Better Together: Attaining the Triad of Byzantine-robust Federated Learning via Local Update Amplification

Anonymous Author(s)

ABSTRACT

Manipulation of local training data and local updates, i.e., the *Byzantine poisoning attack*, is the main threat arising from the collaborative nature of the federated learning (FL) paradigm. Many Byzantine-robust aggregation algorithms (AGRs) have been proposed to filter out or moderate suspicious local updates uploaded by Byzantine participants at the central aggregator. However, they largely suffer from model quality degradation due to the over-removal of local updates or/and the inefficiency caused by the expensive analysis of the high-dimensional local updates.

In this work, we propose AgrAmplifier that aims to simultaneously attain the triad of *robustness*, *fidelity* and *efficiency* for FL. AgrAmplifier features the amplification of the "morality" of local updates to render their maliciousness and benignness clearly distinguishable. It re-organizes the local updates into patches and extracts the most activated features in the patches. This strategy can effectively enhance the robustness of the aggregator, and it also retains high fidelity as the amplified updates become more resistant to local translations. Furthermore, the significant dimension reduction in the feature space greatly benefits the efficiency of the aggregation.

AgrAmplifier is compatible with *any* existing Byzantine-robust mechanism. In this paper, we integrate it with three mainstream ones, i.e., distance-based, prediction-based, and trust bootstrapping-based mechanisms. Our extensive evaluation against five representative poisoning attacks on five datasets across diverse domains demonstrates the consistent enhancement for all of them, with average gains at 57.47%, 30.40% and 10.68% in terms of robustness, fidelity, and efficiency respectively.

1 INTRODUCTION

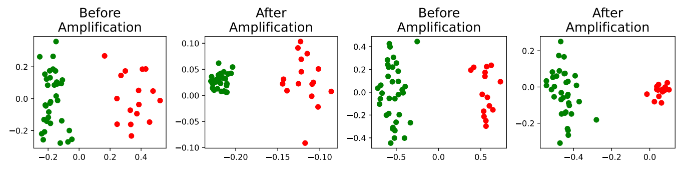
Federated learning (FL) enables participants to jointly train a global model without the necessity of sharing their datasets [24, 29]. It has increasingly spread over various privacy-sensitive tasks such as medical data segmentation [10] and has been incorporated by popular machine learning tools [1]. In FL, each participant holds its own training data and trains local updates based on the current global model. A central server, i.e., the *aggregator*, repeatedly collects and aggregates the local updates, and takes one-step downward in gradient descent to update the *global model*, which is then broadcast to participants for the next training round.

The inherent collaboration paradigm of FL makes it vulnerable to the *poisoning attacks*. A fraction of the participants controlled by an adversary can behave maliciously during the training process to sabotage the trained global model. They may poison their local data, called *data poisoning* [32], or mutate their local updates, called *model poisoning* [4, 6, 11, 27, 31], to corrupt the global model towards deviated predictions over arbitrary test samples, called *untargeted attacks*, or mislead the model to misclassify the attacker-chosen sample classes/sets, called *targeted attacks*.

Many Byzantine-robust mechanisms have been proposed to defend against poisoning attacks. The aggregator is equipped with some robust aggregation algorithm (AGR) [2, 8, 9, 11, 13, 22, 27, 33, 38] to exclude or moderate suspicious local updates to alleviate the impact of negative contributions from the adversary. The most widely used strategy is to cross-check local updates and remove anomalous ones before aggregating them [8, 25, 27]. Distances and similarity (e.g., Euclidean distance and Cosine similarity) among the local updates can be measured for anomaly detection. Another line of work is based on feedback of the global model after absorbing a local update [11, 22]. Those updates that exert negative influence on the model performance would be excluded. A recent study [9] also demonstrates the effectiveness in defense by leveraging a small clean dataset for bootstrapping trust in FL.

Triad of Byzantine-robust federated learning. Besides the essential requirement of retaining *robustness* against poisoned updates, an AGR is desired to preserve model utility in presence of data variety, called *fidelity* [9]. In practice, data variety among different owners is inevitable, and most of the time, beneficial for the generalization of the trained global model. It may render the yielded local updates deviated from their counterparts though, and existing defenses often deny such updates. The third pillar of the AGR is *efficiency* [9]. In each training round, the AGR has to process all local updates, which are often highly dimensional. This is non-trivial in large-scale scenarios. For example, as reported by Shejwalkar et al. [28], some real-world FL scheme involving mobile apps may have 1 billion participants. In light of these, the key question we target to answer is *how can we attain the robustness, fidelity and efficiency triad for Byzantine-robust aggregators?*

Our approach: amplifying morality of local updates. We propose AgrAmplifier to boost Byzantine-robust aggregation in general. Its core idea is to amplify the maliciousness and benignness of the local updates by extracting their most activated feature. Our approach is inspired by the *max pooling operation* [34], which is widely used in modern Convolutional Neural Networks (CNN) models to reduce feature dimension and summarize the most activated presence of a feature [14, 20]. At a high level, AgrAmplifier first re-organizes each local updates into individual patches. It then extracts the largest value in each patch which highlights the most present feature in the patch, and returns it as the amplified result. This seemingly simple strategy is effective in achieving the triad. First, since the most activated features are extracted, the local updates become more distinguishable after the amplification, as shown in Fig. 1, e.g., the feature range of [0.4, 0.8] being reduced to [0, 0.1] (the x-axis of Fig. 1(b)). As such, the AGR can make *robust* decision on the maliciousness and benignness of the local updates (detailed in **Section 5.1**). Second, AgrAmplifier enhances the *fidelity* by providing invariance to distortion caused by local translations. Similar to a max pooling layer in typical CNNs, it can ignore the small changes. This makes AgrAmplifier a noise canceller when no attack is



(a) Untargeted Attack (b) Targeted Attack

Figure 1: Gradients projected to 2-D surface using PCA. We plot 50 local updates at the 70th epoch of the training on the LOCATION30 dataset [35]. Red dots (16/50) are malicious updates and green ones are benign. The untargeted attack is implemented via label flipping [4], and the targeted attack is the Scaling attack proposed by Cao et al. [9].

present (detailed in **Section 5.2**). Third, the significant dimension reduction in the feature space can greatly benefit the *efficiency* of AGR. Considering time complexity in terms of the dimension of local updates, AgrAmplifier runs with linear time complexity and shrinks the input size by a constant value depending on the kernel size in a single loop. Most existing AGRs [9, 11, 38] are of super-linear time complexity, which amplifies the benefit obtained from the input size reduction (detailed in **Section 5.3**).

It is worth highlighting that AgrAmplifier's amplification is universally compatible with any of the existing AGRs regardless of the underlying aggregation rules. In this work, we apply AgrAmplifier to three mainstream mechanisms, including distance-based, prediction-based and trust bootstrapping-based aggregators. Our results demonstrate consistent enhancement for all of them. For example, AgrAmplifier-equipped distance-based mechanism outperforms its base version with 67.2% enhancement in robustness, 29.6% in fidelity, and 12.9% in efficiency. For prediction-based mechanisms, AgrAmplifier outperforms its counterpart [11] with 35.6% in robustness, 47.3% in fidelity, and 7.7% in efficiency. AgrAmplifier also boosts the performance of the trust bootstrapping-based mechanisms in the state-of-the-art aggregator [9] with 16.2% in robustness, 19.4% in fidelity and 12.3% in efficiency.

Contributions. We summarize our contributions as follows.

- **A new Byzantine-robust aggregation method.** We propose AgrAmplifier, the first FL aggregation that attains the robustness, fidelity and efficiency triad simultaneously via local updates amplification.
- **An AgrAmplifier-powered AGR.** We design a new distance-based AGR in which we introduce a top-up component of density measurement to supplement the existing Euclidean distance and Cosine similarity mechanisms. After equipped with AgrAmplifier, it demonstrates a balanced performance on all three properties of the triad.
- **Seamless integration with existing AGRs.** AgrAmplifier has consistently demonstrated its merits after being equipped with three mainstream aggregation mechanisms. We further remark that AgrAmplifier is applicable for general algorithmic FL frameworks, not limited to computer vision applications or CNN architecture.
- **An systematic evaluation.** We conduct an extensive experimental evaluation on five benchmark datasets against five poisoning attacks (including untargeted and targeted attacks). The results show the notable enhancement of AgrAmplifier to original Byzantine-robust methods across all experimented datasets of different heterogeneity.

We release the source code of AgrAmplifier and our artifacts to facilitate future research in this area: https://github.com/AgrAmplifier/AgrAmplifier.

2 BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 Federated Learning. Federated learning (FL) is first proposed by McMahan et al. [16, 24]. A typical FL setting [17, 18, 24, 36] consists of a server and multiple participants. We use p_i to denote one participant from $\mathcal{P}$ where $|\mathcal{P}| = N_p$. Each participant holds a local training dataset d_i extracted from a data distribution D. In the communication round t, the server selects $\mathcal{P}_t \subseteq \mathcal{P}$, and broadcasts the global model parameters ω_t to each member in $\mathcal{P}_t$. Each participant then computes the gradient $g_{i,t} = \dfrac{\partial L(\omega_t, d_i)}{\partial \omega_t}$, where the $L(\omega_t, d_i)$ denotes the loss function. In every communication round, the global model takes one step gradient descent using the collected $g_{i,t}$ following: $\omega_{t+1} \leftarrow \omega_t - \alpha_t \sum_{i=1}^{N_p} (|d_i| \cdot g_{i,t})|_{p_i \in \mathcal{P}_t}$, where α_t is the learning rate at communication round t.

2.1.2 Poisoning Attacks. Poisoning attacks aim to achieve malicious purposes through injecting manipulated information during the training [4–6, 8, 15, 31].

Untargeted attacks. Untargeted attacks aim to corrupt the global model's overall capability [7, 16]. To achieve this goal, the adversary can poison the training data of the controlled malicious participants, which is referred to as *data poisoning attack* [7, 23]. *Local model poisoning* is another common strategy in which the adversary manipulates the local updates of the malicious participants [8, 15, 21, 26]. Recent studies further propose optimized and adaptive poisoning attacks, aiming at maximally perturbing the reference aggregate in the malicious direction while evading the detection of Byzantine-robust aggregator [15].

Targeted attacks. Targeted attacks (or backdoor attacks) aims to alter the global model's prediction towards a selected minority of data samples without harming the overall prediction accuracy [16]. Bagdasaryan et al. [4] propose a general framework of backdoor attack. A common approach [9, 12] is to inject malicious updates into a fraction of local training data controlled by malicious participants (e.g., manipulating a particular pixel of the training images) and mislabel the manipulated data to the targeted class. The poison on the targeted labels is then propagated into the global model while the predictions of other classes are not affected.

2.2 Byzantine-robust Defenses

To eliminate the vulnerability caused by poisoning attack, multiple robust aggregation mechanism have been proposed [2, 8, 9, 11, 13, 22, 27, 33, 38]. One common defense strategy is to cross-check the local updates, detect the anomalies and remove them before aggregation. Krum [8] and Bulyan [25] attempt to address the anomaly detection problem by referencing a distance metric (e.g. Euclidean distance or Cosine similarity). Fang [11] and Lomar [22] first apply the collected local updates to the global model and make decision based on the prediction. Another strategy is to make aggregation on each dimension of input gradients separately using statistical feature like mean or median, examples including Trmean and Median [38]. The *state-of-the-art* approach FLTrust [9] solves it based on trust bootstrapping strategy. It assigns a trust score to each participant according to the distance between the local updates and the reference gradients generated from a clean trusted set. Recently, FedInv [40] combines the distance-based analysis with model inversion techniques. FedInv synthesizes a dummy dataset for each participant by inverse engineering the local updates, and then compares the collected updates with the expected updates generated from the dummy dataset, and removes those with largest distance according to the estimated number of malicious participants.

The existing Byzantine-robust mechanisms' performance in terms of *robustness*, *fidelity* and *efficiency* varies in different scenarios. For example, the most commonly used distance-based methods, e.g., Krum [8] and Bulyan [25], may not preserve robustness and fidelity against a large proportion of malicious participants. Fang [11] and Lomar [22] demonstrate strong robustness but they may not be applicable to large-scale federated learning, as they are usually costly at efficiency because of the extra computation of validating the loss of each individual update using the global model parameters. Trust bootstrapping-based methods, e.g., FLTrust [9], are efficient while its fidelity may be suppressed if the trusted set deviates from the original distribution (especially when training with highly heterogenous datasets).

3 AGRAMPLIFIER

3.1 Attack Model

We consider an attack model which is in favor of the poisoning adversary among prior research [4, 6, 9, 27]. We allow the adversary to have the full knowledge of gradients generated by all participants, including those from benign ones. This enables the optimized and adaptive poisoning attacks and demonstrates compelling attack performance in recent studies [9, 27]. The attacker can take the full control of a proportion of clients, including their local updates, local training data and hyper parameters. The attacker can also decide whether to inject malicious updates for the current round or not. We assume the attacker does not compromise the central aggregator, following the literature of AGRs. The attacker can perform targeted attacks or untargeted attacks.

3.2 Defense Model

3.2.1 Defender Capacity. We follow the defense model commonly used in related studies [9, 11, 22]. The robust aggregation rules are employed on the aggregator, who has the knowledge of the local updates from all participants in each iteration and also has the control on the global model.

3.2.2 Desired Properties: the FL Triad. Byzantine-robust FL methods aim to learn an accurate global model against the poisoning attacks. AgrAmplifier particularly aims to simultaneously achieve the following three desired properties defined by Cao et al. [9].
Robustness. The AGR shall minimize the decrease of the global model's test accuracy caused by malicious updates.
Fidelity. The method shall not harm the performance of the global model when there is no malicious updates, and shall achieve a test accuracy close to that of the plain FedAvg [24].
Efficiency. The method shall retain the computation efficiency and scale in large-scale FL training.

3.3 AgrAmplifier Overview

Fig. 2 and Algorithm 1 present AgrAmplifier's amplification process on the collected local updates. First, AGR collects local updates from local participants and re-organizes the collected gradients into patches (with the kernel size $k \times k$). For example, a gradient g_i of an input size d_{in}^l and output size d_{out}^l at l^{th} hidden layer is sliced into l pieces, each with size $(d_{in}^l \times d_{out}^l)$ in correspondence to its layer number l (line 4 in Algorithm 1). Each layer is then divided into patches and processed by a max filter to extract the most activated features, i.e., calculating the maximum value for patches of a feature map (line 5 in Algorithm 1). Depending on the base AGR, AgrAmplifier can choose whether to restore the original size of the collected gradients by filling up the dropped features with 0 (line 6-8 in Algorithm 1). Then, the amplified gradients are concatenated for follow-up cross check (line 10-13).

3.4 Equipping AgrAmplifier

AgrAmplifier is compatible with most existing mainstream mechanisms. In this section, we describe the algorithmic details of equipping AgrAmplifier for Byzantine-robust aggregators. We first categorize existing methods into three main categories and discuss the workflow of AgrAmplifier for each of them.
Distance-based mechanism. This type of aggregators examines the collected gradients and compares them based on the distances measurement (e.g., Euclidean distance and Cosine similarity), and then removes anomalous ones before aggregating them. Representative Byzantine-robust aggregators in this category include Krum [8], Multi-Krum [8], Bulyan [25] and FedInv [40].
Prediction-based mechanism. This mechanism applies the collected gradients to a validation model to check the model's prediction performance in terms of the loss function value (LRR) and the test accuracy (ERR), and then removes those causing performance degradation before sending them to FedAvg to detox the malicious updates. Methods using this mechanism include Fang [11] and LoMar [22].
Trust bootstrapping-based mechanism. This mechanism leverages on a trusted root set. The aggregator retrieves gradients generated from the trusted set. Each participant is given a trust score by comparing its update with gradient generated from trusted set. The trust score then serves as the weight when averaging updates. FLTrust [9] is a typical aggregator in this category.

Fig. 3 shows the overall workflow of AgrAmplifier for Byzantine-robust mechanisms. In general, the collected gradients G are first amplified to G_{amp}, which are then used to calculate the pair-wise distance, LRR and ERR, and trust scores for distance-based, prediction-based and trust bootstrapping mechanisms respectively. For prediction-based mechanisms, the amplified G_{amp} are required to be restored to its original size. After that, distance-based and prediction-based aggregators output a white-list of benign participants. The original G which belong to the white-list, denoted as G_{detox}, are then

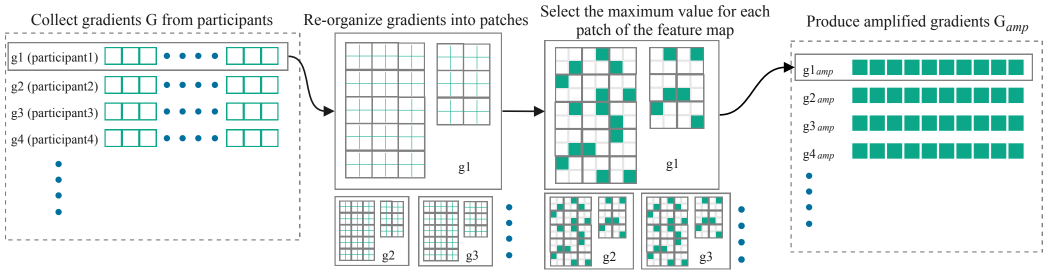

Collect gradients G from participants
g1 (participant1)
g2 (participant2)
g3 (participant3)
g4 (participant4)
Re-organize gradients into patches
g1
g2
g3
Select the maximum value for each patch of the feature map
g1
g2
g3
Produce amplified gradients G_{amp}
$g1_{amp}$
$g2_{amp}$
$g3_{amp}$
$g4_{amp}$

Figure 2: Overview of the amplification process on the collected gradients from FL participants

Algorithm 1 AgrAmplifier's amplification process

```
Input: $L$ - number of layers in the federated learning model; $g_i$ - the collected gradients from participant $i$ $g^{il}$ - the layer $l$ for $g_i$; $d^l_{in}$, $d^l_{out}$ - the input/output size of layer $l$; $N_p$- number of participants; Restore-size - whether restore the amplified gradients to the original size, default to be false here; $p_k$ - the kernel size of the max filter
Output: $G_{amp}$ the amplified gradients collection
 1: function AgrAmplifier($g_1, g_2, g_3, ...$)
 2:     for $i = 1, 2, 3, ..., N_p$ do
 3:         for $l = 1, 2, 3, ..., L$ do
 4:             Re-organize $g^{il}$ to size ($d^l_{in} \times d^l_{out}$)
 5:             $g^{il}_{amp} \leftarrow MaxFilter(g^{il})$
 6:             if Restore-size then
 7:                 Fill the dropped features in $g^{il}_{amp}$ with 0
 8:             end if
 9:         end for
10:         $g^i_{amp} \leftarrow Concatenate(\{g^{il}_{amp} | l = 1, 2, ..., L\})$
11:     end for
12:     $G_{amp} \leftarrow \{g^i_{amp} | i = 1, 2, ..., N_p\}$
13:     return $G_{amp}$
14: end function
15: function MaxFilter($g$)
16:     // We fix stride = kernel size in our implementation
17:     $H_{in} \leftarrow$ the height of $g$
18:     $W_{in} \leftarrow$ the width of $g$
19:     $H_{out} \leftarrow H/p_k$
20:     $W_{out} \leftarrow W/p_k$
21:     $I \leftarrow \emptyset$
22:     for $ho = 1, 2, 3, ...H_{out}$ do
23:         for $wo = 1, 2, 3, ...W_{out}$ do
24:             $g'_{ho,wo} \leftarrow max(\{g_{hi,wi} | hi \in [p_k * ho, pk * (ho + 1)), wi \in [p_k * wo, p_k * (wo + 1))\})$
25:         end for
26:     end for
27:     return $g'$
28: end function
```

fed to FedAvg for the global model computation. For the trust bootstrapping-base mechanisms, G_{detox} are formed by those original G which achieve high trust scores, and then are aggregated into the global model. In the remaining of this section, we give details

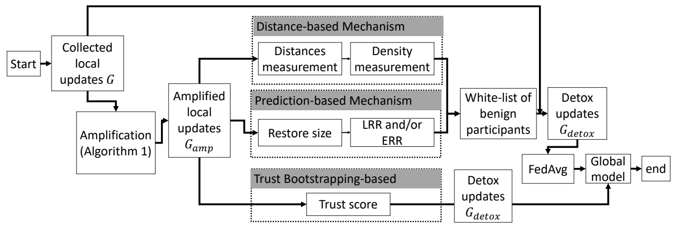

Start
Collected local updates G
Amplification (Algorithm 1)
Amplified local updates G_{amp}
Distance-based Mechanism
Distances measurement
Density measurement
Prediction-based Mechanism
Restore size
LRR and/or ERR
Trust Bootstrapping-based
Trust score
White-list of benign participants
Detox updates G_{detox}
Detox updates G_{detox}
FedAvg
Global model
end

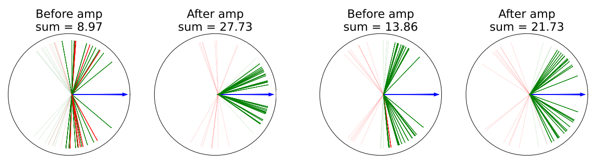
Figure 3: Workflow of AgrAmplifier for Byzantine-robust mechanisms

of AgrAmplifier for distance-based, prediction-based and trust bootstrapping-based mechanism respectively.

3.4.1 AgrAmplifier for Distance-based Mechanism. The process of AgrAmplifier for distance-based mechanism is shown in the upper part in Fig. 3. The original gradients are first processed through the amplifier, and the amplified gradients are into any distance measurements, e.g., Cosine similarity and Euclidean distance (the left box in the upper part in Fig. 3).

Since the traditional distance-based mechanisms mainly focus on anomaly detection and are less effective with the increasing proportion of malicious nodes as the anomalies become less obvious, we therefore additionally design a top-up component of *density measurement* (the right box in the upper part in Fig. 3) to supplement the traditional distance-based mechanisms. Our design is based on the recent findings that the malicious gradients usually distribute sparsely while benign ones are denser [22]. As such, we propose the density measurement, in which we examine the density of each gradient's neighbourhood which consists of k neighbours (k should be $> N_p/2$ where N_p is the number of participants), and regards those gradients residing in a denser neighbourhood as benign ones while remove those with a sparse neighbourhood. In particular, for each participant's update, we select its k-nearest neighbours based on the Cosine similarity score, and sum up the scores in its neighbourhood. Then, the Top-N participants with higher scores will be added into the white list.

AgrAmplifier can make such mechanism more effective as shown in Fig. 4. In both targeted and untargeted attack, the number of malicious participants within the neighbourhood of a benign participant reduces after amplification (i.e., from 8 to 0 for the untargeted attack, and from 1 to 0 for the targeted attack). In addition, the summed Cosine similarity in the neighbourhood significantly



(a) Untargeted attack **(b) Targeted attack**

Figure 4: The k-nearest neighbors to a benign participant (the blue arrow) according to Cosine similarity. Red arrows are malicious participants and green arrows are benign ones belonging to the neighbourhood. The faded ones are not within the k-nearest neighbors. We set $k = 30$. There are overall 50 participants, with 15 of them are malicious. The sum of the Cosine similarity in the neighbourhood is shown in top of each figure. The larger the value, the higher the probability that the AGR will classify the participant as benign.

increases after amplification, from 8.97 to 27.73 for the untargeted attack, and from 13.86 to 21.73 for the targeted attack. The detailed algorithm of AgrAmplifier-powered density-based AGR is given in Appendix A.1.

3.4.2 *AgrAmplifier for Prediction-based Mechanism.*

The middle part of Fig. 3 illustrates the process of AgrAmplifier for prediction-based mechanism. AgrAmplifier first conducts the amplification on the collected gradients. AgrAmplifier then restores the size of the original collected gradients. The restored gradients are then sent to LRR and ERR [11] calculation, and those giving better prediction performance will be added into the white list. The detailed algorithm can be found in Appendix A.2.

3.4.3 *AgrAmplifier for Trust Bootstrapping-based Mechanism.*

In the trust bootstrapping-based mechanism, the server itself collects a small clean root dataset and maintains a trusted validation model. As such, in addition to the amplification on the collected gradients, AgrAmplifier also applies the amplification on the gradient generated from the trusted validation set for the trust bootstrapping. The trust score for each participant is then calculated based on ReLU-clipped Cosine similarity of the amplified gradients. Then the magnitudes of the collected local model update is normalized. The normalized gradients are then weighted by their trust scores and aggregated for the global model. The detailed algorithm can be found in Appendix A.3.

4 EXPERIMENTAL SETTINGS

In this section, we investigate the performance of AgrAmplifier through a serials of experiments. Our experiments aim to answer the following three research questions (RQs).

RQ1. What is the performance of AgrAmplifier in general? How does it improve the existing Byzantine-robust mechanisms against the state-of-the-art poisoning attacks in terms of robustness, fidelity and efficiency?

RQ2. How is the performance of AgrAmplifier affected by the varying number of participants and different proportions of malicious nodes?

RQ3. How is the performance of AgrAmplifier affected by the different settings of the hyper-parameters of the kernel size?

4.1 Datasets

We use five real-world datasets for our evaluation. They are commonly used in AGR literature.

CIFAR-10 [19]. The dataset consists of 60,000 32×32 colour images in 10 classes, with 6,000 images per class. We conduct a pre-training using ResNet56 [14]. The experiments are conducted on the last fully connected layer (FCN) with size $\{64, 1024, 10\}$. This dataset is independent and identically distributed (IID) to each participant.

MNIST [37]. The dataset consists of 60,000 28×28 gray-scale images of handwriting digits in 10 classes (digit 0-9). We use a FCN with size $\{784, 512, 10\}$ for our experiments. This dataset is IID.

LOCATION30 [35]. The dataset consists of publicly available set of mobile users' location "check-ins" in the Foursquare social network compiled by Shokri et al. [30]. The dataset contains 30 classes and 5,010 data samples, each with 446 binary features. We use an FCN with size $\{446, 512, 30\}$ for our experiments. This dataset is non-IID, where count of samples in each label varies from 97 to 308.

PURCHASE100 [30]. The dataset contains the shopping histories of different individuals originated from Kaggle's *acquire valued shopper* challenge. We use the processed set provided by Shorkri et al. [30], which contains 197,324 records with 600 binary features, and classed into 100 different classes. We use a FCN with size $\{600, 1024, 100\}$ for our experiments. This dataset is non-IID, where count of samples in each label varies from 106 to 5214.

TEXAS100 [30]. The dataset contains data of hospital stays released by the Texas Department of State Health Services. We use the processed set provided by Shokri et al. [30], which contains 67,330 records with 6,169 binary features, and classed into 100 different classes. We use a FCN with size $\{6169, 1024, 100\}$ for our experiments. This dataset is non-IID, where count of samples in each label varies from 236 to 3046.

4.2 Evaluated Attacks

We consider five representative untargeted/targeted attacks.

4.2.1 *Untargeted Attacks.*

We evaluate a range of untargeted attacks from the traditional data poisoning [9, 11, 22], model poisoning [21, 26], to the most recent optimized and adaptive attacks [27]. We detail our evaluated attacks as follows.

- **Data poisoning via label-flipping (`L-flip`).** Data poisoning via label-flipping is a widely referenced poisoning attack [9, 11, 22], where the adversary manipulates the local training data, intentionally switching the label for training data.
- **Model poisoning via gradient manipulation (`G-asc`).** The attack is to manipulate the gradient towards the adversarial direction. Prior research shows that manipulating the gradients can effectively degenerate the performance of global model [21, 26].
- **`<L-flip>+<G-asc>`.** To consider a strong attack that can apply both label flipping and gradient manipulation at a time, we propose `<L-flip>+<G-asc>` which refers to the merge of `L-flip` and `G-asc` by summing up the malicious updates from label flipping and gradient manipulation.

- **Optimized and adaptive attack (`S&H Attack`) [27].** This refers to the state-of-the-art Byzantine robust aggregation-tailored attack proposed by Shejwalkar and Houmansadr [27], in which the attack is formalized as an optimization problem aiming at maximally perturbing the reference aggregate in the malicious direction, while being adaptive to evade the detection of AGR. It is reported to outperform LIE [5] and Fang [11] on majority of the experiments with datasets CIFAR-10, PURCHASE100, MNIST, and FEMNIST.

4.2.2 Targeted Attacks. We consider a state-of-the-art targeted attack called scaling attack (**`T-Scal`**), proposed by Cao et al. [9]. The attacker first chooses a specific targeted class, adds triggers to its normal local training samples with data augmentation scheme in [12], and labels them with the targeted class. Due to the limited impact of a single malicious participant, the attacker scales up the poisoned local gradients by a factor larger than 1. In our experiments, the factor is set as $1/M_f$ (recall M_f is the fraction of malicious participants), which retains high attack accuracy even in the presence of some existing Byzantine-robust aggregation mechanisms [8, 9, 38].

4.3 Evaluated Defenses

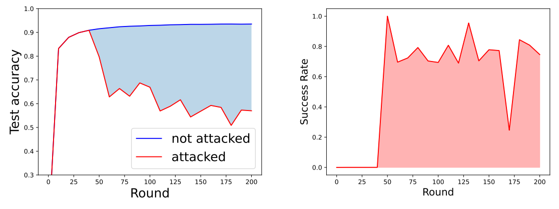
We evaluate our AgrAmplifier for distance-based, prediction-based and trust bootstrapping-based mechanisms respectively. In each category, we take the state-of-the-art base aggregators as the baseline, and compare with the AgrAmplifier equipped version. We detail each of them in the following.

4.3.1 Distance-based Mechanisms.

- **Cosine similarity / Euclidean distance / Merged distance combined with the Density measurement (`CosDen` / `EuDen` / `MgDen`).** This refers to the proposed density-based mechanisms which are evaluated on the original gradients G without amplification. `CosDen` and `EuDen` denote the mechanisms which combine Cosine similarity or Euclidean distance with the Density measurement. `MgDen` denotes the merged use of `CosDen` and `EuDen` which takes an intersection of white-lists from them.
- **`CosDen` / `EuDen` / `MgDen` with AgrAmplifier (`COSDEN_Amp+` / `EUDEN_Amp+` / `MGDEN_Amp+`).** This refers to AgrAmplifier for `CosDen`, `EuDen` and `MgDen` in which the measurements are evaluated on the amplified gradients G_{amp}.

4.3.2 Prediction-based Mechanisms.

- **Fang's defense (`Fang`).** We take Fang's defense [11] (which is one of the state-of-the-art Byzantine-robust aggregators) as the baseline of the prediction-based mechanism. Fang's defense has three variations, i.e., LRR (which is based on the loss value), ERR (which is based on the test accuracy), and the merged algorithm which is reported to have the best performance combining LRR and ERR. We use the merged algorithm of Fang's defense in our experiments.
- **Fang's defense with AgrAmplifier (`Fang_Amp+`).** This refers to AgrAmplifier-equipped Fang's defense, in which the merged LRR and ERR is computed from the amplified gradients.



(a) Untargeted attack **(b) Targeted attack**

Figure 5: Illustration of the evaluation metrics

4.3.3 Trust Bootstrapping-based Mechanisms.

- **`FLTrust`**. FLTrust is the state-of-the-art Byzantine-robust aggregators based on trust bootstrapping [11]. To reach better performance, the size of trust set is recommended to be larger than 300 according to its authors. In our experiments, we set the trust set size as 2000 for CIFAR-10, 2000 for MNIST, 300 for LOCATION30, 2000 for TEXAS100.
- **`FLTrust` with AgrAmplifier (`FLTrust_Amp+`).** This refers to AgrAmplifier-equipped FLTrust, in which the trust score is computed based on the amplified gradients.

4.4 Evaluation Metrics

We observe that the run-time performance of FL tends to fluctuate under attacks. Therefore, in addition to the final test accuracy and attack success rate which might be affected by the fluctuation (which can be huge sometimes as shown in Fig. 5 at round 175), we also evaluate the performance from the aspect of averaged test accuracy and attack success rate over a monitoring period. Such metrics also alleviate the impact of irrelevant factors on the performance evaluation, such as the effect of overfitting to the final performance.

4.4.1 Untargeted Attacks. In accordance to attacker's goal, we evaluate the loss of global model's performance using averaged test accuracy loss comparing to a non-attacked training process. We assume that there are overall T_1 rounds of communication in the FL training process and the attacker starts attack at T_0, and denote the non-attacked test accuracy at round t as a_t and the attacked test accuracy at round t as $\hat{a}_t$. The averaged test accuracy loss $\mathcal{L}$ can be written as

$$\mathcal{L} = \frac{1}{T_1 - T_0} \sum_{t=T_0}^{T_1} a_t - \hat{a}_t \tag{1}$$

Intuitively, $\mathcal{L}$ can be illustrated as the blue shaded area (then divided by the round number) in Fig. 5(a) which is generated on PURCHASE100 with `<L-flip>+<G-asc>` attack without any Byzantine-robust aggregator employed.

To set T_0 and T_1, we observe that most models in our experiments reach more than 95% of their peak performance after round 50, and start to overfit after round 200. We therefore fix $T_0 = 50$ and $T_1 = 200$, and the test accuracy is recorded every 10 rounds during the training process.

4.4.2 Targeted Attacks. We examine the attack success rate by counting the test data with the trigger in other classes being classified to the target class by the global model. Denote the attack success rate at round t as s_t, global model at round t as ω_t, and the test set with trigger added as $\hat{D}$ where $\hat{D}$ does not contain any data samples which is originally the targeted class C. We average s_t over communication rounds to examine the attack performance for the training process. Specifically,

$$s_t = \frac{||\{\hat{d}|_{\hat{d} \in \hat{D}, \omega_t(\hat{d})=C}\}||}{||\hat{D}||} \tag{2}$$

$$\mathcal{A} = \frac{1}{T_1 - T_0} \sum_{t=T_0}^{T_1} s_t \tag{3}$$

We illustrate $\mathcal{A}$ as the red shaded area (then divided by the round number) in Fig. 5(b) which is generated on PURCHASE100 with `T-Scal` attack.

5 PERFORMANCE OF AGRAMPLIFIER (RQ1)

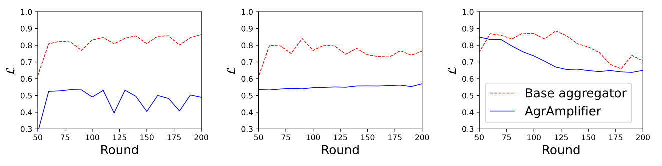
In this section, we discuss the performance of AgrAmplifier against the state-of-the-art poisoning attacks in terms of *robustness*, *fidelity* and *efficiency*. We set 50 participants in this set of experiments, where 30% of them are malicious, exceeding the commonly used 20% setting in the existing literature [9, 11, 15] to give attacker more advantages.

In general, AgrAmplifier benefits Byzantine-robust mechanisms on all three properties and outperforms the existing state-of-the-art methods, with `Fang_Amp+` and `COSDEN_Amp+` performing the best in terms of *robustness* (i.e., `Fang_Amp+` performs the best against targeted attack and `COSDEN_Amp+` performs the best against non-targeted attack). In terms of *fidelity*, `COSDEN_Amp+` performs the best, closely followed by `Fang_Amp+`. In terms of *efficiency*, `FLTrust_Amp+` performs the best with small to medium network size and `COSDEN_Amp+` performs the best with the large network. We recommend `COSDEN_Amp+` as it achieves a desirable trade-off among the three. It performs the best in term of *fidelity*, and the joint best in *robustness* and *efficiency*. We now analyze the performance of AgrAmplifier in detail.

5.1 Robustness Boosting

AgrAmplifier shows its capability to boost the robustness for both untargeted attacks and targeted attacks. We discuss them as follows.

5.1.1 Untargeted Attacks. Overall, AgrAmplifier demonstrates advantages comparing to the base aggregator group without AgrAmplifier. As shown in Fig. 6, in which we average the performance of the aggregators across all datasets in each category against the untargeted attacks, AgrAmplifier-equipped aggregators achieves less performance loss. It outperforms its counterpart base aggregator with 67.2%, 35.6% and 16.2% for distance-based, prediction-based and trust bootstrapping-based mechanism respectively. The detailed experimental results are shown in Table 1. The most robust aggregator against untargeted attack is `COSDEN_Amp+`, achieving the best performance in 7 out of 20 experiment settings (4 attacks × 5 datasets).



(a) Distance-based **(b) Prediction-based** **(c) Trust bootstrapping-based**

Figure 6: Performance of run-time performance loss $\mathcal{L}$ of Byzantine-robust aggregators against untargeted attacks.

Table 1: Averaged test accuracy loss $\mathcal{L}$ in untargeted attacks presented in percentage. The best defense for each attack (row-wise) is in bold.[†]

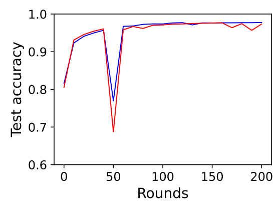
Attack	No def	Distance						Prediction		Trust	
		C	E	M	C+	E+	M+	F	F+	T	T+
CIFAR-10											
L-flip	2.07	3.49	0.32	1.35	9.27	0.75	0.98	0.85	**-0.12**	1.73	2.58
G-asc	1.12	0.71	1.7	0.23	**-0.03**	-0.02	0.07	0.07	0.38	1.44	1.56
S&H	0.12	0.07	-0.09	0.23	-0.1	0.12	**-0.18**	-0.11	0.11	1.96	1.45
L+G	2.87	5.88	2.95	1.7	**0.36**	4.02	0.41	0.5	0.51	2.03	1.82
LOCATION30											
L-flip	4.03	0.82	**-0.51**	3.88	1.96	0.54	1.88	4.48	0.9	3.94	1.31
G-asc	8.48	16.95	**0.5**	3.77	5.96	1.89	3.14	3.4	4.28	5.74	8.67
S&H	0.69	-0.24	1.31	2.76	**-0.65**	0.02	0.2	2.4	1.35	4.26	0.5
L+G	6.5	3.17	11.34	9.37	**-0.45**	6.44	3.23	1.33	1.86	2.74	8.36
MNIST											
L-flip	8.42	2.12	0.35	0.9	9.04	**0.23**	0.85	0.68	0.66	2.31	4.47
G-asc	1.02	1.33	0.24	0.27	0.23	**0.01**	0.14	0.02	0.21	5.63	2.27
S&H	0.41	0.4	0.41	0.3	**0.11**	0.29	0.36	0.25	0.14	1.7	2.18
L+G	0.73	2.63	85.86	2.23	1.39	0.38	0.16	0.2	**-0.08**	5.6	2.35
PURCHASE100											
L-flip	11.9	22.64	1.05	1.6	8.65	0.97	0.79	1.73	**0.43**	1.4	2.55
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											
L-flip	3.85	7.76	2.01	4.18	5.32	2.6	1.56	3.52	**1.07**	2.79	1.25
G-asc	1.98	1.39	1.17	2.64	1.17	1.58	1.49	2.0	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	5.2	**1.25**	4.0	2.79	2.4	1.96	2.2	1.39

[†] In this table, "No def" stands for "No defense". We also use `C` to stand for `CosDen`, `C+` for `CosDen_Amp+`, `E` for `EuDen`, `E+` for `EuDen_Amp+`, `M` for `MergeDen`, `M+` for `MergeDen_Amp+`; `F` for `Fang`, `F+` for `Fang_Amp+`; `T` for `FLTrust`, `T+` for `FLTrust_Amp+`. We use `L+G` for the merged attack `<L-flip>+<G-asc>`.

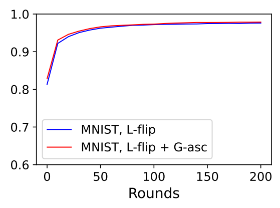
We also examine the effectiveness of the defense against each attack by ranking the defense mechanisms by $\mathcal{L}$ across datasets. For `L-flip` attack and `G-asc` attack, `Fang_Amp+` outperforms the rest. `COSDEN_Amp+` performs the best against stronger attacks including `<L-flip>+<G-asc>` and `S&H Attack`.

We further report several observations in the following.

Negative pulse mitigation. In the existing Byzantine-robust FLs, we observe a phenomenon of "negative pulse", i.e., an abrupt reduction of model's test accuracy caused by the poisoning adversary (usually) at the early stage of the attacks as shown in Fig. 7(a). Such lag in the onset time of the defense is repeatedly observed across different datasets with the existing base aggregators. Notably, our AgrAmplifier can effectively mitigate such "negative pulse"

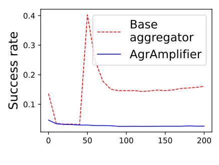


(a) CosDen

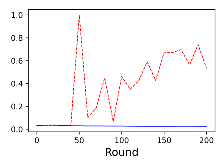


(b) CosDen_Amp+

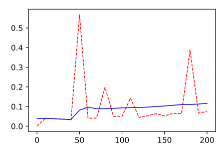
Figure 7: Run-time test accuracy. Left: Negative pulse at round 50 (the attack start round). Right: AgrAmplifier mitigates the negative pulse.



(a) Distance-based



(b) Prediction-based



(c) Trust bootstrapping-based

Figure 8: Run-time success rate of targeted attack `T-Scal` on CIFAR-10.

as shown in Fig. 7(b), as the maliciousness / benignness of local participants becomes more distinguishable.

Base mechanism matters. The performance of the base aggregator determines the performance of the AgrAmplifier-equipped version. For example, in the case that the base aggregators is weak in one specific setting, this weakness can be passed to the amplified version. For example, distance-based `CosDen` reports $\mathcal{L}$ at 16.95% on LOCATION30 against `L-flip` attack. Though `COSDEN_Amp+` achieves $\mathcal{L}$ of 5.96% in the same setting, which is significantly better than its original base aggregator, it still performs worse than the remaining defenses, which achieve averaged $\mathcal{L}$ at 0.28%. Similar observation can be found with trust bootstrapping-based mechanisms (i.e., `FLTrust` and `FLTrust_Amp+`) on LOCATION30.

5.1.2 Targeted Attacks. Fig. 8 illustrates the performance of the targeted attack `T-Scal` against base aggregators and AgrAmplifier on CIFAR-10, in which AgrAmplifier outperforms its counterpart base aggregator with 14.7%, 46.9% and 2.3% for distance-based, prediction-based and trust bootstrapping-based mechanism respectively. In addition, attack success rate pulses (which are similar to the negative pulse in untargeted attack) are observed from all three original defense (round 50 in Fig. 8). Likewise, such phenomenon is also eliminated by AgrAmplifier.

We further examine the performance from the perspective of worst case promise following the original work that proposes `T-Scal` attack, as the success rate of `T-scal` varies across datasets, better performance in terms of worst case ensures the robustness on arbitrary datasets. Table 2 shows the biggest attack gain against each defense across the datasets, where `Fang_Amp+` performs the best with its consistently strong defense across all datasets, followed by `COSDEN_Amp+` which successfully defend against `T-Scal` on MNIST, CIFAR-10, PURCHASE100 and TEXAS100 while is defeated by `Fang_Amp+` on LOCATION30. We however notice an abnormal performance of `FLTrust_Amp+` on TEXAS100. While most of other defenders perform relatively well, both `FLTrust` and `FLTrust_Amp+` fail to provide effective defense in this case. This is because the performance of the trust bootstrapping-based mechanism can largely rely on the representativeness of the trusted set. For dataset with high heterogeneity such as TEXAS100 (detailed in Section 7.1), the distribution of the trusted set may deviate from the original distribution of the targeted class. In such case, the mechanism will possibly mislead the defense to a wrong direction, and AgrAmplifier will further amplify such misleadingness of the original aggregator.

Table 2: Performance of `T-Scal` in terms of the accumulated success rate $\mathcal{A}$ in percentage. The worst-case (i.e., the biggest attack gain) against each defense across the datasets (column-wise) is in bold.[†]

Dataset	No def	Distance		Prediction		Trust	
		C	C+	F	F+	T	T+
CIFAR-10	94.44	**17.35**	2.64	**49.60**	**2.70**	12.25	9.91
LOCATION30	53.45	3.13	**8.46**	5.54	1.39	3.59	4.68
MNIST	16.58	0.37	0.57	0.22	0.24	0.15	0.35
PURCHASE100	74.78	0.56	1.73	0.05	1.02	0.41	1.11
TEXAS100	99.03	11.16	0.06	0.01	0.03	**35.59**	**63.76**

[†] In this table, we use `C` to stand for `CosDen`, `C+` for `CosDen_Amp+`, `F` for `Fang`, `F+` for `Fang_Amp+`; `T` for `FLTrust`, `T+` for `FLTrust_Amp+`.

Table 3: Performance loss $\mathcal{L}$ in terms of fidelity (%).[†]

Dataset	Error Rate	Distance						Prediction		Trust	
		C	E	M	C+	E+	M+	F	F+	T	T+
CIFAR-10	24.51%	-0.07	0.50	0.68	0.08	0.01	0.24	0.15	0.23	1.48	1.38
LOCATION30	28.96%	0.00	2.84	3.59	-0.15	1.35	1.69	3.64	1.29	3.21	1.87
MNIST	2.22%	0.41	0.74	0.62	0.48	0.72	0.88	0.64	0.54	0.62	2.39
PURCHASE100	7.12%	0.53	1.23	1.69	0.28	1.48	1.29	0.69	0.94	3.17	1.98
TEXAS100	35.28%	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`, `C+` for `CosDen_Amp+`, `E` for `EuDen`, `E+` for `EuDen_Amp+`, `M` for `MergeDen`, `M+` for `MergeDen_Amp+`; `F` for `Fang`, `F+` for `Fang_Amp+`; `T` for `FLTrust`, `T+` for `FLTrust_Amp+`.

5.2 Fidelity Boosting

The fidelity indicates the aggregator's capability of preserving useful information. Fig. 9 illustrates the performance of AgrAmplifier and the base aggregators in terms of fidelity (with detailed results given in Table 3). Most of mechanisms benefit significantly from AgrAmplifier, as the amplified version (orange bar) shows far less performance loss than the original aggregators (blue one).

Among all Byzantine-robust aggregators, distance-based `COSDEN_Amp+` performs the best with $\mathcal{L} = 0.18\%$ which is close to fully achieve the fidelity goal. Prediction-based `Fang_Amp+` also achieves desirable fidelity with averaged loss $\mathcal{L} = 0.68\%$. Trust bootstrapping-based mechanism's performance differs across different datasets. For example, it introduces a higher fidelity loss on LOCATION30 and PURCHASE100, since the trust set may have a different distribution than the original training set in these cases which may introduce bias to the global model.

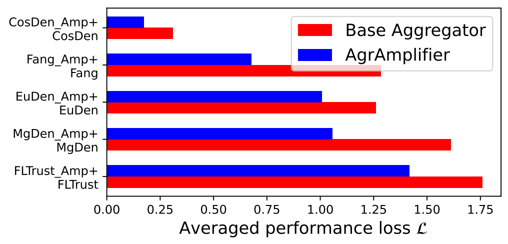


Figure 9: Averaged performance loss $\mathcal{L}$ across all datasets in terms of fidelity, i.e., FL is equipped with Byzantine-robust aggregators while there is no malicious updates.

Table 4: Time consumption (in second) of aggregation on LOCATION30.[†]

Hidden Layer	C	C+	F	F+	T	T+
512	3110.7	3038.6	3916.8	3822.6	3334.1	3283.6
1024	6722.9	6479.4	8612.4	7588.0	6672.6	6339.5
2048	12984.6	10367.1	17039.5	15883.0	13610.2	11084.4

[†] In this table, we use C to stand for CosDen, C+ for CosDen_Amp+, F for Fang, F+ for Fang_Amp+; T for FLTrust, T+ for FLTrust_Amp+

5.3 Efficiency Boosting

Table 4 shows the time consumption for all untargeted experiments on LOCATION30. The experiments are run on a low spec physical device with an Intel Core i7-2670QM CPU and 8GB RAM. In general, AgrAmplifier achieves less time consumption as it shrinks the data size and reduces the amount of calculation. The difference is more obvious when the neural network architecture is larger, which benefits Byzantine-robust mechanisms towards learning on large/deep models. In average, AgrAmplifier reduces total time consumption from 76003.8 seconds to 67886.2 seconds, showing 10.68% improvement.

Among the three mechanisms, the trust bootstrapping-based mechanism (`FLTrust_Amp+`) and distance-based mechanism (`CosDen_Amp+`) are close in efficiency (distance-based mechanism performs slightly better with larger network size), while the the prediction-based approach performs the worst. This is because the prediction-based approach needs extra calculation on the loss/error for each individual update collected. Instead, the trust bootstrapping-based mechanism only conducts the validation on a small trust set, resulting in a desirable performance. The distance-based mechanism does not require validation on any updates at all. However, it does require pair-wise cosine similarity computation while trust bootstrapping based mechanism only does one-to-all.

6 INFLUENCING FACTORS OF AGRAMPLIFIER (RQ2 & RQ3)

6.1 Impact of Varying Number of Participant and Proportion of Malicious Nodes (RQ2)

In this section, we assess the performance of AgrAmplifier with varying number of participants (RQ2.1) and malicious factor (RQ2.2). We fix the attack as `<L-flip>+<G-asc>` in this section. In RQ2.1, we set up the number of participants as 10, 50, 100, 500, 1000 respectively where the percentage of malicious participants fixed on 30%. In RQ2.2, we fix the number of participants as 100, and set up five different proportion of malicious participants as 10%, 20%, 30%, 40%, 50%. Overall, `FLTrust_Amp+` has the best tolerance across various FL environments followed by `Fang_Amp+`.

6.1.1 RQ2.1: Impact of Participant Number. Fig. 10 illustrates the run-time test accuracy of AgrAmplifier on PURCHASE100 with different number of participants. In general, AgrAmplifier outperforms its counterpart base aggregator and approaches the performance of FedAvg (under no attacks) with the number of participants from 50 to 1000. We observe that `FLTrust_Amp+` can defend against the attack for all evaluated number of participants, and it is the only effective Byzantine robust aggregator in the setting of 10 participant, demonstrating the best tolerance across various FL settings due to the amplified advantage of the trust set. The curves of AgrAmplifier are also smoother than the base aggregators in most cases and make model converge earlier. We also observe a reduction of test accuracy in FedAvg under no attacks with large number of participants (i.e., 1000 participants), leading to the reduction of the overall performance of all considered aggregators.

6.1.2 RQ2.2: Impact of Malicious Client Proportion (M_f). In this section we investigate the impact of different ratio of malicious participants. Fig. 11 shows the the run-time test accuracy of AgrAmplifier on PURCHASE100 with M_f ranging from 0.1 to 0.5. As shown in Fig. 11(a), the performance of FegAvg (without defense) significantly drops as the increasing proportion of malicious participants. In terms of AgrAmplifier's performance, both `Fang_Amp+` and `FLTrust_Amp+` have stable high test accuracy even though M_f reaches 50%, which are more robust than the `COSDEN_Amp+`, while for the M_f less than 50%, `COSDEN_Amp+` has a similar performance as the other two aggregators.

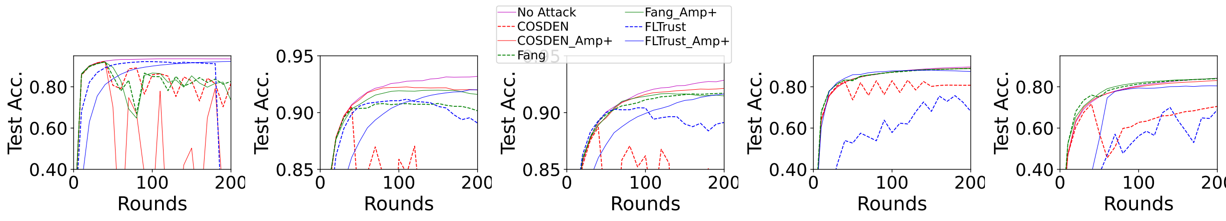
6.2 Impact of the Kernel Size (RQ3)

In this section, we investigate the impact of the kernel size on the performance of AgrAmplifier. In this set of experiment, we examine the kernel size in range of 2×2, 3×3, 5×5, 7×7, 9×9 respectively. In general, as shown in Fig. 12, the parameter does not affect the performance of distance-based `COSDEN_Amp+` and prediction-based `Fang_Amp+`, while it has slightly impact on trust Bootstrapping-based `FLTrust_Amp+`. Specifically, `FLTrust_Amp+` shows a slight drop with large kernel size (e.g., 7×7 and 9×9). Compared to the other two, trust bootstrapping-based mechanism computes the reference updates on a small trust set, which may be affected by the non-trivial information loss introduced by large kernel size. More results on the other datasets are given in Appendix B.

7 DISCUSSION AND LIMITATIONS

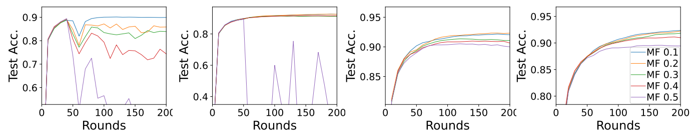
7.1 Effect of Dataset's Heterogeneity

We observe that the performance of the three types of Byzantine-robust mechanisms varies across different datasets. As demonstrated in the prior research [3, 39], the generalization bound of machine learning is theoretically proved to be related to the heterogeneity of the training data. Specifically, the model's generalization



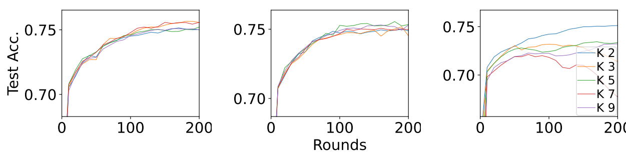
(a) 10 Participants (b) 50 Participants (c) 100 Participants (d) 500 Participants (e) 1000 Participants

Figure 10: Run-time test accuracy of FL with AgrAmplifier against varying number of participants on PURCHASE100



(a) No Defense (b) `COSDEN_Amp+` (c) `Fang_Amp+` (d) `FLTrust_Amp+`

Figure 11: Run-time test accuracy with different proportion of malicious participants on PURCHASE100



(a) `COSDEN_Amp+` (b) `Fang_Amp+` (c) `FLTrust_Amp+`

Figure 12: The impact of kernel size on CIFAR-10

ability decreases when training on the datasets with higher heterogeneity [3]. The poisoning attacks gain more advantages as the generalization ability decreases, since it increases the attacker's capacity in raising the loss on poisoned examples. On the other hand, a well-generalized model is more capable in handling diverse input and thus is more robust to the malicious injection.

Our empirical results also demonstrate such link between the datasets' heterogeneity and the defense performance. We calculate the datasets' heterogeneity based on the average intra-label cosine similarity according to [3]. Denote the training set with label ξ as D_ξ, and the number of classes in the dataset as E, the intra-label cosine similarity is calculated as

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

Table 5 shows the heterogeneity of each dataset. We find that the datasets with lower heterogeneity (larger intra-label cosine similarity) provides advantages for defenders, especially against `T-scal` attack. For example, the `T-scal` attack on PURCHASE100 and LOCATION30 has the worst success rate (less than 10%) when the defensive mechanisms are applied. In addition, the performance of trust bootstrapping-based mechanisms is greatly affected if the heterogeneity of the training data is extremely high (e.g., TEXAS100 with the lowest intra-label cosine similarity).

Table 5: Heterogeneity of the datasets.

MINST	CIFAR-10	LOCATION30	PURCHASE100	TEXAS100
0.85	0.76	0.88	0.97	0.30

7.2 Limitations Inherited from the Base Mechanisms

We acknowledge that AgrAmplifier may not conquer some inherent limitations of the base mechanisms. For example, AgrAmplifier may also fail to work for the trust bootstrapping-based mechanisms if the trusted set itself is poisoned [9]. For some base aggregation rules (e.g., most of distance-based aggregators) which rely on knowledge of malicious fraction M_f, AgrAmplifier also adopts such assumption, whereas the aggregator might not be able to acquire such information in real practices.

8 CONCLUSION AND FUTURE WORKS

This work presents a mechanism named AgrAmplifier to attain the robustness, fidelity and efficiency triad for FL methods against the Byzantine adversary. AgrAmplifier gains 57.47% in average lessening the attack loss. It also retains high fidelity and efficiency in all scenarios. Furthermore, we construct a new approach `COSDEN_Amp+` and its variance `EUDEN_Amp+` and `MGDEN_Amp+` based on the density measurement. AgrAmplifier outperforms the state-of-the-art Byzantine-robust aggregator, and demonstrates the best balanced performance among all desired properties.

A number of future work directions are of interest. In particular, we will investigate the direction towards the theoretically guaranteed defense and convergence provided by the local update amplification mechanism. We also see new research opportunities in leveraging the rich toolbox from explainable artificial intelligence for a better understanding and identification of the significant features in participants' updates, which can potentially improve the distinguishability of malicious updates from the benign ones. We are also considering other directions of defense mechanisms. For example, frequency-domain analysis can be helpful for identifying malicious updates, and style-transformation can possibly turn a malicious update into a benign one.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. 2018. Byzantine stochastic gradient descent. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 4618–4628.

[3] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. 2019. Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks. (2019).

[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[5] Moran Baruch, Gilad Baruch, and Yoav Goldberg. 2019. A Little Is Enough: Circumventing Defenses For Distributed Learning. *http://arxiv.org/licenses/nonexclusive-distrib/1.0* (2019).

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

[7] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[8] 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*. 118–128.

[9] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2020. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. *arXiv preprint arXiv:2012.13995* (2020).

[10] Pierre Courtiol, Charles Maussion, Matahi Moarii, Elodie Pronier, Samuel Pilcer, Meriem Sefta, Pierre Manceron, Sylvain Toldo, Mikhail Zaslavskiy, Nolwenn Le Stang, et al. 2019. Deep learning-based classification of mesothelioma improves prediction of patient outcome. *Nature medicine* 25, 10 (2019), 1519–1525.

[11] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 1605–1622.

[12] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *Machine Learning and Computer Security Workshop* (2017).

[13] Rachid Guerraoui, Sébastien Rouault, et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*. PMLR, 3521–3530.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2016-. IEEE, 770–778.

[15] 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 *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 19–35.

[16] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[17] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[18] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

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

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[21] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial Machine Learning at Scale. *http://arxiv.org/licenses/nonexclusive-distrib/1.0* (2016).

[22] Xingyu Li, Zhe Qu, Shangqing Zhao, Bo Tang, Zhuo Lu, and Yao Liu. 2021. LoMar: A Local Defense Against Poisoning Attack on Federated Learning. *IEEE Transactions on Dependable and Secure Computing* (2021).

[23] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2017. Trojaning attack on neural networks. (2017).

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[25] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. (2018).

[26] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil Lupu, and Fabio Roli. 2017. Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization. In *Proceedings of the 10th ACM Workshop on artificial intelligence and security (AISec '17)*. ACM.

[27] Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning. *Internet Society* (2021), 18.

[28] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage. 2022. Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning. In *2022 2022 IEEE Symposium on Security and Privacy (SP)*. 1117–1134.

[29] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 1310–1321.

[30] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[31] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. 2019. Can You Really Backdoor Federated Learning? (2019).

[32] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data Poisoning Attacks Against Federated Learning Systems. In *ESORICS 2020*. 480–501.

[33] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2018. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116* (2018).

[34] Kouichi Yamaguchi, Kenji Sakamoto, Toshio Akabane, and Yoshiji Fujimoto. 1990. A neural network for speaker-independent isolated word recognition. In *Proc. First International Conference on Spoken Language Processing (ICSLP 1990)*. 1077–1080.

[35] Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2016. Participatory cultural mapping based on collective behavior data in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 3 (2016), 1–23.

[36] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

[37] LeCun Yann, Cortes Corinna, and Christopher J. Burges. 1998. Mnist handwritten digit database. (1998). Available: http://yann.lecun.com/exdb/mnist.

[38] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. PMLR, 5650–5659.

[39] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. (2016).

[40] Bo Zhao, Peng Sun, Tao Wang, and Keyu Jiang. 2022. FedInv: Byzantine-robust Federated Learning by Inversing Local Model Updates. (2022).

A ALGORITHMS EQUIPPING AGRAMPLIFIER

A.1 Algorithm: AgrAmplifier for Distance-based Mechanism

The original gradients are first processed through the amplifier (line 1 in Algorithm 2). Then the amplified gradients G_{amp} can be fed into any distance measurements (e.g., Cosine similarity and Euclidean distance) and output pair-wise distance/similarity score C_{ij} for each gradient (line 2-5, in which we use Cosine similarity in the algorithm as an example).

Specifically, this measurement examines the density of each gradient's neighbourhood which consists of k neighbours (k should be $> N_p/2$ where N_p is the number of participants), and regards those gradients residing in a denser neighbourhood as benign ones. In particular, for each participant's update, we select its k-nearest neighbours based on the similarity score, and sum up the scores in its neighbourhood (line 6-12). Then, the Top-N participants with higher scores will be added into the white list (N can be a hyper-parameter decided by the practitioner), and submitted to federated learning (line 14-22).

Algorithm 2 AgrAmplifier for Distance-based Mechanism

```
Input: $g_i, g_j$ the gradients collected from $i$-th and $j$-th participant; $N_p$ number of participants; $M_f$ fraction of malicious participants; $k$ number of neighbours examined
Output: $G_{detox}$ the detoxed gradients collection
1:  $\{g^1_{amp}, g^2_{amp}, ...\} \leftarrow$ AgrAmplifier $(g_1, g_2, ...)$
2:  for $i = 1, 2, ..., N_p$ do
3:      for $j = 1, 2, ..., N_p$ do
4:          // Pair-wise cosine similarity
5:          $C_{ij} \leftarrow \frac{g^i_{amp} \cdot g^j_{amp}}{||g^i_{amp}|| \cdot ||g^j_{amp}||}$
6:      end for
7:      // Sum up the similarity of the $k$-nearest neighborhood as $S_i$
8:      $C_i \leftarrow Descending\text{-}sort(C_{ij} | j = 1, 2, 3, ...N_p)$
9:      $S_i \leftarrow \sum C_{ij} | C_{ij} \in C_i, j \leq k$
10: end for
11: $Whitelist \leftarrow \emptyset$
12: for $i = 1, 2, ..., N_p$ do
13:     //Add into white-list if $i$ is with larger $S_i$
14:     if $S_i$ in the largest $(1 - M_f) * N_p$ values $\forall S_i$ then
15:         $Whitelist \leftarrow Whitelist \cup \{i\}$
16:     end if
17: end for
18: $G_{detox} \leftarrow \{g_i | i \in Whitelist\}$
19: return $G_{detox}$
```

A.2 Algorithm: AgrAmplifier for Prediction-based Mechanism

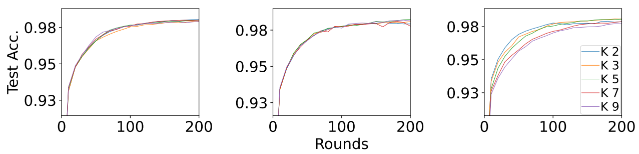
AgrAmplifier first conducts the amplification on the collected gradients specifying that the gradient size shall be restored (line 1 in Algorithm 3). As aforementioned, AgrAmplifier restores the size of the original collected gradients by keeping the maximum value in every feature map and filling up the dropped features with 0. The restored gradients are then sent to LRR and ERR [11] calculation (line 3), and those giving better prediction performance will be added into the white list (line 4).

Algorithm 3 AgrAmplifier for Prediction-based Mechanisms

```
Input: $G$ the collected gradients, equivalent to $\{g_1, g_2, g_3, ...\}$; $G_{amp}$ the amplified gradients; $g_i$ the collected from $i$-th participant in $G$
Output: $G_{detox}$ the detoxed gradients collection
1: $G_{amp} \leftarrow$ AgrAmplifier$(G, Restore\text{-}size{=}True)$
2: $Whitelist \leftarrow$ LRR$(G_{amp}) \cap$ ERR$(G_{amp})$
3: $G_{detox} \leftarrow \{g_i | i \in Whitelist, g_i \in G\}$
4: return $G_{detox}$
```

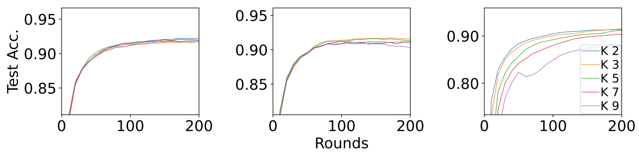
A.3 Algorithm: AgrAmplifier for Trust Bootstrapping-based Mechanism

In addition to the amplification on the collected gradients which outputs g^i_{amp} (line 1 in Algorithm 4), AgrAmplifier also applies the amplification on the gradient generated from the trusted validation set for the trust bootstrapping which outputs g^0_{amp} (line 2). The trust score TS_i for each participant is then calculated based on ReLU-clipped Cosine similarity of the amplified gradients g^i_{amp} and g^0_{amp} (line 4). Then the magnitudes of the collected local model update is normalized (line 5). The normalized gradients are then weighted by their trust scores and aggregated for the global model (line 7).



(a) COSDEN_Amp+ (b) Fang_Amp+ (c) FLTrust_Amp+

Figure 13: The impact of kernel size on MNIST



(a) COSDEN_Amp+ (b) Fang_Amp+ (c) FLTrust_Amp+

Figure 14: The impact of kernel size on PURCHASE100

Algorithm 4 AgrAmplifier for Trust Bootstrapping-based Mechanism

```
Input: $G$ the collected gradients; $g_i$ the collected from $i$-th participant in $G$; $g_0$ the gradient generated from the trusted root set for the trust bootstrapping
Output: $g_{detox}$ the detoxed gradients aggregated for the global model
1: $\{g^1_{amp}, g^2_{amp}, ...\} \leftarrow$ AgrAmplifier$(G)$
2: $g^0_{amp} \leftarrow$ AgrAmplifier$(g_0)$
3: for $i = 1, 2, ..., N_p$ do
4:     $TS_i \leftarrow \mathbf{ReLU}\left(\frac{g^i_{amp} \cdot g^0_{amp}}{||g^i_{amp}|| \cdot ||g^0_{amp}||}\right)$
5:     $\bar{g}_i \leftarrow \frac{||g_0||}{||g_i||} \cdot g_i$
6: end for
7: $g_{detox} \leftarrow \frac{1}{\sum_{j=1}^{N_p} TS_j} \sum_{i=1}^{N_p} TS_i \cdot \bar{g}_i$
8: return $g_{detox}$
```

B THE IMPACT OF KERNEL SIZE ON MNIST AND PURCHASE100

Fig. 13 and 14 shows the impact of different kernel size on MNIST and PURCHASE100.