existing defenses are implemented on the server side during the aggregation process. However, as data remains inherently local in FL, these defenses either rely on unreliable statistical or structural properties of local updates, or make strong assumptions that rely on dataset information. As a result, the unique advantage of clients having access to trusted local data and training dynamics has been largely overlooked. In this work, we propose ClieND[1], a novel *client-side* detection framework that shifts the detection of poisoning attacks from the server to the client. Specifically, ClieND enables each client to maintain the *neuron importance scores* of the aggregated global model in each round by leveraging its local dataset. Through tracking the inter-round changes in these scores, each client detects abnormal behavior by identifying significant *discrepancy spikes*, which serve as indicators of potential poisoning attacks. To evaluate the effectiveness of ClieND, we compare it against three state-of-the-art baselines under both targeted and untargeted poisoning attacks, and also assess its ability to detect attacks at an early stage. Experimental results show that ClieND achieves a false positive rate (FPR) as low as 0.01 and a true positive rate (TPR) of up to 0.99, which significantly outperforms server-side defenses, demonstrating its high detection accuracy. Additionally, the existence of poisoning attacks, even those launched by adaptive settings, can be detected in an early stage within 5 communication rounds.

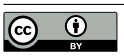
CCS Concepts

• **Security and privacy**; • **Computing methodologies** → **Machine learning**;

Keywords

Federated Learning, Poisoning Attack, Client-Side Detection

[1]The code is available at https://github.com/Trusted-System-Lab/ClieND.



ASIA CCS '26, June 01–05, 2026, Bangalore, India

ACM ISBN 979-8-4007-2356-8/2026/06
https://doi.org/10.1145/3779208.3785390

ACM Reference Format:
Mengyao Ma, Shuofeng Liu, Viet Vo, Minghong Fang, Surya Nepal, and Guangdong Bai. 2026. ClieND: Client-Side Neuron-Level Detection against Poisoning Attacks on Cross-Silo Federated Learning. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '26), June 01–05, 2026, Bangalore, India.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3779208.3785390

1 Introduction

Federated learning (FL) [36] is a distributed machine learning framework that enables multiple clients to collaboratively train a shared model (called *global model*) [33]. Unlike the centralized training paradigm that gathers raw data centrally, FL adopts a *model update aggregation* paradigm, wherein each client trains a local model on its private data and only shares model updates with the server. Among various FL deployment scenarios, *cross-silo FL* [25, 38, 60] has emerged as a particularly practical and widely adopted setting, in domains such as healthcare and finance [23, 38]. Specifically, the server starts the process by randomly initializing the global model and sending its parameters to all clients. Each client then updates the model using their local data and sends the updated model parameters back to the server. The server aggregates the received local updates to refine the global model, and the training process repeats until the global model converges.

Although FL facilitates collaborative training across distributed clients, its inherently decentralized nature renders it vulnerable to attacks from malicious adversaries. *Poisoning attack* [3, 4, 6, 15, 35, 47, 51, 52] is one such attack of severe threats. It is generally divided into two types, *untargeted* [15, 30, 47, 48] and *targeted poisoning attacks* [3, 4, 6, 11, 12, 21, 46, 52, 56]. The goal of untargeted attacks is to reduce the global model's general performance, leading to uniformly poor predictions across the input distribution. In contrast, targeted attacks aim to manipulate the model or data to produce incorrect predictions on specific inputs, often with the goal of misclassifying particular data points of interest. Both types of poisoning attacks can mislead the global model to update its parameters incorrectly during the aggregation.

In response to these attacks, many server-side defenses have been proposed. They can be broadly categorized into two categories based on their detection logic, i.e., statistics-based [4, 10, 16, 18, 20, 37] and behavior-based defenses [15, 26, 42, 43]. The first category, such as FLTrust [10] and FLAME [37], relies on the statistical properties or structural similarity (e.g., cosine similarity, gradient direction, or frequency spectrum) of local model updates to mitigate

poisoning through statistical features directly. However, statistical signals are inherently unreliable, as it becomes difficult to distinguish between client updates resulting from data heterogeneity and those induced by adversarial behavior from a single server-side view. Adversaries can also behave similarly to benign clients in statistical patterns, further misleading the server's decisions. The second category, such as Fang [15] and CrowdGuard [42], detects malicious clients by evaluating each client's local model performance or behavior on specific inputs, such as validation accuracy or class-wise prediction consistency. These methods typically require auxiliary information, such as server access to clean validation datasets or a trusted client-side behavior, which introduces strong deployment constraints and limits their practicality in real-world FL. Both categories operate from a constrained server-side perspective, overlooking the intrinsic advantage of the self-trustworthy client side in having access to authentic local datasets and rich training dynamics (e.g., model parameters and neuron importance scores). As a result, the potential of client-side in detecting poisoning attacks has been largely underestimated.

Our work. We propose CLIEND, a **clie**nt-side **n**euron-level **d**etection framework, to detect the presence of poisoning attacks in FL. CLIEND aims to achieve two goals, i.e., the poisoning attacks can be effectively detected (***Goal #1***) and the detection is accomplished in an early stage (***Goal #2***). The core idea of CLIEND is to shift the detection from server to the client side, capitalizing on the unique advantage of clients. This advantage enables each client to monitor the global model update process by evaluating the shift in neuron importance scores of the global model using local data across different training rounds. The magnitude of change in neuron importance scores (i.e., *discrepancy spike*) across consecutive rounds serves as an indicator of whether a poisoning attack has occurred, thereby contributing to the achievement of ***Goal #1***. Since neuron importance scores directly capture the contribution of individual neurons to the model's predictions on the local data, they are more sensitive to training dynamics than other metrics. This heightened sensitivity allows for more accurate and timely detection of abnormal behavior, thus contributing to ***Goal #2***. The detection is guided by thresholds established during the pre-training rounds [2], which serve as a warm-up phase to validate the system configuration. Besides, we also propose a potential mitigation concept by leveraging dynamic aggregated weighting, following our detection, to mitigate the impact of poisoning attacks detected by CLIEND.

We evaluate CLIEND with both theoretical and empirical analyses. On the theoretical front, we establish a formal foundation to establish the client-side advantage in poisoning attack detection and prove the effectiveness of CLIEND. We begin by proving that poisoning attacks introduce statistically abnormal shifts in neuron importance score dynamics. This is achieved by modeling score discrepancies under clean training as bounded smooth processes and analyzing their deviations under adversarial conditions using concentration inequalities and smoothness-based sensitivity analysis. We then prove the local dataset serves as an additional advantage in client-side detection by proving that the dataset is necessary to observe the discrepancies. We also formally demonstrate the effectiveness of CLIEND, including false alarm avoidance and true attack detection by applying Hoeffding's inequality.

We experimentally benchmark CLIEND's effectiveness against three baseline methods [37, 42, 43] on both widely used untargeted poisoning attacks [15, 47, 59] and targeted attacks, including adaptive adversary settings [4, 6, 12, 28, 59], across four benchmarking datasets. The results show that CLIEND significantly outperforms the baseline methods, with the highest true positive rate of 0.99 and the lowest false positive rate of 0.01. For its early detection capability, CLIEND can detect the presence of untargeted attacks within three rounds and targeted attacks, including adaptive adversaries, within five rounds after the poisoning attack occurs. To further evaluate its scalability and generalization, we conduct ablation studies to assess CLIEND with varying proportions of the malicious clients, different numbers of pre-training rounds, and timing of the attack start. Our results show that CLIEND's performance remains stable, demonstrating its practical use and reliable deployment in real-world scenarios. We also conduct an experimental evaluation for our proposed mitigation. The results show that our mitigation maintains the model's accuracy at nearly the same level as in the absence of attacks for untargeted attacks. While for targeted attacks, it reduces the attack success rate (ASR) to 3%, and even in the presence of adaptive adversaries, the ASR can still be effectively lowered to 7.5%.

Contributions. We list our main contributions as follows.

- **A valuable step forward in advocating the advantages of the client side in poisoning attack detection.** We demonstrate the effectiveness of detecting poisoning attacks from the client side in cross-silo FL. The local dataset and training dynamics can serve as additional advantages for the detection, a factor that is being largely overlooked in existing studies.
- **A practical solution for detecting poisoning attacks.** We propose CLIEND, a novel and effective method designed for detecting poisoning attacks from the client side. It leverages the noticeable discrepancy of neuron importance score spikes between two consecutive training rounds as an indicator to successfully detect poisoning attacks in an early stage.
- **A comprehensive study and evaluation.** We theoretically prove the effectiveness of CLIEND and comprehensively evaluate its performance, including the detection effectiveness and early detection ability. We also show its scalability and generalization, demonstrating its practical application scenarios.

2 Preliminaries and Related Work

2.1 Federated Learning

Federated learning (FL) [36] is a decentralized machine learning paradigm where multiple clients collaboratively train a shared global model while keeping their local data private. Instead of exchanging raw data, clients compute updates (e.g., gradients or model parameters) on their local datasets and only share these updates with a central server. The server then aggregates these updates by using the traditional mean aggregation mechanism (e.g., FedAvg [36]) and subsequently shares the model parameters back with all clients. Recent FL frameworks can be categorized into cross-device FL and cross-silo FL. The cross-device FL usually involves a massive number of mobile or IoT devices that are typically

resource-constrained, intermittently available, and prone to unreliable communication [27]. In contrast, the cross-silo FL operates across a number of stable, resource-rich institutional clients, such as hospitals or banks, with high-quality communication channels and fully-involved consistent participation [25, 38, 60]. Due to its robustness and operational feasibility, cross-silo FL is considered more practical and has seen broader adoption in real-world deployments.

2.2 Poisoning Attacks in FL

Studies have indicated that FL frameworks are susceptible to poisoning attacks [3–6, 15, 47, 52, 55]. We categorize these attacks into two types and provide a brief overview of the techniques applied in each of them.

Untargeted attacks. Untargeted attacks primarily focus on manipulating model parameters to degrade overall model performance. Techniques such as Fang [15] and Min-max attacks [47] involve malicious clients altering calculated gradients, while Min-activation attack [59] suppress the highest activations in dropout layers, thereby minimizing the model's overall performance.

Targeted attacks. Current targeted attacks can be conducted through both data and model perspectives. From the data perspective, Label-flipping [6, 24] and Distributed backdoor attacks [55] enable malicious clients to manipulate specific data labels or embed triggers in the data, thereby degrading the performance of targeted classes. From the model perspective, techniques such as LIE [4] manipulate the gradients calculated by malicious clients to carry out attacks, while Neuron-separation [59] methods modify the neurons in dropout layers to compromise the performance of specific classes. Additionally, the adaptive attack [3, 5, 28] is also a kind of targeted attacks in FL, which refers to sophisticated strategies where adversaries dynamically adjust their behavior to exploit weaknesses in the system while evading detection. A classical adaptive adversary creates a loss function for the deployed defense and bypass the defensive measure, such as constrain-and-scale [3, 5].

2.3 Existing Defenses and Limitations

Existing defenses. Existing server-side defenses can be broadly categorized into two classes based on their underlying detection logic, i.e., *statistics-based* and *behavior-based* defenses. Below, we summarize representative methods for each category.

Statistics-based defenses. This category relies on the structural properties or statistical patterns of model updates, such as gradient similarity, frequency spectrum, or clustering characteristics, to identify potentially malicious behavior. Representative methods in this category include FLTrust [10], FoolsGold [20], FreqFed [18], FLAME [37] and robust aggregation rules [4, 16]. For instance, FLTrust [10] uses a trusted bootstrapping dataset to compute a reference update and assigns a similarity-based trust score to each client's model. Updates with low similarity are down-weighted during aggregation to mitigate their impact. FLAME [37] clusters updates based on their direction, and suppresses those that diverge significantly from the majority.

Behavior-based defenses. This class of defenses evaluates client models based on their performance on specific inputs, such as validation accuracy, prediction distributions, or client-reported loss statistics. These approaches typically rely on auxiliary validation data or feedback to assess client reliability. Notable examples include Fang [15], CrowdGuard [42], FLDetector [61], SHERPA [43]and FLShield [26], For example, CrowdGuard [42] collects local loss statistics from clients as auxiliary signals, and analyzes them to identify inconsistency patterns that indicate malicious behavior.

Limitations. Despite their acceptable performance under controlled settings, existing server-side defenses face two fundamental limitations, hindering their practical deployment in real-world FL.

Lack of robustness due to unreliable assumptions. Most statistics-based defenses [10, 16, 18, 20, 37, 62] rely on model updates from all clients, which forces the server to make empirical assumptions when identifying malicious clients, which may be unreliable. Specifically, these methods often presume that malicious clients exhibit consistent patterns, such as similar gradients or abnormal frequency components. However, such patterns can be easily obfuscated by stealthy or adaptive attackers, rendering the detection ineffective. Moreover, under non-IID data distributions, the natural divergence among benign clients' data can lead them to exhibit similar patterns to those of malicious clients, resulting in misleading and reducing accuracy.

Auxiliary dependencies and excessive overhead. Many behavior based defenses rely on strong auxiliary assumptions, such as access to clean validation datasets [10, 15, 43], auxiliary network [39], or the deployment of trusted execution environments (TEEs) [42]. Such auxiliary dependencies significantly limit the practicality of these methods in large-scale federated deployments, where trusted datasets are hard to obtain and secure hardware is costly. In addition, using auxiliary datasets may raise concerns regarding client data exposure, which does not fully align with the core principles of FL. Besides auxiliary dependencies, many defenses introduce considerable communication and computation overhead. For example, FLShield [26] and CrowdGuard [42] require validator clients to repeatedly evaluate models on their local data and report loss differences to the server. This validator-based feedback mechanism incurs high interaction costs and reduces scalability.

2.4 Neuron Importance Scoring

Neuron importance scoring is a technique used for evaluating the importance of neurons in a neural network by assessing the impact of their removal (or pruning) on the overall model performance. In this work, we adopt the scoring method from Hydra [45], which has been used widely, proposing a scaled initialization for the importance scores of neurons. The equation for computing the importance score is shown below,

$$\hat{s} = \arg\min_{s} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}_{\text{prune}}(\theta_{\text{pretrain}}, s, x, y) \right]. \tag{1}$$

Here $\hat{s}$ represents the optimal importance score for neurons, s represents the importance score being optimized, $\mathbb{E}_{(x,y)\sim\mathcal{D}}$ represents the expected value over the data distribution $\mathcal{D}$, where x and y are the input data samples and their corresponding labels, respectively. L_{prune} is the pruning loss function that measures how much the performance of the model is impacted by removing or pruning neurons [53] based on the importance score, and θ_{pretrain} refers to the pre-trained model parameters.

3 Problem Formulation

3.1 System Settings

Configuration. We configure our system settings within a common realistic FL application scenario. We consider the existence of a few pre-training rounds as a warm-up phase before the actual FL training begins, during which no attacks occur [2]. In real-world deployments, the initial rounds of training are often treated as a secure warm-up phase, where the primary objective is to validate the system configuration and ensure model convergence before transitioning into actual production use. During this phase, all participating clients are closely monitored, minimizing the likelihood of attacks. Therefore, this phase provides an opportunity for benign clients to observe the natural dynamics of neuron importance scores and establish a baseline reference for detecting abnormal updates in subsequent rounds, notably without requiring potentially compromised clients to establish the baseline. For example, in Google's deployment of FL for the Gboard mobile keyboard [8], the early training rounds are conducted to validate the end-to-end FL pipeline. Besides, the data distribution among clients can be arbitrary, supporting both IID (independent and identically distributed) and non-IID settings during FL training. Once the pre-training phase starts, ClieND is activated.

3.2 Threat Model

Adversary's capabilities. Following the existing work [10, 15, 16, 47], we consider a classic adversary setting who controls a subset of malicious clients in the FL system. These clients can be fake participants injected by the adversary or benign clients that have been compromised. These controlled clients upload the manipulated model updates to launch the poisoning attacks. The adversary's capabilities range from weak label-flipping adversaries with minimal control over training [4, 7, 15], to strong adaptive adversaries capable of crafting updates optimized for both model performance [47, 55, 59]. In the worst case (e.g., constrain-and-scale attack [3, 5]), the strongest adversary is assumed to possess full knowledge of the global aggregation rule, the server-side defense metrics, and has the ability to design multi-objective loss functions.
Benign clients' capabilities. For all benign clients, they have the same view. First, they have full access to and control over their local data and model updates, including parameters, neuron indices (i.e., model structures), and neuron importance scores. Second, the clients have access to the global model's parameters as well as the training algorithm. These are the standard practices in FL being adopted. Additionally, following prior work [7, 16, 18, 37, 42], we assume that the majority of participating clients are benign.
Goals. ClieND is designed to assist the benign clients in achieving the detection of whether poisoning attacks exist in an FL framework from the client side. Specifically, we define two goals for ClieND.

- ***Goal #1 (Effective detection)***. Successfully detect the presence of poisoning attacks.
- ***Goal #2 (Early detection)***. A successful detection should be fulfilled at an early stage.

We provide the formal definitions of the two goals below.

Definition 1 (Effective detection). *Let $\mathcal{T}$ be the total set of training rounds, and $\mathcal{T}_{attack} \subseteq \mathcal{T}$ be the set of rounds during which poisoning attacks occur. Let $\mathcal{D}_i^{(t)} \in \{0, 1\}$ denote the detection outcome of client i at round t, where $\mathcal{D}_i^{(t)}=1$ indicates a detected attack, and $\mathcal{D}_i^{(t)}=0$ indicates a clean round. An effective detection is if there exists a detection function $\mathcal{D}_i^{(t)}$ such that:*

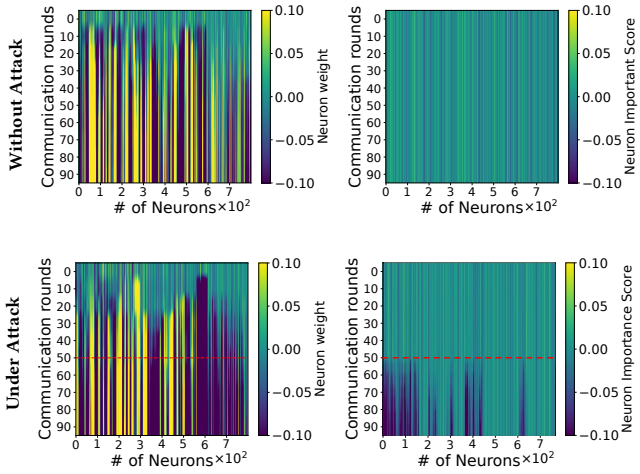
$$\begin{cases} \Pr(\mathcal{D}_i^{(t)} = 0 \mid t \notin \mathcal{T}_{attack}) \geq 1 - \epsilon_1; \\ \Pr(\mathcal{D}_i^{(t)} = 1 \mid t \in \mathcal{T}_{attack}) \geq 1 - \epsilon_2, \end{cases} \tag{2}$$

where ϵ_1 and ϵ_2 denote acceptable upper bounds on the false positive and false negative rates, respectively, with $\epsilon_1, \epsilon_2 \ll 1$.

Definition 2 (Early detection). *Let t_0 denote the starting round of a poisoning attack, and t_{det} be the earliest round at which the client i achieves effective detection, i.e., $\mathcal{D}_i^{(t_{det})}=1$. Detection is considered early if:*

$$\Pr\left(t_{det} - t_0 \leq \Delta\right) \geq 1 - \epsilon, \tag{3}$$

where Δ is a small constant bounding the detection delay, ϵ bounds the probability of not achieving an early detection with $\epsilon \ll 1$.



(a) Weights **(b) Importance scores**

Figure 1: A comparison of neuron weights (a) and importance scores (b) with and without poisoning attacks. The red dashed line indicates the commencement of the attack.

4 Approach

In this section, we delve into the details of ClieND. We begin by providing the insight behind the approach design and outlining the workflow of ClieND, and then introducing its detailed techniques.

4.1 Overview

Insight and motivation. According to the existing work regarding the poisoning attacks in FL [3, 4, 6, 12, 15, 47], such attacks often cause unusual shifts in the weights of affected neurons in the global model, indicating changes in their importance. However, neuron importance is better evaluated using an *importance score*, based on behavior across data samples [45, 58], rather than weights. Fluctuations in these scores can serve as clearer indicators of poisoning attacks. Figure 1 compares neuron weight and importance

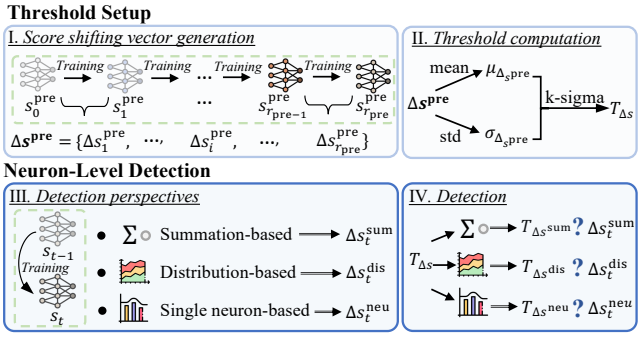


Figure 2: The overall workflow of CLIEND.

score changes under attack and non-attack conditions, with both standardized to the range between [-0.10, 0.10]. Obviously, without attacks, neuron weights fluctuate, while importance scores remain stable. When an attack occurs at the 50$^{\text{th}}$ round (marked by a red dashed line), weight changes are subtle, making attacks hard to detect. In contrast, importance scores show a sharp decline, clearly indicating the attack. Thus, tracking changes in neuron importance scores can help detect poisoning attacks at specific communication rounds. Theoretical analysis as a foundation is listed in Section 5.1.

Overall workflow. Figure 2 illustrates the overall workflow of CLIEND. Once the model begins training within the FL framework, the first step is that each client must set a *threshold* for neuron importance score changes during the pre-training rounds. This involves each client calculating the range of shifting in neuron importance scores between every two consecutive iterations. These values collectively form a score shifting vector. Then, CLIEND establishes a statistical-based threshold $T_{\Delta s}$ based on this score shifting vector. In the second step, each client executes the detection during the FL training phase. CLIEND introduces three detection perspectives, i.e., *summation-based*, *distribution-based*, and *single neuron-based* detection (detailed in Section 4.3), as the score shifting computing methods, ensuring the detection covers almost all types of poisoning attacks. Then CLIEND calculates the values of score shifting between each two consecutive training iterations under each detection perspective and compares them with their respective thresholds. If any of these score-shifting values exceed the thresholds, i.e., $\Delta s_t > T_{\Delta s}$, CLIEND concludes that a poisoning attack is present.

4.2 Threshold Setup

The threshold setup is conducted during the pre-training rounds, where a baseline is established to evaluate whether a client may be involved in a poisoning attack in FL training. This process mainly consists of two components, i.e., score shifting vector generation and threshold computation.

Score shifting vector generation. To set up a threshold, we need to obtain the neuron importance score shifting between each training round. Thus, we first introduce the way that CLIEND calculates the neuron importance score. Following the scoring method proposed in Hydra[45], CLIEND initializes the neuron importance scores on each client and iteratively updates them in every local training round based on the latest global model. Eq. 4 presents the specific scoring method for CLIEND:

$$s_{t+1} = \arg\min_{s_t} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(\theta_t, s_t, x, y)\right], \tag{4}$$

where s_t and s_{t+1} represent the score at round t and t+1 respectively, $\mathcal{L}$ represents the local models' loss function. The remaining terms are consistent with those in Eq. 1. Then CLIEND computes the neuron importance scores for each pre-training round using Eq. 4, and calculates the score shifts between every two consecutive rounds (as detailed in Section 4.3.2). This results in a vector Δs^{pre}, which captures the per-round score variations during the pre-training phase and is then used for threshold computation.

Threshold computation. Once the pre-training rounds come to an end, CLIEND computes the threshold $T_{\Delta s}$ of score shifting based on the values in Δs^{pre}, by leveraging the k-sigma rule principle. Eq. 5 formally presents the computation expression:

$$T_{\Delta s} = \mu_{\Delta s^{\text{pre}}} + k \cdot \sigma_{\Delta s^{\text{pre}}} \leq \max \Delta s_t^{\text{pre}}, \text{ for } t \in [1, r_{\text{pre}}-1], \tag{5}$$

where r^{pre} represents the total number of pre-training rounds, $\mu_{\Delta s^{\text{pre}}} = \frac{1}{r_{\text{pre}}-1} \sum_{i=1}^{r_{\text{pre}}-1} \Delta s_i^{\text{pre}}$ is the mean of score changes of all rounds in pre-training, $\sigma_{\Delta s^{\text{pre}}} = \sqrt{\frac{1}{r_{\text{pre}}-1} \sum_{i=1}^{r_{\text{pre}}-1} \left(\Delta s_i^{\text{pre}} - \mu_{\Delta s^{\text{pre}}}\right)^2}$ is the standard deviation of them, and k is the confidence interval coefficient to control the sensitivity of anomaly detection. We typically set k=2, which corresponds to a 95% confidence interval assuming a near-normal distribution [40]. This setting ensures that the threshold remains below the maximum score change while effectively reducing the impact of anomalies in Δs^{pre}.

4.3 Neuron-Level Detection

Upon receiving the latest global model, each client performs the detection locally during its training round. After obtaining the threshold, by comparing it with the neuron importance score shifting, CLIEND can achieve the neuron-level detection. We provide three potential cases of score spike that may occur during targeted and/or untargeted poisoning attacks and formulate detection perspectives accordingly.

4.3.1 Score spike cases. According to the existing studies [45, 58], there are three possible patterns of score shifting: *1) a spike in the scores of most neurons*; *2) a spike in the scores of some non-specific neurons*; and *3) a spike in the scores of certain neurons of the targeted class*. These cases correspond to different types of poisoning attacks, including both targeted and untargeted variants, covering the most mainstream attack cases. We list the detailed cases below.

- **(*Pattern #1*) Corrupting overall model performance.** This is the most severe case that can occur when poisoning attacks aim to corrupt the entire model. In this case, the model's predictions across all classes are undermined, leading to an abnormal spike in the importance scores of almost all neurons. This case typically occurs in the context of untargeted attacks.
- **(*Pattern #2*) Partial degradation with retained learning.** This case occurs when some neurons maintain their functionality while others are compromised, leading to the importance scores of certain neurons spiking unusually, while others remain stable or exhibit fluctuation. This case is observed in both early-stage untargeted and targeted attack scenarios.

- **(*Pattern #3*) Misleading model performance on targeted samples.** This case arises when the attacks are aimed at misleading the model's predictions on specific data samples, affecting only certain neurons while leaving others nearly unaffected. Specifically, only the scores of neurons related to targeted classes are affected following the attack, which commonly occurs in typical targeted poisoning attacks.

To more intuitively present the score spike of multiple poisoning attacks, we visualize the score shifting patterns caused by them in Figure 4 in Appendix B including a detailed explanation.

4.3.2 Detection perspectives. Based on the three widely adopted cases and their score spike patterns, we systematically design three detection perspectives for poisoning attack detection.

- **Summation-based detection (Sum.).** To quantify the spike of neuron scores holistically, we consider the collective scores of all neurons as a unified entity for the first case. Specifically, we start by calculating the total sum of all neuron scores in each round. We then measure the difference Δs_t^{sum} between the current and previous round scores using the L1-norm:

$$\Delta s_t^{\text{sum}} = \sum_{i=1}^{N} \left| s_t^i - s_{t-1}^i \right|, \tag{6}$$

  where s_t^i is the current-round (i.e., t^{th} iteration) importance score of the i^{th} neuron, s_{t-1}^i is the previous-round (i.e., $(t-1)^{\text{th}}$ iteration) importance score of the i^{th} neuron and N is the total number of neurons.
- **Distribution-based detection (Dis.).** In the second case, a significant spike in the scores of certain neurons leads to a noticeable shift in the overall score distribution. Therefore, detecting poisoning attacks based on distributional changes becomes a reliable strategy. To quantify this shift, we employ the Kullback–Leibler (KL) divergence as a metric. CLIEND identifies potential poisoning attacks by comparing the neuron score distribution of the current round with that of the preceding round.

$$\Delta s_t^{\text{dis}} = \sum_{i=1}^{N} P_{t-1}(s^i) \log \left(\frac{P_{t-1}(s^i)}{P_t(s^i)} \right), \tag{7}$$

  where $P_{t-1}(s^i)$ and $P_t(s^i)$ are the probability distributions of the importance score of the i^{th} neuron in the previous and current round, respectively.
- **Single neuron-based detection (Neu.).** We apply the single neuron-based detection to the third case, as only the neurons associated with the affected classes exhibit abnormal spikes. We use a single neuron-based method to detect potential attacks by monitoring the changes in the importance scores of individual neurons, comparing each of the top 10% affected neurons' current score with the previous-round score to identify the spikes. The formal expression can be presented as Eq. 8.

$$\Delta s_t^{\text{neu}} = \left| s_t^i - s_{t-1}^i \right|, \tag{8}$$

  where $i \in$ top 10% affected neurons. The top 10% neurons are selected based on their ability to retain model performance, which are easily affected by poisoning attacks, as demonstrated in prior studies [45, 58].

4.3.3 Detection. In practice, the specific type of poisoning attack is typically unknown. Therefore, to ensure comprehensive detection coverage, CLIEND performs detection from three aforementioned complementary perspectives. The core principle of CLIEND is to compare the observed score shifts in each training round with a pre-computed threshold to determine the presence of poisoning attacks. Based on the threshold computation method in Eq. 5 and the defined detection perspectives, CLIEND independently computes a dedicated threshold for each perspective, i.e.,$T_{\Delta s^{\text{sum}}}$, $T_{\Delta s^{\text{dis}}}$, and $T_{\Delta s^{\text{neu}}}$, and conducts detection by comparing the corresponding score shifts with their respective thresholds. CLIEND determines the existence of a poisoning attack if the neuron score shift under any detection perspective exceeds its corresponding threshold; Otherwise, the round is considered benign.

Computational cost. To demonstrate the low overhead of the detection, we quantitatively analyze its computational complexity of $O(N \cdot \log N)$, which is presented in Appendix C in detail.

5 Theoretical Analysis

In this section, we rigorously establish the theoretical foundations of CLIEND. We begin by formalizing the client-side advantage in FL, proving that the local dataset is crucial for observing the *discrepancy spike* under the attack. Then we prove that this advantage enables effective attack detection through *neuron importance score shifting*.

5.1 Client-Side Advantage

DEFINITION 3 (DISCREPANCY METRICS). *Let $\theta^{(t)}$ be the global model at round t and $\theta_i^{(t)}$ be the local model trained by client i at round t. Denote the local data distribution of client i by $P_i(x, y)$. We define the global discrepancy and local discrepancy as:*

$$d_{global}^{(t)} := \mathcal{L}(\theta^{(t)}; P_i); \quad d_{local}^{(t)} := \mathcal{L}(\theta_i^{(t)}; P_i), \tag{9}$$

where $\mathcal{L}(\theta; P_i)$ denotes the expected predictive loss over distribution P_i, defined as:

$$\mathcal{L}(\theta; P_i) := \mathbb{E}_{(x,y)\sim P_i} \left[\ell(f_\theta(x), y) \right]. \tag{10}$$

Here, $\ell(\cdot, \cdot)$ is the pointwise loss function*, such as cross-entropy or squared error, which measures the prediction error of $f_\theta(x)$ against ground truth y on a single data point. In contrast, $\mathcal{L}(\theta; P_i)$ aggregates this error over the entire distribution P_i, and serves as a discrepancy measure between model θ and client i's local data.*

This discrepancy aligns with the classical notion of population risk in statistical learning theory [50]. Specifically, the local and global discrepancies quantify a model's expected predictive loss over a client's true data distribution. In the context of FL, these discrepancy metrics provide a principled way to assess model alignment from the perspective of individual clients.

ASSUMPTION 1 (SMOOTHNESS OF LOSS). *The expected loss $\mathcal{L}(\theta; P_i)$ is β-smooth for all clients i, i.e.,*

$$|\mathcal{L}(\theta'; P_i) - \mathcal{L}(\theta; P_i)| \le \beta \|\theta' - \theta\|, \quad \forall \theta, \theta'. \tag{11}$$

This is a standard and mild assumption in optimization theory. Specifically, it requires the expected loss function $\mathcal{L}(\theta; P_i)$ to be β-smooth [9], i.e., its gradients are Lipschitz continuous [9], which is widely adopted [31, 34]. Such smoothness conditions are commonly assumed in non-convex learning theory and are generally

satisfied by modern deep networks employing smooth activation functions (e.g., ReLU). In our context, this property enables bounding the change in loss values across model updates.

ASSUMPTION 2 (BOUNDED UPDATE STEPS). *The global and local model updates are bounded by constants ρ_1 and ρ_2 such that:*

$$\|\theta^{(t+1)} - \theta^{(t)}\| \le \rho_1, \quad \|\theta_i^{(t+1)} - \theta_i^{(t)}\| \le \rho_2, \quad \forall t. \tag{12}$$

This is also a reasonable and practical assumption, as it bounds the magnitude of model updates across training rounds, both at the global and local levels. In typical FL systems such as FedAvg, this condition is naturally satisfied by constraining the number of local SGD steps, applying appropriate learning rate schedules, or employing gradient clipping.

THEOREM 1 (DISCREPANCY DYNAMICS UNDER ATTACK). *Given $d_{global}^{(t)}$ and $d_{local}^{(t)}$, both discrepancies are smooth during clean training, i.e. $|d_{global}^{(t+1)} - d_{global}^{(t)}| \le \delta_{clean}$, and $|d_{local}^{(t+1)} - d_{local}^{(t)}| \le \delta_{clean}$, where $\delta_{clean} := \max(\beta\rho_1, \beta\rho_2)$ and β is the smoothness constant of the loss function. If a poisoning attack at round t^* introduces a malicious perturbation Δ_{attack} to the global model, such that $\tilde{\theta}^{(t^*)} = \theta^{(t^*)} + \Delta_{attack}$ with $\|\Delta_{attack}\| = \delta_{adv}$, then when the attack strength satisfies:*

$$\delta_{adv} > \sqrt{2\delta_{clean}/\beta}. \tag{13}$$

We have the following spike in both discrepancies:

$$|d_{global}^{(t^*)} - d_{global}^{(t^*-1)}| > \delta_{clean}, \quad |d_{local}^{(t^*)} - d_{local}^{(t^*-1)}| > \delta_{clean}. \tag{14}$$

Theorem 1 demonstrates that under clean training, discrepancy changes remain bounded due to smoothness, while poisoning attacks induce detectable spikes in both global and local discrepancy values. This validates the use of discrepancy-based thresholds as reliable indicators for identifying abnormal updates. The detailed proof is presented in Appendix D.1.

THEOREM 2 (DATASET AS AN ADDITIONAL ADVANTAGE IN CLIENT-SIDE DETECTION). *Given $d_{global}^{(t)}$ and $d_{local}^{(t)}$, client i has access to a dataset $\mathcal{D}_i \sim P_i$ of size n_i, and that the loss function ℓ is bounded. Then:*

(i) Client i can compute unbiased empirical estimates of both discrepancies using $\mathcal{D}_i$;

(ii) If an attack causes the discrepancies to spike beyond δ_{clean}, the client can observe such abnormal changes.

This enables the client to monitor model alignment concerning its own distribution, offering a key advantage in client-side detection.

Theorem 2 mainly indicates that clients' local data can serve as an advantage to detect the presence of poisoning attacks. The detailed proof is listed in Appendix D.2.

5.2 Effectiveness of CLIEND

THEOREM 3 (DETECTION GUARANTEE OF CLIEND). *Let $\mathcal{D}_i^t = \mathbb{I}[\Delta s_t > T_{\Delta s}]$ express the detection of CLIEND, where $\Delta s_t = \sum_{i=1}^{N} |s_t^i - s_{t-1}^i|$ is the L1-norm neuron score shift at round t, with N being the number of neurons. Let $T_{\Delta s} = \max_{t_{pre} \in [1, r_{pre}-1]} \Delta s_t$ denote the maximum shifting observed in the first r_{pre} pre-rounds. Let $\kappa > 0$ be a positive constant characterizing the sensitivity of neuron importance scores to loss discrepancy. Given that the neuron scores are estimated with bounded error $|\hat{s}_t^i - s_t^i| \le \delta_s$, and the expected clean-round shift as $\mu_{clean} = \mathbb{E}[\Delta s_t \mid t \notin \mathcal{T}_{attack}]$. If a poisoning attack at round $t \in \mathcal{T}_{attack}$ causes a discrepancy spike $|d_t - d_{t-1}| \ge \Delta_d$, then CLIEND satisfies Eq. 2 in Definition 1, where $\epsilon_1 = \exp\left(-\frac{2(T_{\Delta s} - \mu_{clean})^2}{N\delta_s^2}\right)$, and $\epsilon_2 = \exp\left(-\frac{2(\kappa\Delta_d - T_{\Delta s})^2}{N\delta_s^2}\right)$.*

Theorem 3 establishes the theoretical effectiveness of CLIEND by demonstrating that it satisfies the two probabilistic guarantees defined in Definition 1, i.e., low false alarm rate in clean rounds and high detection rate under attack. In the clean case, the neuron score shift remains bounded, and the threshold $T_{\Delta s}$ is conservatively chosen based on the maximum observed shift during the warm-up phase. In the attack case, a significant discrepancy spike results in a neuron score shift that exceeds the clean threshold with high probability. Together, these results provide formal guarantees that CLIEND can reliably distinguish between clean and malicious rounds by monitoring neuron score dynamics, thus achieving provably effective poisoning attack detection. The detailed proof is presented in Appendix D.3.

6 Evaluation

6.1 Experimental Settings

Datasets and model architectures. In our evaluation, we utilize four widely adopted datasets, CIFAR-10 [29], MNIST [14], TEXAS hospital dataset [1] and ImageNet [13]. For model architectures, we employ a convolutional neural network (CNN) for CIFAR-10, a ResNet for ImageNet, and a fully connected neural network (FCN) for both MNIST and TEXAS datasets. Detailed descriptions of the datasets and model architectures are provided in Appendix E.

FL settings. We adopt a standard cross-silo FL setup [10, 17, 34, 37] with 100 clients (20% of which are malicious) and one server. We provide the details of FL settings in Appendix F

Poisoning attack settings. The poisoning attacks are initiated immediately after pre-training phase concludes. We choose three untargeted and five targeted attacks to evaluate CLIEND's capabilities. We adpot Min-Max [47], Fang [15], and Min-Activation attacks (Min-Act.) [59] for untargeted attacks, Distributed backdoor attack (DBA) [55], LIE [4], Label-flipping (Lab-flip.) [6], and Neuron-separation (Neu-sep.) [59] for targeted attacks. Additionally, we also adopt constrain-and-scale attack [5] for adaptive attack (Adap.). The details of the attack are in Appendix G.

Baselines. We select three detection based server-side defenses, FLAME [37], CrowdGuard [42], and SHERPA [43]. We re-implement FLAME based on the description provided in its original paper. The implementation of CrowdGuard is adopted directly from the publicly available source code [19], using the default hyperparameter settings. For SHERPA, we employ the Deep Explainer and use HDBSCAN as the clustering method.

Metrics. We adopt true positive rate (TPR) and false positive rate (FPR) to evaluate the effectiveness of CLIEND detection. Additionally, we measure the number of communication rounds required for detecting the existence of the attack to assess its early detection capability. We provide the detailed explanation of these metrics in Appendix H.

6.2 CLIEND Effectiveness

To assess CLIEND's capability of achieving ***Goal #1***, we benchmark the effectiveness of CLIEND on detecting three untargeted and five targeted poisoning attacks against three baseline methods,

Table 1: The average TPR over all training rounds under poisoning attacks for each method.

(a) TPR.

	Attacks	CIFAR-10						MNIST						TEXAS						ImageNet					
		FLAME	CG.	SHERPA	CLIEND			FLAME	CG.	SHERPA	CLIEND			FLAME	CG.	SHERPA	CLIEND			FLAME	CG.	SHERPA	CLIEND		
					Sum.	Dis.	Neu.				Sum.	Dis.	Neu.				Sum.	Dis.	Neu.				Sum.	Dis.	Neu.
U.	Min-max	0.64	0.72	0.69	**0.98**	0.96	0.82	0.72	0.82	0.80	**0.99**	0.98	0.98	0.65	0.79	0.72	**0.93**	0.89	0.91	0.68	0.77	0.72	**0.95**	0.90	0.92
U.	Fang	0.59	0.65	0.61	**0.98**	0.96	0.83	0.78	0.88	0.81	**0.99**	**0.99**	0.98	0.63	0.75	0.69	**0.97**	0.92	0.88	0.65	0.78	0.72	**0.96**	0.93	0.91
U.	Min-act.	0.53	0.56	0.51	**0.98**	0.97	0.89	0.77	0.79	0.73	0.98	**0.99**	0.97	0.57	0.71	0.64	**0.99**	0.97	0.95	0.62	0.72	0.69	**0.94**	0.89	0.87
T.	DBA	0.66	0.69	0.73	0.73	0.83	**0.86**	0.64	0.69	0.75	0.76	0.87	**0.93**	0.61	0.69	0.72	0.82	0.87	**0.92**	0.60	0.65	0.73	0.82	0.83	**0.84**
T.	LIE	0.65	0.58	0.69	0.83	0.86	**0.88**	0.78	0.81	0.86	0.89	0.90	**0.91**	0.68	0.77	0.79	0.92	**0.97**	0.95	0.71	0.75	0.73	0.84	**0.85**	0.83
T.	Lab-flip.	0.67	0.62	0.80	0.79	0.87	**0.91**	0.72	0.80	0.89	0.87	0.92	**0.94**	0.71	0.69	0.76	0.82	**0.96**	0.94	0.69	0.70	0.76	0.85	**0.95**	0.92
T.	Neu-sep.	0.64	0.67	0.65	0.72	**0.98**	0.84	0.72	0.75	0.72	0.78	**0.89**	0.86	0.61	0.72	0.74	0.86	0.89	**0.93**	0.62	0.68	0.69	0.87	0.91	**0.90**
T.	Adap.	0.51	0.57	0.72	**0.78**	0.77	0.62	0.42	0.52	0.65	0.62	**0.72**	0.63	0.51	0.47	0.59	**0.79**	0.72	0.73	0.53	0.50	0.63	**0.80**	0.73	0.71

(b) FPR.

	Attacks	CIFAR-10						MNIST						TEXAS						ImageNet					
		FLAME	CG.	SHERPA	CLIEND			FLAME	CG.	SHERPA	CLIEND			FLAME	CG.	SHERPA	CLIEND			FLAME	CG.	SHERPA	CLIEND		
					Sum.	Dis.	Neu.				Sum.	Dis.	Neu.				Sum.	Dis.	Neu.				Sum.	Dis.	Neu.
U.	Min-max	0.65	0.41	0.45	**0.08**	0.10	0.18	0.56	0.33	0.42	**0.01**	0.02	**0.01**	0.51	0.38	0.46	**0.12**	0.15	0.13	0.60	0.40	0.39	**0.10**	0.14	0.12
U.	Fang	0.45	0.39	0.34	**0.05**	0.09	0.11	0.48	0.35	0.36	**0.01**	**0.01**	**0.01**	0.48	0.22	0.21	**0.13**	0.11	0.13	0.49	0.37	0.29	**0.11**	0.14	0.10
U.	Min-act.	0.68	0.34	0.32	**0.05**	0.06	0.12	0.50	0.22	0.28	0.02	**0.01**	**0.01**	0.46	0.31	0.34	**0.07**	0.15	0.14	0.51	0.32	0.31	**0.08**	0.15	0.10
T.	DBA	0.42	0.22	0.25	**0.05**	0.07	0.14	0.47	0.32	0.30	0.02	**0.01**	0.03	0.39	0.37	0.26	**0.09**	0.18	0.16	0.51	0.32	0.31	**0.08**	0.15	0.13
T.	LIE	0.43	0.21	0.29	**0.06**	0.13	0.09	0.52	0.32	0.32	**0.01**	**0.01**	**0.01**	0.42	0.29	0.20	**0.09**	0.12	0.18	0.43	0.39	0.33	**0.12**	0.15	0.14
T.	Lab-flip.	0.46	0.32	0.26	0.11	**0.07**	0.15	0.43	0.20	0.29	**0.01**	**0.01**	0.03	0.41	0.23	0.24	**0.10**	0.13	0.11	0.45	0.33	0.26	**0.09**	0.12	0.10
T.	Neu-sep.	0.69	0.28	0.29	0.14	0.12	**0.06**	0.48	0.31	0.34	**0.02**	0.03	0.08	0.39	0.26	0.32	0.16	0.18	**0.14**	0.48	0.37	0.33	**0.10**	0.12	0.11
T.	Adap.	0.48	0.52	0.32	0.15	0.17	**0.12**	0.49	0.42	0.38	**0.12**	**0.12**	0.13	0.59	0.57	0.51	**0.11**	0.12	0.13	0.50	0.48	0.38	**0.13**	0.14	0.12

FLAME, GrowdGuard, and SHERPA. We first report the average TPR and FPR over all training rounds for each method in Table 1, and then provide the impact of different data distributions on the effectiveness of CLIEND in Table 2.

TPR. According to Table 1a, we observe that CLIEND consistently outperforms FLAME, CrowdGuard and SHERPA across three benchmarking datasets regarding the TPR for detecting both untargeted and targeted attacks. Specifically, when detecting untargeted attacks, such as the Min-activation attack, CLIEND achieves a TPR exceeding 0.89 from all three detection perspectives on CIFAR-10, with the sum-based perspective reaching the highest TPR of 0.98, significantly surpassing FLAME's 0.53, CrowdGuard's 0.56 and SHERPA's 0.51. Similar results are observed for the Fang and Min-Max attacks across the MNIST, TEXAS and ImageNet datasets, where CLIEND achieves a TPR as high as 0.99, far exceeding the performance of baselines.

Additionally, when detecting targeted attacks, CLIEND's three detection perspectives consistently exhibit the highest TPRs. For example, under the LIE attack, CLIEND's TPR exceeds 0.92 across all three perspectives, with the distribution-based perspective reaching as high as 0.97 when evaluating on the TEXAS dataset, significantly outperforming the three baselines. Similar results are observed across the other three datasets for other targeted attacks. We also evaluate CLIEND's detection performance under an adaptive adversary in targeted attacks. The results display that three baseline methods can only achieve a TPR around 0.50, but CLIEND can achieve the highest TPR of 0.80.

FPR. We also benchmark CLIEND's effectiveness from the perspective of false positives. The results presented in Table 1b demonstrate that CLIEND consistently achieves a significantly lower FPR compared to the three baselines in detecting both untargeted and targeted attacks, highlighting its stability in the absence of attacks. Specifically, when detecting Fang attack, CLIEND achieves an FPR of 0 on MNIST using the sum-based perspective, which is far lower than FLAME's 0.48, CrowdGuard's 0.35 and SHERPA's 0.36. Similar results are observed for the Min-Max and Min-Activation attacks across all four datasets, where CLIEND maintains exceptionally low FPRs under its three detection perspectives.

Furthermore, in detecting targeted attacks, CLIEND demonstrates similar results to those observed for untargeted attacks. For example, in detecting the Distributed attack on CIFAR-10, CLIEND achieves FPRs of 0.05, 0.07, and 0.14 across the three detection perspectives, all of which are substantially lower than the three baselines. Comparable results are observed across four datasets for the other targeted attacks. Notably, even in the presence of adaptive adversaries, CLIEND effectively maintains the FPR below 0.17, with the best performance achieved on the TEXAS dataset at 0.11. In contrast, the three baselines exhibit significantly higher FPRs, reaching over 0.5. Therefore, we conclude that CLIEND maintains high detection accuracy regarding FPR in the absence of attacks.

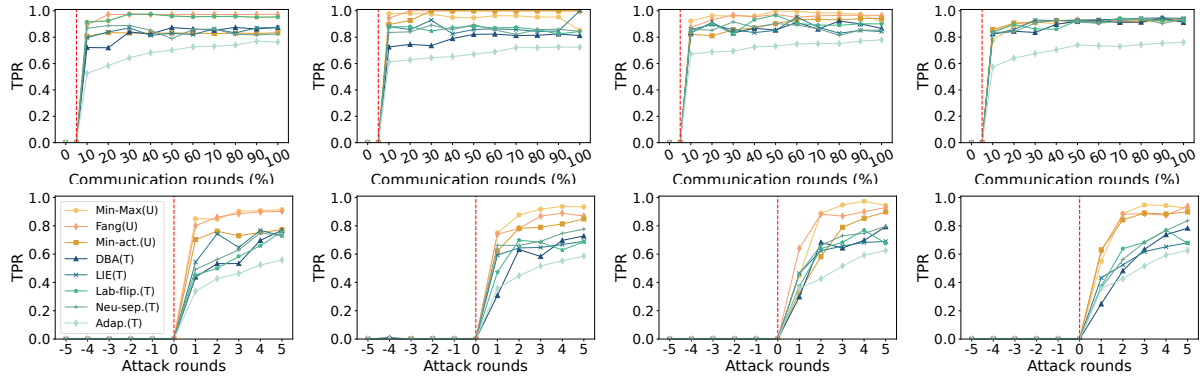
Table 2: Impact of non-IID rates of data.

(a) TPR.

Poisoning Attacks		$q = 0.3$			$q = 0.6$			$q = 0.9$		
		Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.
Unta.	Min-max	0.97	0.96	0.80	0.97	0.95	0.81	0.90	0.77	0.79
Unta.	Fang	0.94	0.93	0.82	0.92	0.94	0.80	0.89	0.86	0.75
Unta.	Min-act.	0.96	0.95	0.84	0.91	0.90	0.82	0.86	0.83	0.80
Targ.	DBA	0.73	0.80	0.82	0.71	0.77	0.77	0.70	0.76	0.74
Targ.	LIE	0.82	0.82	0.83	0.83	0.80	0.79	0.78	0.77	0.76
Targ.	Lab-flip	0.86	0.89	0.73	0.85	0.89	0.70	0.82	0.85	0.68
Targ.	Neu-sep.	0.71	0.97	0.83	0.70	0.95	0.83	0.70	0.94	0.80
Targ.	Adap.	0.69	0.70	0.72	0.69	0.69	0.73	0.68	0.71	0.72

(b) FPR.

Poisoning Attacks		$q = 0.3$			$q = 0.6$			$q = 0.9$		
		Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.
Unta.	Min-max	0.08	0.10	0.17	0.10	0.12	0.17	0.12	0.14	0.19
Unta.	Fang	0.05	0.10	0.12	0.08	0.09	0.12	0.09	0.12	0.15
Unta.	Min-act.	0.04	0.05	0.14	0.05	0.08	0.15	0.08	0.10	0.17
Targ.	DBA	0.06	0.07	0.15	0.08	0.07	0.16	0.08	0.09	0.18
Targ.	LIE	0.07	0.12	0.10	0.06	0.15	0.11	0.09	0.16	0.11
Targ.	Lab-flip.	0.14	0.09	0.12	0.15	0.10	0.14	0.15	0.09	0.18
Targ.	Neu-sep.	0.15	0.11	0.07	0.18	0.13	0.09	0.17	0.13	0.12
Targ.	Adap.	0.10	0.08	0.09	0.11	0.09	0.06	0.09	0.11	0.14



(a) CIFAR-10 (b) MNIST (c) TEXAS (d) ImageNet

Figure 3: The performance of ClieND on early detection across four datasets under eight poisoning attacks.

Impact on non-IID rate. We then explore the impact of data distribution (non-IID rates) on ClieND's detection effectiveness on CIFAR-10. We set q=0.3, 0.6, and 0.9 to simulate real-world scenarios where each client has differently distributed data. A higher q indicates a more pronounced non-IID distribution. As shown in Table 2a, ClieND demonstrates excellent performance across various data distributions. Overall, both in untargeted and targeted attacks, as q increases, ClieND's TPR experiences a slight decline but remains at a consistently high level. For instance, in detecting Min-Max attack, the TPR of the sum-based perspective reaches 0.97 for q=0.3 and 0.6, only 0.01 lower than in the IID setting (where q=0), and it remains as high as 0.90 for q=0.9. Table 2b further presents the FPR of ClieND under different non-IID rates of data. The results indicate that varying non-IID rates q have minimal impact on the FPR of ClieND. Similar conclusions are observed for targeted attacks. For more extreme data heterogeneity (e.g., single class), we plot ClieND's FNR and FPR curves under extreme data heterogeneity across three detection perspectives in Appendix I.

> **Remark**. ClieND achieves ***Goal #1*** by demonstrating superior detection effectiveness, as evidenced by both TPR and FPR, consistently outperforming the three baseline methods. Moreover, its performance remains stable across different data distributions, further validating ClieND's strong detection capability.

6.3 Early Detection

We adopt communication times to evaluate ClieND's ability to achieve ***Goal #2***. Figure 3 presents the highest TPR among the three detection perspectives of ClieND every 10% of the total training rounds under eight different poisoning attacks across four datasets. The red dashed vertical line indicates the point at which the attack begins. In our settings, a TPR close to 0 indicates that the majority of clients believe no attack is present, while a TPR close to 1 indicates that most clients recognize the presence of an attack. Before the attack begins, the TPR should be 0.

The first line of Figure 3a illustrates the results on CIFAR-10. We observe that within 5% rounds (50 rounds) after the attack commencement, the TPR for all the attacks excluding adaptive attack exceeds 0.7, with the TPRs for the Min-Max and Label-flipping attacks even surpassing 0.9. As the rounds progress, there is still a slight upward trend in TPR overall. For the adaptive attack, its strong stealth at the onset makes it extremely challenging to detect. Nevertheless, ClieND achieves a TPR exceeding 0.5 within the first 5% attack rounds. As the attack intensifies with increasing communication rounds, ClieND is able to ultimately achieve a TPR close to 0.8 in the later stages. The first line of Figure 3b, 3c, and 3d illustrates the early detection performance of ClieND on MNIST, TEXAS, and ImageNet, respectively.

To more clearly illustrate how many communication rounds ClieND are required for detection, the second line of Figure 3 displays the its TPR values in the first 5 rounds following the onset of the attack. Figure 3a demonstrates that on CIFAR-10, within the first round after the attack begins, ClieND achieves a TPR exceeding 0.6 under untargeted attacks. By the third round, the TPR for targeted attacks surpasses 0.5, while the TPR for untargeted attacks reaches approximately 0.7. By the fifth round, ClieND achieves a TPR exceeding 0.5 for detecting adaptive attacks. Similarly, by analyzing the remaining plots in Figure 3, we draw a conclusion that ClieND can detect untargeted attacks within three communication rounds and targeted attacks, including adaptive attacks, within five rounds. We also illustrate that non-IID rate has little impact on ClieND's early detection. The details are in Appendix J.

> **Remark**. ClieND is capable of detecting the presence of poisoning attacks at an early stage, with untargeted attacks and targeted attacks being identified within three and five rounds, respectively. This demonstrates ClieND's effectiveness in achieving ***Goal #2***.

6.4 Ablation Study

To assess the robustness of ClieND, we conduct ablation studies on CIFAR-10, further assessing ClieND's performance across the following aspects.

Proportion of the malicious clients. We first investigate the impact of the proportion of adversaries on ClieND's detection effectiveness. We denote N to represent the proportion of malicious clients, with N=5%, 30%, and 45% being selected for analysis. Table 3 presents our experimental results. Overall, the variation in N has

Table 3: Impact of proportion of malicious clients.

(a) TPR.

Poisoning Attacks		$N = 5\%$			$N = 30\%$			$N = 45\%$		
		Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.
Unta.	Min-max	0.93	0.90	0.81	0.93	0.90	0.80	0.97	0.94	0.82
	Fang	0.95	0.94	0.80	0.95	0.81	0.81	0.97	0.95	0.82
	Min-act.	0.92	0.93	0.82	0.94	0.92	0.85	0.95	0.93	0.87
Targ.	DBA	0.70	0.79	0.82	0.73	0.81	0.83	0.74	0.82	0.85
	LIE	0.82	0.86	0.83	0.83	0.87	0.87	0.84	0.82	0.88
	Lab-flip.	0.76	0.86	0.88	0.75	0.87	0.88	0.79	0.86	0.89
	Neu-sep.	0.70	0.95	0.82	0.72	0.97	0.80	0.74	0.98	0.79
	Adap.	0.67	0.69	0.60	0.72	0.72	0.62	0.78	0.80	0.66

(b) FPR.

Poisoning Attacks		$N = 5\%$			$N = 30\%$			$N = 45\%$		
		Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.
Unta.	Min-max	0.05	0.07	0.09	0.06	0.08	0.10	0.08	0.11	0.12
	Fang	0.02	0.04	0.05	0.05	0.07	0.08	0.08	0.09	0.11
	Min-act.	0.02	0.03	0.06	0.04	0.06	0.08	0.05	0.07	0.11
Targ.	DBA	0.04	0.06	0.05	0.04	0.06	0.09	0.06	0.09	0.13
	LIE	0.08	0.06	0.09	0.08	0.07	0.10	0.09	0.08	0.11
	Lab-flip.	0.09	0.11	0.10	0.10	0.12	0.09	0.12	0.14	0.13
	Neu-sep.	0.08	0.10	0.12	0.08	0.09	0.13	0.09	0.11	0.13
	Adap.	0.15	0.10	0.12	0.13	0.11	0.10	0.09	0.09	0.10

Table 4: Impact of numbers of pre-training rounds.

(a) TPR.

Poisoning Attacks		$t_{pre} = 0.5\%$			$t_{pre} = 1\%$			$t_{pre} = 2\%$			$t_{pre} = 10\%$			$t_{pre} = 20\%$		
		Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.
Unta.	Min-max	0.74	0.73	0.65	0.87	0.82	0.78	0.96	0.93	0.85	0.98	0.95	0.86	0.97	0.96	0.83
	Fang	0.77	0.71	0.66	0.84	0.82	0.73	0.95	0.88	0.84	0.97	0.96	0.79	0.96	0.94	0.81
	Min-act.	0.70	0.74	0.58	0.82	0.86	0.72	0.97	0.94	0.86	0.93	0.95	0.91	0.98	0.98	0.83
Targ.	DBA	0.52	0.56	0.63	0.58	0.69	0.71	0.71	0.84	0.84	0.76	0.83	0.89	0.74	0.81	0.83
	LIE	0.60	0.52	0.59	0.74	0.69	0.71	0.82	0.85	0.88	0.81	0.87	0.84	0.85	0.83	0.89
	Lab-flip.	0.55	0.66	0.63	0.64	0.77	0.71	0.79	0.89	0.86	0.78	0.85	0.90	0.76	0.88	0.89
	Neu-sep.	0.54	0.60	0.61	0.62	0.73	0.73	0.75	0.95	0.83	0.72	0.96	0.83	0.73	0.98	0.82
	Adap.	0.54	0.53	0.51	0.62	0.59	0.55	0.74	0.76	0.70	0.72	0.71	0.64	0.70	0.71	0.65

(b) FPR.

Poisoning Attacks		$t_{pre} = 0.5\%$			$t_{pre} = 1\%$			$t_{pre} = 2\%$			$t_{pre} = 10\%$			$t_{pre} = 20\%$		
		Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.
Unta.	Min-max	0.24	0.20	0.20	0.18	0.16	0.17	0.12	0.14	0.11	0.08	0.09	0.10	0.04	0.06	0.04
	Fang	0.17	0.22	0.21	0.15	0.17	0.19	0.09	0.12	0.15	0.08	0.08	0.06	0.03	0.04	0.09
	Min-act.	0.23	0.18	0.19	0.19	0.13	0.16	0.14	0.08	0.09	0.08	0.09	0.10	0.03	0.05	0.07
Targ.	DBA	0.24	0.21	0.24	0.20	0.16	0.15	0.13	0.11	0.09	0.08	0.06	0.08	0.03	0.07	0.05
	LIE	0.20	0.16	0.19	0.16	0.15	0.16	0.14	0.09	0.10	0.11	0.10	0.09	0.04	0.08	0.07
	Lab-flip.	0.21	0.23	0.23	0.20	0.17	0.18	0.15	0.12	0.14	0.12	0.10	0.13	0.07	0.04	0.06
	Neu-sep.	0.24	0.23	0.25	0.24	0.20	0.18	0.18	0.14	0.16	0.16	0.10	0.13	0.11	0.06	0.09
	Adap.	0.27	0.24	0.29	0.26	0.20	0.27	0.19	0.16	0.19	0.16	0.15	0.12	0.13	0.10	0.08

a minimal impact on ClieND's effectiveness, as the TPR remains relatively consistent across the three values of N, with a slight increase as N grows. Specifically, for the three untargeted attacks, the sum-based detection perspective achieves a TPR of approximately 0.95 for N=5%, 30%, and 45%, with maximum TPRs of 0.95, 0.95, and 0.97, respectively. Although the TPRs for the distribution-based and neuron-based perspectives show a slight decline, they follow the same general trend. For the targeted attacks including adaptive adversaries, the neuron-based detection perspective maintains a similar TPR across all three values of N, mirroring the trend observed for detecting untargeted attacks.

We also examine the effect of varying N on ClieND's early detection capabilities. Our results show that as N increases, ClieND achieves an earlier detection. The detailed analysis and explanation is in Appendix K.

Number of pre-training rounds. We next explore the impact of using different numbers of pre-training rounds to calculate the threshold of score shifting on ClieND's effectiveness. According to the results in Table 4, ClieND demonstrates excellent detection effectiveness in terms of TPR and FPR across different values of t_{pre}=2%, t_{pre}=10%, and t_{pre}=20%. For instance, under untargeted attacks such as Fang attack, ClieND achieves TPR values of 0.95, 0.97, and 0.96 and FPR values of 0.09, 0.08, and 0.03 for t_{pre}=2%, 10%, 20%, respectively. These results indicate that once a moderate warm-up period is available, the detector becomes stable and highly reliable. However, when t_{pre} is very small (e.g., 0.5% or 1%), the detection effectiveness degrades, reflected in lower TPR and higher FPR across both untargeted and targeted attacks. This is expected, as insufficient pre-training limits the establishment of a stable baseline for neuron-importance dynamics. Overall, these results show that while ClieND requires a non-trivial warm-up phase, a relatively small number of pre-training rounds (e.g., $t_{\text{pre}} \geq 2\%$) is already sufficient to achieve robust detection.

Table 5: Impact of attack starting time.

(a) TPR.

Poisoning Attacks		$t_{start} = 0\%$			$t_{start} = 3\%$			$t_{start} = 10\%$		
		Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.
Unta.	Min-max	0.92	0.89	0.83	0.95	0.92	0.85	0.98	0.94	0.90
	Fang	0.94	0.89	0.86	0.94	0.92	0.86	0.99	0.98	0.95
	Min-act.	0.91	0.90	0.84	0.92	0.90	0.93	0.99	0.96	0.93
Targ.	DBA	0.72	0.79	0.82	0.76	0.85	0.88	0.79	0.91	0.90
	LIE	0.81	0.84	0.85	0.84	0.85	0.87	0.89	0.85	0.91
	Lab-flip.	0.79	0.85	0.86	0.81	0.87	0.92	0.83	0.89	0.91
	Neu-sep.	0.75	0.94	0.80	0.76	0.99	0.84	0.81	0.99	0.85
	Adap.	0.70	0.72	0.61	0.74	0.75	0.64	0.78	0.79	0.72

(b) FPR.

Poisoning Attacks		$t_{start} = 0\%$			$t_{start} = 3\%$			$t_{start} = 10\%$		
		Sum.	Dis.	Neu.	Sum.	Dis.	Neu.	Sum.	Dis.	Neu.
Unta.	Min-max	0.17	0.20	0.21	0.07	0.10	0.16	0.05	0.08	0.14
	Fang	0.15	0.14	0.20	0.05	0.08	0.09	0.04	0.08	0.08
	Min-act.	0.14	0.15	0.20	0.05	0.07	0.12	0.04	0.06	0.12
Targ.	DBA	0.16	0.14	0.21	0.06	0.08	0.14	0.04	0.06	0.10
	LIE	0.14	0.20	0.17	0.06	0.12	0.08	0.05	0.10	0.09
	Lab-flip.	0.18	0.15	0.20	0.10	0.06	0.14	0.08	0.05	0.11
	Neu-sep.	0.23	0.20	0.14	0.13	0.10	0.08	0.11	0.09	0.04
	Adap.	0.22	0.22	0.21	0.17	0.13	0.12	0.10	0.09	0.05

Regarding the early detection, our results show that even with varying t_{pre}, ClieND can detect the presence of a poisoning attack as early as the first round after the attack begins, demonstrating its strong early detection ability. The detailed analysis is in Appendix K.

Timing of attacks. We finally investigate the impact of the attack timing (starting round including the pre-training phase) on ClieND's effectiveness. This is to show whether ClieND remains effective when attacks are launched at different stages of training, even in the pre-training rounds. We set t_{pre}=5%, and evaluate the attack starting from t_{attack}=0%, 3%, and 10%. The results in Table 5 present that ClieND performs excellent detection effectiveness on different timings of attacks. Overall, as t_{attack} increases, the TPRs show a slight upward trend. Specifically, when attack is launched in the pre-training phase, ClieND Specifically, when the attack is launched during the pre-training phase, ClieND still demonstrates strong detection capability. Despite a slight increase in FPR, it achieves TPRs as high as 0.94 for both untargeted and targeted attacks. The effectiveness of ClieND in detecting targeted including adaptive attacks shows a similar trend but with a lower FPR.

Table 6: Impact of top-k% neuron selection.

Poisoning Attacks		TPR			FPR		
		k = 1%	k = 5%	k = 20%	k = 1%	k = 5%	k = 20%
Unta.	Min-max	0.71	0.79	0.88	0.26	0.21	0.16
	Fang	0.73	0.82	0.85	0.20	0.14	0.10
	Min-act.	0.70	0.83	0.94	0.18	0.13	0.09
Targ.	DBA	0.71	0.77	0.89	0.23	0.19	0.10
	LIE	0.77	0.86	0.90	0.16	0.11	0.01
	Lab-flip.	0.69	0.84	0.93	0.16	0.15	0.14
	Neu-sep.	0.70	0.78	0.87	0.12	0.09	0.06
	Adap.	0.50	0.58	0.65	0.19	0.13	0.08

Our results on CLIEND's early detection capability show that the attack timing has limited impact. The detection can be slightly earlier when the attack starts in a later round. The detailed explanation is listed in Appendix K.

Top k% neuron selection. For single neuron-based detection, we also explore how top k% neuron selection would affect CLIEND's detection effectiveness. Table 6 presents the detailed results of CLIEND's TPR and FPR under untargeted and targeted attacks. Overall, increasing k improves TPR and reduces FPR. For example, under Min-Max attack, TPR rises from 0.79 (5%) to 0.82 (10% in Table 1) and 0.88 (20%), while FPR decreases from 0.21 (5%) to 0.18 (10%) and 0.16 (20%). The differences across k=5%, 10%, 20% are modest, indicating that once a sufficient subset of important neurons is monitored, detection becomes stable. In contrast, k=1% leads to a relatively clear degradation: in the range from k=5% to k=20%, the FPR decreases by nearly the same amount as the drop from k=1% to k=5%, indicating a consistently high rate of reduction once k exceeds very small values. This is because monitoring too few neurons might reduce the chance of capturing attack-induced score spikes. These observations justify our default choice of k=10%, which provides a strong balance between detection sensitivity and stability.

6.5 Quantitative Overhead Analysis

To evaluate the overhead introduced by CLIEND, we measure wall-clock time per round, memory footprint, and communication cost under three detection perspectives, and compare them against a standard FedAvg round. Overall, CLIEND introduces a slight RAM overhead (from 908 MB to 1.23-1.42 GB) due to maintaining historical score information, while keeping computation (around 79 s) and communication costs (around 6×10^5 Bytes) close to the FedAvg baseline. The detailed quantitative analysis is in Appendix L.

Remark. CLIEND demonstrates strong scalability and generalization, maintaining high detection effectiveness across four key aspects, varying proportions of malicious clients, different pre-training rounds for score shifting threshold determination, different timings of attacks in the pre-training rounds, and varying top k% neuron selection for single neuron-based detection. These together with CLIEND's low overhead highlight its practical deployment in real-world scenarios.

7 Possible Mitigation after Detection

To further mitigate the impact of poisoning attacks that CLIEND detect, we put forward a mitigation concept that is client-driven and governed by a client-defined protocol. We introduce the method and provide evaluation results.

7.1 Mitigation Pipeline

Voting. When clients determine that they have been attacked by CLIEND, they report this to the server by voting. The server will compile these reports, and if the number of votes exceeds half of the total number of clients, the server declares an attack has occurred.

Protocol. Each client, regardless of whether it has voted for an attack in the current round, participates in a unified reporting protocol after the local training. Specifically, clients submit the indices of the top-k neurons with the largest importance score changes and the corresponding score changes relative to the previous training round. Neuron indices are encoded in the format "x_y", where x denotes the layer number and y denotes the neuron's position within the layer. These reports reflect the client's local observation of suspicious behavior and form the foundation for subsequent server response.

Validation. The server first verifies protocol compliance, which involves submitting the top-k neuron indices and the corresponding importance score changes. Non-compliant clients are immediately eliminated. For the remaining clients, the server assesses the consistency between the reported neuron importance scores and the actual model parameter changes. Specifically, it compares the vector of score changes with the corresponding parameter differences between the client's local model and the previous global model. This alignment is quantified using the Pearson correlation coefficient:

$$r = \frac{\sum_i (\Delta s_i - \overline{\Delta s})(\Delta \theta_i - \overline{\Delta \theta})}{\sqrt{\sum_i (\Delta s_i - \overline{\Delta s})^2} \cdot \sqrt{\sum_i (\Delta \theta_i - \overline{\Delta \theta})^2}}, \tag{15}$$

where Δs_i and $\Delta \theta_i$ denote the score and parameter changes for the i^{th} neuron, and $\overline{\Delta s}$, $\overline{\Delta \theta}$ are their respective means. If the resulting correlation $|r|$ is below 0.5, the update is deemed inconsistent with expected learning behavior and the corresponding client is excluded from aggregation [41, 44].

Weighting. To further mitigate the impact of poisoning attacks on affected benign clients during aggregation, the server employs a dynamic weighting mechanism for their updates. This mechanism assigns weights based on how far a client's score change deviates from the median of those who did not raise an attack vote. Specifically, the weight assigned to client i is defined as:

$$w_i = \frac{1}{1 + \gamma \cdot ||\Delta s_i| - \text{median}(\Delta S_{\text{raise}})|}, \tag{16}$$

where ΔS_{raise} is the set of score changes from non-voting clients and γ is a tunable sensitivity factor. Clients with apparent small or big deviations are considered affected by an attack and are accordingly down-weighted to reduce their influence on the global model.

The server aggregates the clients' models using score-aware weighted aggregation. First, the raw weights are normalized using a softmax function, i.e., $w_i^{\text{norm}} = \frac{\exp(w_i)}{\sum_{j=1}^{N} \exp(w_j)}$. Then, the global model is updated via $\theta_{\text{global}} = \sum_{i=1}^{N} w_i^{\text{norm}} \cdot \theta_i$, where θ_i denotes the local model from client i. This aggregation process ensures that model updates from more reliable clients have proportionally greater influence.

Table 7: ClieND's mitigation across CIFAR-10 and TEXAS.

Defense	Metric	CIFAR-10				Texas			
		Fang	Min-Act.	DBA	Adap.	Fang	Min-Act.	DBA	Adap.
Benign setting	Acc	**76.3%**				**58.2%**			
	ASR	-				-			
FedAvg	Acc	10.3%	23.0%	69.0%	67.5%	6.7%	38.4%	51.4%	52.2%
	ASR	-	-	66.0%	76.7%	-	-	86.3%	92.0%
FLTrust	Acc	41.3%	42.0%	61.0%	61.5%	34.5%	35.0%	53.0%	52.2%
	ASR	-	-	26.0%	28.7%	-	-	30.0%	32.0%
Flame	Acc	23.9%	10.6%	54.1%	62.0%	13.6%	16.8%	49.7%	52.9%
	ASR	-	-	20.2%	22.8%	-	-	14.5%	19.2%
CG	Acc	52.8%	33.2%	70.3%	69.5%	35.1%	31.5%	49.3%	49.5%
	ASR	-	-	16.9%	20.5%	-	-	18.3%	20.0%
FLShield	Acc	54.2%	40.0%	71.7%	70.2%	32.0%	37.5%	56.3%	55.5%
	ASR	-	-	14.5%	26.0%	-	-	15.5%	17.5%
FreqFed	Acc	66.0%	53.7%	73.1%	72.9%	42.0%	41.7%	54.8%	53.8%
	ASR	-	-	14.3%	19.0%	-	-	28.0%	20.5%
FedDefender	Acc	63.8%	59.9%	69.7%	70.8%	41.8%	42.3%	52.6%	53.1%
	ASR	-	-	13.9%	19.9%	-	-	27.3%	23.1%
Our Mitigation	Acc	**74.2%**	**73.3%**	**76.1%**	**75.7%**	**56.9%**	**57.5%**	**56.6%**	**58.0%**
	ASR	-	-	**3.0%**	**7.5%**	-	-	**5.3%**	**7.8%**

7.2 Evaluation on Mitigation

We evaluate the effectiveness of mitigation by comparing its performance against five state-of-the-art defenses under four poisoning attacks. We measure the attack success rate (ASR) for the targeted attack and the model accuracy (Acc) for both the untargeted and targeted attacks. Table 7 presents the model Acc under benign settings for CIFAR-10 and TEXAS, as well as the Acc and ASR of our proposed mitigation and other baseline defense methods under two untargeted poisoning attacks and two targeted poisoning attacks, including those involving adaptive adversaries.

Untargeted attacks. Overall, both Fang and Min-Activation attacks significantly degrade the model's accuracy. Even with existing defenses applied, baseline methods still suffer from significant accuracy drops. In contrast, our mitigation achieves superior performance in defending against these attacks. Specifically, on the TEXAS dataset, our mitigation renders Fang and Min-Activation attacks nearly ineffective, with the model's accuracy dropping by only 1.3% and 0.7%. Similar conclusions can be drawn for CIFAR-10.

Targeted attacks. For targeted attacks, although the overall accuracy of the model does not experience a significant drop due to the label-specific attack target, the ASR is high without defense mechanisms (i.e., in FedAvg). After applying defense mechanisms, the ASR is dropped to some extent, such as FedDefender [39], which lowers the ASR of adaptive attacks from 76.5% to 19.9%. Among all methods, our mitigation significantly outperforms the baseline methods. It maintains model accuracy closer to that in the benign setting, with only a 0.2% drop, while achieving the lowest ASR. For example, on CIFAR-10, the ASR for the distributed backdoor attack and adaptive attack is reduced to 3.0% and 7.5%, respectively. These results highlight the effectiveness of our mitigation.

8 Discussions and Limitations

In this section, we discuss the potential limitations of our work.

Risks of sharing neuron important scores. Although ClieND does not transmit raw data, gradients, or sample-level representations, sharing neuron-importance scores with the server introduces a potential privacy consideration. Even these scores are high-level statistical indicators derived from aggregated model updates and are not directly invertible to reconstruct inputs, they still constitute an additional channel of information beyond standard federated updates. Therefore, we consider a formal privacy guarantee a meaningful future direction, to mathematically provide theoretical foundations for ClieND.

Adaptive vs. defense-aware adversaries. In this work, "adaptive adversary" follows its conventional use in prior FL attack literature [28], referring to an attacker capable of optimizing malicious updates while remaining unaware of any defense mechanisms. However, in real world, there exists a stronger variant, called "defense-aware adversary" that can realize the defense mechanisms. In our attack scenario, they may attempt to circumvent the mitigation mechanism by manipulating the neuron scores submitted to the server. We discuss two possible stealthy evasion strategies. First, the adversary may retain a clean local model in parallel with the poisoned one and submit neuron indices and scores extracted from the clean model. Although this creates the illusion of benign behavior in score reporting, it fails the validation that compares the reported scores against the actual parameter changes in the submitted (poisoned) model update. As a result, such updates are detected and eliminated during aggregation. Second, the adversary may upload a normally trained model update without any injection in the current round. In this case, the actual model behavior is benign, so the impact on the global model is negligible regardless of the reported scores. If clean scores are reported, the client is treated as benign, otherwise they are either excluded or down-weighted.

Benign warm-up assumption. The secure warm-up phase is a hard requirement for ClieND. The warm-up allows each client to establish a stable, attack-free estimation of neuron-importance distributions, which subsequently serves as the baseline for detecting anomalous deviations. Although the required number of pre-training rounds is small, this phase must exist, as it anchors ClieND 's ability to distinguish benign behaviors from malicious ones. If this assumption is violated, the importance-score baseline may be skewed before it stabilizes. This may reduce the separability between benign and malicious updates, weakening ClieND 's detection capability. Therefore, designing mechanisms that can bootstrap reliable baselines under early (i.e., first-round) corruption constitutes a promising direction for future work.

9 Conclusion

We propose ClieND, a practical solution for detecting the presence of poisoning attacks within the FL framework. Leveraging the client-side advantage of having access to local datasets and training dynamics, ClieND is conducted on the client side to execute the detection process by observing discrepancy spike of the global model neuron importance score changes between two consecutive training rounds. We provide formal proof demonstrating that the local dataset is an inherent advantage in detecting poisoning attacks, and the effectiveness of ClieND using neuron importance scores for detection. Our experimental results indicate that ClieND successfully detects untargeted and targeted poisoning attacks, including adaptive settings, within an average of five communication rounds. The ablation study further demonstrates the scalability and generalization of ClieND in real-world application scenarios.

References

[1] 2025. TEXAS hospital dataset. https://healthdata.dshs.texas.gov/dashboard/hospitals/texas-hospital-data.

[2] Sebastien Andreina, Giorgia Azzurra Marson, Helen Möllering, and Ghassan Karame. 2021. Baffle: Backdoor detection via feedback-based federated learning. In *Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 852–863.

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

[4] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A little is enough: Circumventing defenses for distributed learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 32.

[5] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing federated learning through an adversarial lens. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 634–643.

[6] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning (ICML)*. 1467–1474.

[7] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*.

[8] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, H Brendan McMahan, Tim Van Overveldt, Dimitrios Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards federated learning at scale: System design. In *Proceedings of the 2nd SysML Conference*.

[9] Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[10] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2021. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS)*.

[11] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[12] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. 2018. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency-choose two. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 248–255.

[14] Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[15] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local Model Poisoning Attacks to {Byzantine-Robust} Federated Learning. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security)*.

[16] Minghong Fang, Seyedsina Nabavirazavi, Zhuqing Liu, Wei Sun, S.S. Iyengar, and Haibo Yang. 2025. Do We Really Need to Design New Byzantine-robust Aggregation Rules?. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[17] Xinguo Feng, Zhongkui Ma, Zihan Wang, Eu Joe Chegne, Mengyao Ma, Alsharif Abuadbba, and Guangdong Bai. 2024. Uncovering Gradient Inversion Risks in Practical Language Model Training. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 3525–3539.

[18] Hossein Fereidooni, Alessandro Pegoraro, Phillip Rieger, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. 2024. FreqFed: A Frequency Analysis-Based Approach for Mitigating Poisoning Attacks in Federated Learning. In *Proceedings of the 31th Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society.

[19] Patrick Foley, Micah J Sheller, Brandon Edwards, Sarthak Pati, Walter Riviera, Mansi Sharma, Prakash Narayana Moorthy, Shi-han Wang, Jason Martin, Parsa Mirhaji, Prashant Shah, and Spyridon Bakas. 2022. OpenFL: the open federated learning library. *Physics in Medicine & Biology* (2022).

[20] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. 2020. The limitations of federated learning in sybil settings. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. 301–316.

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

[22] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* 58, 301 (1963), 13–30.

[23] Chao Huang, Jianwei Huang, and Xin Liu. 2022. Cross-silo federated learning: Challenges and opportunities. *arXiv preprint arXiv:2206.12949* (2022).

[24] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*. 19–35.

[25] Yangfan Jiang, Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2024. Protecting Label Distribution in Cross-Silo Federated Learning. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 113–113.

[26] Ehsanul Kabir, Zeyu Song, Md Rafi Ur Rashid, and Shagufta Mehnaz. 2024. FLShield: A Validation Based Federated Learning Framework to Defend Against Poisoning Attacks. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*. 141–141.

[27] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning* 14, 1–2 (2021), 1–210.

[28] Torsten Krauß and Alexandra Dmitrienko. 2023. Mesas: Poisoning defense for federated learning resilient against adaptive attackers. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1526–1540.

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

[30] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. 2019. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 33. 1544–1551.

[31] Shuofeng Liu, Mengyao Ma, Minhui Xue, and Guangdong Bai. 2025. Modifier Unlocked: Jailbreaking Text-to-Image Models Through Prompts. In *Proceedings of the 2025 IEEE Symposium on Security and Privacy (SP)*. 355–372.

[32] Shuofeng Liu, Zihan Wang, Minhui Xue, Long Wang, Yuanchao Zhang, and Guangdong Bai. 2024. Being transparent is merely the beginning: enforcing purpose limitation with polynomial approximation. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security)*. 6507–6524.

[33] Mengyao Ma, Shuofeng Liu, MAP Chamikara, Mohan Baruwal Chhetri, and Guangdong Bai. 2024. Unveiling intellectual property vulnerabilities of gan-based distributed machine learning through model extraction attacks. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM)*. 1617–1626.

[34] Mengyao Ma, Yanjun Zhang, Pathum Chamikara Mahawaga Arachchige, Leo Yu Zhang, Mohan Baruwal Chhetri, and Guangdong Bai. 2023. Loden: Making every client in federated learning a defender against the poisoning membership inference attacks. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security (ASIACCS)*. 122–135.

[35] Zhongkui Ma, Xinguo Feng, Zihan Wang, Shuofeng Liu, Mengyao Ma, Hao Guan, and Mark Huasong Meng. 2023. Formalizing Robustness Against Character-Level Perturbations for Neural Network Language Models. In *Proceedings of the International Conference on Formal Engineering Methods (ICFEM)*. Springer, 100–117.

[36] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics (AISTATS)*. PMLR.

[37] Thien Duc Nguyen, Phillip Rieger, Roberta De Viti, Huili Chen, Björn B Brandenburg, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, et al. 2022. {FLAME}: Taming backdoors in federated learning. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security)*. 1415–1432.

[38] Jean Ogier du Terrail, Samy-Safwan Ayed, Edwige Cyffers, Felix Grimberg, Chaoyang He, Regis Loeb, Paul Mangold, Tanguy Marchand, Othmane Marfoq, Erum Mushtaq, et al. 2022. Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings. *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), 5315–5334.

[39] Sungwon Park, Sungwon Han, Fangzhao Wu, Sundong Kim, Bin Zhu, Xing Xie, and Meeyoung Cha. 2023. Feddefender: Client-side attack-tolerant federated learning. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 1850–1861.

[40] Friedrich Pukelsheim. 1994. The three sigma rule. *The American Statistician* 48, 2 (1994), 88–91.

[41] Bruce Ratner. 2009. The correlation coefficient: Its values range between+ 1/- 1, or do they? *Journal of Targeting, Measurement and Analysis for Marketing* 17, 2 (2009), 139–142.

[42] Phillip Rieger, Torsten Krauß, Markus Miettinen, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. 2024. CrowdGuard: Federated Backdoor Detection in Federated Learning. In *Proceedings of the 31th Annual Network and Distributed System Security Symposium (NDSS)*.

[43] Chamara Sandeepa, Bartlomiej Siniarski, Shen Wang, and Madhusanka Liyanage. 2024. SHERPA: Explainable robust algorithms for privacy-preserved federated

learning in future networks to defend against data poisoning attacks. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*. 4772–4790.
[44] Patrick Schober, Christa Boer, and Lothar A Schwarte. 2018. Correlation coefficients: appropriate use and interpretation. *Anesthesia & Analgesia* 126, 5 (2018), 1763–1768.
[45] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. 2020. Hydra: Pruning adversarially robust neural networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 19655–19666.
[46] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. {Explanation-Guided} backdoor poisoning attacks against malware classifiers. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security)*. 1487–1504.
[47] Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS)*.
[48] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1354–1371.
[49] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Proceedings of the 2017 IEEE symposium on security and privacy (SP)*.
[50] Vladimir N Vapnik. 1999. An overview of statistical learning theory. *IEEE Transactions on Neural Networks* 10, 5 (1999), 988–999.
[51] Viet Vo, Mengyao Ma, Guangdong Bai, Ryan Ko, and Surya Neplal. 2025. Practical Poisoning Attacks with Limited Byzantine Clients in Clustered Federated Learning. In *Proceedings of the 2025 IEEE Symposium on Security and Privacy (SP)*. 1751–1769.
[52] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. 2020. Attack of the tails: Yes, you really can backdoor federated learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 16070–16084.
[53] Zihan Wang, Zhongkui Ma, Xinguo Feng, Ruoxi Sun, Hu Wang, Minhui Xue, and Guangdong Bai. 2024. Corelocker: Neuron-level usage control. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*. 2497–2514.
[54] Zihan Wang, Zhongkui Ma, Xinguo Feng, Chuan Yan, Dongge Liu, Ruoxi Sun, Derui Wang, Minhui Xue, and Guangdong Bai. 2025. Re-Key-Free, Risky-Free: Adaptable Model Usage Control. *arXiv preprint arXiv:2511.18772* (2025).
[55] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. 2019. Dba: Distributed backdoor attacks against federated learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
[56] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2019. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2041–2055.
[57] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR.
[58] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. 2018. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 9194–9203.
[59] Andrew Yuan, Alina Oprea, and Cheng Tan. 2024. Dropout attacks. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1255–1269.
[60] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. 2020. {BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC)*. 493–506.
[61] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2022. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2545–2555.
[62] Lingchen Zhao, Shengshan Hu, Qian Wang, Jianlin Jiang, Chao Shen, Xiangyang Luo, and Pengfei Hu. 2021. Shielding Collaborative Learning: Mitigating Poisoning Attacks Through Client-Side Detection. *IEEE Transactions on Dependable and Secure Computing (TDSC)* 18, 5 (2021), 2029–2041.

A Threat Model Comparison

We compare the threat model of ClieND with existing server-side defense in Table 8.

B Patterns on Neuron Scores

We provide the detailed patterns on neuron scores here.

Untargeted attacks. The results in Figure 4.(a), 4.(b), and 4.(c) illustrate the score shifting patterns in untargeted attacks. Specifically, in the Min-max attack (a) and Fang attack (b), most neurons exhibit a decline in importance scores, which corresponds to *Pattern #1*. In the Min-activation attack (c), some neurons are affected experiencing the importance score decline, which aligns with *Pattern #2*.

Targeted attacks. The results depicted in Figure 4.(d), 4.(e), 4.(f), and 4.(g) demonstrate the score shifting patterns in targeted attacks. In the Neuron-separation (g), some neuron importance scores exhibit fluctuations with a higher value, while others experience decline or remain unchanged, which aligns with *Pattern #2*. In the remaining three targeted attacks (d), (e), and (f), the importance scores of the targeted neurons clearly drop off, aligning with *Pattern #3*.

Therefore, the detection perspectives we establish based on the three patterns are rational and comprehensive, effectively enabling the detection of poisoning attacks.

C Computational Cost

We explain the computational cost of ClieND in this section. First, based on Eq. 4, we derive that the complexity of computing the importance score is $O(N)$, indicating a minimal computational cost. We then analyze the computational complexity for three detection perspectives. For sum-based detection, the complexity is derived as $O(N)$ according to Eq. 6. Similarly, for distribution-based detection, the complexity is also $O(N)$ as established by Eq. 7. Finally, for single neuron-based detection, the computational complexity is $O(N \cdot \log N)$, as shown in Eq. 8, since sorting all N neurons to get top 10% requires $O(N \cdot \log N)$ in complexity. In summary, the detection phase exhibits a minimum computational overhead of $O(N)$ and a maximum of $O(N \cdot \log N)$, highlighting its low computational complexity across all perspectives.

D Proof of Theorems

D.1 Proof of Theorem 1

Proof. **Clean training.** We first show that both discrepancy sequences are smooth in the absence of attacks. Since $\mathcal{L}(\theta; P_i)$ is β-smooth, we have:

$$|\mathcal{L}(\theta'; P_i) - \mathcal{L}(\theta; P_i)| \leq \beta \|\theta' - \theta\|, \quad \forall \theta, \theta'. \tag{17}$$

Applying this to the global and local model updates, we obtain:

$$\begin{aligned} |d_{\text{global}}^{(t+1)} - d_{\text{global}}^{(t)}| &= |\mathcal{L}(\theta^{(t+1)}; P_i) - \mathcal{L}(\theta^{(t)}; P_i)| \\ &\leq \beta \|\theta^{(t+1)} - \theta^{(t)}\| \leq \beta \rho_1, \end{aligned} \tag{18}$$

$$\begin{aligned} |d_{\text{local}}^{(t+1)} - d_{\text{local}}^{(t)}| &= |\mathcal{L}(\theta_i^{(t+1)}; P_i) - \mathcal{L}(\theta_i^{(t)}; P_i)| \\ &\leq \beta \|\theta_i^{(t+1)} - \theta_i^{(t)}\| \leq \beta \rho_2. \end{aligned} \tag{19}$$

Define $\delta_{\text{clean}} := \max(\beta\rho_1, \beta\rho_2)$, and we have:

$$|d_{\text{global}}^{(t+1)} - d_{\text{global}}^{(t)}| \leq \delta_{\text{clean}}, \quad |d_{\text{local}}^{(t+1)} - d_{\text{local}}^{(t)}| \leq \delta_{\text{clean}}. \tag{20}$$

Under attack. Assume a poisoning attack occurs at round t^* such that the global model becomes

$$\tilde{\theta}^{(t^*)} = \theta^{(t^*)} + \Delta_{\text{attack}}, \quad \|\Delta_{\text{attack}}\| = \delta_{\text{adv}}. \tag{21}$$

Using the lower bound from β-smoothness:

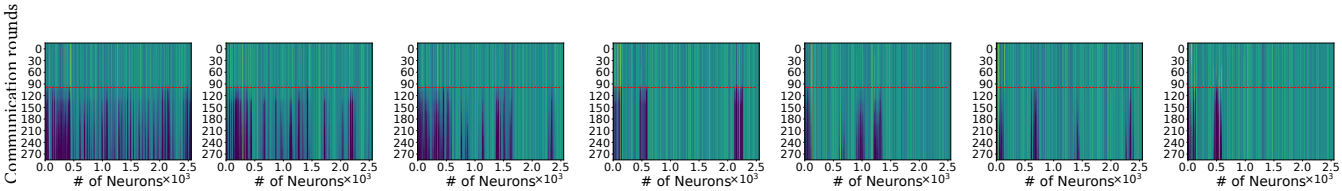
$$\mathcal{L}(\tilde{\theta}^{(t^*)}; P_i) \geq \mathcal{L}(\theta^{(t^*)}; P_i) + \frac{\beta}{2} \|\Delta_{\text{attack}}\|^2, \tag{22}$$

and combining with:

$$|\mathcal{L}(\theta^{(t^*)}; P_i) - \mathcal{L}(\theta^{(t^*-1)}; P_i)| \leq \delta_{\text{clean}}, \tag{23}$$

Table 8: Threat model comparison between CLIEND and representative server-side defenses.

Method	Deployment	Uses Validation Data	Server Resource Required	Client Behavior Used	Non-IID Robustness	Additional Communication with Server
FLTrust [10]	Server	Trusted Dataset	Bootstrap Data	✗	✓	✗
Fang [15]	Server	Accuracy/Loss	Aggregation Only	✗	✓	✗
FLShield [26]	Server	Validator Feedback	Validator Clients + LIPC Matrix	Validator-side feedback	✓	✓
FLAME [37]	Server	✗	Gradient Statistics	✗	✓	✗
FreqFed [18]	Server	✗	Fourier Analysis	✗	✓	✗
CrowdGuard [42]	Server	Local Loss Stats	Client Feedback Signals	feedback-driven	✓	✓
Robust Aggregation Rules[4]	Server	✗	Aggregation-Only	✗	✗	✗
CLIEND (ours)	Client	✗	✗	Neuron scores	✓	✗



(a) Min-max **(b) Fang** **(c) Min-act.** **(d) DBA** **(e) LIE** **(f) Lab-flip.** **(g) Neu-sep.**

Figure 4: The neuron score shifting patterns on untargeted and targeted poisoning attacks. (a), (b), and (c) are the patterns on untargeted attacks, while (d), (e), (f), (g) are on targeted attacks. The red dashed vertical line indicates the attack begins.

We derive:

$$d_{\text{global}}^{(t^*)} \geq d_{\text{global}}^{(t^*-1)} + \frac{\beta}{2}\delta_{\text{adv}}^2 - \delta_{\text{clean}}. \tag{24}$$

To ensure a spike beyond the clean threshold, we require:

$$\frac{\beta}{2}\delta_{\text{adv}}^2 > 2\delta_{\text{clean}} \quad \Rightarrow \quad |d_{\text{global}}^{(t^*)} - d_{\text{global}}^{(t^*-1)}| > \delta_{\text{clean}}. \tag{25}$$

Next, we analyze the local discrepancy. The client uses $\tilde{\theta}^{(t^*)}$ as initialization for local training:

$$\theta_i^{(t^*)} = \tilde{\theta}^{(t^*)} - \eta \nabla \mathcal{L}(\tilde{\theta}^{(t^*)}; \mathcal{D}_i). \tag{26}$$

Then:

$$\begin{aligned} \|\theta_i^{(t^*)} - \theta_i^{(t^*-1)}\| &\geq \|\tilde{\theta}^{(t^*)} - \theta^{(t^*-1)}\| - \eta \|\nabla \mathcal{L}(\tilde{\theta}^{(t^*)}; \mathcal{D}_i)\| \\ &\geq \delta_{\text{adv}} - \epsilon. \end{aligned} \tag{27}$$

for some small ϵ depending on the gradient norm. Applying β-smoothness:

$$|d_{\text{local}}^{(t^*)} - d_{\text{local}}^{(t^*-1)}| \geq \beta(\delta_{\text{adv}} - \epsilon). \tag{28}$$

To ensure a spike:

$$\beta(\delta_{\text{adv}} - \epsilon) > \delta_{\text{clean}} \quad \Rightarrow \quad |d_{\text{local}}^{(t^*)} - d_{\text{local}}^{(t^*-1)}| > \delta_{\text{clean}}. \tag{29}$$

Therefore, if $\delta_{\text{adv}} > \sqrt{2\delta_{\text{clean}}/\beta}$ and ϵ is small (e.g., via learning rate adjustment or gradient clipping), both discrepancies will exceed their clean bounds and serve as a indicator for poisoning attacks. □

D.2 Proof of Theorem 2

PROOF. Let $\mathcal{D}_i = \{(x_j, y_j)\}_{j=1}^{n_i} \sim P_i$ be the local dataset of client i. Define the empirical loss estimator:

$$\hat{\mathcal{L}}(\theta; \mathcal{D}_i) := \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(f_\theta(x_j), y_j). \tag{30}$$

Then we have:

$$\mathbb{E}_{\mathcal{D}_i}[\hat{\mathcal{L}}(\theta; \mathcal{D}_i)] = \mathcal{L}(\theta; P_i), \tag{31}$$

so $\hat{d}_{\text{global}}^{(t)} := \hat{\mathcal{L}}(\theta^{(t)}; \mathcal{D}_i)$ and $\hat{d}_{\text{local}}^{(t)} := \hat{\mathcal{L}}(\theta_i^{(t)}; \mathcal{D}_i)$ are unbiased estimators.

According to Theorem 1, when a poisoning attack occurs at round t^*, both $d_{\text{global}}^{(t)}$ and $d_{\text{local}}^{(t)}$ experience a significant increase exceeding δ_{clean}. Given that client i can empirically track these quantities over time, it is capable of detecting the spike and identifying the anomalous round. This validates that the client's access to $\mathcal{D}_i$ enables it to detect abnormal discrepancy shifts using local information only. □

D.3 Proof of Theorem 3

PROOF. We prove the two probabilistic guarantees separately using the Hoeffding inequality [22].

False alarm avoidance in clean rounds. For any clean round $t \notin \mathcal{T}_{\text{attack}}$, the model is assumed to converge smoothly. Let $\mu_{\text{clean}} = \mathbb{E}[\Delta s_t]$ be the expected neuron score shift during clean training. By construction, the threshold $T_{\Delta s}$ is defined as the maximum clean shift across the pre-rounds, so:

$$T_{\Delta s} \geq \mu_{\text{clean}}. \tag{32}$$

Applying Hoeffding's inequality again under bounded estimation error δ_s, we bound the false positive probability as:

$$\Pr(\Delta s_t > T_{\Delta s}) \leq \exp\left(-\frac{2(T_{\Delta s} - \mu_{\text{clean}})^2}{N\delta_s^2}\right) = \epsilon_1. \tag{33}$$

Therefore, the probability of avoiding false alarms is:

$$\Pr(\mathcal{D}_i^t = 0 \mid t \notin \mathcal{T}_{\text{attack}}) \geq 1 - \epsilon_1, \tag{34}$$

which is identical to the first equation of Eq. 2 in Definition 1.

Detection under attack. Suppose a poisoning attack occurs at round $t^* \in \mathcal{T}_{\text{attack}}$, and it induces a discrepancy spike such that $|d_{t^*} - d_{t^*-1}| \geq \Delta_d$. By the definition of the sensitivity constant κ, the neuron score shift satisfies:

$$\Delta s_{t^*} \geq \kappa \cdot |d_{t^*} - d_{t^*-1}| \geq \kappa\Delta_d. \tag{35}$$

Let $T_{\Delta s}$ be the threshold computed from the maximum shift in clean pre-rounds. To ensure detection, we require $\Delta s_{t^*} > T_{\Delta s}$. The score Δs_{t^*} is computed from estimated scores $\hat{s}_t^i$, each with bounded error $|\hat{s}_t^i - s_t^i| \leq \delta_s$. Applying Hoeffding's inequality to the sum of N independent neuron score deviations, we obtain:

$$\Pr(\Delta s_{t^*} \leq T_{\Delta s}) \leq \exp\left(-\frac{2(\kappa\Delta_d - T_{\Delta s})^2}{N\delta_s^2}\right) = \epsilon_2. \tag{36}$$

Thus, the probability of successful detection is:

$$\Pr(\mathcal{D}_i^{t^*} = 1 \mid t^* \in \mathcal{T}_{\text{attack}}) \geq 1 - \epsilon_2, \tag{37}$$

which is identical to the second equation of Eq. 2 in Definition 1.

Therefore, ClieND achieves provably effective poisoning attack detection by leveraging neuron score shifting, and fulfills the requirements of Definition 1. This completes the theoretical effectiveness justification of our method. □

E Datasets and Model Architecture

We provide the details of the datasets evaluated and the model architecture employed in this work.

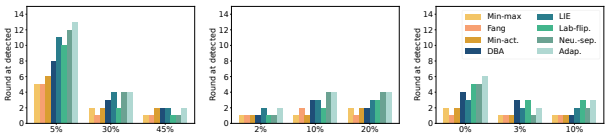
- **CIFAR-10** [29] is a well-known dataset for image classification tasks, consisting of 60,000 color images, each with a resolution of 32×32 pixels [29]. The dataset is categorized into 10 classes with 6,000 samples per class. We employ a convolutional neural network (CNN) architecture for this dataset, consisting of three convolutional layers with ReLU activations and max-pooling, followed by two fully connected layers. The layer sizes are $\{3 \times 32 \times 32, 32, 64, 128, 256, 10\}$.
- **MNIST** [14] dataset serves as a standard benchmark for handwritten digit recognition. It includes 60,000 training samples and 10,000 test samples, each represented as a 28×28 grayscale image with 784 features. We use a fully connected neural network (FCN) consisting of two fully connected layers with ReLU activations and a softmax output layer. The layer sizes are $\{1 \times 28 \times 28, 32, 64, 128, 10\}$.
- **Texas Hospital Dataset** [1] is derived from the Hospital Discharge Data public use files released by the Texas Department of State Health Services. Following the work of Shokri et al. [49], we use a processed version of the dataset containing 67,330 data samples with 6,170 binary features. We utilize a fully connected neural network (FCN) with ReLU activations of layer sizes $\{6169, 1024, 512, 256, 100\}$.
- **ImageNet** [13] dataset serves as a large-scale benchmark for image classification tasks, containing over a million images categorized into 1,000 classes. We use the ResNet-18 model, a convolutional neural network with 18 layers and residual connections, which is widely recognized for its efficiency and high performance in complex image classification tasks.

F FL settings

We conduct training for 2,000 rounds on the MNIST dataset, and 1,000 rounds each on the TEXAS Hospital, CIFAR-10, and ImageNet datasets. The pre-training phase is fixed to span 5% of the total rounds for each dataset at default. The data non-IID rate is controlled by q, which specifies the proportion of samples assigned to a primary label. By default, q=0 corresponds to an IID data distribution [10, 37, 42].

G Poisoning Attacks Settings

- **Untargeted attacks.** We adopt Min-max [47], Fang [15], and Min-activation attacks (Min-act.) [59] for untargeted attacks. We set the Inverse Unit Vector for Min-max [47] as the perturbation vector type, and set the dropout rate for Min-activation to 0.5.



(a) Adv. proportions (b) Num. pre-rounds (c) Attack timing

Figure 5: Ablation on ClieND's early detection.

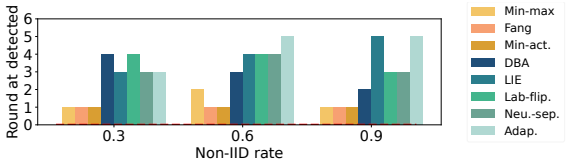


Figure 6: The impact of non-IID rates on the ClieND's early detection.

- **Targeted attacks.** We use Distributed backdoor attack (DBA) [12], LIE [4], Label-flipping (Lab-flip.) [6], and Neuron-separation (Neu-sep.) [59] for targeted attacks. Overall, we set the target label being attacked to class-0. For Label-flipping, we set the sample flipping rate to 0.2. For distributed attack [12], we set the size of the trigger pattern to 6×6 squared pixels in the top-left corner of the image, ensuring the trigger color is consistent across all channels for ImageNet, CIFAR-10 and MNIST, and we set a feature-pattern trigger by setting every 20^{th} features to 0 in TEXAS. We set the dropout rate to 0.5 for Neuron-separation [59]. Additionally, we also adopt constrain-and-scale attack [3] in adaptive attack. The settings for the adaptive adversary are the same with that in Mesas [28].

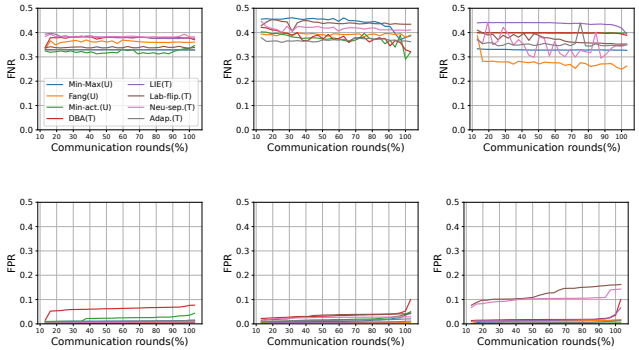
H Evaluation Metrics Details

We provide the details of true positive rate (TPR) and false positive rate (FPR) which evaluate the effectiveness of ClieND detection, and the number of communication rounds.

- **True positive rate (TPR)**. TPR measures ClieND's ability to detect attacks, defined as $\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$, where TP is the number of correctly identified attacks, and FN is the number of missed attacks. A higher TPR indicates better detection effectiveness.
- **False positive rate (FPR)**. FPR quantifies the likelihood of ClieND falsely identifying an attack, defined as $\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$, where FP is the number of false alarms, and TN is the number of correct non-attack identifications. Lower FPR indicates better performance.
- **Communication times**. We evaluate ClieND's detection speed by the number of communication rounds, where each round includes local training, sending updates to the server, and receiving the aggregated global model.

I Impact on extreme data heterogeneity

As shown in Figure 7, under the extreme data heterogeneity setting with single-class clients, ClieND maintains detection performance across all three detection perspectives. Among them, the distribution-based detection exhibits the most stable behavior, achieving consistently low FPR and competitive FNR throughout the training process. The summation-based perspective also performs well, while the single neuron–based detection shows slightly



(a) Sum-based (b) Dis.-based (c) Neu.-based

Figure 7: FNR and FPR curves under extreme Non-IID rate with single-class clients under three detection perspectives.

more fluctuation due to its fine-grained sensitivity to neuron-level score dynamics. Compared with the standard Non-IID setting, the overall detection effectiveness experiences a mild degradation. This is expected, as extreme heterogeneity can amplify the divergence between client updates and increase the natural variability in neuron-importance scores, making it harder to separate benign and malicious changes, especially for fine-grained detectors. Interestingly, we observe a gradual increase in FPR toward the final communication rounds for certain attacks. This trend is also attributable to extreme heterogeneity: as local updates become increasingly dominated by class-specific information, the score distributions shift more sharply across rounds, occasionally causing benign updates to trigger small false alarms.

J Impact of non-IID Rate on Early Detection

Figure 6 shows results under different q values, where the red dashed line marks the attack onset. The value of q has little effect: untargeted attacks are typically detected within the first round, targeted attacks within three to four rounds, and adaptive attacks within five rounds.

K Ablation Study

Proportion of the malicious clients. We examine the effect of varying N on ClieND's early detection capabilities. According to the results shown in Figure 5.(a), we observe that with N = 5%, ClieND can detect untargeted attacks, such as the Min-max attack, as early as the 5^{th} round, and targeted attacks, like the Distributed backdoor attack, by the 8^{th} round. The latest detection occurs by the 13^{th} round for the adaptive attack. As N increases to 30, ClieND detects the presence of poisoning attacks earlier. The earliest attack is detected in the 1^{st} round and the latest by the 4^{th} round. When N = 45%, ClieND's early detection performance is further enhanced, successfully detecting all eight attacks including adaptive adversary within just 2 rounds. Therefore, As N increases, ClieND achieves an earlier detection.

Number of pre-training rounds. Figure 5.(c) illustrates ClieND's early detection capability with varying t_{pre}. We observe that even with varing t_{pre}, ClieND can detect the presence of a poisoning attack as early as the first round after the attack begins, demonstrating its strong early detection ability. Comparing different values of t_{pre}, ClieND performs the best with t_{pre}=2%, detecting the LIE and adaptive attack within 2 rounds and the other 6 attacks within just 1 round. When t_{pre}=10% and 20%, ClieND still shows excellent performance, detecting all attacks by the 4^{th} round at the latest, which is close to the performance with our default setting, t_{pre}=5%. Since the detection might not be very stable under t_{pre}=0.5% or 1%, we do not plot figures under these two hyperparameter selections.

Timing of attacks. Figure 5.(d) shows ClieND's early detection capability when the attack starts at different timings. We observe that when the attack starts early at t_{attack} = 50, ClieND can still detect all untargeted attacks within 3 communication rounds. For targeted attacks, ClieND can detect them in 5 communication rounds. When the attack starts in round 150 and 200, ClieND detects all untargeted attacks in the first round and detects targeted attacks including adaptive adversary within 3 rounds. The results indicate that the attack timing does not significantly affect the ClieND's early detection capability, but it can detect the presence of attacks slightly earlier when the attack starts in a later round.

Table 9: Quantitative overhead analysis of ClieND.

Method	Scenario	Time (s/round)	RAM (MB)	Comm. (Bytes)
FedAvg	–	74.2	808.2	5.93×10^5
ClieND	Sum.	79.3	1330.2	5.93×10^5
	Dis.	79.5	1333.9	5.93×10^5
	Neu.	86.2	1421.4	6.52×10^5

L Quantitative Overhead Analysis

As shown in Table 9, across the three detection perspectives, the per-round time overhead remains close to the FedAvg baseline. The summary-based and distribution-based detectors add only lightweight computations, resulting in time costs of 79.3–79.5 s, only slightly above the 74.2 s of FedAvg. The single neuron-based detection requires tracking the score dynamics of the top k% neurons, leading to a moderately higher time cost of 86.2 s. On the other hand, all three perspectives show an increase in RAM usage, rising from 908 MB in FedAvg to 1.23-1.42 GB. This is expected, as ClieND stores the previous-round neuron-importance scores and additional cached statistics needed for temporal comparison, with the Neu. variant consuming the most memory due to its fine-grained neuron-level operation. Communication overhead remains essentially unchanged for Sum. and Dis., and increases only slightly for Neu., because it reports a small set of neuron indices and scores.

M Future Direction

Our mitigation may weaken when a large proportion of clients (e.g., approaching to 50%) are compromised, as voting alone can be manipulated by colluding adversaries. A promising direction is to combine voting with robust aggregation rules such as Krum [7] and Median [57], using voting as a verification layer rather than the sole determinant. This design strengthens resilience to large-scale collusion [32, 54] and suggests avenues for more comprehensive defenses.