AGRAMPLIFIER: Defending Federated Learning Against Poisoning Attacks Through Local Update Amplification

Zirui Gong*, Liyue Shen*, Yanjun Zhang, Leo Yu Zhang, Jingwei Wang, Guangdong Bai, and Yong Xiang

***Abstract*—The collaborative nature of federated learning (FL) poses a major threat in the form of manipulation of local training data and local updates, known as the Byzantine poisoning attack. To address this issue, many Byzantine-robust aggregation rules (`AGRs`) have been proposed to filter out or moderate suspicious local updates uploaded by Byzantine participants.**

This paper introduces a novel approach called AGRAMPLIFIER, aiming to simultaneously improve robustness, fidelity, and efficiency of the existing `AGRs`. The core idea of AGRAMPLIFIER is to amplify the "morality" of local updates by identifying the most repressive features of each gradient update, which provides a clearer distinction between malicious and benign updates, consequently improving the detection effect. To achieve this objective, two approaches, namely `AGRMP` and `AGRXAI`, are proposed. `AGRMP` organizes local updates into patches and extracts the largest value from each patch, while `AGRXAI` leverages explainable AI methods to extract the gradient of the most activated features. By equipping AGRAMPLIFIER with the existing Byzantine-robust mechanisms, we successfully enhance the model robustness, maintaining its fidelity and improving overall efficiency.

AGRAMPLIFIER is universally compatible with the existing Byzantine-robust mechanisms. The paper demonstrates its effectiveness by integrating it with all mainstream `AGR` mechanisms. Extensive evaluations conducted on seven datasets from diverse domains against seven representative poisoning attacks consistently show enhancements in robustness, fidelity, and efficiency, with average gains of 40.08%, 39.18%, and 10.68%, respectively.

***Index Terms*—Federated Learning, Byzantine-robust Aggregation, Poisoning Attack, Explainable AI.**

I. INTRODUCTION

FEDERATED learning (FL) [2], [3] is a subset of the deep learning system, where several clients collaboratively train a central model. It has become more prevalent across a range of privacy-sensitive tasks and has been implemented by well-known machine learning techniques. In FL, each client maintains a private training dataset and trains a local model based on their datasets. After each local training, clients only update the gradient to the central server (i.e., the aggregator) for the global model aggregation. On the server side, the global model is updated by taking one step downward in gradient descent; then, the global model is further broadcasted to clients for the subsequent training cycle. FL enables the training of high-quality Machine Learning (ML) models with massive data and eliminates the exposition of private raw data to the server.

However, the distributed nature of FL makes it susceptible to client-side poisoning attacks [4]–[13]. One kind called untargeted attacks [6], [7], [10]–[12], [14], which aims to corrupt the global model to low test accuracy, therefore causes the model unusable and eventually leads to denial-of-service attacks (e.g., an attacker may perform such attacks on its competitor's FL system). On the other hand, targeted attacks [4], [5], [15], also known as the backdoor attack, where the attacker corrupts the global model to predict an attacker-chosen label for any testing input embedded with a trigger while maintaining high test accuracy on other input.

Several Byzantine-robust defense mechanisms have been proposed to defend against poisoning attacks, [1], [6], [7], [10], [16]–[21] Essentially, the server employs the robust aggregation algorithm (`AGR`) to filter out or moderate suspicious local updates to mitigate the malicious impact of adversary contributions. There are three widely used `AGR` mechanisms, i.e., distance-based [16], [17], prediction-based [7], [10], [18], and trust bootstrapping-based techniques [6], [22], [23]. Specifically, the distance-based mechanism compares the collected gradients based on distance measurements, i.e., Euclidean distance and cosine similarity, and removes anomalous updates before aggregating the remaining ones. The prediction-based mechanism checks the model's prediction performance and removes the updates causing performance degradation, while the trust bootstrapping-based mechanism computes trust scores for each participant and uses them as weights when averaging updates.

Triad of Byzantine-robust FL: We articulate the triad of desirable properties of `AGRs` as follows. (1) ***Robustness***. The `AGRs` shall minimize the decrease of the global model's test

This article extends the preliminary results presented in [1]. In our prior work, we focus on a basic approach `AGRMP`, which draws inspiration from the max pooling operation to amplify the salient features of local updates. In this work, we introduce a more advanced approach named `AGRXAI` to enhance the distinctions between benign gradients and malicious gradients. The paper also substantially extends experimental evaluation on seven datasets across diverse domains, demonstrating the consistent improvement of robustness and fidelity.

Correspondence to Dr. Y. Zhang and Dr. L. Zhang.

* These authors contributed equally to this work.

Zirui Gong and Leo Yu Zhang are with the School of Information and Communication Technology, Griffith University, Southport, Queensland, Australia. Email: z.gong@griffith.edu.au, leo.zhang@griffith.edu.au.

Liyue Shen, Jingwei Wang, and Guangdong Bai are with the School of Information Technology and Electrical Engineering, The University of Queensland, St Lucia, Queensland, Australia. Email: liyue.shen@uq.net.au, jingwei.wang@uq.net.au, g.bai@uq.edu.au.

Yanjun Zhang is with the School of Computer Science, University of Technology Sydney, Sydney, New South Wales, Australia. Email: Yanjun.Zhang@uts.edu.au.

Yong Xiang is with the School of Information Technology, Deakin University, Melbourne, Victoria, Australia. E-mail: yong.xiang@deakin.edu.au.

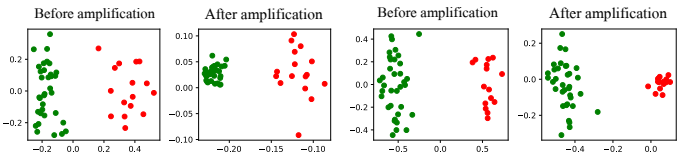
accuracy caused by malicious updates. (2) ***Fidelity***. The AGRs shall not harm the performance of the global model when there are no malicious updates and shall achieve a test accuracy close to that of the plain FedAvg [6]. Additionally, the AGRs should be designed to handle the inherent diversity of data from different owners. While this data variety can enhance the generalization of the global model, it can also result in deviations in the local updates, which existing defenses may deny. (3) ***Efficiency***. The method shall retain the computation efficiency and scale in large-scale FL training, particularly in processing the high-dimensional local updates from a large number of participants. The primary question this paper addresses is how to achieve and enhance the triad of robustness, fidelity, and efficiency for Byzantine-robust aggregators in FL.

Our work: We introduce a novel approach called AGRAMPLIFIER, which is built upon the base AGRs, aiming to achieve robustness, fidelity, and efficiency in FL. The central idea behind AGRAMPLIFIER is to amplify the influence of both malicious and benign local updates by focusing on the gradients of the most prominent features. Through this, we can make better decisions about removing malicious updates before global model aggregation. To achieve this goal, we develop two strategies, AGRMP and AGRXAI.

The AGRMP draws inspiration from the widely used max pooling operation [24] in Convolutional Neural Networks (CNN), which can reduce the dimensionality of features and summarize their most activated presence. By leveraging this technique, AGRMP seeks to improve the aggregation process by amplifying the salient features of local updates, thereby enhancing their impact on the final aggregated output. Specifically, AGRMP initially rearranges each updated gradient into individual patches. Subsequently, it identifies the most prominent feature in each patch by extracting the largest value and returns it as the amplified outcome.

The AGRXAI is motivated by recent progress in the integration of XAI within the cybersecurity domain [25], [26] and applies the concept of Explainable AI (XAI) [27] to amplify the difference between benign gradients and malicious gradients. Specifically, we employ Grad-CAM [28] to determine the feature maps that learned the key features. After getting the importance weights, we rank them in descending order and select the top p feature maps with the highest weights (p indicates the percentage we extracted). This selection allows us to focus on the most significant feature maps. By indexing the gradients corresponding to these selected feature maps, we can extract the top p most significant gradients as the amplified outcome.

The seemingly straightforward strategies are effective in achieving the triad of Byzantine-robust FL. Firstly, the extraction of the most activated gradients renders the local updates more distinguishable after the amplification. As illustrated in Fig. 1, the benign gradient is reduced from [-0.3, 0.4] to [0, 0.06] (the y-axis of green dots in Fig. 1a), while the malicious gradient is reduced from [-0.4, 0.3] to [-0.1, 0.1] (the y-axis of red dots in Fig. 1b). Therefore, the AGRs can make robust decisions of detecting maliciousness local updates (detailed in Section V-A). Secondly, AGRAMPLIFIER enhances fidelity by providing invariance to distortion arising from local



(a) Untargeted attack (b) Targeted attack

Fig. 1: Utilizing the PCA method to project gradients onto a two-dimensional surface. Specifically, we plot a total of 50 local updates at the 70th epoch of the training process using the LOCATION30 dataset [29]. Within the plotted updates, red dots represent malicious updates, while green dots represent benign ones. The attack employs the untargeted label flip [5] and targeted scale attack [6].

translations. Analogous to a max pooling layer in typical CNNs, AGRAMPLIFIER can suppress small changes. Hence, when no attack is present, it functions as a noise canceler (detailed in Section V-B). Thirdly, the significant dimension reduction in the feature space provides substantial benefits to the efficiency of AGRs. As many existing AGRs [6], [7] exhibit superlinear time complexity and AGRAMPLIFIER amplifies the advantages derived from the input size reduction (detailed in Section V-C).

It is important to note that the proposed amplification process is universally compatible with any existing AGRs. Specifically, we applied AGRAMPLIFIER to three widely used mechanisms, including distance-based, prediction-based, and trust bootstrapping-based aggregators. Specifically, we introduce ten variations of AGRAMPLIFIER, each of these is suitable for different use cases, which are explored and discussed in Section VI-A. Overall, our results consistently demonstrate improvement for all three mechanisms. For instance, when the distance-based mechanisms are equipped with AGRAMPLIFIER, they outperform their base versions by 66.26% in robustness, 29.6% in fidelity, and 12.9% in efficiency. Similarly, in the case of prediction-based mechanisms, AGRAMPLIFIER yields 35.59% improvement in robustness, 47.3% improvement in fidelity, and 7.7% improvement in efficiency compared to its counterpart. Furthermore, AGRAMPLIFIER also enhances the performance of the trust bootstrapping-based mechanisms in the state-of-the-art aggregator by 18.37% in robustness, 19.4% in fidelity, and 12.3% in efficiency.

The paper presents several significant contributions to the field of FL with a focus on robustness, fidelity, and efficiency. We summarize our contributions as follows.

- **A novel Byzantine-robust aggregation method.** We propose AGRAMPLIFIER, containing two approaches, i.e., AGRMP and AGRXAI. It achieves the triad of robustness, fidelity, and efficiency through the amplification of local updates.
- **A novel Aggregation Rule (AGR).** We design a distance-based AGR, which incorporates a top-up component of density measurement to supplement existing Euclidean distance and cosine similarity mechanisms.
- **The seamless integration of AGRAMPLIFIER with existing AGRs.** We propose ten versions of AGRAMPLIFIER

that integrated three kinds of AGRs.
- **A systematic evaluation.** We conduct a systematic evaluation on seven benchmark datasets against five poisoning attacks, demonstrating the notable enhancement of AGRAMPLIFIER over original Byzantine-robust methods across all experimented datasets.

II. BACKGROUND AND RELATED WORK

A. Poisoning Attacks in FL

FL enables multiple clients to train a model collaboratively. Specifically, let $r \in \{1, 2, \cdots, R\}$ be the current iteration of the training process, at $r = 0$, the server initializes the global model and selects $\mathcal{C}_r$ clients to broadcast the global model $\mathcal{W}_r$. Then each client c_i calculates the local updates g_i^r based on its local datasets and uploads g_i^r to the server. The server then applies an aggregation rule on the local updates to get the global model $\mathcal{W}_{r+1}$ for the next round of training.

However, this process makes the FL system susceptible to client-side poisoning attack, which can be divided into untargeted attacks [6], [7], [10]–[12], [14] and targeted attacks [4], [5]. Untargeted attacks aim to corrupt the global model to make incorrect predictions for any testing examples, therefore leading to undermining the test accuracy and integrity of FL models. This can be achieved by either poisoning the training dataset (data poisoning attacks) [30] or manipulating the local updates directly (model poisoning attack) [31]. The targeted attack is designed to influence the global model's predictions towards a particular class while maintaining overall prediction accuracy [32]. For example, an attacker might inject specific patterns (trigger or backdoor) into the training data and change the label of those images with the trigger to a target class. In the inference stage, all images with the trigger will be mislabeled as the target class, while others will remain unaffected [5].

B. Byzantine-robust Aggregation Rules

Several Byzantine-robust AGRs have been proposed to defend against the poisoning attacks [1], [6], [7], [17], [18], [23], [33], which can be generally divided into three categories.

- **Distance-based mechanism.** Most distance-based AGRs rely on measuring the pairwise distance between local updates to identify and discard malicious updates. Krum and Multi-Krum [33] identify one or m local updates that are similar to others as the global model. Trimmed Mean and Median [16] employ approach involves coordinate-wise aggregation, where the AGR separately identifies and removes the outliers in each dimension and aggregates the remaining benign update. Another approach, *Adaptive federated average (AFA)* [34], compares the cosine similarity between the weighted average of collected gradients and each individual gradient. Gradients with cosine similarities outside a specified range are discarded, helping eliminate malicious gradients.
- **Prediction-based mechanism.** The prediction-based AGR test the gathered updates on a validation set to assess its prediction performance. The updates that lead to a decline in performance are subsequently removed before aggregation. One example is LoMar [10], which evaluates the quality of clients' model updates by analyzing the relative distribution of neighboring updates and determining an optimal threshold for differentiating between malicious and clean updates. Another representative mechanism is Fang [7], which calculates losses and errors on the validation set of the updated model to determine whether the updates originate from malicious or benign clients. Additionally, Zhang [18] identifies malicious clients by checking the consistency of their model updates. To elaborate, the server predicts a client's model update in each iteration based on historical model updates and flags a client as malicious if the received model update is inconsistent with the predicted update across multiple iterations.
- **Trust bootstrapping-based mechanism.** *FLTrust* [6] is a typical trust-bootstrapping based AGR. Each client is given a trust score based on the distance between the local updates and the reference gradients produced from a clean, trustworthy dataset. The trust score then serves as the weight when averaging updates.

However, the performance of the current Byzantine-robust techniques varies in terms of robustness, fidelity, and efficiency. For instance, distance-based techniques, such as Krum [33] and Bulyan [17], might not maintain robustness and fidelity against a large number of malicious participants. Fang [7] may not be applicable to large-scale FL because it requires the extra computation cost for validating the loss of each individual update using the global model parameters. The integrity of trust bootstrapping-based approaches, such as FLTrust [6], may be suppressed if the clean dataset deviates from the initial distribution.

C. Explainable Artificial Intelligence

Explainable AI (XAI) [27], [28], [35], [36] is a collection of techniques that aim to increase the reliability and transparency of AI systems. Among them, Gradient-Weighted Class Activation Map (Grad-CAM) [28] is a widely used XAI technique to identify the importance of the image region by projecting back the weights of the output layer onto the convolutional feature maps. Specifically, it first computes the result of the gradient of the class score (y^c) w.r.t. the feature map in the last convolution layer (A^k), as $\frac{\partial y^c}{\partial A^k}$. Thus, for a given class c, it calculates the weights α_k^c corresponding to class c for feature map k as:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{i,j}^k}, \tag{1}$$

where Z denotes the number of pixels in the feature map, and i and j respectively index the width and height of the k-th feature map A^k.

III. AGRAMPLIFIER

A. Problem Statement

1) Attack Model: We evaluate the proposed AGRAMPLIFIER method against untargeted and targeted attacks. Specifically, we consider an attack model which is in favor of the

poisoning adversary among prior research [4]–[6], [31] that allows the adversary to have the full knowledge of gradients uploaded by all participants, including those from benign ones. This enables the optimized and adaptive poisoning attacks and demonstrates compelling attack performance in recent studies [6], [31]. The attacker can also decide whether to inject malicious updates for the current round. Here, we assume the attacker does not compromise the central aggregator.

2) Defense Model: We consider the defense strategy to be implemented on the server side. The server can gather a small, clean dataset for validation. Since we only need a small dataset, e.g., 100 training examples, manual gathering and labeling on the server side is feasible.

The defense aims to achieve Byzantine robustness against malicious clients and maintain fidelity and efficiency. Specifically, the strategy should not compromise the global model's classification accuracy, and it should be as accurate in non-adversarial conditions as the global model learned by FedAvg. Also, the method should retain computation efficiency and scalability in large-scale FL training.

B. AGRAMPLIFIER *Overview*

In AGRAMPLIFIER, the server first collects local updates from participants and then extracts the most activated features (i.e., using AGRMP or AGRXAI) to amplify the differences between benign gradients and malicious gradients. Then, the amplified gradients are concatenated for the following check. The detailed steps of AGRMP and AGRXAI are as follows.

1) AGRMP: Fig. 2 and Algorithm 1 demonstrate the AGRMP method. Initially, the server acquires local updates g_i from N participants. Then, it divides each g_i into patches using a kernel size of $k_p \times k_p$ to do the max filter. This involves computing the maximum value of each patch, which indicates the gradient of the most activated feature (function MAXFILTER in Algorithm 1). After the max filter process, the amplified gradients are concatenated to facilitate cross-checking in the ensuing steps.

2) AGRXAI: Fig. 3 and Algorithm 2 demonstrate the AGRXAI method to extract the most activated gradients. Given that the last convolutional layer of the CNN typically has a satisfactory balance between high-level semantics and detailed spatial information, the gradient information flowing into this layer is utilized to understand the importance of each neuron for a specific decision of interest.

Initially, a small validation set D is collected by the server (line 4 in Algorithm 2). After the server reviews gradients g_i from N clients, it will make a copy of the original gradients as g_i^{og}. The server then deploys D to the received local model, where the gradient of the score w.r.t. the feature maps A^k is computed (line 6). These gradients flowing back are global-average-pooled over the width and height dimensions (indexed by i and j respectively) to obtain the importance weights $\alpha_k = \frac{1}{Z}\sum_i \sum_j \frac{\partial y}{\partial A^k_{i,j}}$ (line 7). After getting the importance weights, we rank them in descending order and select the top p feature maps with the highest weights. This selection allows us to focus on the most significant feature maps (need to note that AGRXAI only focuses on the gradients of the last convolutional layer). By indexing the gradients corresponding to these selected feature maps, represented as G_{amp}, we can utilize them for conducting the following defense check.

Algorithm 1 AGRMP's amplification process.

Input: g_i - received gradients from client i; N - number of participants; k_p - the kernel size of each patch; H_{in} - the height of g; W_{in} - the width of g; $Restore-size$ - whether restore the amplified gradients in the original size, default to being false.

Output: G_{amp} the amplified gradients collection.

1: **function** AGRMP($g_1, g_2, g_3, ...$)
2: **for** $i = 1, 2, 3, ..., N$ **do**
3: $g^i_{amp} \leftarrow$ MAXFILTER(g_i)
4: **if** $Restore-size$ **then**
5: Fill the dropped gradients in g^i_{amp} with 0s
6: **end if**
7: **end for**
8: $G_{amp} \leftarrow \{g^i_{amp} \mid i = 1, 2, \ldots, N\}$
9: **return** G_{amp}
10: **end function**
11: **function** MAXFILTER(g)
12: $H_{out} \leftarrow H_{in}/k_p$
13: $W_{out} \leftarrow W_{in}/k_p$
14: **for** $h_o = 1, 2, 3, ..., H_{out}$ **do**
15: **for** $w_o = 1, 2, 3, ..., W_{out}$ **do**
16: $g_{amp} \leftarrow \max(\{g_{hi,wi} | hi \in [k_p * (ho-1)+1, k_p * ho], wi \in [k_p * (wo-1)+1, k_p * wo]\})$
17: **end for**
18: **end for**
19: **return** g_amp
20: **end function**

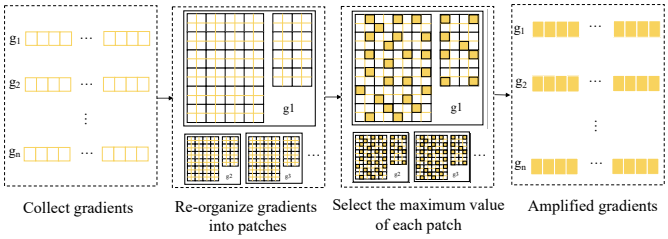


Fig. 2: The amplification process on the collected gradients using AGRMP.

C. *Equipping* AGRAMPLIFIER *with Byzantine-robust Aggregation Rules*

This section outlines the technical details of adapting AGRAMPLIFIER to support Byzantine-robust aggregators. An overview of AGRAMPLIFIER is provided in Fig. 4. Specifically, the collected gradients G are first amplified to G_{amp} and used to calculate pairwise distances, loss value (LRR), and test accuracy (ERR), as well as trust scores for distance-based, prediction-based, and trust bootstrapping mechanisms, respectively. In the case of prediction-based mechanisms, the amplified gradients G_{amp} are restored to their original size by filling the dropped gradients in g^i_{amp} with 0. The distance-

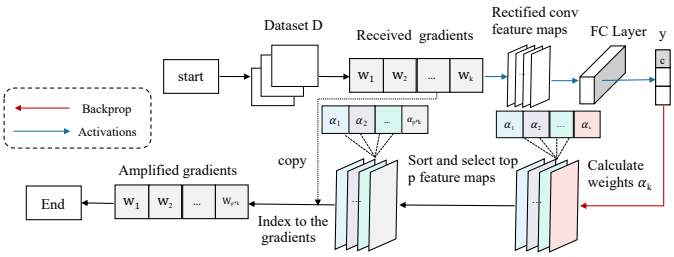


Fig. 3: The amplification process on the collected gradients using AGRXAI.

Algorithm 2 AGRXAI's amplification process.

Input: g_i - received gradients from client i; g_i^{og} - a copy of original gradients N - number of participants; D - server-side clean dataset; α_k - contribution weight of feature map k; A^k - feature maps of a convolutional layer.

Output: G_{amp} the amplified gradients collection.

function AGRXAI($g_1, g_2, g_3, ...$)
  for $i = 0, 1, 2, ..., N$ **do**
    $g_i^{og} \leftarrow$ copy g_i
    Pass D through the received model
    $y \leftarrow$ final output logits
    Backpropagate the weight $\frac{\partial y}{\partial A^k}$
    $\alpha_k \leftarrow \frac{1}{Z} \sum_i \sum_j \frac{\partial y}{\partial A^k_{i,j}}$
    Sort and select the top p feature maps
    $g^i_{amp} \leftarrow$ index to the corresponding gradients stored in g_i^{og}
    if $Restore - size$ **then**
      Fill the dropped gradients in g^i_{amp} with 0s
    end if
    $G_{amp} \leftarrow \left\{ g^i_{amp} \mid i = 1, 2, \ldots, N \right\}$
  end for
  return G_{amp}
end function

based and prediction-based aggregators output a white list of benign participants denoted as G_{detox}, and the original gradients belonging to G_{detox} are fed to FedAvg for global model computation. For trust bootstrapping-based mechanisms, it will first compute a trust score for each g_i, and the trust score then serves as the weight when averaging updates. Subsequent sections provide detailed information on AGRAMPLIFIER's implementation for each of the three categories of aggregation mechanisms.

1) **AGRAMPLIFIER** ***for Distance-based Aggregation Rules****:* Before aggregating the gradients, distance-based aggregation rules [16], [17] examine the collected gradients and use distance measurements (e.g., Euclidean distance and cosine similarity) to measure the difference between malicious and benign ones. However, previous distance-based techniques primarily focus on anomaly detection, which becomes less effective as the number of malicious nodes rises. To address this issue, we design a *density measurement component* to supplement the traditional distance-based mechanisms by using the knowledge that malicious gradients tend to be sparsely distributed, while benign ones are denser [10]. Our proposed method involves evaluating the density of the neighborhood surrounding each gradient. The neighborhood is defined as a set of K neighbors, where K must exceed half of the total number of participants N. The approach considers gradients residing in denser neighborhoods benign, whereas those in sparser areas are malicious. To implement this method, we first select the K nearest neighbors of each participant's update based on distance measurement and record the scores of each measurement. Then, we sum up the score of the K neighbors as the density score. The top N_t participants with the highest density scores are then included in the whitelist. This density measurement is integrated into the distance-based mechanisms as a top-up component, illustrated in the right box in the upper part of Fig. 4, to improve the efficacy of the original approach.

Fig. 5 demonstrates the effectiveness of the AGRAMPLIFIER mechanism, as observed in both targeted and untargeted attacks. The number of malicious participants in the neighborhood of a benign participant is notably reduced after amplification, as evidenced by a decrease from 8 to 0 for the untargeted attack and from 1 to 0 for the targeted attack. Additionally, the summed cosine similarity in the neighborhood undergoes a significant increase after amplification, with values escalating from 8.97 to 27.73 for the untargeted attack and from 13.86 to 21.73 for the targeted attack. The pseudocode of AGRAMPLIFIER for Distance-based Aggregation Rules is displayed in Supplemental Document I-A.

2) **AGRAMPLIFIER** ***for Prediction-based Aggregation Rules****:* The Prediction-based mechanism [7] applies the collected gradients to a validation dataset to evaluate the model's prediction performance in terms of LRR and ERR. It removes those clients, degrading the model performance before aggregating the gradients. The middle section of Fig. 4 indicates how AGRAMPLIFIER works for a prediction-based mechanism. The amplification of the gathered parameters is first carried out by AGRAMPLIFIER. Then, we calculate the LRR and ERR, and those that perform better in prediction are added to the white list, which will be sent to global aggregation. The pseudocode of AGRAMPLIFIER for Prediction-based Aggregation Rules is displayed in Supplemental Document I-B.

3) **AGRAMPLIFIER** ***for Trust Bootstrapping-based Aggregation Rules****:* This mechanism [6] utilizes a trustworthy clean dataset to generate clean gradients. By contrasting each participant's update with the clean gradients, a trust score is assigned to each participant. Then, the server utilizes the trust score as the weight to aggregate updates for the global model. The bottom section of Fig. 4 indicates how AGRAMPLIFIER works for a Trust Bootstrapping-based mechanism. In addition to amplifying the collected gradients g_i, AGRAMPLIFIER also applies amplification to the gradient produced from the trust validation set. The trust score for each participant is calculated by utilizing the ReLU-clipped cosine similarity of the amplified gradients. The local model update data magnitudes are then normalized and combined to form the global model, weighted by their corresponding trust scores. The pseudocode of AGRAMPLIFIER for Trust Bootstrapping-based Aggregation Rules is displayed in Supplemental Document I-C.

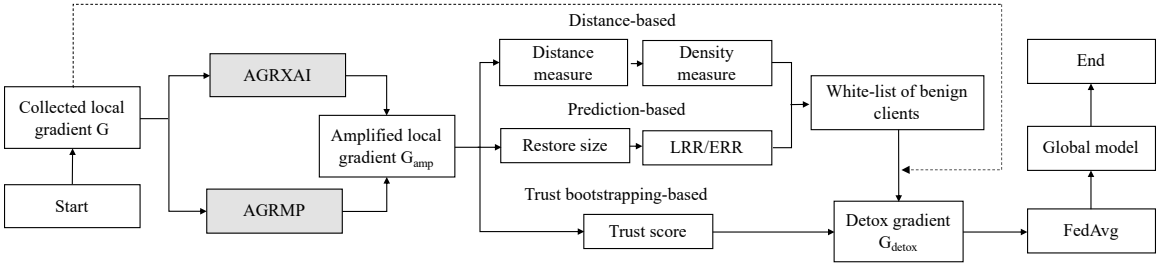


Fig. 4: Workflow of AGRAMPLIFIER for Byzantine-robust mechanisms.

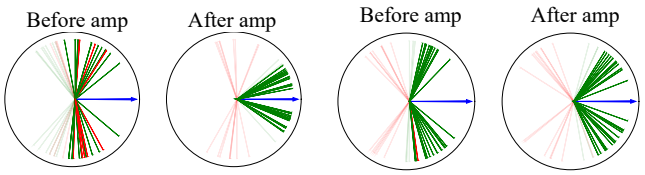


Fig. 5: A benign participant is selected (the blue arrow), and the K-nearest neighbors are determined using cosine similarity. The neighboring participants are further categorized as malicious (the red lines) or benign (the green lines).

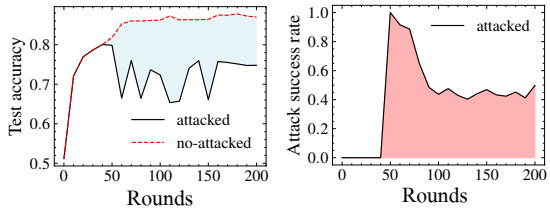


Fig. 6: Illustration of the evaluation metrics.

IV. EXPERIMENTAL SETUP

In this section, we conduct experiments to determine the triad of Byzantine-robust for the proposed method.

A. Datasets

We use seven real-world datasets: CIFAR-10, MNIST, Fashion-MNIST, CATvsDOG Kaggle, PURCHASE100, LOCATION30, and TEXAS100 to evaluate our method. The details are shown in TABLE I.

TABLE I: Description of dataset.

Dataset	Samples	Class	Distribution	Data Type
CIFAR-10	60,000	10	IID	Images
MNIST	70,000	10	IID	Images
Fashion-MNIST	60,000	10	IID	Images
CATvsDOG Kaggle	25,000	2	IID	Images
PURCHASE100	197,324	100	Non-IID	Non-Images
LOCATION30	5,010	30	Non-IID	Non-Images
TEXAS100	67,330	100	Non-IID	Non-Images

B. Poisoning Attacks

We evaluate AGRAMPLIFIER under both untargeted and targeted attacks. For untargeted attacks, we consider Grad-Ascent attack (`G-asc`) [14], Optimized and adaptive attack (`S&H`) [37], Label-flip attack (`L-flip`) [6] combined with Grad-Ascent attack (`L+G`). For targeted attacks, we consider Scale attack (`Scale`) [6] and Distributed Backdoor Attack (`DBA`) [38] to manipulate the model update.

C. Evaluated Defenses

We introduce ten variations of AGRAMPLIFIER, which integrate three categories: distance-based mechanisms, prediction-based mechanisms, and trust bootstrapping-based mechanisms.

TABLE II: Ten versions of our proposed methods and their base methods.

	Base	AGRMP	AGRXAI
Distance-based	`CosDen` [17]	`CosDen_MP`	`CosDen_XAI`
	`EuDen` [33]	`EuDen_MP`	`EuDen_XAI`
	`MergeDen`	`MergeDen_MP`	`MergeDen_XAI`
Prediction-based	`Fang` [7]	`Fang_MP`	`Fang_XAI`
Trust-based	`FLTrust` [6]	`FLTrust_MP`	`FLTrust_XAI`

For each category, we compare our `AGRMP` and `AGRXAI` equipped versions of its base `AGRs`.

TABLE II indicates the name of each base method and its equipped version. Additionally, for distance methods, we improve them by adding a density-measure component for all methods. For the prediction-based mechanism, we merge LRR and ERR for optimal performance in Fang's defense [7]. For bootstrapping-based, we set the validation set size as 2000 for CIFAR-10 [39], MNIST [40], Fashion-MNIST [41], CATvsDOG Kaggle [42], PURCHASE100 [40] and TEXAS100 [40], and 300 for the LOCATION30 dataset [29].

D. Evaluation Metrics

For untargeted attacks, we use the *test accuracy rate (TA)* of the global model to evaluate the performance. We denote a_r as the TA at iteration r without attack, while $\hat{a}_r$ denote the test accuracy at iteration r under attack. The running time has a great impact on the TA [1]. Therefore, we utilize the averaged TA over a monitoring period as the evaluation metric, which can also alleviate the impact of irrelevant factors such as overfitting on the performance evaluation. We calculate this averaged metric $\mathcal{L}$ as:

$$\mathcal{L} = \frac{1}{R_1 - R_0} \sum_{r=R_0}^{R_1} a_r - \hat{a}_r.$$

We illustrate $\mathcal{L}$ as the blue shaded area (then divided by the round number) in Fig. 6a, which is generated on the CATvsDOG Kaggle dataset with the `G-asc` attack.

In targeted attacks, the goal is to maintain the test accuracy rate while inducing the global model to predict towards the target class. Therefore, we employ *Attack Success Rate (ASR)* as the evaluation metric. The ASR is determined by analyzing the test data containing a trigger that is misclassified as the target class by the global model. By measuring the ASR, we are able to evaluate the extent to which the poisoning attack influences the global model's behavior and its ability to induce incorrect predictions toward the target class. We denote the ASR at iteration r as Eq. (2):

$$s_r = \|\{\hat{d} \mid_{\hat{d}\in\hat{D},\omega_r(\hat{d})=c}\}\|/\|\hat{D}\|, \tag{2}$$

where $\hat{D}$ denote the malicious set, ω_r denote the global model at iteration r and c is the target label. Then we calculate the average ASR as Eq. (3) to evaluate the performance:

$$S = \frac{1}{R_1 - R_0} \sum_{r=R_0}^{R_1} (s_r)\,. \tag{3}$$

We illustrate S as the red shaded area (then divided by the round number) in Fig. 6b, generated on the CATvsDOG Kaggle dataset under `Scale` attack.

E. FL Settings

We train a five-layer CNN with two convolution layers, one max-pooling layer, and two dense layers. By default, each round involves the participation of 50 clients. We employ the global communication rounds are $Rg = 200$. Each local client conducts 2 epochs with a batch size of 64. The attacker initiates the attack at round $Rg = 50$, and we record test accuracy systematically every 10 round.

It's worth noting that in the default setting, we choose a higher malicious rate (i.e., 30%) compared to previous works [6], [7] (around 20%) to demonstrate the advantage of our method in defending against a stronger attack which compromises high proportion of malicious clients. In Supplemental Document II-B, we also discuss the influence of the proportion of malicious clients, which ranges from 0.2 to 0.4.

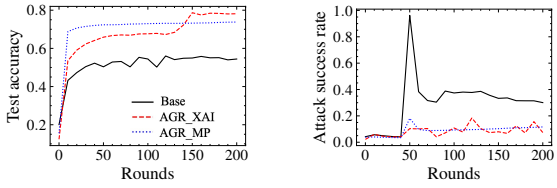
V. Experiment Results

In this section, we discuss the performance of ten variations of AGRAMPLIFIER against the five state-of-the-art poisoning attacks in terms of *robustness*, *fidelity* and *efficiency*. The results show that AGRAMPLIFIER outperforms its base mechanisms on all three properties.

In particular, `EuDen_XAI` shows the highest robustness against targeted attacks, while `Fang_XAI` performs the best against untargeted attacks. Furthermore, the fidelity of `CosDen_MP` is superior, followed closely by `CosDen_XAI`. Additionally, `FLTrust_MP` exhibits superior efficiency with small to medium network sizes, whereas `CosDen_MP` delivers the best efficiency with large networks. Now, we demonstrate a comprehensive performance analysis of AGRAMPLIFIER.

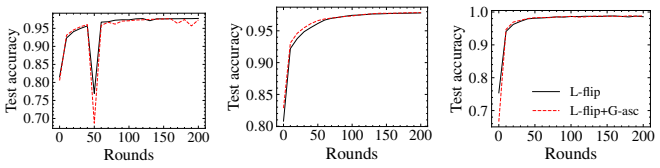
A. Robustness Boosting

AGRAMPLIFIER shows its capability to boost the robustness of untargeted and targeted attacks. We examine it as follows.



(a) Untargeted attack (b) Targeted attack

Fig. 7: TA for untargeted attacks and ASR for targeted attacks.



(a) `CosDen` (b) `CosDen_MP` (c) `CosDen_XAI`

Fig. 8: (a) Negative pulse at round 50. (b, c) AGRAMPLIFIER mitigates the "negative pulse".

1) Untargeted attack: The AGRAMPLIFIER demonstrates its ability to improve the robustness of the FL system in general, as evidenced by the results presented in Fig. 7a. Specifically, the AGRAMPLIFIER-equipped `AGR` achieves higher TA than its counterpart, outperforming it by 45.21%. These findings are supported by the experimental results presented in Table III. Specifically, `AGRMP` outperforms its counterpart base aggregator in distance-based, prediction-based, and trust bootstrapping-based mechanisms with 62.21%, 35.63%, and 16.19%, respectively. Similarly, `AGRXAI` exhibits superior performance over its base aggregator in distance-based, prediction-based, and trust bootstrapping-based mechanisms by 88.68%, 2.41%, and 71.11%, respectively. We further report several observations.

Negative pulse mitigation. In FL training, the presence of these malicious actors can cause significant disruptions to the training process and can severely degrade the accuracy and performance of the shared model. One of the observed phenomena associated with such attacks is the "negative pulse", which is characterized by an abrupt reduction in the model's TA caused by the poisoning adversary at the early stage of the attacks. This phenomenon is consistently observed across different datasets and FL frameworks, indicating the need for effective defense mechanisms to mitigate it. We observe that AGRAMPLIFIER shows promise in effectively mitigating negative pulses as shown in Fig. 8.

Base mechanism matters. Also, it should be noted that the efficacy of the AGRAMPLIFIER-equipped version is predicated on the effectiveness of its base aggregator. If the base aggregator exhibits a weakness in a specific context, this vulnerability may propagate to the amplified version. For instance, the distance-based `CosDen_MP` defense demonstrates an $\mathcal{L}$ error rate of 16.95% against `L-flip` attack on LOCATION30. While the `CosDen_MP` defense improves its performance, achieving an $\mathcal{L}$ error rate of 5.96%, it still falls behind other defenses that achieve an average $\mathcal{L}$ error rate of 0.28%.

2) Targeted attack: Targeted attacks in FL show a significant challenge for detection due to the subtle and incon-

spicuous nature of the triggers that are designed to blend in with normal model updates. However, our AGRAMPLIFIER exhibits promising performance in detecting targeted attacks. Specifically, Fig. 7b presents the results of a targeted attack against base aggregators and AGRAMPLIFIER, demonstrating that AGRXAI outperforms its base aggregator counterpart by achieving a 34.94% lower S. Specifically, AGRMP outperforms its counterpart base aggregator in distance-based, prediction-based, and trust bootstrapping-based mechanisms with 25.02%, 48.4%, and 5.5%, respectively. Similarly, AGRXAI exhibits superior performance over its base aggregator in distance-based, prediction-based, and trust bootstrapping-based mechanisms by 59.22%, 18.89%, and 39.70%, respectively.

It has been observed that the AGRXAI-based AGRAMPLIFIER outperforms the AGRMP-based AGRAMPLIFIER, indicating the potential of XAI techniques in improving targeted attack detection in FL. Specifically, this enhanced performance can be attributed to AGRXAI's ability to capture the significant features of an image that has a high probability of containing the trigger. By selectively manipulating these features, AGRXAI is able to effectively detect the presence of triggers in model updates. These findings highlight the potential of XAI techniques for improving targeted attack detection in FL and suggest that AGRXAI may be a promising approach for enhancing the robustness of FL models against targeted attacks.

The Scale attack and DBA attack against each defense mechanism for seven benchmark datasets are presented in Table III. Among the evaluated defenses, Fang_XAI exhibits consistently strong defense performance across all datasets, followed by Fang_MP, which demonstrates effective defense against non-image datasets. However, FLTrust_MP shows an abnormal performance on TEXAS100, with a 63.76% ASR for scale attacks. This can be attributed to the fact that the trust bootstrapping-based mechanism heavily relies on the representativeness of the validation set. Supplemental Document II-B-4 presents a detailed evaluation of how the distribution of the validation dataset can impact the performance of our proposed method.

TABLE III: Averaged TA loss ($\mathcal{L}$) for untargeted attacks and averaged ASR (S) for targeted attacks (presented in percentage). The best defense for each attack (row-wise) is in bold.

Attack	No def	Distance-based									Prediction-based			Trust-based		
		C	E	M	CM	EM	MM	CX	EX	MX	F	FM	FX	T	TM	TX
Untargeted attacks																
CIFAR-10																
G-asc	1.12	0.71	1.70	0.23	**-0.03**	-0.02	0.07	1.07	0.21	1.06	0.07	0.38	1.02	1.44	1.56	1.92
S&H	0.12	0.07	-0.09	0.23	-0.10	0.12	**-0.18**	0.01	0.09	0.23	-0.11	0.11	0.12	1.96	1.45	1.01
L+G	2.87	5.88	2.95	1.70	0.36	4.02	0.41	1.70	**-1.52**	1.25	0.50	0.51	1.74	2.03	1.82	1.18
MNIST																
G-asc	1.02	1.33	0.24	0.27	0.23	0.01	0.14	0.07	**-0.06**	0.30	0.21	0.21	1.72	5.63	2.27	0.62
S&H	0.41	0.40	0.41	0.30	0.11	0.29	0.36	0.88	0.82	**-0.17**	0.14	0.14	0.03	1.70	2.18	1.27
L+G	0.73	2.63	85.86	2.23	1.39	0.38	0.16	0.14	**-1.06**	-0.58	-0.08	-0.08	0.10	5.60	2.35	-0.54
CAT-DOG																
G-asc	6.80	6.92	4.36	4.02	2.65	5.36	1.94	8.91	3.04	3.09	6.14	9.67	6.42	5.60	**-0.48**	0.93
S&H	3.19	4.28	4.65	3.92	**-2.83**	2.06	-0.21	-0.18	0.50	-2.58	1.74	-0.55	-0.54	8.43	5.54	-0.13
L+G	28.54	0.07	3.42	5.45	2.89	-2.04	1.49	-3.22	**-4.69**	3.46	1.32	0.14	1.32	7.18	-3.32	0.08
FASHION-MNIST																
G-asc	5.02	5.02	3.20	3.02	2.55	4.29	1.90	1.07	0.19	0.90	4.42	9.77	3.92	2.89	**-0.12**	1.22
S&H	2.10	3.09	3.29	2.08	2.01	2.09	-0.31	2.88	**-3.82**	0.27	1.59	0.01	0.83	5.09	-0.42	2.27
L+G	20.04	2.61	4.98	2.01	2.01	0.32	0.21	**-3.14**	1.02	-0.28	2.09	0.12	0.60	6.47	2.31	0.34
LOCATION30																
G-asc	8.48	16.95	**0.50**	3.77	5.98	1.89	3.14	-	-	-	3.40	4.28	-	5.74	8.67	-
S&H	0.69	-0.24	1.31	2.76	**-0.65**	0.02	0.20	-	-	-	2.40	1.35	-	4.26	0.50	-
L+G	6.50	3.17	11.34	9.37	**-0.45**	6.44	3.23	-	-	-	1.33	1.86	-	2.74	8.36	-
PURCHASE100																
G-asc	12.82	17.49	17.61	1.64	0.96	2.25	2.52	-	-	-	**0.75**	1.04	-	2.69	2.55	-
S&H	**0.04**	0.34	1.34	1.81	0.16	1.54	1.69	-	-	-	0.66	0.89	-	3.52	1.87	-
L+G	33.78	14.22	19.22	2.75	**0.54**	1.07	0.75	-	-	-	1.06	1.48	-	4.82	2.64	-
TEXAS100																
G-asc	1.98	1.39	1.17	2.64	1.17	1.58	1.49	-	-	-	2.00	0.52	-	1.66	**-0.15**	-
S&H	1.46	0.58	1.87	2.51	0.53	1.91	1.99	-	-	-	1.93	0.51	-	0.51	**-0.45**	-
L+G	2.33	3.23	5.52	5.20	**1.25**	4.00	2.79	-	-	-	2.40	1.96	-	2.20	1.39	-
Targeted attacks																
CIFAR-10																
Scale	94.44	17.35	14.60	10.03	2.64	1.89	1.09	2.81	**0.25**	1.74	49.60	2.70	0.49	12.25	9.91	4.21
DBA	85.64	12.64	10.43	15.01	9.03	9.22	8.61	8.99	6.00	3.95	15.12	5.32	**3.88**	13.29	18.00	13.38
MNIST																
Scale	16.58	0.37	0.89	2.19	0.57	0.37	0.39	2.81	0.25	1.74	5.54	**0.24**	0.49	3.59	0.35	0.79
DBA	16.95	2.95	2.41	1.96	0.34	10.09	0.63	7.83	0.90	0.42	5.39	1.27	**0.23**	3.20	0.46	1.80
CAT-DOG																
Scale	80.26	37.14	27.90	26.13	27.14	16.40	16.13	16.18	14.36	12.50	24.23	14.03	10.68	20.53	10.30	**6.88**
DBA	95.23	24.47	23.53	22.24	14.03	14.37	13.83	12.24	13.28	17.63	19.52	10.43	20.07	12.79	10.39	**10.23**
FASHION-MNIST																
Scale	79.95	22.76	19.85	14.29	2.70	9.85	4.12	14.70	13.96	3.44	12.45	**2.41**	14.3	14.39	4.83	3.20
DBA	80.01	13.32	14.34	17.95	3.89	4.51	4.18	15.02	12.09	**0.22**	23.21	3.32	10.32	13.45	3.24	2.33
LOCATION30																
Scale	53.45	3.13	7.32	2.34	8.46	12.6	13.6	-	-	-	5.54	**1.39**	-	3.59	4.68	-
DBA	65.32	3.21	3.66	2.40	2.74	2.98	**1.93**	-	-	-	7.43	2.34	-	6.23	3.56	-
PURCHASE100																
Scale	74.78	0.56	3.21	2.67	1.73	2.82	**0.04**	-	-	-	0.05	1.02	-	0.41	1.11	-
DBA	69.03	4.21	3.23	1.35	3.89	4.51	4.18	-	-	-	2.23	3.12	-	4.26	**1.23**	-
TEXAS100																
Scale	99.03	11.16	32.02	12.43	33.1	14.4	0.03	-	-	-	**0.01**	0.03	-	35.59	63.76	-
DBA	89.21	13.35	12.32	10.78	21.02	12.65	3.54	-	-	-	**2.01**	3.12	-	34.04	15.54	-

† In this table, we use No def to stand for no defense, C to stand for CosDen, CM for CosDen_MP, CX for CosDen_XAI, E to stand for EuDen, EM for EuDen_MP, EX for EuDen_XAI, M to stand for MergeDen, MM for MergeDen_MP, MX for MergeDen_XAI, F to stand for Fang, FM for Fang_MP, FX for Fang_XAI, T to stand for FLTrust, TM for FLTrust_MP, TX for FLTrust_XAI (The abbreviation also applies to Tables IV).

B. Fidelity Boosting

Fidelity is a crucial factor in assessing an aggregator's effectiveness in preserving helpful information during the FL process. Fig. 9 presents a comparison of the fidelity scores between AGRAMPLIFIER and their base AGRs. Our results demonstrate that AGRAMPLIFIER (indicated by the pink and green bars) outperforms the original base aggregators (indicated by the blue bars) by showing significantly less fidelity loss. These findings suggest that AGRAMPLIFIER is effective in improving the fidelity of base aggregators, which is essential for preserving the quality of the trained model and enhancing the robustness of the federated learning process.

Our evaluation reveals that the distance-based CosDen_MP aggregator performs the most promisingly, achieving an average fidelity loss of 0.18%. Furthermore, the prediction-based Fang_XAI aggregator also demonstrates desirable fidelity with an average loss of 0.68%. However, the performance of trust bootstrapping-based FLTrust_XAI highly depends on the dataset used. Notably, this approach can significantly reduce the fidelity loss in some datasets. Still, it may yield a higher fidelity loss on others, such as LOCATION30 and PURCHASE100, where the validation dataset distribution differs from the original training set, potentially leading to biased global models. The detailed results are given in Supplemental Document II-A.

C. Efficiency Boosting

Table IV presents the time consumption of AGRAMPLIFIER experiments carried out on CIFAR-10 utilizing the NVIDIA GeForce RTX 3080 device. For AGRMP-based methods, the time complexity of the amplification process is $\mathcal{O}(N * P_n)$ in each iteration, (where N is the number of clients and

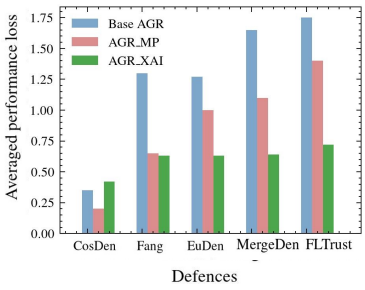


Fig. 9: Averaged performance loss $\mathcal{L}$ across all datasets in terms of fidelity.

TABLE IV: Time consumption (in second) of aggregation on CIFAR-10.

Hidden layer	C	CM	CX	F	FM	FX	T	TM	TX
512	330.8	241.0	2128.4	439.8	341.1	2383.3	316.4	233.2	2340.8
1024	328.6	240.6	2308.1	426.9	340.2	2352.7	324.8	233.1	2401.1
2048	374.3	306.8	3542.3	442.6	412.3	2342.8	320.2	295.3	2538.7

P_n is the number of parameters in the model). Since the value of P_n is significantly reduced by the amplification, our AGRMP effectively lowers the computational cost for AGR, particularly for large-size neural networks. Specifically, as is shown in Table IV, AGRMP reduces total time consumption by 20% from 369.38 seconds to 293.73 seconds, on average. However, AGRXAI-based methods exhibit increased time consumption due to the additional computation involved in collecting feature weights. The time complexity of the AGRXAI amplification process is $\mathcal{O}(N*(N_c+P_n^2))$, where N_c is the total number of computations involved in the forward pass and gradient calculation [28].

The space complexity of AGRAMPLIFIER is equivalent to that of the traditional FL method, i.e., $\mathcal{O}(N * P_n)$, which exhibits linear growth with respect to both N and P_n.

D. Influence Factors

We conduct extensive experiments to evaluate AGRAMPLIFIER's performance in varying scenarios concerning 1) the number of participants, 2) the malicious factor, and 3) the proportion of the extracted gradient. We also investigate the effect of Dataset Heterogeneity. Experiment results demonstrate our advantage under different settings. (Detailed results are displayed in the Supplemental Document II-B.)

VI. DISCUSSION

A. Comparison of AGRMP and AGRXAI

In terms of robustness, AGRXAI outperforms AGRMP, especially for targeted attacks. The enhanced performance of AGRXAI can be attributed to its ability to capture significant features containing triggers, thereby improving detection performance. In terms of fidelity, AGRXAI also outperforms AGRMP in general, but both AGRMP and AGRXAI show significantly less fidelity loss compared to the base AGRs. In terms of efficiency, AGRAMPLIFIER reduces gradient size, resulting in reduced time consumption, particularly for larger neural network architectures. However, the AGRXAI approach demonstrates higher time consumption due to additional computations required for individual update feature weights.

Specifically, EuDen_XAI demonstrates the highest robustness against targeted attacks, while Fang_XAI performs the best against untargeted attacks. In terms of fidelity, CosDen_MP exhibits superior performance, closely followed by CosDen_XAI. Furthermore, FLTrust_MP demonstrates superior efficiency with small to medium network sizes, while CosDen_MP delivers the best efficiency with large networks.

In conclusion, investigating the trade-off between robustness, fidelity, and efficiency in the ten proposed versions is a promising direction for future research. By considering the factors mentioned above and conducting rigorous experiments, researchers can gain valuable insights into the interplay between these aspects and make informed decisions to achieve the desired balance in real-world scenarios.

B. Furture Works

Several promising directions for future research are of interest. Firstly, we aim to explore and establish a better trade-off between robustness, fidelity, and efficiency. Secondly, ongoing efforts are dedicated to enhancing the explainability of Natural language processing (NLP) tasks [43], [44]. Therefore, our AGRXAI-based approach, which leverages Explainable AI (XAI), holds promise for extension into NLP tasks and improves the identification and understanding of significant features within participants' updates. We will also investigate other defense mechanisms, such as frequency-domain analysis for detecting malicious updates and style transformation techniques for converting malicious updates into benign ones.

VII. CONCLUSION

This study presents AGRAMPLIFIER, a mechanism specifically designed to enhance the robustness, fidelity, and efficiency of FL methods against Byzantine adversaries. Through comprehensive evaluations across various scenarios, our approach demonstrates an average reduction of 40.07% in the ASR while maintaining high levels of fidelity and efficiency. These results highlight the effectiveness of AGRAMPLIFIER in mitigating the impact of Byzantine adversaries in FL. Furthermore, AGRAMPLIFIER presents the first work that incorporates XAI into the domain of malicious detection. This integration not only enhances the performance of existing methods but also opens up new avenues for future research and exploration.

REFERENCES

[1] L. Shen, Y. Zhang, J. Wang, and G. Bai, "Better together: Attaining the triad of byzantine-robust federated learning via local update amplification," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 201–213.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2017, pp. 1273–1282.

[3] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2015, pp. 1310–1321.

[4] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[6] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," in *Proceedings of the 2021 Network and Distributed System Security Symposium*. Internet Society, 2021.

[7] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proceedings of the 29th USENIX Conference on Security Symposium*, 2020, pp. 1623–1640.

[8] M. Ma, Y. Zhang, P. C. M. Arachchige, L. Y. Zhang, M. B. Chhetri, and G. Bai, "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*, 2023, pp. 122–135.

[9] J. Wei, Y. Zhang, L. Y. Zhang, C. Chen, S. Pan, K.-L. Ong, J. Zhang, and Y. Xiang, "Client-side gradient inversion against federated learning from poisoning," *arXiv preprint arXiv:2309.07415*, 2023.

[10] X. Li, Z. Qu, S. Zhao, B. Tang, Z. Lu, and Y. Liu, "Lomar: A local defense against poisoning attack on federated learning," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[11] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice *et al.*, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 27–38.

[12] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proceedings of 28th Annual Network and Distributed System Security Symposium (NDSS)*, 2021.

[13] H. Zhang, Z. Yao, L. Y. Zhang, S. Hu, C. Chen, A. Liew, and Z. Li, "Denial-of-service or fine-grained control: Towards flexible model poisoning attacks on federated learning," *arXiv e-prints*, pp. arXiv–2304, 2023.

[14] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2016.

[15] D. Wang, S. Wen, A. Jolfaei, M. S. Haghighi, S. Nepal, and Y. Xiang, "On the neural backdoor of federated generative models in edge computing," *ACM Transactions on Internet Technology (TOIT)*, vol. 22, no. 2, pp. 1–21, 2021.

[16] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018.

[17] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018.

[18] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "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*, 2022, pp. 2545–2555.

[19] Y. Zhang, G. Bai, M. A. P. Chamikara, M. Ma, L. Shen, J. Wang, S. Nepal, M. Xue, L. Wang, and J. Liu, "Agrevader: Poisoning membership inference against byzantine-robust federated learning," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2371–2382.

[20] W. Wan, S. Hu, M. Li, J. Lu, L. Zhang, L. Y. Zhang, and H. Jin, "A four-pronged defense against byzantine attacks in federated learning," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 7394–7402.

[21] J. Shi, W. Wan, S. Hu, J. Lu, and L. Y. Zhang, "Challenges and approaches for mitigating byzantine attacks in federated learning," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 139–146.

[22] G. Geng, T. Cai, and Z. Yang, "Better safe than sorry: Constructing byzantine-robust federated learning with synthesized trust," *Electronics*, vol. 12, no. 13, p. 2926, 2023.

[23] J. Xu, X. Tong, and S.-L. Huang, "Communication-efficient and byzantine-robust distributed stochastic learning with arbitrary number of corrupted workers," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 5415–5420.

[24] K. Yamaguchi, K. Sakamoto, T. Akabane, and Y. Fujimoto, "A neural network for speaker-independent isolated word recognition." in *Proceedings of the First International Conference on Spoken Language (ICSLP)*. ISCA, Nov. 1990, pp. 1077–1080.

[25] F. Charmet, H. C. Tanuwidjaja, S. Ayoubi, P.-F. Gimenez, Y. Han, H. Jmila, G. Blanc, T. Takahashi, and Z. Zhang, "Explainable artificial intelligence for cybersecurity: a literature survey," *Annals of Telecommunications*, vol. 77, no. 11-12, pp. 789–812, 2022.

[26] A. Kuppa and N.-A. Le-Khac, "Adversarial xai methods in cybersecurity," *IEEE transactions on information forensics and security*, vol. 16, pp. 4924–4938, 2021.

[27] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, Dec. 2020.

[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam *et al.*, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

[29] D. Yang, D. Zhang, and B. Qu, "Participatory cultural mapping based on collective behavior data in location-based social networks," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 7, no. 3, pp. 1–23, 2016.

[30] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, Jun. 2012.

[31] M. Jagielski, A. Oprea, B. Biggio, C. Liu *et al.*, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[32] P. Kairouz, H. B. McMahan, B. Avent *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[33] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information*, vol. 30. Curran Associates, Inc., 2017.

[34] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *Clinical Orthopaedics and Related Research (CoRR)*, 2019.

[35] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," in *Proceedings of the 29th British Machine Vision Conference*, 2018.

[36] C. Burns, J. Thomason, and W. Tansey, "Interpreting black box models via hypothesis testing," in *Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference*, 2020, pp. 47–57.

[37] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *NDSS*, 2021.

[38] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International conference on learning representations*, 2019.

[39] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller *et al.*, "Learning how to explain neural networks: Patternnet and patternattribution," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2017.

[40] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[41] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *Clinical Orthopaedics and Related Research (CoRR)*, 2017.

[42] Kaggle, "Dogs vs. Cats Competition," Available online: https://www.kaggle.com/competitions/dogs-vs-cats.

[43] M. H. Shakil and M. G. R. Alam, "Hate speech classification implementing nlp and cnn with machine learning algorithm through interpretable explainable ai," in *2022 IEEE Region 10 Symposium (TENSYMP)*, 2022, pp. 1–6.

[44] Y. Sisodia, "Explainable ai for nlp: Decoding black box."

[45] S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks," in *Proceedings of the International Conference on Machine Learning*. PMLR, May 2019, pp. 322–332.

AGRAMPLIFIER: Defending Federated Learning Against Poisoning Attacks Through Local Update Amplification

Supplementary Material

I. ALGORITHMS FOR AGRAMPLIFIER

A. AGRAMPLIFIER *for Distance-based Mechanism*

The proposed algorithm involves processing the original gradients through an amplifier (line 1 in Algorithm 1), followed by a distance measurement step that employs a pair-wise comparison approach, such as Cosine similarity or Euclidean distance. This distance measurement step determines the density of each gradient's neighborhood, consisting of K neighbors, where K is set to a value greater than $N/2$, with N representing the number of participants. The algorithm considers gradients residing in denser neighborhoods as benign ones. For each participant's update, the algorithm selects the K-nearest neighbors based on their similarity score and calculates the cumulative score in their neighborhood (lines 4-10). The top N_t participants with higher scores are then added to the white list, which can be considered a hyperparameter chosen by the practitioner and submitted for aggregation (lines 12-19). Therefore, the proposed algorithm utilizes a novel approach to improve the performance of FL by selecting benign participants through a similarity-based analysis of gradient neighborhoods.

Algorithm 1 AGRAMPLIFIER for Distance-based Mechanism.

Input: g_i, g_j - the gradients collected from i-th and j-th clients; g^i_{amp} - the amplified gradients from client i; N - number of participants; M_f - the fraction of malicious participants; K - number of neighbors examined.
Output: G_{detox} the detoxed gradients collection.

1: $\left\{g^1_{amp}, g^2_{amp}, \ldots\right\} \leftarrow$ AGRAMPLIFIER $(g_1, g_2, \ldots)$
2: **for** $i = 0, 1, 2, ..., N$ **do**
3: **for** $j = 0, 1, 2, ..., N$ **do**
4: // Pair-wise cosine similarity
5: $C_{i,j} \leftarrow \frac{g^i_{amp} \cdot g^j_{amp}}{\left\|g^i_{amp}\right\| \cdot \left\|g^j_{amp}\right\|}$
6: **end for**
7: // Sum up the similarity of the K-nearest neighborhood as S_i
8: $C_i \leftarrow$ Descending-sort $(C_{i,j} \mid j = 1, 2, 3, \ldots N)$
9: $S_i \leftarrow \sum(C_{i,j} \mid j = 1, 2, 3, \ldots K)$
10: **end for**
11: Whitelist $\leftarrow \emptyset$
12: **for** $i = 0, 1, 2, ..., N$ **do**
13: // Add into white-list of i is with larger S_i
14: **if** S_i in the largest $(1 - M_f) * N$ values $\forall S_i$ **then**
15: Whitelist $\leftarrow$ Whitelist $\cup \{i\}$
16: **end if**
17: **end for**
18: $G_{\text{detox}} \leftarrow \{g_i \mid i \in$ Whitelist $\}$
19: **return** G_{detox}

B. AGRAMPLIFIER *for Prediction-based Mechanism*

The AGRAMPLIFIER algorithm initially performs gradient amplification while ensuring that the size of the collected gradients is restored (line 1 in Algorithm 2). This is accomplished by retaining the maximum value in each feature map and replacing the dropped features with zeros. Subsequently, the restored gradients undergo calculation of LRR and ERR [7], and those leading to superior prediction performance are included in the white list (line 3).

Algorithm 2 AGRAMPLIFIER for Prediction-based Mechanism.

Input: G - the collected gradients, equivalent to $\{g_1, g_2, g_3, \ldots\}$; G_{amp} - the amplified gradients; g_i - the collected gradient from i-th client in G.
Output: G_{detox} the detoxed gradients collection.

1: $G_{amp} \leftarrow$ AGRAMPLIFIER$(G, Restore - size = True)$
2: Whitelist $\leftarrow \text{LRR}\left(G_{amp}\right) \cap \text{ERR}\left(G_{amp}\right)$
3: $G_{\text{detox}} \leftarrow \{g_i \mid i \in$ Whitelist, $g_i \in G\}$
4: **return** G_{detox}

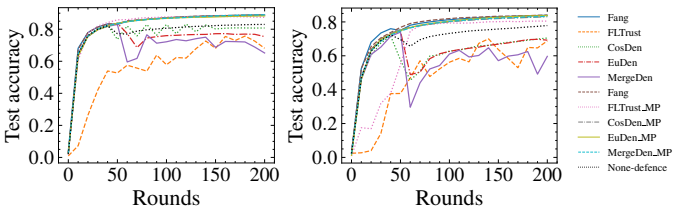
C. AGRAMPLIFIER *for Trust Bootstrapping-based Mechanism*

The AGRAMPLIFIER algorithm not only conducts gradient amplification on the collected gradients to produce g^i_{amp} (line 1 in Algorithm 3) but also amplifies the gradient generated from the trusted validation set for trust bootstrapping to produce g^0_{amp} (line 2). The trust score TS_i for each participant is subsequently determined by computing the ReLU-clipped cosine similarity between the amplified gradients g^i_{amp} and g^0_{amp} (line 4). The collected local model update magnitudes are then normalized (line 5), weighted by their respective trust scores, and aggregated to form the global model (line 7).

Algorithm 3 AGRAMPLIFIER for Trust Bootstrapping-based Mechanism.

Input: g^i_{amp} - the amplified gradients from client i; g_i - the collected gradient from client i; g_0 - the gradient generated from the trusted root dataset.
Output: G_{detox} the detoxed gradients collection.

1: $\left\{g^1_{amp}, g^2_{amp}, \ldots\right\} \leftarrow$ AGRAMPLIFIER $(g_1, g_2, \ldots)$
2: $g^0_{amp} \leftarrow$ AGRAMPLIFIER (g_0)
3: **for** $i = 1, 2, \ldots, N$ **do**
4: $TS_i \leftarrow \mathbf{ReLU}\left(\frac{g^i_{amp} \cdot g^0_{amp}}{\left\|g^i_{amp}\right\| \cdot \left\|g^0_{amp}\right\|}\right)$
5: $\bar{g}_i \leftarrow \frac{\|g_0\|}{\|g_i\|} \cdot g_i$
6: **end for**
7: $g_{\text{detox}} \leftarrow \frac{1}{\sum_{j=1}^{N} TS_j} \sum_{i=1}^{N} TS_i \cdot \bar{g}_i$
8: **return** G_{detox}



(a) 500 Participants (b) 1000 Participants

Fig. 1: Performance of run-time TA for untargeted attacks with a large number of participants.

II. Experiments Results

A. Fidelity Boosting

Table I shows the detailed results of the performance loss in terms of fidelity.

TABLE I: Performance loss $\mathcal{L}$ in terms of fidelity.

Datasets	Distance									Prediction			Trust		
	C	E	M	CM	EM	MM	CX	EX	MX	F	FM	FX	T	TM	TX
CIFAR-10	-0.07	0.50	0.68	0.08	0.01	0.24	0.17	0.27	0.29	0.15	0.23	0.29	1.48	1.38	1.29
MNIST	0.41	0.74	0.62	048	0.72	0.88	0.14	0.13	0.10	0.64	0.54	0.15	0.62	1.39	0.13
CAT-DOG	0.43	0.42	0.44	0.42	0.41	0.40	0.42	0.45	0.41	0.38	0.39	0.38	0.48	0.62	0.42
FASHION-MNIST	0.51	0.63	0.55	0.46	0.39	0.23	0.21	0.30	0.24	0.34	0.25	0.21	0.35	0.43	0.20
LOCATION30	0.00	2.84	3.59	-0.15	1.35	1.69	-	-	-	3.64	1.29	-	3.21	1.87	-
PURCHASE100	0.53	1.23	1.69	0.28	1.48	1.29	-	-	-	0.69	0.94	-	3.17	1.98	-
TEXAS100	0.68	0.98	1.49	0.18	1.48	1.18	-	-	-	1.30	0.38	-	0.31	-0.53	-

† In this table, we use `C` to stand for `CosDen`, `CM` for `CosDen_MP`, `CX` for `CosDen_XAI`, `E` to stand for `EuDen`, `EM` for `CosDen_MP`, `EX` for `EuDen_XAI`, `M` to stand for `MergeDen`, `MM` for `MergeDen_MP`, `MX` for `MergeDen_XAI`, `F` to stand for `Fang`, `FM` for `Fang_MP`, `FX` for `Fang_XAI`, `T` to stand for `FLTrust`, `TM` for `FLTrust_MP`, `TX` for `FLTrust_XAI`. (The abbreviation also applies to Table II).

B. Influence Factors

This section presents the detailed evaluation of AGRAMPLIFIER's performance in varying scenarios concerning 1) the number of participants, 2) the malicious factor, and 3) the proportion of the extracted gradient. Specifically, in Section II-B1, we consider 30, 50, and 100 participants while fixing the percentage of malicious participants at 30%. In Section II-B2, we consider 20%, 30%, and 40% proportions of malicious participants, with a fixed number of participants at 50. In Section II-B3, we consider the kernel size to be within the range of 2×2, 3×3, 5×5, 7×7, and 9×9 for `AGRMP`, and the proportion of the extracted gradient to be top 10%, 25%, 50%, and 75% for `AGRXAI`.

1) Impact of Participant Number: Fig. 2 demonstrates the run-time ASR of AGRAMPLIFIER for targeted attacks and TA for untargeted attacks on MNIST, varying the participant count. Our findings reveal that AGRAMPLIFIER outperforms the base aggregator and achieves comparable performance to FedAvg when the number of participants ranges from 30 to 100. Furthermore, we find that Base aggregators underperform when there is a large number of participants (500 and 1000 participants), leading to a reduction in overall performance for all considered aggregators. However, our proposed `AGRMP` still maintains good performance, as shown in Fig. 1.

2) Impact of Malicious Client Proportion: This section examines the effect of varying ratios of malicious participants (M_f) on the performance of AGRAMPLIFIER. The results are presented in Fig. 3, which depicts the run-time TA and ASR of AGRAMPLIFIER on MNIST with M_f values ranging from 0.2 to 0.4. The performance of FedAvg (without defense) is shown to drop significantly as the proportion of malicious participants increases. However, the performance of AGRAMPLIFIER remains stable and exhibits high TA even when M_f reaches 40%.

3) Impact of the Proportion of the Extracted Gradient: In this section, we investigate how the proportion of the extracted gradient affects the performance of AGRAMPLIFIER. First, for `AGRMP`, the varying kernel sizes can affect the selection of the extracted gradients. Specifically, the kernel size is experimented within the range of 2×2, 3×3, 5×5, 7×7, and 9×9. The results indicate that the distance-based `CosDen_MP` and prediction-based `Fang_MP` are not significantly impacted by the kernel size, whereas the trust bootstrapping-based `FLTrust_MP` shows a slight decrease in performance with larger kernel sizes such as 7×7 and 9×9. This mechanism computes the reference updates on a small trust set, which may result in non-negligible information loss when the kernel size is too large. The findings are presented in Fig. 4.

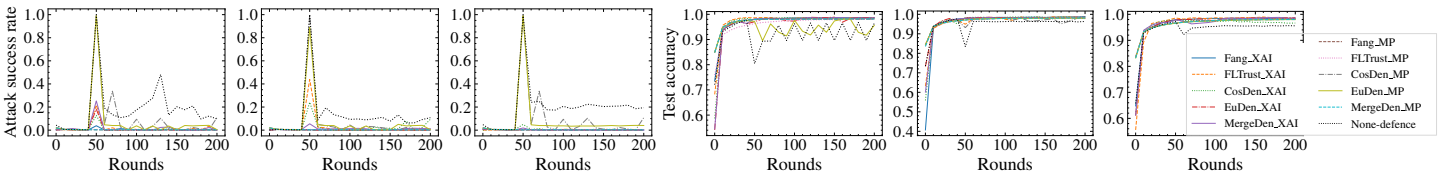
We also investigate how the proportion p affects the performance of `AGRXAI`. We conduct experiments by extracting the top 10%, 25%, 50%, and 75% of the gradient g_i. It shows a similar result as `AGRMP`, where the trust bootstrapping-based `FLTrust_XAI` shows a slight decrease in performance with a large reduction size such as 10%. The detailed findings are presented in Fig. 5 and 6. The results indicate that extracting the top 50% of the gradients has the best performance for both untargeted and targeted attacks. This choice not only achieves the best evaluation metric but also exhibits the smoothest trend.

4) Impact of the validation datasets: In the case of `Fang` and `FLTrust`-based defenses, which require the aggregator to hold a small validation dataset, we consider two scenarios [6] to evaluate the impact for validation dataset distribution:

- Case I: In this scenario, we assume the aggregator has the capability to construct a well-balanced validation dataset that represents the distribution of the training data. To achieve this, we uniformly and randomly sample the validation dataset from the clean local training data of clients.

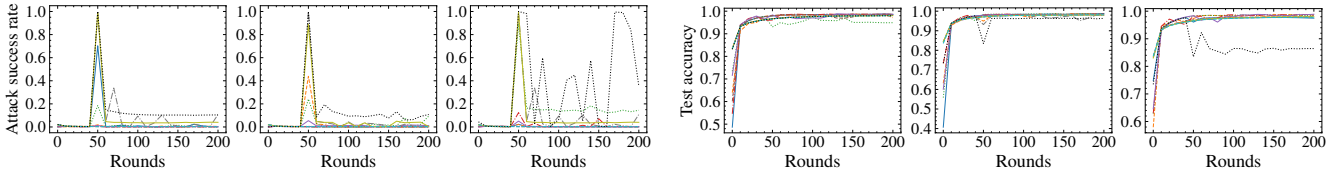
- Case II: We assume a validation dataset with a different distribution compared to the training data, specifically biased towards a particular class (class 1 in our case). We use bias probability (θ) to represent the fraction of examples from that specific class and set the other $1-\theta$ examples to be uniformly sampled from the other classes.

Table II presents the model TA based on different bias probabilities (θ), and the result shows that the impact of data distribution on these methods differs:
For the `Fang`-based methods, they consistently perform well across various bias probabilities. In contrast, the `FLTrust`-based methods demonstrate varying performance. The base `FLTrust` method fails when the bias probability is 0.4. However, our AGRAMPLIFIER-equipped approach displays



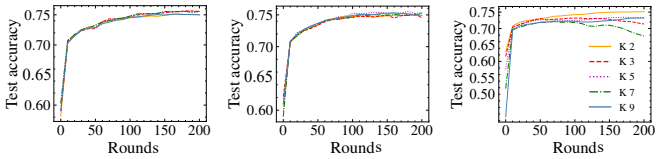
(a) 30-targeted 50-targeted 100-targeted (b) 30-untargeted 50-untargeted 100-untargeted

Fig. 2: (a) run-time ASR for targeted attacks, (b) run-time TA for untargeted attacks with different numbers of participants on MNIST.



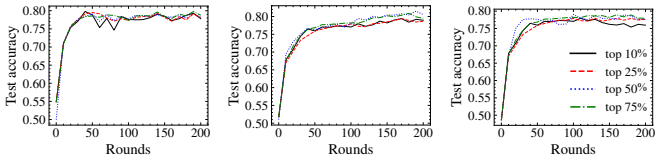
(a) 0.2-targeted 0.3-targeted 0.4-targeted (b) 0.2-untargeted 0.3-untargeted 0.4-untargeted

Fig. 3: (a) run-time ASR for targeted attacks, (b) run-time TA for untargeted attacks with different proportions of malicious participants on MNIST.



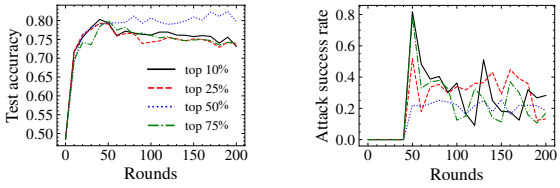
(a) `CosDen_MP` (b) `Fang_MP` (c) `FLTrust_MP`

Fig. 4: The impact of kernel size for `AGRMP` on CIFAR-10.



(a) `CosDen_XAI` (b) `Fang_XAI` (c) `FLTrust_XAI`

Fig. 5: The impact of the proportion of the extracted gradient for `AGRXAI` on CIFAR-10.



(a) Untargeted attacks (b) Targeted attacks

Fig. 6: The impact of the proportion of the extracted gradient on CATvsDOG Kaggle.

improved resilience and can handle a bias probability up to 0.9. This difference in performance can be attributed to the underlying mechanisms of the base methods. The `Fang`-based approach evaluates local models based on their error rates on the clean validation dataset. It remains effective even in the presence of dataset bias, as the benign model continues to make accurate predictions. On the other hand, the `FLTrust`-based methods train a clean model using its own clean dataset and assign trust scores to each update by comparing them with the clean model. Consequently, it becomes vulnerable to highly biased validation datasets. However, our approach mitigates this vulnerability by amplifying the differences between benign and malicious gradients. Even when trained on biased data, the gradients remain sufficiently close to the benign ones, facilitating the distinction of malicious gradients. This insight highlights the effectiveness of our approach in handling validation dataset biases.

TABLE II: The TA of `FLTrust` and `Fang` based methods under G-asc attacks on the CIFAR dataset when the validation dataset is sampled with different bias probabilities. We highlight 0.1 in bold, which indicates that the training fails to converge.

Bias	0	0.2	0.4	0.6	0.8	0.9	1
None	0.60	0.61	0.59	0.60	0.60	0.60	0.59
T	0.57	0.57	**0.1**	0.55	**0.1**	**0.1**	**0.1**
TM	0.59	0.58	0.58	0.57	0.57	**0.1**	**0.1**
TX	0.58	0.58	0.58	0.57	0.58	0.58	**0.1**
F	0.58	0.57	0.56	0.55	0.54	0.53	0.54
FM	0.59	0.58	0.56	0.55	0.55	0.52	0.55
FX	0.60	0.59	0.59	0.58	0.56	0.56	0.55

5) Effect of Dataset's Heterogeneity: Our observations indicate that the performance of the three categories of Byzantine-robust mechanisms varies depending on the dataset employed. Previous studies have shown that the generalization bound of ML is linked to the diversity of the training data [45]. As the model's generalization capacity decreases, poisoning attacks become more effective, as it enhances the attacker's ability to increase the loss on poisoned examples. Conversely, a well-generalized model is better equipped to handle varied input and thus exhibits greater resilience to malicious injections.

Our experimental findings provide evidence of a correlation between the heterogeneity of datasets and the efficacy of

TABLE III: Heterogeneity of the datasets.

MNIST	CIFAR-10	LOCATION30	PURCHASE100	TEXAS100	CAT-DOG	FASHION-MNIST
0.15	0.24	0.12	0.03	0.61	0.41	0.21

defense mechanisms. We measure the heterogeneity of the datasets by computing the average intra-label cosine similarity using the approach outlined in [45]. Specifically, given a training set label ξ denoted as D_ξ, and the number of classes in the dataset denoted as E, the heterogeneity scores is computed as:

$$1 - \frac{1}{E} \sum_{\xi=1}^{E} \frac{\sum D_\xi \cdot D_\xi^\top}{\|D_\xi\|^2}. \tag{4}$$

Table III presents the heterogeneity scores for each dataset. Our analysis indicates that datasets with lower heterogeneity, as measured by larger intra-label cosine similarity, confer advantages to defenders, particularly against the scale attack. For instance, the scale attack on PURCHASE100 and LOCATION30 exhibits the lowest success rates (less than 10%) when defensive mechanisms are applied. Furthermore, we find that the performance of trust bootstrapping-based mechanisms is significantly impacted when the training data's heterogeneity is extremely high, as observed in TEXAS100, with the highest heterogeneity score.