ACE: A Model Poisoning Attack on Contribution Evaluation Methods in Federated Learning

Zhangchen Xu
University of Washington
zxu9@uw.edu

Fengqing Jiang
University of Washington
fqjiang@uw.edu

Luyao Niu
University of Washington
luyaoniu@uw.edu

Jinyuan Jia
Pennsylvania State University
jinyuan@psu.edu

Bo Li
University of Chicago
bol@uchicago.edu

Radha Poovendran
University of Washington
rp3@uw.edu

Abstract

In *Federated Learning (FL)*, a set of clients collaboratively train a machine learning model (called *global model*) without sharing their local training data. The local training data of clients is typically non-i.i.d. and heterogeneous, resulting in varying contributions from individual clients to the final performance of the global model. In response, many contribution evaluation methods were proposed, where the server could evaluate the contribution made by each client and incentivize the high-contributing clients to sustain their long-term participation in FL. Existing studies mainly focus on developing new metrics or algorithms to better measure the contribution of each client. However, the security of contribution evaluation methods of FL operating in adversarial environments is largely unexplored. In this paper, we propose the *first* model poisoning attack on contribution evaluation methods in FL, termed ACE. Specifically, we show that any malicious client utilizing ACE could manipulate the parameters of its local model such that it is evaluated to have a high contribution by the server, even when its local training data is indeed of low quality. We perform both *theoretical* analysis and *empirical* evaluations of ACE. Theoretically, we show our design of ACE can effectively boost the malicious client's perceived contribution when the server employs the widely-used cosine distance metric to measure contribution. Empirically, our results show ACE effectively and efficiently deceive five state-of-the-art contribution evaluation methods. In addition, ACE preserves the accuracy of the final global models on testing inputs. We also explore six countermeasures to defend ACE. Our results show they are inadequate to thwart ACE, highlighting the urgent need for new defenses to safeguard the contribution evaluation methods in FL.

1 Introduction

Federated learning (FL) [42] enables a set of clients to collaboratively train a machine learning model, denoted as the *global model*, using their local training data in an iterative manner. At each communication round, a cloud server first broadcasts the current global model to the clients. Each client then adopts the global model as its local model, locally minimizes an empirical loss function (e.g., cross-entropy function) over its local training data to compute a *local model update*, and finally sends the local model update to the server. The server aggregates the local model updates from the clients according to an aggregation rule (e.g., FedAvg [42]) to update the current global model. FL has been widely deployed in real-world cross-silo settings where data is spread across multiple isolated organizations [40, 52, 85].

In practice, the local training data possessed by the clients in FL is non-i.i.d. and heterogeneous [11, 35, 37, 59], and thus inherently of varying qualities. Therefore, it is crucial to understand and evaluate the contribution of each client toward the performance (e.g., accuracy on testing inputs) of the global model. Accurate contribution evaluation facilitates the designs of incentive mechanisms to encourage the clients, especially those owning high-quality data, to participate into FL [15, 22, 26, 73], which could further enhance the performance of the global model. To this end, contribution evaluation methods in FL has been extensively studied [54, 57]. The existing studies [15, 16, 22, 26, 39, 62, 65, 73, 81] primarily focus on the development of novel metrics or algorithms to measure the contributions of clients in FL. At present, however, the security of contribution evaluation methods in FL remains largely unexplored.

Our Contribution. In this paper, we propose the *first* model poisoning attack on contribution evaluation methods in FL. We term this attack as ACE. We consider that an attacker owns a subset of clients in FL, denoted as malicious clients. These malicious clients are evaluated as low-contributing participants by the server. Specifically, the malicious clients manipulate the parameters of their local models, with the objective of elevating their contributions evaluated by the server. Accomplishing this attack goal can result in monetary advantages for the attacker when contribution-based incentive mechanisms are employed in FL. For example, the FL server may distribute a certain amount of budget to the clients in propor-

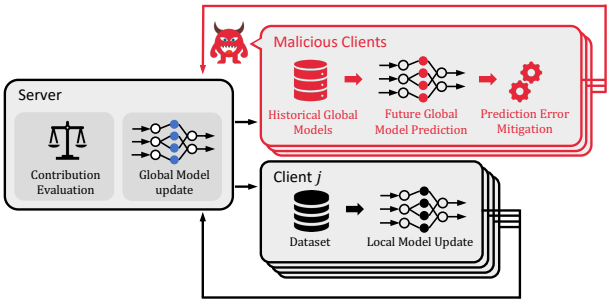


Figure 1: An illustration of ACE consisting of two components: future global model prediction and prediction error mitigation.

tion to their individual contributions in order to encourage clients with high contributions to remain in the FL [46]. The attack studied in this paper can boost the contributions of malicious clients, and thus increase their shares of the budget. This lowers the shares available to the other clients, thereby jeopardizing their interests.

We highlight that the attack goal of ACE is significantly different from existing attacks in FL, including untargeted poisoning attacks [8, 18, 30, 51, 70, 86] and backdoor attacks [1, 2, 58, 63, 68, 82]. In particular, existing untargeted poisoning attacks and backdoor attacks aim to make the global model exhibit low performance for indiscriminate testing inputs or predict an attacker-chosen target class for any inputs embedded with a backdoor trigger [27]. By contrast, ACE aims to deceive the contribution evaluation method employed by the server to increase the contributions of malicious clients (with low-quality local training data). Our empirical studies demonstrate that ACE retains the performance of the global models in different settings. We defer the detailed discussion on the difference to Section 8.

A major challenge in developing ACE is how the malicious clients should strategically manipulate the parameters of malicious clients' local models to increase the perceived contribution by the server. Our insight to address this challenge is that the FL procedure provides the malicious clients information on the global model as shown in Figure 1, allowing the malicious client to predict how the global model evolves over communication rounds. Therefore, the malicious clients can craft local model updates to better align with the prediction of the global model, making them more likely to be perceived by the server as having higher contributions. ACE uses the Cauchy mean value theorem [33] to predict the global model at each communication round. We show that predicting the global model significantly reduces the computation complexity for the malicious clients compared to iteratively learning local model updates from the local training datasets, allowing ACE to boost the perceived contributions at negligible cost.

We *theoretically* analyze the effectiveness of ACE when the server measures contributions using cosine distance. We prove that ACE allows the malicious clients to always increase the perceived contribution by appropriately scaling up the predicted global model. We further *empirically* evaluate the effectiveness and efficiency of ACE using five state-of-the-art contribution evaluation methods in FL [41, 65, 72, 73, 81]. We compare ACE with four baselines using three models across three datasets including MNIST [14], CIFAR-10 [31], and Tiny-ImageNet [34]. We show that ACE consistently outperforms all baselines when the local training data of clients is non-i.i.d., yielding the highest perceived contribution by the FL server. This demonstrates the severity of ACE on contribution evaluation methods in FL. We evaluate ACE against countermeasures including extended Multi-Krum [3] and Trimmed-Mean [75], which have been widely used in FL. Our empirical evaluations demonstrate that these countermeasures are not effective against ACE. These results underscore the need for the development of new defenses to thwart ACE.

To summarize, this paper makes the following major contributions:

- We propose the first model poisoning attack on contribution evaluation methods in FL, termed as ACE.
- We present theoretical analysis and perform extensive empirical evaluations of ACE to demonstrate its effectiveness and efficiency.
- We investigate the countermeasures to mitigate ACE. We show that ACE can remain stealthy against the existing mitigation strategies, highlighting the needs for new defense mechanisms.

2 Background and Related Work

In this section, we present background on federated learning and contribution evaluation methods in FL.

2.1 Federated Learning

We consider an FL setting where N clients collaboratively train a machine learning model, called *global model*. Let $\mathcal{D}_i$ represent the local training dataset of the i-th client and $|\mathcal{D}_i|$ denote its size, where $i = 1, 2, \cdots, N$. We denote the set of clients as Γ, and thus the joint training dataset can be represented as $\mathcal{D} = \cup_{i\in\Gamma}\mathcal{D}_i$. To learn the global model, the set of clients collaboratively minimizes a loss function over their local training datasets as follows:

$$\min_{\mathbf{w}} \sum_{i\in\Gamma} L(\mathcal{D}_i; \mathbf{w}),$$

where $\mathbf{w}$ represents the parameters of the global model and $L(\mathcal{D}_i; \mathbf{w})$ is the empirical loss (e.g., cross-entropy loss) evaluated using the global model with parameters $\mathbf{w}$ on the local training dataset $\mathcal{D}_i$. The clients iteratively solve the optimization problem through multiple communication rounds with

an FL server. Specifically, there are three steps at each communication round t, which are detailed below.

Step I. The server broadcasts the current global model, denoted as $\mathbf{w}^t$, to the clients.

Step II. For each client i, it first uses the global model $\mathbf{w}^t$ to initialize its local model, and then uses the local training dataset $\mathcal{D}_i$ to update its local model by minimizing the empirical loss function L, i.e., $\mathbf{w}_i^{t+1} = \mathbf{w}^t - \eta_i \nabla L(\mathcal{D}_i; \mathbf{w}^t)$, where η_i is the learning rate and $\nabla L(\mathcal{D}_i; \mathbf{w}^t) = \frac{\partial L(\mathcal{D}_i; \mathbf{w}^t)}{\partial \mathbf{w}^t}$. Finally, it sends the *local model update* $\mathbf{g}_i^t = \mathbf{w}^t - \mathbf{w}_i^{t+1}$ back to the server. Note that it is equivalent for the client to send the local model update or local model due to the above relationship.

Step III. The server aggregates the local model updates from the clients, and updates the global model of the $(t+1)$-th communication round as $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \mathbf{g}^t$, where $\mathbf{g}^t = \mathcal{A}(\mathbf{g}_1^t, \mathbf{g}_2^t, \cdots, \mathbf{g}_N^t)$ is the global model update at communication round t and $\mathcal{A}$ is an aggregation rule. A typical example of $\mathcal{A}$ is FedAvg [42], defined as $\mathcal{A}(\mathbf{g}_1^t, \mathbf{g}_2^t, \cdots, \mathbf{g}_N^t) = \sum_{i \in \Gamma} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \mathbf{g}_i^t$.

2.2 Contribution Evaluation Methods in FL

Contribution evaluation in FL aims to quantify the *contribution* made by each client to the performance (e.g., accuracy on testing inputs) of the global model. Following previous studies [39, 54], we divide the existing contribution evaluation methods in FL into three categories: *self-reporting based contribution evaluation* [76, 77], *individual performance based contribution evaluation* [12, 22, 29, 41, 55, 72, 73, 78, 79], and *game theory based contribution evaluation* [16, 23, 28, 39, 62, 64, 65, 81, 84].

Self-Reporting based Contribution Evaluation. The methods in this category utilize the information reported by each client to measure its contribution. For instance, Zeng et al. [77] propose to use self-reported local data size and communication bandwidth to evaluate the contribution. Yu et al. [76] proposed to use the data quantity and quality (e.g., measured by the marginal revenue generated by the global model) reported by each client to measure its contribution. Since those methods heavily rely on self-reported information, they are susceptible to malicious clients who can untruthfully report their information to the server.

Individual Performance based Contribution Evaluation. This category of methods proposes some performance metrics defined on the local model updates of clients to measure their contributions. Some studies [12, 41] assume the server has a clean validation dataset in order to calculate the proposed performance metrics. For instance, Lyu et al. [41] evaluate the contribution of a client at each communication round using the validation accuracy of its local model on the validation dataset of the server. Chen et al. [12] propose to quantify the contribution of a client at each individual communication round by computing the mutual cross-entropy loss between the local model of the client and the global model on the validation dataset. However, these methods assume the server has a clean, small validation dataset, which has been shown to be impractical [50]. When the server does not have any validation data, existing studies propose to utilize the distance (e.g., cosine distance [29, 55, 72, 73, 78] and Euclidean distance [22, 79]) between the global model and the local model of a client to measure its contribution at each communication round. The key assumption is that a client whose local model is more similar to the aggregated global model makes more contribution to it. Therefore, the contribution of a client is smaller if the distance becomes larger.

Game Theory based Contribution Evaluation. This category of methods formulates the FL as a cooperative game, where all clients collaboratively learn a global model using their local training data. Then the aggregated contribution of the clients is represented by the utility of the game (e.g., accuracy [39] or empirical loss [16, 62, 81] of the global model on testing inputs), and the contribution of each individual client is modeled by its payoff received in this game.

Wang et al. [62] propose to quantify the contribution (averaged over all communication rounds) of each client in FL using its marginal loss, which is defined as the utility difference when the client joins the game as opposed to not joining the game. Similarly, Zhang et al. [81] measure the contributions of clients at each communication round using their marginal performance loss evaluated on a held-out validation dataset. However, these methods are sensitive to the order in which the clients join the game, and may not yield consistent results for contribution evaluation when the order changes.

To eliminate the effect from the order of joining the FL, Shapley value (SV) is adopted to measure the contribution of each client under the cooperative game framework [23, 28]. Wang et al. [65] propose the federated SV (FedSV) to quantify the contribution of each client at each communication round, which retains the key features of the traditional SV without introducing additional communication costs. However, computing SVs is generally computationally expensive. Consequently, numerous efforts have been made to reduce the computation complexity of SV [16, 17, 39, 64, 84].

3 Problem Formulation

In this section, we characterize the threat model by presenting the capabilities, background knowledge, and goals of an attacker. We then formally formulate the model poisoning attack on contribution evaluation methods in FL. We finally describe the design goals of the attack.

3.1 Threat Model

Attacker's capabilities and background knowledge. We consider an attacker owns a set of clients, referred to as *ma-*

licious clients. As a result, the attacker could (1) access the local training datasets of malicious clients and the global model sent by the server, (2) control the training processes of local models of malicious clients, and (3) manipulate the parameters of the malicious clients' local models before sending the local model updates to the server. However, the attacker lacks the necessary background knowledge (e.g., local training datasets) of all other clients and the capabilities to manipulate parameters in other clients' local models. Moreover, we assume the attacker knows the contribution evaluation method employed by the server. This assumption is realistic in practice since the server normally shares such information with all clients for transparency and trustworthiness purposes during the initiation of the FL system [18, 42]. However, we consider that the attacker does not know the hyperparameters of the contribution evaluation method and thus does not know the contributions of malicious clients computed by the server in each communication round. For example, the server may use a validation dataset to evaluate the local model of a client, and use the validation accuracy as the contribution of the client, which is not broadcast to the clients [41, 65]. We also assume that the attacker possesses some storage capacity to retain the global models broadcast by the server in the previous m (e.g., $m = 3$) communication rounds.

Attacker's goal. Suppose the server employs a contribution evaluation method (as reviewed in Section 2) to quantify the contribution of each client. We consider that the goal of the attacker is to elevate the malicious clients' contributions computed by the server, compared to truthfully sending the local model updates learned using their local training data to the server. By accomplishing the attack goal, the attacker can get extra rewards from the existing incentive mechanisms deployed in FL [15, 26, 41, 78], even though it holds low-quality local training data. For instance, in a cross-silo FL system where multiple banks collaborate to jointly learn a global model for commercial purposes [40], if the profit earned using the global model is distributed among the banks based on their contributions, the bank launching ACE can gain extra profit while reducing the shares of others. Such malicious behavior further discourages the long-term participation of banks possessing high-quality training data and undermines the fairness of the federated learning system.

3.2 A Model Poisoning Attack on Contribution Evaluation Methods in FL

In what follows, we formally formulate the model poisoning attack on contribution evaluation methods in FL. We denote the set of malicious clients as $\hat{\Gamma}$. For each malicious client $i \in \hat{\Gamma}$, we use $\mathbf{g}_i^t$ to denote the local model update learned using the local training data without attacks. Moreover, we denote the manipulated parameters of the local model update from client i at communication round t as $\hat{\mathbf{g}}_i^t$. We use $\mathcal{E}$ to denote the contribution evaluation method deployed by the server, where $\mathcal{E}(\mathbf{g}_i^t)$ (or $\mathcal{E}(\hat{\mathbf{g}}_i^t)$) denotes the contribution calculated by the server when the client i sends the local model update $\mathbf{g}_i^t$ (or $\hat{\mathbf{g}}_i^t$) to the server. The goal of the attacker is to craft $\hat{\mathbf{g}}_i^t$ for each malicious client $i \in \hat{\Gamma}$ such that the accumulated contribution $\sum_{i\in\hat{\Gamma}} \mathcal{E}(\hat{\mathbf{g}}_i^t)$ (or equivalently $\sum_{i\in\hat{\Gamma}}(\mathcal{E}(\hat{\mathbf{g}}_i^t) - \mathcal{E}(\mathbf{g}_i^t))$) is maximized. We call such an attack *model poisoning attack on contribution evaluation methods in FL*. Formally, we formulate the attack as the following optimization problem in the t-th communication round:

$$\{\hat{\mathbf{g}}_i^t | i \in \hat{\Gamma}\} = \underset{\{\mathbf{g}_i | i\in\hat{\Gamma}\}}{\operatorname{argmax}} \sum_{i\in\hat{\Gamma}} \mathcal{E}(\mathbf{g}_i). \tag{1}$$

3.3 Design Goals

We aim to design a model poisoning attack on contribution evaluation methods in FL. In particular, we aim to accomplish the following goals in our attack design:

Effective. Our first goal is that the attack should be effective, i.e., it could significantly increase the contributions of malicious clients calculated by the server, compared to the scenario without attacks (i.e., the malicious clients learn their local model updates using the local training datasets).

Efficient. Our second goal is that the attack should be efficient, i.e., it should incur small computation and communication costs, compared to the baseline when there is no attack. The reason for incorporating efficiency into the design goals is that clients in FL are often resource-constrained, e.g., mobile phones and IoT devices [42, 45].

Performance Preserving. We note that the attacker's goal is not to disrupt the convergence or performance of the final global model learned from FL. Thus, we aim to design an attack that preserves the performance of the final global model. As a result, the malicious clients, who own low-quality local training data, could obtain a global model with comparable performance to the one learned without attacks.

Aggregation Rule Independent. We note that many aggregation rules [3, 42, 73, 75] have been proposed to aggregate local models in FL. We aim to design an attack that is agnostic to the aggregation rules such that our attack is generalizable to a wide range of FL systems.

4 Description of ACE

4.1 Overview of ACE

A key challenge in solving the optimization problem in Eq. (1) is that an attacker does not know $\mathcal{E}(\hat{\mathbf{g}}_i^t)$ for an arbitrary local model update $\hat{\mathbf{g}}_i^t$. The reason is that the attacker lacks necessary information such as the local models of other clients or validation dataset of the server that is utilized to calculate contribution (see Section 2.2 for details). To address the challenge, our key insight is that a client is more likely to have a high contribution if its local model is more similar to the

aggregated global model in each communication round. The reason is that the aggregated global model is obtained from local models learned on local training datasets of all clients, including those whose local training data is considered more valuable by the server. Based on this insight, we propose ACE, a poisoning attack on contribution evaluation methods in FL.

ACE consists of two components: *future global model prediction* and *prediction error mitigation*. The future global model prediction component aims to predict the global model in each communication round based on the historic global models (i.e., the global models in previous communication rounds) received by the attacker. We note that the attacker could incur certain prediction errors and they would accumulate over communication rounds, resulting in low contributions for the malicious clients. In response, we further propose two strategies to mitigate the impact of errors in our prediction error mitigation component. We further discuss how ACE can improve the attack effectiveness on certain contribution evaluation methods. Finally, we analyze both the time and space complexities of ACE. Our analysis shows ACE incurs negligible computation and storage costs. The reason is that it is very efficient to predict the future global model and ACE only requires saving a few snapshots of historic global models.

4.2 Detailed Design of ACE

We first describe the *future global model prediction* component, followed by the *prediction error mitigation* component. Then, we discuss how to adapt ACE to further enhance the effectiveness of our attack.

4.2.1 Future Global Model Prediction

In what follows, we describe how the malicious clients predict the global model update. Note that given the predicted global model update, the predicted global model could be obtained by adding the predicted global model update and the current global model together.

Predicting the Global Model Updates using the Cauchy Mean Value Theorem. We use $\hat{\mathbf{g}}^t$ to denote the predicted global model update at communication round t. According to the Cauchy mean value theorem [33], the *predicted global model update* $\hat{\mathbf{g}}^t$ at communication round t is calculated as

$$\hat{\mathbf{g}}^t = \mathbf{g}^{t-1} + H^t(\mathbf{w}^t - \mathbf{w}^{t-1}), \tag{2}$$

where $\mathbf{g}^{t-1}$ is the global model update in the communication round $t-1$, $\mathbf{w}^t$ (or $\mathbf{w}^{t-1}$) is the global model at communication round t (or $t-1$), and $H^t = \int_0^1 H(\mathbf{w}^{t-1} + z(\mathbf{w}^t - \mathbf{w}^{t-1}))dz$ is an *integrated Hessian matrix*. Based on Eq. (2), predicting the global model update only requires the current and previous global models, the integrated Hessian matrix, and the previous global model update. Although each malicious client has background knowledge on the previous global model update, previous global model, and current global model, computing the integrated Hessian matrix, however, is computationally expensive. In light of this, we utilize the L-BFGS algorithm [4, 5], which is widely used to approximate the integrated Hessian matrix.

Approximating the Integrated Hessian Matrix via the L-BFGS Algorithm. We approximate the integrated Hessian matrix H^t using the L-BFGS algorithm [4, 5], as outlined in Algorithm 1 in Appendix C. The L-BFGS algorithm takes two buffers $\Delta\mathbf{W}^t = [\Delta\mathbf{w}^{t-m}, \Delta\mathbf{w}^{t-m+1}, \cdots, \Delta\mathbf{w}^{t-1}]$ and $\Delta\mathbf{G}^t = [\Delta\mathbf{g}^{t-m}, \Delta\mathbf{g}^{t-m+1}, \cdots, \Delta\mathbf{g}^{t-1}]$ as inputs, and approximates the integrated Hessian matrix, where $\Delta\mathbf{w}^t = \mathbf{w}^t - \mathbf{w}^{t-1}$ is the change in the global model and $\Delta\mathbf{g}^t = \mathbf{g}^t - \mathbf{g}^{t-1}$ is the change in global model update. In practical implementation [5], the L-BFGS algorithm consumes an additional vector $\mathbf{v}$ of appropriate dimension as an input, and returns a product $H^t\mathbf{v}$, termed *Hessian-vector product*, as the output. This is sufficient for the malicious clients to predict the global model update since Eq. (2) only requires a Hessian-vector product $H^t(\mathbf{w}^t - \mathbf{w}^{t-1})$ to predict the global model update $\hat{\mathbf{g}}^t$. In the remainder of this paper, we represent the L-BFGS algorithm as $\text{L-BFGS}(\Delta\mathbf{W}^t, \Delta\mathbf{G}^t, \mathbf{v})$. We note that the L-BFGS algorithm is employed by the server to estimate clients' local models, as countermeasures against model poisoning attacks in FL in existing studies [9, 80]. This paper, however, considers scenarios where malicious clients use the L-BFGS algorithm to estimate the global model, thereby elevating their contribution evaluated by the server.

4.2.2 Prediction Error Mitigation

Insufficiency of Future Global Model Prediction. We remark that it is inadequate for the malicious clients to only predict the global model update. The reason is that the L-BFGS algorithm may incur large prediction errors, i.e., the deviation between the predicted global model update and its true value, at some communication rounds. As we will demonstrate in Section 6, the prediction error results in low contribution evaluated by the server for the malicious clients. We identify two major reasons for prediction errors. The first issue arises because the L-BFGS algorithm requires historical information from past global models and their updates to construct buffers $\Delta\mathbf{W}^t$ and $\Delta\mathbf{G}^t$. To compensate for the absence of this historical information, we adopt a strategy used in previous studies [9], *preliminary iteration*, during the initial communication rounds, where malicious clients either learn from their local datasets or use the previous round's global model updates as proxies for their current local updates. In the meantime, the malicious clients collect the global models and global model updates to construct the buffers $\Delta\mathbf{W}^t$ and $\Delta\mathbf{G}^t$. After that, the malicious clients proceed with the L-BFGS algorithm as discussed in Section 4.2.1. The second reason for prediction errors is the potential accumulation of errors over successive communication rounds. To address this, we develop a *threshold-based filtering* strategy to mitigate the

impact of error accumulation.

Threshold based Filtering. As the L-BFGS algorithm cannot predict the exact global model update, it is unavoidable that there would be prediction errors. Moreover, those errors could accumulate over communication rounds, which could lower the contributions of the malicious clients. We note that the malicious clients may not be aware when a large prediction error occurs since it does not have access to the true global model update for the future communication rounds, and thus cannot calculate the prediction errors. To tackle this challenge, we develop a threshold based filtering strategy to estimate whether the predicted global model update incurs a large prediction error. Our intuition is that the prediction error is more likely to be larger if the magnitude of the Hessian-vector product is larger. Therefore, if the ℓ_2-norm of the Hessian-vector product is less than a threshold, i.e.,

$$\|\text{L-BFGS}(\Delta\mathbf{W}^t, \Delta\mathbf{G}^t, \mathbf{w}^t - \mathbf{w}^{t-1})\| \leq \tau, \tag{3}$$

where $\|\cdot\|$ represents ℓ_2-norm and τ is a threshold (we defer the detailed discussion on it), we consider that the prediction error is tolerable, and thus the malicious clients use the predicted global model update as their local model updates. Otherwise, each malicious client calculate its local model update by using its local training dataset or utilizing the global model update from the previous communication round.

A key question in our design is how to set the threshold. Our idea is that the magnitude of the predicted global model update should be on a similar scale as the previous global model updates. Thus, we set the threshold τ as $l\|\mathbf{w}^t - \mathbf{w}^{t-1}\|$, where l is a positive coefficient that can be tuned by each malicious client to control its tolerance on prediction errors. Our experimental results show that $l = 1$ is sufficient to mitigate prediction errors.

4.2.3 Strategies to Enhance ACE

In this subsection, we discuss how to adapt ACE to further enhance its effectiveness when certain a contribution evaluation method is employed by the server.

Adaptation to Contribution Evaluation Methods using Cosine Distance. In the following, we discuss how to adapt ACE when the server uses the cosine distance between the local model and global model to measure each client's contribution. This class of methods is widely used in existing studies [29, 55, 72, 73] since cosine distance is efficient to calculate. In this case, we can rewrite the optimization in Eq. (1) as the following equivalent formulation:

$$\{\hat{\mathbf{g}}_i^t | i \in \hat{\Gamma}\} = \underset{\{\mathbf{g}_i | i \in \hat{\Gamma}\}}{\text{argmax}} \sum_{i \in \hat{\Gamma}} (1 - \cos(\mathbf{g}_i, \mathbf{g}^t)). \tag{4}$$

Here, $\cos(\mathbf{a}, \mathbf{b}) = 1 - \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$ is the cosine distance function, where $\mathbf{a} \cdot \mathbf{b}$ represents the inner product between vectors $\mathbf{a}$ and $\mathbf{b}$. Global model update $\mathbf{g}^t$ is obtained following the aggregation rule (Step III in Section 2.1).

According to Eq. (4), we note that the malicious clients can enhance the attack effectiveness by sending an amplified prediction of the global model update $c\hat{\mathbf{g}}_i^t$ to the server as their local model updates, where $c > 1$ is an amplifying coefficient. The insight is that increasing the magnitude of the predicted global model update will make it more dominant compared with other clients' local model updates. As a result, the angle deviation between the predicted global model update and the aggregated global model update decreases, leading to a smaller cosine distance between them. Furthermore, such dominance effects will accumulate over communication rounds, thereby elevating the malicious clients' contributions perceived by the server. We finally remark that introducing the coefficient c does not lower the accuracy of the global model, which will be demonstrated in Section 6.

Adaptation to Contribution Evaluation Methods with Validation Dataset. Some contribution evaluation methods [12, 41, 62, 65, 81] require the server to have a validation dataset, and use the validation accuracy or loss of local models to measure contributions. In this case, the malicious clients can execute the L-BFGS algorithm multiple times within one communication round. We term this operation as the *local evolution*. The insight behind local evolution is to mimic the normal training process over multiple epochs at one communication round. When the L-BFGS algorithm yields limited prediction error, the local evolution will allow the malicious clients to craft local model updates of higher validation accuracy and hence increase the associated contributions.

4.3 Complete Algorithm

Algorithm 2 in Appendix C shows our complete algorithm for attacking contribution evaluation methods in FL. The attacker can initiate attack at any communication round t. If communication round t is a preliminary iteration, then the malicious clients compute their local model updates by learning from the local training data or sending the previous global model update to the server (see Section 4.2.3). If communication round t is not a preliminary iteration, then the malicious clients leverage the L-BFGS algorithm to predict the next global model update (see Section 4.2.1). Given the predicted global model update, the malicious clients estimate whether the prediction error can be tolerated or not using the threshold based filtering as shown in Eq. (3). If the prediction error is tolerable by the malicious clients (Eq. (3) holds true), then they set the prediction of global model update to be their local model updates which will be sent to the server. Otherwise, the malicious clients compute the local model updates using their local training data or the previous global model update.

4.4 Complexity Analysis

Time Complexity. According to [5], the time complexity to calculate the Hessian-vector product using Algorithm 1 is $O(m^3) + 6mp + p$, where p is the model size and m is the buffer length. If the malicious clients does not launch the attack and learn the local model update using the local training dataset over e epochs, then the time complexity of learning the local model update using the local training dataset $\mathcal{D}_i$ over e epochs is $6|\mathcal{D}_i|eR(p)$, where $R(p)$ is the time complexity for forward propagation [24]. Although the expression of $R(p)$ is architecture dependent, we note that the time complexity of ACE is in general significantly less than that of learning the local model update using local training data [20], achieving the design goal of being efficient in Section 3.3. For instance, when the global model is a CNN (see Appendix B.1 for the detailed architecture), the computation time of ACE using an RTX 6000 Ada GPU is 0.004s per communication round. By contrast, training the same model using the local training dataset of a client takes 1.10s, which is 270× slower compared to ACE.

Space Complexity. The space complexity for each malicious client is $O(mp)$ when it launches ACE. Note that in practice, we typically choose $m = 2$ or $m = 3$. Therefore, the proposed attack in this paper imposes a small storage constraint on the malicious clients. For instance, when $m = 3$ and the architecture of the global model is CNN, our attack only requires 42.43MB of additional storage space.

5 Theoretical Analysis

In this section, we characterize the strategies to enhance the attack effectiveness. Specifically, we focus on the cases where cosine distance is used by the server to measure contributions. All the proofs can be found in Appendix D. The effectiveness of amplifying the predicted global model update with coefficient c is stated as follows.

Proposition 1. *Let* $\mathbf{g}' = \mathcal{A}(\mathbf{g}_1, \cdots, c\hat{\mathbf{g}}_i, \cdots, \mathbf{g}_N)$ *and* $\mathbf{g} = \mathcal{A}(\mathbf{g}_1, \cdots, \hat{\mathbf{g}}_i, \cdots, \mathbf{g}_N)$ *be the global model updates obtained using the predicted global model update* $c\hat{\mathbf{g}}$ *and* $\hat{\mathbf{g}}$*, respectively. When* $c \geq 1$*, we have the following relationship:*

$$\cos(\mathbf{g}', c\hat{\mathbf{g}}_i) \leq \cos(\mathbf{g}, \hat{\mathbf{g}}_i). \tag{5}$$

Remark 1. *Proposition 1 shows that the cosine distance between the amplified global model update* $c\mathbf{g}_i$ *and the global model update* $\mathbf{g}'$ *is no larger than the cosine distance before applying the amplification with coefficient* $c \geq 1$*. Therefore, our strategy developed for contribution evaluation using cosine distance will not degrade the attack effectiveness.*

Proposition 1 yields the following corollary.

Corollary 1. *Let* $\mathbf{g}'$ *and* $\mathbf{g}$ *be defined as in Proposition 1. If* $\cos(\mathbf{g}, \hat{\mathbf{g}}_i) \leq \cos(\mathbf{g}, \mathbf{g}_j)$*, then* $\cos(\mathbf{g}', c\hat{\mathbf{g}}_i) \leq \cos(\mathbf{g}', \mathbf{g}_j)$.

Remark 2. *Corollary 1 indicates that if a malicious client* i *surpasses the contribution of another client* j *by sending the predicted global model update as its local model update, then amplifying the predicted global model update will not make the contribution of malicious client* i *less than client* j*. Therefore, the malicious client* i *is always perceived as a high-contributing client compared to client* j *(note that a smaller cosine distance means a higher contribution).*

We finally show that the parameter c can be tuned such that the contributions of malicious clients become larger than any arbitrary client.

Proposition 2. *Let* $\mathbf{g}'$ *and* $\mathbf{g}$ *be defined as in Proposition 1. Suppose that* $\cos(\mathbf{g}, \hat{\mathbf{g}}_i) > \cos(\mathbf{g}, \mathbf{g}_j)$ *holds for some client* j *and malicious client* i*, and* $\mathcal{A}(\mathbf{g}_1, \cdots, \mathbf{g}_N) = \sum_{j \in \Gamma} \alpha_j \mathbf{g}_j$*, where* $\alpha_j \in [0, 1]$ *is a weight coefficient and* $\sum_{j \in \Gamma} \alpha_j = 1$*. If the malicious client* i *chooses the coefficient* c *such that* $c \geq \frac{\|\hat{\mathbf{g}}_i\| \mathbf{g} \cdot \mathbf{g}_j - \|\mathbf{g}_j\| \mathbf{g} \cdot \hat{\mathbf{g}}_i}{\alpha_i \|\hat{\mathbf{g}}_i\| (\|\mathbf{g}_j\| \|\hat{\mathbf{g}}_i\| - \hat{\mathbf{g}}_i \cdot \mathbf{g}_j)} + 1$*, then* $\cos(\mathbf{g}', c\hat{\mathbf{g}}_i) \leq \cos(\mathbf{g}', \mathbf{g}_j)$.

Remark 3. *Proposition 2 focuses on servers that utilize linear combinations as the aggregation rule, which are widely used in practice [3, 42, 73, 75]. Proposition 2 shows that even if the predicted global model may not be sufficient for a malicious client* i *to surpass the contribution of another client* j*, choosing a proper coefficient* c *allows the malicious client* i *to make a higher contribution (evaluated by the server) compared to client* j*.*

6 Empirical Evaluations

We perform extensive experiments to evaluate ACE. In Section 6.1, we show the experimental setup. Section 6.2 presents the results. Ablation analysis is presented in Section 6.3.

6.1 Experimental Setup

Datasets and Models. We consider three benchmark datasets MNIST [14], CIFAR-10 [31], and Tiny-ImageNet [34]. Specifically, MNIST is a 10-class digit image classification dataset, which contains 60,000 training images and 10,000 testing images of dimension 28×28 in grayscale. CIFAR-10 is a 10-class dataset with 50,000 training images and 10,000 testing images uniformly distributed across the classes, where the size of each image is $32 \times 32 \times 3$. Tiny-ImageNet is a color image classification dataset covering 200 classes, with 100,000 training images, 10,000 validation images, and 10,000 testing images. Each image in Tiny-ImageNet is of dimension $64 \times 64 \times 3$. We use two Convolution Neural Network (CNN) variants on MNIST and CIFAR-10 datasets respectively, and a pre-trained VGG11 model [56] on Tiny-ImageNet. The model structures are shown in Appendix B.1.

Data Partition. For each dataset, we consider one homogeneous data partition (denoted as UNI) and two heterogeneous

data partitions (denoted as POW and CLA) among clients by following previous studies on contribution evaluation methods in FL [41, 72, 73]. The heterogeneous data partitions yield non-i.i.d. data distributions. We detail each data partition as follows.

- **UNI**: This data partition uniformly splits the training images in each dataset among all clients, yielding an i.i.d. data distribution among the clients.

- **POW**: Following [41, 72, 73], the sizes of local training datasets of all clients are sampled from a parameterized power law distribution. We set the parameter of power law distribution as two. This leads to a data-size heterogeneous setting among the clients. For example, on MNIST, the numbers of training images of 10 clients are 110, 219, 328, 437, 546, 655, 764, 873, 982, and 1,086, respectively. The details for the power law distribution can be found in Appendix B.2.

- **CLA**: Following [41, 72, 73], we use CLA to create a class imbalance setting where the local training datasets of different clients cover heterogeneous numbers of classes. For example, on MINST, the local training datasets of 10 clients contain training images from 6, 6, 7, 7, 8, 8, 9, 9, 10, and 10 classes, respectively. We show the details in Appendix B.2.

FL Setup. By default, we assume there are $N = 10$ clients. Note that we set $N = 10$ since some contribution evaluation methods calculate the Shapley value to measure contributions, whose computation cost grows exponentially as the number of clients increases. As some contribution evaluation methods require the server to have a validation dataset to compute contributions, we reserve 20% of the data samples from the training images of MNIST, CIFAR-10, and Tiny ImageNet as the validation dataset. Each client uses stochastic gradient descent (SGD) to update its local model for ϵ epochs with a batch size of 128. We set $\epsilon = 3$ for MNIST and CIFAR-10, and $\epsilon = 5$ for Tiny-ImageNet, considering that the classification tasks on MNIST and CIFAR-10 are easier than that on Tiny-ImageNet. Following previous studies [37, 68, 83], we set the learning rate η as 0.03 for MNIST, 0.05 for CIFAR-10, and 0.001 for Tiny-ImageNet, with the learning rate exponentially decaying at rate $\gamma = 0.995$. The total number of communication rounds is $T = 60$.

Evaluation Metrics. We define the following two metrics to demonstrate the effectiveness of ACE. The first metric is called *contribution score*, which measures the fraction of contribution from each individual client. Formally, the contribution score for the client i is computed as follows:

$$CS_i = \frac{\sum_{t=1}^{T} e_i^t}{\sum_{j\in\Gamma}\sum_{t=1}^{T} e_j^t}, \tag{6}$$

where Γ is the set of all clients and e_i^t (or e_j^t) is the contribution computed by the server using a contribution evaluation method for the client i (or j) in the t-th communication round.

We note that elevation in the contribution scores does not necessarily imply that the malicious clients' contributions can surpass those of other clients. Therefore, we propose the second metric named *rank gain* to measure the change in the ranks of a client's contribution score with and without attacks. Formally, the rank gain for the client i is computed as follows:

$$\Delta R_i = \widehat{R}_i - R_i, \tag{7}$$

where $\widehat{R}_i$ and R_i represent the ranking (in ascending order) of the contribution score of the client i among all clients with and without attack, respectively. An attack is more effective when CS_i and ΔR_i are larger for a malicious client i.

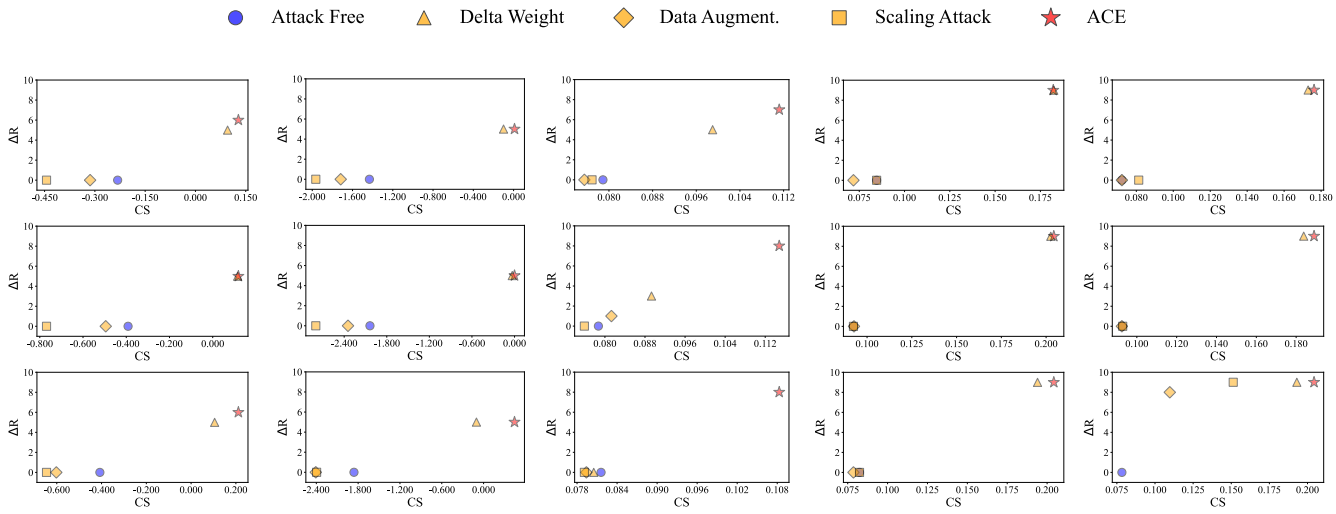
We use *ACC* to measure the performance preserving property of ACE. In particular, ACC measures the classification accuracy of the final global model on testing inputs. Additionally, we will compare the computation cost of ACE with the attack free setting (i.e., a malicious client uses its local training dataset to learn a local model update) to demonstrate the efficiency of ACE.

Compared Baselines. We note that there are no existing studies on attacking contribution evaluation methods in FL. In response, we generalize several existing methods as baselines [1, 38, 74]. First, we consider the scenario where the malicious clients do not launch any attack and follow the procedure outlined in Section 2.1 to learn their local model updates (denoted as *Attack Free*). In addition to Attack Free, we compare ACE with the following three baselines [1, 38, 74]:

- **Delta Weight**: In this baseline [38], the malicious client i crafts local model updates as $\mathbf{g}_i^t = \mathbf{w}^{t-1} - \mathbf{w}^t + \delta$, where $\mathbf{w}^{t-1} - \mathbf{w}^t$ is the global model update in the previous communication round and each entry of δ follows a zero-mean Gaussian distribution with a standard deviation $\sigma = 5 \times 10^{-5}$.

- **Data Augment**: This baseline utilizes data augmentation to increase the sizes of local training datasets of the malicious clients. In particular, each malicious client randomly rotates, scales, and crops the data samples from its local training dataset to form a new dataset, and then merges the newly generated dataset with the original one. At each communication round, each malicious client learns its local model update using the augmented dataset, aiming to increase its contribution [74].

- **Scaling Attack**: In this baseline, each malicious client first learns a local model update using its local training dataset. Then the magnitudes of these local model updates are amplified using a scaling attack [1]. In our experiments, the scaling factor is set to 2.



(a) FedSV (b) LOO (c) CFFL (d) GDR (e) RFFL

Figure 2: Comparing the contribution score *CS* and rank gain ΔR of the attacker when using ACE and baselines under three datasets, i.e., MNIST (first row), CIFAR-10 (second row), and Tiny-ImageNet (third row), and five contribution evaluation methods, i.e., FedSV, LOO, CFFL, GDR, and RFFL. The data partition method is CLA (a heterogeneous setting). Our results show ACE is more effective than baselines. The results for data partitions UNI and POW are in Figure 3 and 4 of Appendix A.

Contribution Evaluation Methods. We consider five state-of-the-art contribution evaluations that can be employed by the server. We briefly introduce these methods as follows. The detailed description can be found in Appendix B.3.

- **FedSV** [65]: FedSV is a game theory based contribution evaluation. It measures the contribution of each client at each communication round following two steps. It first calculates an empirical loss evaluated on a validation dataset held by the server. Then FedSV computes contributions by splitting the empirical loss among all clients using the Shapley value. FedSV uses FedAvg [42] as the aggregation rule.

- **LOO** [65, 81]: This is a game theory based contribution evaluation. To quantify the contribution from a client i, it computes two global models at each communication round, with and without the client's local model update. Then the client's contribution is calculated as the difference of the values of an empirical loss function evaluated on a validation dataset using these two global models. LOO uses FedAvg [42] as the aggregation rule.

- **CFFL** [41]: This is an individual performance based contribution evaluation. The server calculates the contribution of a client i using the accuracy of its local model evaluated on a validation dataset. The aggregation rule of CFFL is a variant of FedAvg [42], which considers both the data size and the number of classes of a client. Specifically, when the data size is imbalanced, the aggregation rule follows FedAvg. When the class numbers are imbalanced, the aggregation rule assigns weights to the local model updates of clients according to the number of classes in their local training datasets.

- **GDR** [73]: This is an individual performance based contribution evaluation. The server utilizes the cosine distance between the aggregated global model updates and local model updates to estimate the Shapley value, and assigns the Shapley value to each client as its contribution. GDR uses a weighted sum of all local model updates to compute the global model update, where the weight associated with each client's local model update is the rolling mean of its contribution.

- **RFFL** [72]: This is an individual performance based contribution evaluation. The contribution of a client is quantified by the cosine similarity between the client's local model update and the aggregated global model update. RFFL uses a similar aggregation rule as RFFL.

ACE Setup. By default, we consider a single malicious client. We note that when a client has a high contribution even if there is no attack, it is very challenging to improve the ranking of its contribution score. In response, by default, we select a client whose contribution is the lowest without attacks as the malicious client. Unless otherwise mentioned, the buffer length is set to $m = 3$, the threshold is chosen as $l = 1$ to mitigate prediction error, and the tunable amplifying coefficient is set as $c = 1$. When the server employs contribution evaluation methods using cosine distance (GDR and RFFL) and validation datasets (FedSV, LOO, and CFFL), we set the

local evolution rounds to one and two, respectively. During the preliminary iteration and when the L-BFGS algorithm incurs a large prediction error, the malicious client executes the delta weight attack (using the global model update in the previous communication round with Gaussian noise). In Section 6.3, we perform ablation analysis and evaluate alternative strategies when the L-BFGS incurs a large prediction error.

6.2 Experimental Results

We evaluate ACE using the metrics in Section 6.1. When the context is clear, we drop the subscript i for the malicious client.

ACE is Effective and Aggregation Rule Independent. Figure 2 and 3, 4 (in Appendix A) compare the contribution score and rank again of the malicious client when using ACE and baselines. We have the following key observations. First, the contribution score and rank gain of ACE consistently outperform those of all baselines. For example, when the server employs CFFL [41] as the contribution evaluation method and the data partition is CLA, the malicious client is perceived with contribution score 0.108 and rank gain 8 by the server for the classification task on Tiny-ImageNet dataset. This moves the malicious client from being the lowest-contributing to becoming the second highest-contributing client. All the baselines in this case, however, give rank gain zero, indicating that the malicious client is still evaluated as the lowest-contributing client by the server under those baseline attacks. Our second observation is that under non-i.i.d. data distribution, some baselines yield the same rank gain as ACE, e.g., when the server employs RFFL and the data partition is CLA. However, we note that the malicious client attains the highest contribution score using ACE in these scenarios. Therefore, ACE is more effective compared with all baselines under non-i.i.d. data distributions.

We note that ACE is effective across all contribution evaluation methods in Figure 2, which utilize different aggregation rules. This indicates that ACE is independent of the aggregation rules used in FL.

ACE Preserves the Performance. We show the accuracy on testing inputs of the final global model in Table 1. We observe that the accuracy of the final global model under ACE remains within a negligible 1% deviation from highest ACC in the worst-case. Therefore, ACE preserves the performance of the final global model learned in FL. Note that ACE can potentially lead to the higher ACC under some contribution evaluation methods than Attack Free, e.g., FedSV, CFFL, and GDR. The reason is that ACE replaces the malicious clients' original local model updates (that should have been learned from their low-quality local training data) with the predicted global model update which integrates updates from other clients learned using local training data of higher qualities.

ACE is Efficient. We compare the computation cost of ACE with Attack Free (when the malicious client utilizes its local training dataset to learn a local model update). Table 2 shows the ratio between the computation cost of learning a local model update using a local training dataset and ACE, i.e., (computation cost of learning a local model update)/(computation cost of ACE). Our key observation is that ACE is significantly more efficient than learning a local model update using the local training dataset. The reason is that it is very efficient to predict the future global model. Note that ACE does not incur extra communication cost since the malicious client simply replaces its local model update with the predicted global model update.

6.3 Ablation Analysis

We perform ablation analysis on the CIFAR-10 dataset under CFFL and RFFL contribution evaluation methods. Here CFFL and RFFL are chosen as the representative contribution evaluation methods for servers with and without validation datasets, respectively.

Effect of Client Number N. Table 3 evaluates the effect of client number N. We compare the contribution score CS and the rank gain ΔR of ACE with Attack Free (the malicious client uses its local training dataset to learn a local model update). We have the following observations. First, ACE is consistently effective. In particular, ACE consistently makes the malicious client, who has the lowest contribution score without attack, the high-contributing client regardless of the client number N. In our experiments, the malicious client is evaluated as the highest-contributing client ($\Delta R = 9, 19, 49$) in 14 out of 18 settings. This observation aligns with our results shown in Figure 2, and indicates that the design of ACE is insensitive to FL systems with different numbers of clients. We note that ACE consistently preserves the ACC, i.e., the ACC under ACE is similar to that of Attack Free. In other words, ACE preserves the performance of the final global model for FL with different number of clients.

Effect of the Threshold Based Filtering. We evaluate the effect of the threshold based filtering discussed in Section 4.2.2, which is used when ACE potentially incurs high prediction error. We compare the ACC, CS, ΔR with and without the threshold based filtering in Table 4. We observe that when the data distribution is non-i.i.d. (e.g., POW data partition), the threshold based filtering can significantly improve the attack effectiveness and the accuracy of the final global model (ACC) under CFFL. Hence, the threshold based filtering is necessary to the design of ACE.

Effect of Amplifying Coefficient c. In Table 5, we evaluate the effect of the amplifying coefficient c. We observe that as we increase the coefficient c, the contribution score of the malicious client increases under all data partitions when the server utilizes cosine distance to measure the contributions (RFFL). This observation aligns with our theoretical analysis

Table 1: This table summarizes the ACC of the final global model learned with ACE and all baselines, evaluated under three data partitions (UNI, POW, and CLA), three datasets (MNIST, CIFAR-10, and Tiny ImageNet), and five contribution evaluation methods (FedSV, LOO, CFFL, GDR, and RFFL). The accuracy of the final global model under ACE remains within a negligible 1% deviation from highest ACC in the worst-case. Thus ACE preserves the accuracy of the final global model.

Contribute Evaluation	Attack	MNIST UNI	MNIST POW	MNIST CLA	CIFAR-10 UNI	CIFAR-10 POW	CIFAR-10 CLA	Tiny-ImageNet UNI	Tiny-ImageNet POW	Tiny-ImageNet CLA
FedSV	Attack Free	95.86%	95.69%	89.89%	71.16%	70.82%	56.32%	46.37%	47.84%	44.98%
	Delta Weight	95.68%	95.46%	90.39%	70.89%	71.02%	57.16%	46.10%	47.80%	45.27%
	Data Augment.	95.87%	95.67%	89.88%	71.63%	70.27%	56.05%	46.77%	48.37%	45.26%
	Scaling Attack	95.85%	95.81%	89.66%	71.58%	71.01%	55.29%	46.59%	48.07%	45.01%
	ACE	95.81%	95.53%	91.27%	71.30%	71.45%	57.60%	46.35%	48.23%	45.94%
LOO	Attack Free	95.86%	95.69%	89.89%	71.16%	70.82%	56.32%	46.37%	47.84%	44.98%
	Delta Weight	95.88%	95.69%	90.39%	70.89%	71.02%	57.16%	46.10%	47.80%	45.27%
	Data Augment.	95.94%	95.73%	89.88%	71.63%	70.27%	56.05%	46.77%	48.37%	45.26%
	Scaling Attack	95.78%	95.66%	89.66%	71.58%	71.01%	55.29%	46.59%	48.07%	45.01%
	ACE	96.06%	95.51%	91.27%	71.30%	71.45%	57.60%	46.35%	48.23%	45.94%
CFFL	Attack Free	96.83%	94.71%	79.43%	71.84%	60.65%	49.99%	51.77%	48.23%	39.96%
	Delta Weight	96.58%	91.89%	82.26%	70.66%	59.37%	50.62%	51.30%	44.18%	40.54%
	Data Augment.	97.44%	94.49%	79.19%	73.08%	60.93%	50.62%	51.92%	47.83%	40.04%
	Scaling Attack	97.01%	94.59%	79.28%	71.55%	60.41%	49.91%	52.22%	44.23%	39.87%
	ACE	96.61%	95.35%	83.18%	70.44%	62.03%	52.45%	51.53%	49.20%	42.02%
GDR	Attack Free	96.26%	96.23%	85.41%	70.97%	71.33%	56.66%	51.80%	51.96%	44.78%
	Delta Weight	96.84%	96.43%	89.02%	70.32%	70.76%	59.18%	52.19%	52.57%	46.01%
	Data Augment.	96.43%	96.18%	87.42%	72.01%	71.12%	57.38%	51.79%	52.04%	44.84%
	Scaling Attack	96.26%	96.23%	85.42%	71.01%	71.36%	56.63%	51.84%	51.89%	44.78%
	ACE	96.78%	96.53%	89.12%	70.27%	70.60%	59.23%	52.64%	52.77%	46.61%
RFFL	Attack Free	96.78%	96.85%	92.67%	71.78%	71.03%	57.66%	52.35%	52.43%	46.72%
	Delta Weight	96.66%	96.85%	91.83%	70.69%	71.07%	56.95%	51.89%	52.49%	46.84%
	Data Augment.	96.25%	96.08%	92.67%	71.84%	71.04%	57.60%	51.83%	52.50%	46.31%
	Scaling Attack	95.96%	95.97%	91.73%	71.73%	71.07%	56.60%	50.84%	52.50%	46.17%
	ACE	96.64%	96.87%	92.30%	70.72%	70.90%	57.36%	51.75%	52.31%	46.54%

Table 2: This table shows the ratio between the computation costs of using a local training dataset to learn a local model update and ACE. The data partition method is UNI. We observe that ACE is significantly more efficient.

Dataset	FedSV	LOO	CFFL	GDR	RFFL
MNIST	30.88×	30.88×	7.48×	16.15×	18.26×
CIFAR-10	270.81×	270.81×	21.25×	86.48×	101.44×
Tiny-ImageNet	35.35×	35.35×	13.26×	29.22×	24.79×

in Section 5. Moreover, ACE preserves the ACC under all data partitions with different choices of amplifying coefficient c.

More Experiments. Due to space constraint, we defer the results of the attack effectiveness under UNI and POW data partitions under all datasets (in Figure 3 and 4), effects of strategies for preliminary iteration and threshold based filtering (in Table 7), buffer length (in Figure 5), local evolution (in Figure 6), the fraction of clients selected by the server in each communication round (in Table 8), and the fraction of malicious clients (in Figure 7) to Appendix A. We also show experimental results when the attacker does not know the contribution evaluation methods deployed by the server in Appendix A. In summary, our results show that ACE is relatively insensitive to these factors, and is effective in boosting the malicious clients' contributions perceived by the server.

7 Countermeasures to ACE and Evaluations

7.1 Countermeasures to ACE

We focus on defense developed for FL against model poisoning attacks to thwart ACE. According to [53], existing defenses against model poisoning attacks can be divided into three categories: *performance based defense* [7, 10, 36, 71], *distance based defense* [3, 6, 13, 19, 21, 61, 67, 80] and *statistics based defense* [25, 43, 44, 49, 69, 75]. As demonstrated in Section 6, ACE successfully deceives the methods that utilize validation accuracy to measure contribution (e.g., FedSV and LOO), which invalidates the performance based defenses. We thus mainly focus on distance and statistics based defenses. In particular, we choose Multi-Krum [3], Trimmed-Mean [75], FABA [67], Sniper [6], and Foolsgold [21] as representative countermeasures. These defenses do not require a validation dataset in the server and are commonly employed by the community to defend against poisoning and Sybil attacks in FL.

Table 3: This table evaluates the effect of the number of clients in FL. Regardless of the number of clients, ACE consistently makes the attacker the highest-contributing agent. AF is the abbreviation for Attack Free.

Contribution evaluation	Data partition	#Clients	*CS* AF	*CS* ACE	Relative improv.	ΔR
CFFL	UNI	10	0.099	0.105	*+6.41%*	9
		20	0.050	0.053	*+7.14%*	19
		50	0.020	0.023	*+19.69%*	49
	POW	10	0.086	0.106	*+22.73%*	8
		20	0.043	0.056	*+30.02%*	19
		50	0.016	0.024	*+54.03%*	49
	CLA	10	0.079	0.115	*+45.80%*	8
		20	0.040	0.058	*+47.21%*	18
		50	0.016	0.024	*+49.94%*	49
RFFL	UNI	10	0.096	0.192	*+100.26%*	9
		20	0.048	0.124	*+160.88%*	19
		50	0.018	0.060	*+237.81%*	49
	POW	10	0.045	0.196	*+335.49%*	9
		20	0.021	0.125	*+502.81%*	19
		50	0.007	0.064	*+872.81%*	49
	CLA	10	0.096	0.189	*+97.41%*	9
		20	0.045	0.123	*+174.98%*	16
		50	0.017	0.064	*+284.43%*	49

- **Multi-Krum [3].** We choose Multi-Krum [3] from the category of distance based defense as the first countermeasure to ACE, which has been widely used to mitigate model poisoning attacks in FL. Multi-Krum focuses on identifying and eliminating local model updates from malicious clients by analyzing the distance between local model updates from different clients. We extend it to defend against ACE. In particular, the server computes the Euclidean distance between each pair of local model updates at each communication round to measure how similar or dissimilar the local model updates are to each other. The server then considers the k most dissimilar local model updates, i.e., k local model updates that has the largest sum of distance to other local model updates, to be sent by the malicious clients.

- **Trimmed-Mean [75].** We extend Trimmed-mean from the class of statistics based defense as the second countermeasure to ACE. In each communication round, Trimmed-Mean first sorts each dimension of the local model updates from all clients. Subsequently, it identifies the largest k and smallest k entries for each dimension in the sorted local model updates, considering these entries to be potentially sent by malicious clients. Finally, the top k clients that are identified by the server most frequently across all dimensions are considered to be malicious by the server.

- **FABA [67].** We apply FABA [67], an efficient defense algorithm against Byzantine attacks in FL as the third countermeasure. In each communication round, the server first computes the mean of local model updates, and then computes the difference between the mean and each local model update. The server subsequently identifies the local model update with the largest difference from the mean as malicious.

- **Sniper [6].** We consider a clustering-based approach, Sniper, to defend against ACE. Sniper clusters benign local model updates by solving a maximum clique problem in each communication round. Specifically, the server first calculates the Euclidean distances between each pair of local model updates, then constructs a graph such that each local model update is a vertex in the graph. An edge exists if the Euclidean distance between two local model updates is smaller than a pre-determined threshold. In the end, the server finds the maximum clique in the graph and identifies vertices (local model updates) in the clique as benign. The remaining local model updates are identified as malicious.

- **Foolsgold [21].** We evaluate ACE against Foolsgold [21], a Sybil detection countermeasure. Foolsgold maintains aggregate historical updates of local model updates from

Table 4: This table shows the effect of the threshold based filtering. We compare the ACC, *CS*, and ΔR of ACE with and without the threshold based filtering. We observe that solely relying on future global model prediction is not sufficient for attack effectiveness. The prediction error mitigation with threshold based filtering is necessary to guarantee the effectiveness of ACE.

Contri. Eval.	Metric	UNI with	UNI without	POW with	POW without	CLA with	CLA without
CFFL	ACC	70.44%	70.50%	62.03%	56.87%	52.45%	52.51%
	CS	0.1051	0.1049	0.1055	0.0526	0.1148	0.1149
	ΔR	9	9	8	0	8	8
RFFL	ACC	70.72%	70.72%	70.90%	70.90%	57.36%	57.36%
	CS	0.1917	0.1917	0.1963	0.1963	0.1890	0.1890
	ΔR	9	9	9	9	9	9

Table 5: This table presents the effect of amplifying coefficient c when the server employs RFFL as the contribution evaluation method under all data partitions. The contribution score of the malicious client increases as the amplifying coefficient increases. AF is the abbreviation of Attack Free.

Data Partition	Metric	AF	$c = 1$	$c = 1.5$	$c = 2$	$c = 2.5$
UNI	ACC	71.78%	70.72%	70.90%	70.83%	70.59%
	CS	0.096	0.192	0.199	0.205	0.207
POW	ACC	71.03%	70.90%	70.78%	70.70%	70.73%
	CS	0.045	0.196	0.204	0.211	0.212
CLA	ACC	57.66%	57.36%	58.33%	59.70%	60.13%
	CS	0.093	0.189	0.196	0.203	0.205

each client to better estimate the similarity of the overall contributions made by clients. Foolsgold detects malicious clients by calculating the pairwise cosine similarity of historical updates as a representation of how strongly two clients are acting similarly. The clients with high pairwise cosine similarity are identified as malicious.

- **Random Guess.** The last countermeasure considered in this paper is Random Guess. In particular, the server randomly selects k local model updates as manipulated by the malicious clients.

We note that k is a hyper-parameter for those countermeasures. We consider a strong defense scenario where the server knows the total number of local model updates (which is used to set k) from the malicious clients.

7.2 Evaluations of Countermeasures to ACE

We empirically evaluate the six detection countermeasures against ACE under our default setting.

Evaluation Metrics. We use three metrics, **precision**, **recall**, and **F1-Score** to evaluate the detection performance of Multi-Krum, Trimmed-Mean, and Random Guess when the attacker launches ACE.

- **Precision.** Precision is defined as the fraction of local model updates that are indeed from the malicious clients among all predicted ones.
- **Recall.** Recall is the fraction of local model updates from the malicious clients that are successfully predicted by a countermeasure.
- **F1-Score.** F1-Score measures the harmonic mean of precision and recall, i.e., $\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$.

We note that each defense will detect local model updates from the malicious clients in each communication round. Thus, we report the average precision, recall, and F1-Score overall all communication rounds.

Evaluation Results. We report the precision, recall, and F1-score when the server utilizes Multi-Krum, Trimmed-Mean, FABA, Sniper, Foolsgold, or Random Guess to mitigate ACE in Table 6 on CIFAR-10, where the contribution evaluation methods are CFFL and RFFL. For UNI data distribution, we observe that the malicious client is rarely detected by any of these countermeasures, indicating that none of these are effective and adequate to thwart ACE. In particular, the performances of these detection methods are worse than the naive Random Guess. The reason is that the local model updates sent by the malicious client do not significantly diverge from the global model update, leading the server to perceive the malicious client as a benign client. For POW and CLA, two more realistic data partitions, there is only a marginal increase in detection performance. However, the performance remains worse than Random Guess, highlighting the stealthiness of ACE in non-i.i.d. settings. We defer the evaluation results when parameter $c \geq 1$ to Appendix A.

We note that the insights of detecting malicious clients behind Multi-Krum, Trimmed Mean, FABA, Sniper, and Foolsgold can be classified into two categories, whereas neither of these was adequate to detect ACE. The inadequacy of Multi-Krum, Trimmed Mean, FABA, and Sniper can be attributed to their insight that malicious clients likely to send local model updates significantly different from other clients' updates. Foolsgold identifies malicious clients whose local model updates are too similar to others as malicious. However, as discussed in Section 3.2, our ACE does not exhibit either excessive similarity or dissimilarity to benign clients, and thus can evade detection by all these countermeasures. This highlights the urgent need to develop new mitigation strategies to defend against ACE.

Table 6: This table summarizes the precision, recall, and F1-score when Multi-Krum, Trimmed-Mean, FABA, Sniper, Foolsgold, or Random Guess is employed to mitigate ACE under both i.i.d. (i.e., UNI) and non-i.i.d. (i.e., POW and CLA) data distributions. We observe that none of these countermeasures are effective and adequate to defend ACE. None of these countermeasures are effective and adequate to defend ACEin all data distributions. If Precision and Recall are 0, F1-Score is not defined and denoted as N/A.

Data Partition	Countermeasure	CFFL			RFFL		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
UNI	Multi-Krum	0.017	0.017	0.017	0.017	0.017	0.017
	Trimmed-Mean	0.017	0.017	0.017	0.017	0.017	0.017
	FABA	0.017	0.017	0.017	0	0	N/A
	Sniper	0.017	0.017	0.017	0	0	N/A
	Foolsgold	0	0	N/A	0	0	N/A
	Random Guess	0.100	0.100	0.100	0.100	0.100	0.100
POW	Multi-Krum	0.017	0.017	0.017	0.017	0.017	0.017
	Trimmed-Mean	0.017	0.017	0.017	0.017	0.017	0.017
	FABA	0.017	0.017	0.017	0.017	0.017	0.017
	Sniper	0.022	0.033	0.027	0	0	N/A
	Foolsgold	0	0	N/A	0	0	N/A
	Random Guess	0.100	0.100	0.100	0.100	0.100	0.100
CLA	Multi-Krum	0.017	0.017	0.017	0.017	0.017	0.017
	Trimmed-Mean	0.017	0.017	0.017	0	0	N/A
	FABA	0.017	0.017	0.017	0.017	0.017	0.017
	Sniper	0.022	0.033	0.026	0	0	N/A
	Foolsgold	0	0	N/A	0	0	N/A
	Random Guess	0.100	0.100	0.100	0.100	0.100	0.100

8 Discussion and Limitation

Untargeted Poisoning and Backdoor Attacks in FL. Attacks against FL have been extensively studied, such as untargeted poisoning attacks [8, 18, 30, 51, 60, 70, 86] and backdoor attacks [1, 2, 58, 63, 68, 82]. For image classification tasks, the final global model of FL learned under untargeted poisoning attacks produces incorrect predictions indiscriminately on the testing inputs, whereas the global model learned under backdoor attacks outputs attacker-chosen class when

the inputs are embedded with the attacker-chosen triggers. We note that the untargeted poisoning attacks have a different goal from our attack. Scaling Attack [1] is a state-of-the-art backdoor attack. We extend it to attack contribution evaluation methods in FL. Our experimental results in Section 6 demonstrate that ACE consistently outperforms Scaling Attack. The reason is that backdoor attacks are not designed to improve the contribution of a malicious client.

Defenses against Untargeted Poisoning and Backdoor Attacks in FL. To mitigate these attacks, various defenses have been proposed [3, 7, 9, 13, 21, 32, 43, 47, 48, 50, 58, 66, 75] to identify and remove the local model updates from malicious clients. However, most of these defenses are ineffective to mitigate ACE. The reason is that these existing defenses identify the malicious clients by detecting abnormal local model updates from the clients. For example, FLTrust [7] assigns a trust score to each local model update by comparing its direction with the global model update. As the deviation in the directions between the local model update and the global model increases, the client's trust score decreases, indicating that it is more likely to be malicious. Our design of ACE, however, does not lead to abnormal local model updates. Instead, the manipulated local model updates sent by the malicious clients using ACE tend to be similar to the global model update in order to elevate their contributions. This makes ACE particularly challenging to be detected as shown in Section 7.

Limitation of ACE. One limitation of ACE is how to select the *optimal* parameters and strategies for ACE. Our evaluation results in Figure 5 and Table 7 show that the effectiveness of ACE is relatively insensitive to these choices under most of the cases with different datasets, data partitions, and contribution evaluation methods. In order to further enhance the attack effectiveness of ACE, a future direction is to explore efficient techniques to compute the optimal parameters and strategies.

9 Conclusion and Future Work

In this paper, we proposed a new model poisoning attack called ACE on contribution evaluation methods in federated learning (FL). We showed that the malicious clients in FL could utilize ACE to predict the future global model update with small errors, and elevate their contributions evaluated by the server in FL. We empirically evaluated ACE using five state-of-the-art contribution evaluation methods with three datasets and three data partition methods. Our results showed that ACE effectively increased the contributions of malicious clients with negligible costs, and it preserved the performance of the final global model learned from FL. We also evaluated ACE against six countermeasures, and showed that none of them can effectively mitigate ACE. An interesting future work is to develop new mitigation strategies against our attack.

Acknowledgement

We are grateful for the time and insightful comments provided by the reviewers and shepherd, which have significantly enhanced the quality of our work.

This work is partially supported by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-23-1-0208, National Science Foundation (NSF) under grants IIS 2229876, Office of Naval Research (ONR) under grant N00014-23-1-2386, DARPA GARD, the National Aeronautics and Space Administration (NASA) under grant No.80NSSC20M0229, Alfred P. Sloan Fellowship, and the Amazon research award.

This work is supported in part by funds provided by the National Science Foundation, Department of Homeland Security, and IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or its federal agency and industry partners.

References

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *AISTATS*, pages 2938–2948. PMLR, 2020.

[2] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *NeurIPS*, 32, 2019.

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *NeurIPS*, 30, 2017.

[4] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J Sci Comput*, 16(5):1190–1208, 1995.

[5] Richard H Byrd, Jorge Nocedal, and Robert B Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1-3):129–156, 1994.

[6] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. Understanding distributed poisoning attack in federated learning. In *ICPADS*, pages 233–239. IEEE, 2019.

[7] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.

[8] Xiaoyu Cao and Neil Zhenqiang Gong. Mpaf: Model poisoning attacks to federated learning based on fake clients. In *CVPR*, pages 3396–3404, 2022.

[9] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *IEEE S&P*, pages 1366–1383. IEEE, 2023.

[10] Xinyang Cao and Lifeng Lai. Distributed gradient descent algorithm robust to an arbitrary number of Byzantine attackers. *IEEE Trans. Signal Process.*, 67(22):5850–5864, 2019.

[11] Zheng Chai, Hannan Fayyaz, Zeshan Fayyaz, Ali Anwar, Yi Zhou, Nathalie Baracaldo, Heiko Ludwig, and Yue Cheng. Towards taming the resource and data heterogeneity in federated learning. In *OpML*, pages 19–21, 2019.

[12] Yiqiang Chen, Xiaodong Yang, Xin Qin, Han Yu, Piu Chan, and Zhiqi Shen. Dealing with label quality disparity in federated learning. *Federated Learning: Privacy and Incentive*, pages 108–121, 2020.

[13] Tianyue Chu, Alvaro Garcia-Recuero, Costas Iordanou, Georgios Smaragdakis, and Nikolaos Laoutaris. Securing federated sensitive topic classification against poisoning attacks. *arXiv preprint arXiv:2201.13086*, 2022.

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

[15] Yongheng Deng, Feng Lyu, Ju Ren, Yi-Chao Chen, Peng Yang, Yuezhi Zhou, and Yaoxue Zhang. Fair: Quality-aware federated learning with precise user incentive and model aggregation. In *INFOCOM*, pages 1–10. IEEE, 2021.

[16] Zhenan Fan, Huang Fang, Zirui Zhou, Jian Pei, Michael P Friedlander, Changxin Liu, and Yong Zhang. Improving fairness for data valuation in horizontal federated learning. In *ICDE*, pages 2440–2453. IEEE, 2022.

[17] Zhenan Fan, Huang Fang, Zirui Zhou, Jian Pei, Michael P Friedlander, and Yong Zhang. Fair and efficient contribution valuation for vertical federated learning. *arXiv preprint arXiv:2201.02658*, 2022.

[18] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to Byzantine-robust federated learning. In *USENIX Security*, pages 1605–1622, 2020.

[19] Minghong Fang, Jia Liu, Neil Zhenqiang Gong, and Elizabeth S Bentley. AFLGuard: Byzantine-robust asynchronous federated learning. In *ACSAC*, pages 632–646, 2022.

[20] Pedro J Freire, Sasipim Srivallapanondh, Antonio Napoli, Jaroslaw E Prilepsky, and Sergei K Turitsyn. Computational complexity evaluation of neural network applications in signal processing. *arXiv preprint arXiv:2206.12191*, 2022.

[21] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating Sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[22] Liang Gao, Li Li, Yingwen Chen, Wenli Zheng, ChengZhong Xu, and Ming Xu. Fifl: A fair incentive mechanism for federated learning. In *ICPP*, pages 1–10, 2021.

[23] Amirata Ghorbani and James Zou. Data Shapley: Equitable valuation of data for machine learning. In *ICML*, pages 2242–2251. PMLR, 2019.

[24] Andreas Griewank and Andrea Walther. *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. SIAM, 2008.

[25] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in Byzantium. In *ICML*, pages 3521–3530. PMLR, 2018.

[26] Miao Hu, Di Wu, Yipeng Zhou, Xu Chen, and Min Chen. Incentive-aware autonomous client participation in federated learning. *IEEE TPDS*, 33(10):2612–2627, 2022.

[27] Malhar S Jere, Tyler Farnan, and Farinaz Koushanfar. A taxonomy of attacks on federated learning. *IEEE S&P*, 19(2):20–28, 2020.

[28] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *AISTATS*, pages 1167–1176. PMLR, 2019.

[29] Meirui Jiang, Holger R Roth, Wenqi Li, Dong Yang, Can Zhao, Vishwesh Nath, Daguang Xu, Qi Dou, and Ziyue Xu. Fair federated medical image segmentation via client contribution estimation. In *CVPR*, pages 16302–16311, 2023.

[30] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Byzantine-robust learning on heterogeneous datasets via bucketing. *arXiv preprint arXiv:2006.09365*, 2020.

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

[32] Kavita Kumari, Phillip Rieger, Hossein Fereidooni, Murtuza Jadliwala, and Ahmad-Reza Sadeghi. Baybfed: Bayesian backdoor defense for federated learning. *arXiv preprint arXiv:2301.09508*, 2023.

[33] Serge Lang. *Real and functional analysis*, volume 142. Springer Science & Business Media, 2012.

[34] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[35] Li Li, Moming Duan, Duo Liu, Yu Zhang, Ao Ren, Xianzhang Chen, Yujuan Tan, and Chengliang Wang. FedSAE: A novel self-adaptive federated learning framework in heterogeneous systems. In *IJCNN*, pages 1–10. IEEE, 2021.

[36] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen. Abnormal client behavior detection in federated learning. *arXiv preprint arXiv:1910.09933*, 2019.

[37] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Machine Learning and Systems*, 2:429–450, 2020.

[38] Jierui Lin, Min Du, and Jian Liu. Free-riders in federated learning: Attacks and defenses. *arXiv preprint arXiv:1911.12560*, 2019.

[39] Zelei Liu, Yuanyuan Chen, Han Yu, Yang Liu, and Lizhen Cui. Gtg-shapley: Efficient and accurate participant contribution evaluation in federated learning. *ACM Trans. Intell. Syst. Technol.*, 13(4):1–21, 2022.

[40] Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In *Federated Learning: Privacy and Incentive*, pages 240–254. Springer, 2020.

[41] Lingjuan Lyu, Xinyi Xu, Qian Wang, and Han Yu. Collaborative fairness in federated learning. *Federated Learning: Privacy and Incentive*, pages 189–204, 2020.

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

[43] Hamid Mozaffari, Virat Shejwalkar, and Amir Houmansadr. Every vote counts: Ranking-based training of federated learning to resist poisoning attacks. In *USENIX Security*, 2023.

[44] Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125*, 2019.

[45] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Commun. Surv. Tutor.*, 23(3):1622–1658, 2021.

[46] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Trade-off between payoff and model rewards in Shapley-fair collaborative machine learning. *NeurIPS*, 35:30542–30553, 2022.

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

[48] Mustafa Safa Ozdayi, Murat Kantarcioglu, and Yulia R Gel. Defending against backdoors in federated learning with robust learning rate. In *AAAI*, volume 35, pages 9268–9276, 2021.

[49] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE T Signal Proces*, 70:1142–1154, 2022.

[50] Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, and Ahmad-Reza Sadeghi. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. *arXiv preprint arXiv:2201.00763*, 2022.

[51] Virat Shejwalkar and Amir Houmansadr. Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.

[52] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(1):12598, 2020.

[53] Junyu Shi, Wei Wan, Shengshan Hu, Jianrong Lu, and Leo Yu Zhang. Challenges and approaches for mitigating Byzantine attacks in federated learning. In *TrustCom*, pages 139–146. IEEE, 2022.

[54] Yuxin Shi, Han Yu, and Cyril Leung. Towards fairness-aware federated learning. *IEEE TNNLS*, 2023.

[55] Zhuan Shi, Lan Zhang, Zhenyu Yao, Lingjuan Lyu, Cen Chen, Li Wang, Junhao Wang, and Xiang-Yang Li. Fedfaim: A model performance-based fair incentive mechanism for federated learning. *IEEE Trans. Big Data*, 2022.

[56] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[57] Behnaz Soltani, Yipeng Zhou, Venus Haghighi, and John Lui. A survey of federated evaluation in federated learning. *arXiv preprint arXiv:2305.08070*, 2023.

[58] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[59] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. FedProto: Federated prototype learning across heterogeneous clients. In *AAAI*, volume 36, pages 8432–8440, 2022.

[60] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *ESORICS*, pages 480–501. Springer, 2020.

[61] Wei Wan, Jianrong Lu, Shengshan Hu, Leo Yu Zhang, and Xiaobing Pei. Shielding federated learning: A new attack approach and its defense. In *IEEE WCNC*, pages 1–7. IEEE, 2021.

[62] Guan Wang, Charlie Xiaoqian Dang, and Ziye Zhou. Measure contribution of participants in federated learning. In *IEEE BigData*, pages 2597–2604. IEEE, 2019.

[63] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *NeurIPS*, 33:16070–16084, 2020.

[64] Junhao Wang, Lan Zhang, Anran Li, Xuanke You, and Haoran Cheng. Efficient participant contribution evaluation for horizontal and vertical federated learning. In *ICDE*, pages 911–923, 2022.

[65] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A principled approach to data valuation for federated learning. *Federated Learning: Privacy and Incentive*, pages 153–167, 2020.

[66] Chen Wu, Xian Yang, Sencun Zhu, and Prasenjit Mitra. Mitigating backdoor attacks in federated learning. *arXiv preprint arXiv:2011.01767*, 2020.

[67] Qi Xia, Zeyi Tao, Zijiang Hao, and Qun Li. FABA: an algorithm for fast aggregation against byzantine attacks in distributed neural networks. In *IJCAI*, 2019.

[68] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *ICLR*, 2019.

[69] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized Byzantine-tolerant SGD. *arXiv preprint arXiv:1802.10116*, 2018.

[70] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation. In *Uncertainty in Artificial Intelligence*, pages 261–270. PMLR, 2020.

[71] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *ICML*, pages 6893–6901. PMLR, 2019.

[72] Xinyi Xu and Lingjuan Lyu. A reputation mechanism is all you need: Collaborative fairness and adversarial robustness in federated learning. *arXiv preprint arXiv:2011.10464*, 2020.

[73] Xinyi Xu, Lingjuan Lyu, Xingjun Ma, Chenglin Miao, Chuan Sheng Foo, and Bryan Kian Hsiang Low. Gradient driven rewards to guarantee fairness in collaborative machine learning. *NeurIPS*, 34:16104–16117, 2021.

[74] Xinyi Xu, Zhaoxuan Wu, Chuan Sheng Foo, and Bryan Kian Hsiang Low. Validation free and replication robust volume-based data valuation. *NeurIPS*, 34:10837–10848, 2021.

[75] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, pages 5650–5659. PMLR, 2018.

[76] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. A sustainable incentive scheme for federated learning. *IEEE Intell. Syst.*, 35(4):58–69, 2020.

[77] Rongfei Zeng, Shixun Zhang, Jiaqi Wang, and Xiaowen Chu. Fmore: An incentive scheme of multi-dimensional auction for federated learning in mec. In *ICDCS*, pages 278–288. IEEE, 2020.

[78] Jingwen Zhang, Yuezhou Wu, and Rong Pan. Incentive mechanism for horizontal federated learning based on reputation and reverse auction. In *WWW*, pages 947–956, 2021.

[79] Wei Zhang, Zhuo Li, and Xin Chen. Quality-aware user recruitment based on federated learning in mobile crowd sensing. *Tsinghua Science and Technology*, 26(6):869–877, 2021.

[80] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *KDD*, pages 2545–2555, 2022.

[81] Zhebin Zhang, Dajie Dong, Yuhang Ma, Yilong Ying, Dawei Jiang, Ke Chen, Lidan Shou, and Gang Chen. Refiner: A reliable incentive-driven federated learning

system powered by blockchain. *VLDB Endowment*, 14(12):2659–2662, 2021.

[82] Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael Mahoney, Prateek Mittal, Ramchandran Kannan, and Joseph Gonzalez. Neurotoxin: Durable backdoors in federated learning. In *ICML*, pages 26429–26446. PMLR, 2022.

[83] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[84] Shuyuan Zheng, Yang Cao, and Masatoshi Yoshikawa. Secure shapley value for cross-silo federated learning. *VLDB Endowment*, 16(7):1657–1670, 2023.

[85] Zhaohua Zheng, Yize Zhou, Yilong Sun, Zhang Wang, Boyi Liu, and Keqiu Li. Applications of federated learning in smart cities: Recent advances, taxonomy, and open challenges. *Connection Science*, 34(1):1–28, 2022.

[86] Xingchen Zhou, Ming Xu, Yiming Wu, and Ning Zheng. Deep model poisoning attack on federated learning. *Future Internet*, 13(3):73, 2021.

Appendix A Additional Experimental Results

In this appendix, we present more experimental results.

ACE is Effective under UNI and POW data partitions. We show the contribution score versus the rank gain in Figure 3 and 4 under UNI and POW data partitions, respectively. We observe that ACE consistently outperforms all baselines when the data distribution is non-i.i.d. (e.g., POW). We note that, in a few cases, ACE exhibits slightly lower attack performance compared with the baseline using data augmentation when the data is uniformly distributed among all clients. The reason is that UNI data partition yields an i.i.d. data distribution among the clients. By executing data augmentation, the malicious clients expand their local training datasets and disrupt the i.i.d. pattern, thereby gaining extra advantage compared with other clients.

Effect of the Strategies for Preliminary Iteration & Threshold based filtering. We evaluate the effect of the strategies that can be taken for preliminary iteration and threshold based filtering in Table 7. In particular, we consider two strategies for the malicious clients: denoted as *strategy one* (s_1) and *strategy two* (s_2). By using strategy one, the malicious clients execute the delta weight attack. When taking strategy two, the malicious clients learn local model updates using their local training datasets. These two strategies yield four possible combinations of strategies that can be taken for preliminary iteration and threshold based filtering as follows:

- $s_1 \times s_1$: Strategy one for both preliminary iteration and threshold based filtering.
- $s_1 \times s_2$: Strategy one for preliminary iteration and strategy two for threshold based filtering.
- $s_2 \times s_1$: Strategy two for preliminary iteration and strategy one for threshold based filtering.
- $s_2 \times s_2$: Strategy two for both preliminary iteration and threshold based filtering.

We observe that overall the combination $s_2 \times s_2$ provides the malicious clients the best performance in terms of accuracy, contribution score, and rank gain.

Effect of Buffer Length m. We evaluate the effect of buffer length m by varying it from two to five. We compare ACC and the rank gain with Attack Free (buffer length $m = 0$) in Figure 5. We observe that ACE with non-zero buffer lengths significantly improve the rank gain, and hence the contributions of the malicious clients, without degrading ACC. Furthermore, as we vary the buffer length from two to five, ACE exhibits a consistent rank gain.

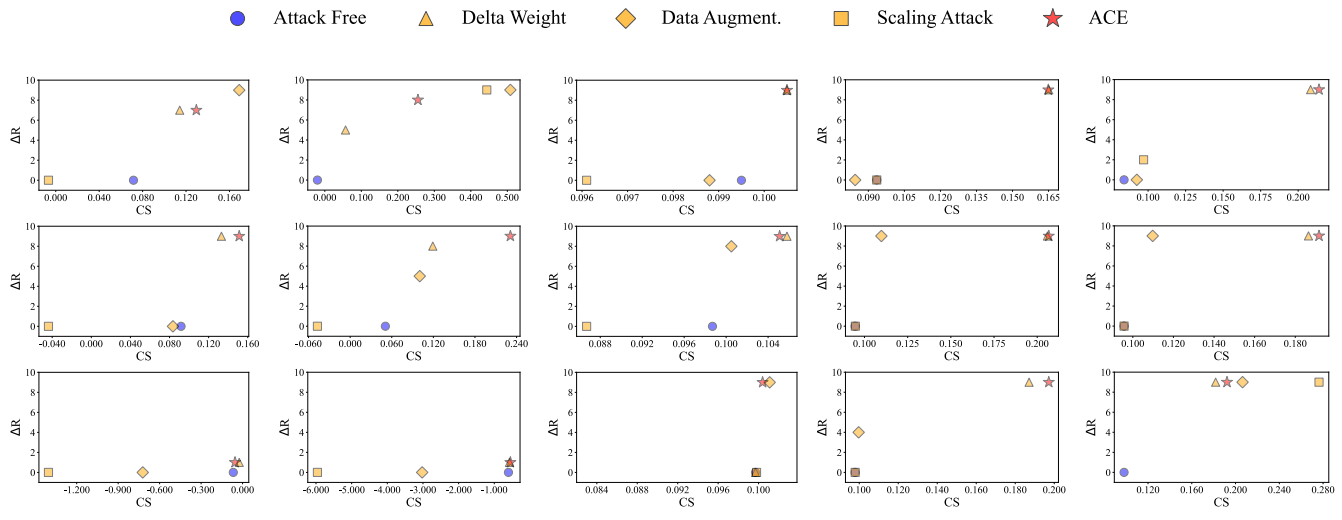
Effect of Local Evolution. Figure 6 evaluates the effect of local evolution (see Section 4.2.3) when validation accuracy is used to measure contribution. We have the following observations. First, ACE is effective to increase the rank gains for the malicious clients by predicting the future global model. However, running the L-BFGS algorithm with different number of rounds can lead to distinct contribution score. For CFFL, the local evolution decreases the contribution score under all three data partitions. We observe that predicting the global model update for future two communication rounds yields the best performance under FedSV. While the contribution scores generally increases when the malicious clients run more rounds of the L-BFGS algorithm under UNI data partition for LOO, the pattern on POW and CLA data partitions is similar to CFFL and FedSV.

Effect of Client Selection. In Table 8, we evaluate the effect of the fraction of clients selected at each communication round. We vary the fraction of clients that is selected by the server from 50% to 100%. We make the following observations. First, ACE is consistently effective under all fractions of client selections in terms of ACC, CS, and ΔR. Therefore, ACE is insensitive to the client selection used by the server. Furthermore, as the fraction of clients being selected at each communication round increases, the effectiveness of ACE increases. The reason is that the malicious clients can make more precise predictions of the global model updates, and therefore make the local model updates of malicious clients better aligned with the global model update.

Effect of the Fraction of Malicious Clients. In the following, we evaluate the effect of the fraction of malicious clients. We vary the fraction of malicious clients from 10% to 30%, and present the accumulated CS (the summation of contribution scores of all malicious clients) in Figure 7. We observe that the accumulated CS using ACE is always higher than the Attack Free case under all data partitions when RFFL is used for contribution evaluation. This indicates that ACE is effective under all settings as we vary the fraction of malicious clients.

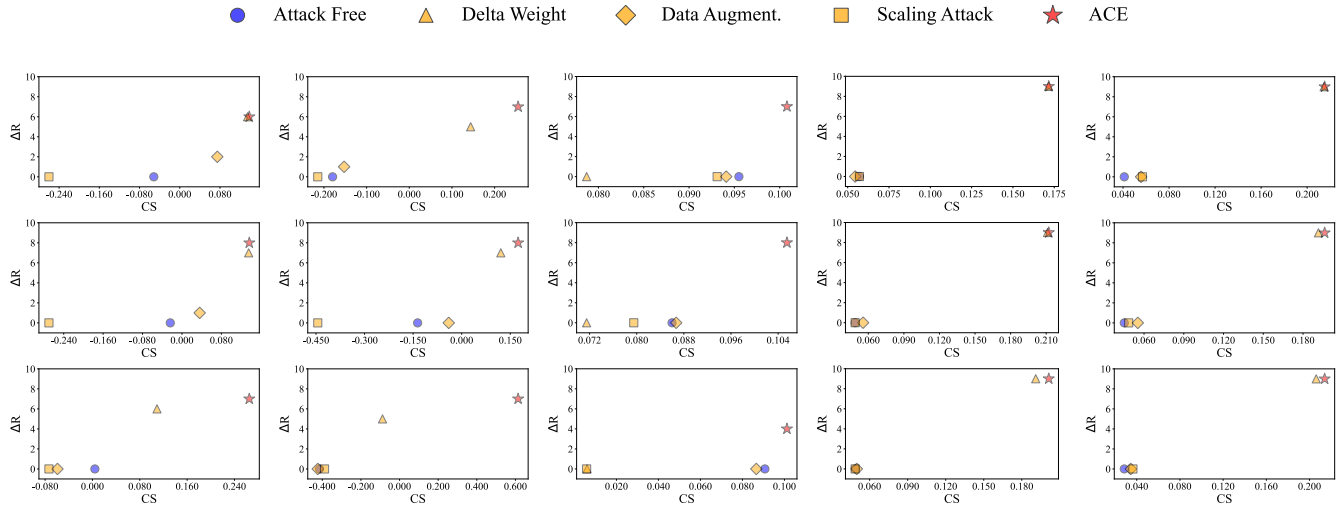
Effect of Different c on Defense Performances. We present the precision, recall and F1-score when the server utilizes Multi-Krum, Trimmed-Mean, FABA, Sniper, Foolsgold to mitigate ACE in Table 9, where we apply ACE under UNI data partition with RFFL contribution evaluation method employed. We observe that the defense methods show marginal performance gain as c increases. This indicates the stealthiness of ACE towards the choice of c.

Experimental Results When the Attacker is Unware of the Contribution Evaluation Methods. In Figure 8, we evaluate the scenario where the attacker does not know the specific contribution evaluation method used by the server. Specifically, the attacker makes a guess on the contribution evaluation method used by the server, which may not necessarily be identical to the method employed. Our results indicate that the client launching ACE can still successfully elevate its contribution evaluated by the server even when it is unaware of the contribution evaluation method.



(a) FedSV (b) LOO (c) CFFL (d) GDR (e) RFFL

Figure 3: Comparing the contribution score *CS* and rank gain ΔR of the attacker when using ACE and baselines under three datasets, i.e., MNIST (first row), CIFAR-10 (second row), and Tiny-ImageNet (third row), and five contribution evaluation methods, i.e., FedSV, LOO, CFFL, GDR, and RFFL. The data partition method is UNI (i.i.d. data distribution). Our results show ACE is more effective than baselines under most of the contribution evaluation methods.



(a) FedSV (b) LOO (c) CFFL (d) GDR (e) RFFL

Figure 4: Comparing the contribution score *CS* and rank gain ΔR of the attacker when using ACE and baselines under three datasets, i.e., MNIST (first row), CIFAR-10 (second row), and Tiny-ImageNet (third row), and five contribution evaluation methods, i.e., FedSV, LOO, CFFL, GDR, and RFFL. The data partition method is POW (non-i.i.d. data distribution). Our results show ACE is consistently more effective than baselines.

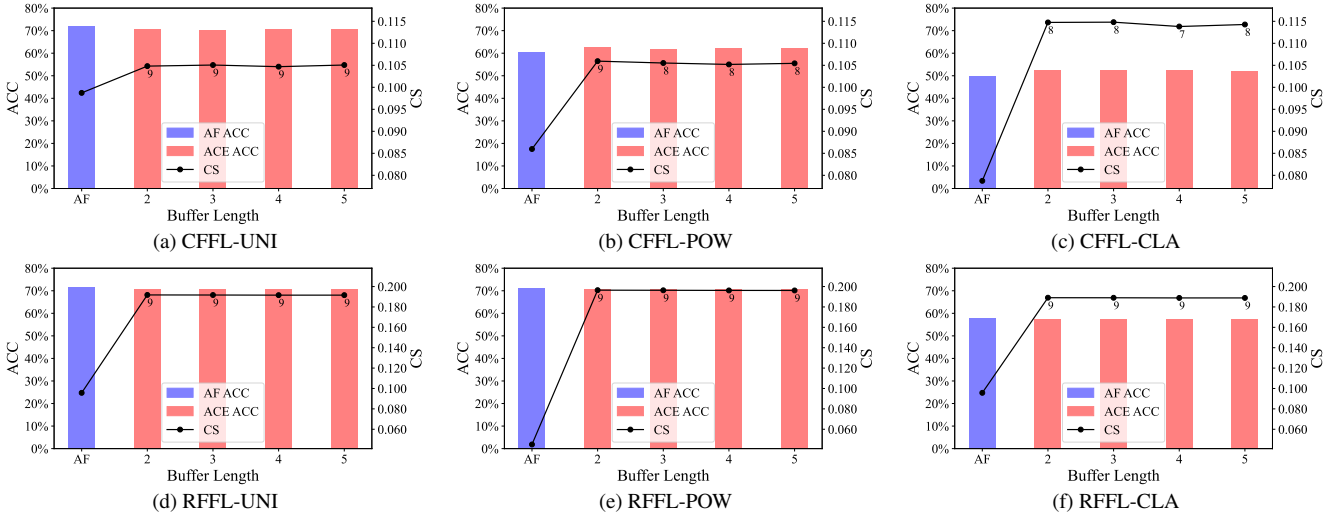


Figure 5: Ablation study with different buffer lengths $m = 2, 3, \ldots, 5$. The numbers annotated in the figure are the rank gains. `ACE` with non-zero buffer lengths significantly improves the rank gain, without degrading the accuracy. AF is the abbreviation for Attack Free.

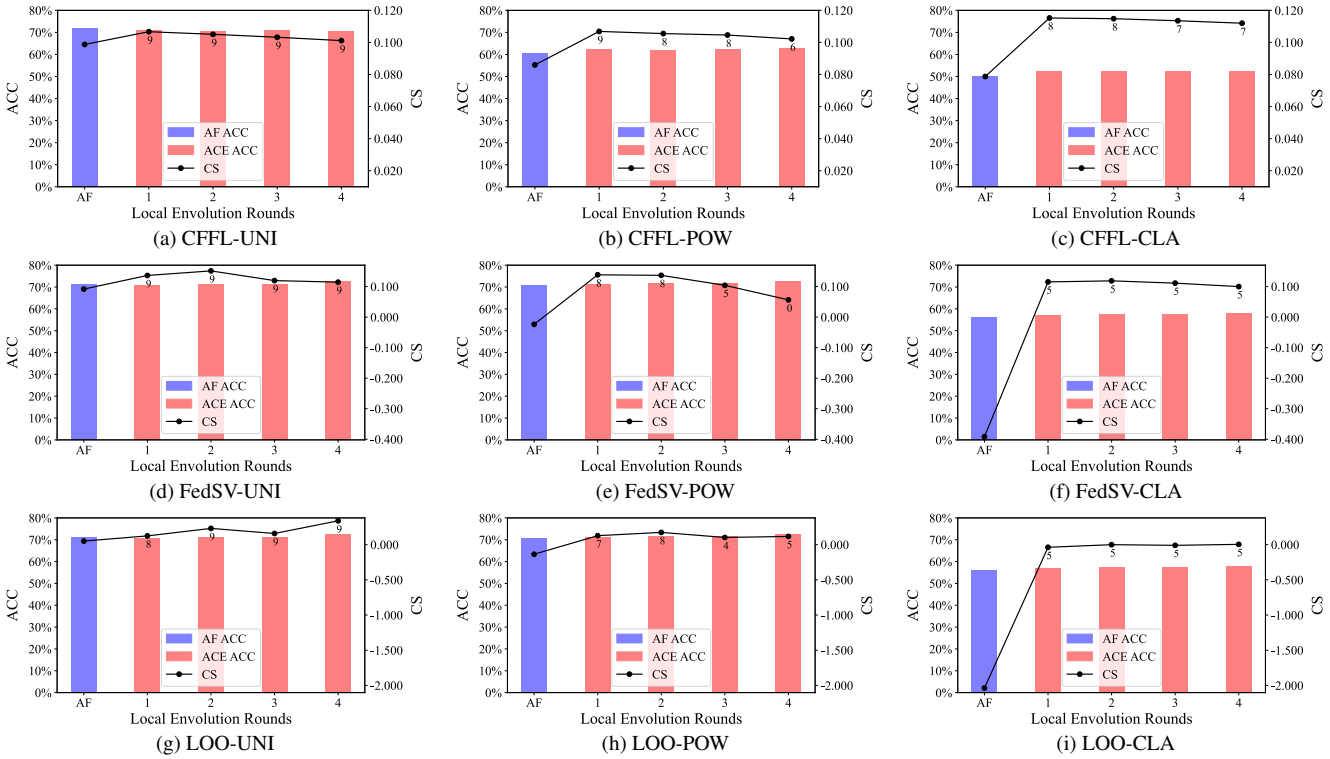


Figure 6: Ablation analysis on the effect of local evolution rounds when CFFL, FedSV, and LOO are used as contribution evaluation methods. The numbers in the figure annotates the rank gain ΔR. AF is the abbreviation for Attack Free.

Table 7: Effect of the strategies that each malicious client take for preliminary iteration and threshold based filtering. In *strategy one* (s_1), the attacker could learn the local model update from the local training dataset of a client. In *strategy two* (s_2), the attacker could use Delta Weight. We have four combinations as the attacker could either use strategy one or two for preliminary iteration and threshold based filtering. We denote these combinations by {strategy for preliminary iteration} × {strategy for threshold based filtering}. This leads to combinations $s_1 \times s_1$, $s_1 \times s_2$, $s_2 \times s_1$, and $s_2 \times s_2$. Overall $s_2 \times s_2$ provides the malicious client the best performance in terms of accuracy, contribution score, and rank gain.

Contribution evaluation	Metric	UNI				POW				CLA			
		$s_1 \times s_1$	$s_1 \times s_2$	$s_2 \times s_1$	$s_2 \times s_2$	$s_1 \times s_1$	$s_1 \times s_2$	$s_2 \times s_1$	$s_2 \times s_2$	$s_1 \times s_1$	$s_1 \times s_2$	$s_2 \times s_1$	$s_2 \times s_2$
CFFL	ACC	71.77%	71.46%	71.50%	70.44%	62.33%	62.35%	62.18%	62.03%	52.12%	52.51%	52.61%	52.45%
	CS	0.102	0.103	0.103	0.105	0.104	0.104	0.104	0.106	0.095	0.115	0.111	0.115
	ΔR	9	9	9	9	7	7	7	8	3	8	7	8
RFFL	ACC	70.81%	70.81%	70.72%	70.72%	70.88%	70.88%	70.90%	70.90%	57.34%	57.34%	57.36%	57.36%
	CS	0.192	0.192	0.192	0.192	0.194	0.194	0.196	0.196	0.188	0.188	0.189	0.189
	ΔR	9	9	9	9	9	9	9	9	9	9	9	9

Table 8: Effect of the fraction of selected clients by the server in each communication round on ACE under contribution evaluation methods SV and LOO. Note that other three contribution evaluation methods require the server to select all clients in each communication round. The results show ACE is consistently effective.

Contribution evaluation	Metrics	Fraction of selected clients	UNI		POW		CLA	
			Attack Free	ACE	Attack Free	ACE	Attack Free	ACE
FedSV	ACC	50%	70.87%	71.73%	69.93%	71.69%	57.30%	59.90%
		70%	70.86%	71.30%	70.50%	71.39%	55.64%	58.17%
		100%	71.16%	71.30%	70.82%	71.45%	56.32%	57.60%
	CS	50%	0.0825	0.1157	-0.0162	0.1142	-0.1825	0.0812
		70%	0.0882	0.1358	-0.0187	0.1316	-0.3108	0.0882
		100%	0.0918	0.1513	-0.0237	0.1367	-0.3916	0.1187
	ΔR	50%	0	8	0	6	0	5
		70%	0	9	0	8	0	5
		100%	0	9	0	8	0	5
LOO	ACC	50%	70.87%	71.73%	69.93%	71.69%	57.30%	59.90%
		70%	70.86%	71.30%	70.50%	71.39%	55.64%	58.17%
		100%	71.16%	71.30%	70.82%	71.45%	56.32%	57.60%
	CS	50%	0.0120	0.2641	-0.1088	0.1606	-2.0835	-0.0693
		70%	0.0346	0.2111	-0.1362	0.1528	-1.9926	-0.0598
		100%	0.0508	0.2311	-0.1361	0.1743	-0.3916	0.1187
	ΔR	50%	0	9	0	7	0	5
		70%	0	9	0	7	0	5
		100%	0	9	0	8	0	5

Appendix B More Experiment Details

B.1 Model Structures

We give the details of the CNN models for experiments on MNIST and CIFAR-10 in Table 10. For Tiny-ImageNet, we use the pretrained *VGG11* implementation from PyTorch[1].

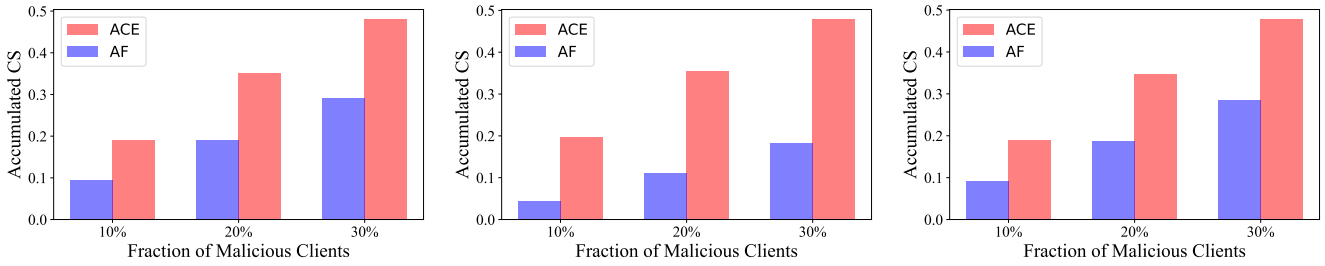
[1] https://pytorch.org/vision/0.16/models/generated/torchvision.models.vgg11.html

B.2 Data Partitions

POW. The data partition method POW distributes the training images of each dataset to the clients using a parameterized power law distribution. The probability density function for the parameterized power law distribution is given as follows

$$f(x;a) = ax^{a-1}, \tag{8}$$

where $0 \leq x \leq 1$ is a random variable, and $a > 1$ is a shape parameter. Let $F(x;a)$ be the corresponding cumulative density function, which is illustrated in Figure 9. Then the data partition method POW requires that the



(a) UNI (b) POW (c) CLA

Figure 7: This figure presents the accumulated CS (summation of the contribution scores of all malicious clients) under UNI, POW, and CLA data partitions when the fraction of malicious clients varies from 10% to 30%. The server employs RFFL as the contribution evaluation method. The results indicate that ACE is effective under different fractions of malicious clients.

Table 9: This table shows the effect of values of c on defense performance under UNI data partition. The contribution evaluation method is RFFL. We observe that the defense methods show marginal performance gain as c increases. This indicates the stealthiness of ACE towards the choice of c. If Precision and Recall are 0, F1-Score is not defined and denoted as N/A.

Detection	Metric	$c=1$	$c=1.5$	$c=2$	$c=2.5$
Multi-Krum	Precision	0.017	0.017	0.017	0.017
	Recall	0.017	0.017	0.017	0.017
	F1-Score	0.017	0.017	0.017	0.017
Trimmed-Mean	Precision	0.017	0.017	0.017	0.017
	Recall	0.017	0.017	0.017	0.017
	F1-Score	0.017	0.017	0.017	0.017
FABA	Precision	0	0.017	0.017	0.017
	Recall	0	0.017	0.017	0.017
	F1-Score	N/A	0.017	0.017	0.017
Sniper	Precision	0	0	0	0
	Recall	0	0	0	0
	F1-Score	N/A	N/A	N/A	N/A
Foolsgold	Precision	0	0	0	0
	Recall	0	0	0	0
	F1-Score	N/A	N/A	N/A	N/A

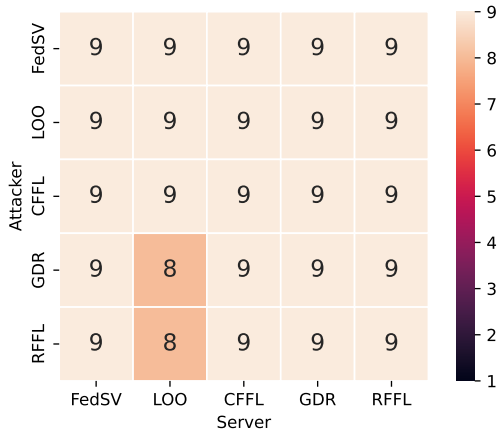


Figure 8: This figure shows the rank gain ΔR when the attacker is unaware of the contribution evaluation method applied by the server. In this figure, the y-axis represents the attacker's guess on the contribution evaluation method used by the server, while the x-axis represents the actual contribution evaluation method employed by the server. We observe that the client launching ACE can still successfully elevate its contribution evaluated by the server even when it is unaware of the contribution evaluation method.

Table 10: CNN model architectures for MNIST and CIFAR-10 datasets.

MNIST	CIFAR-10
Conv3-64 + ReLU	Conv5-64 + ReLU
Max Pool, 2x2	Max Pool, 2x2
Conv7-16 + ReLU	Conv5-128 + ReLU
Max Pool, 2x2	Max Pool, 2x2
FC-64	FC-64
FC-10	FC-10
Softmax	Softmax

number of clients that owns no less than x fraction of the training images follows $F(x;a)$. In our experiments, we set $a = 2$. For the CIFAR-10 and Tiny-ImageNet datasets, the numbers of training samples of the ten clients are $731, 1458, 2184, 2911, 3637, 4364, 5090, 5817, 6543$ and 7265, respectively.

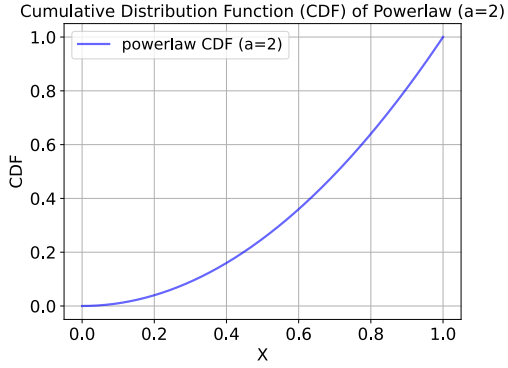


Figure 9: This figure shows the cumulative distribution function of a parameterized power law distribution with parameter $a = 2$.

CLA. The data partition method CLA splits the data samples into the clients' local training datasets as follows. We first categorize the dataset based on labels. Then CLA specifies the numbers of classes that should be covered by the local training dataset of each client. We then sample a subset of data samples these specified classes, and allocate these data samples to the local training dataset of the client. Using CLA, the numbers of classes of 10 clients are $6, 6, 7, 7, 8, 8, 9, 9, 10$, and 10 for the CIFAR-10 dataset, and $100, 111, 122, 133, 144, 155, 166, 177, 188$, and 200 for Tiny-ImageNet dataset, respectively.

B.3 Details on Contribution Evaluation Methods

In our experiments, we consider five state-of-the-art contribution evaluations that can be employed by the server. We detail each of these methods in the following.

- **FedSV** [65]: FedSV is a game theory based contribution evaluation method. The contribution of client i is then calculated as

$$\mathcal{E}(\mathbf{g}_i^t) = \frac{1}{|\Gamma|} \sum_{S \subseteq \Gamma \setminus \{i\}} \frac{U^t(S \cup \{i\}) - U^t(S)}{\binom{|\Gamma|-1}{|S|}},$$

  where the utility function $U^t(S)$ is defined as $U^t(S) = L(\mathcal{D}_s; \mathbf{w}^t) - L(\mathcal{D}_s; \frac{1}{|S|}\sum_{k \in S} \mathbf{w}_k^{t+1})$, $\mathcal{D}_s$ is a validation dataset kept by the server, and $S \subseteq \Gamma \setminus \{i\}$ is a subset of clients not including client i. FedSV uses FedAvg [42] as the aggregation rule.

- **LOO** [65, 81]: This is a game theory based contribution evaluation. It measures the contribution of each client i as $\mathcal{E}(\mathbf{g}_i^t) = L(\mathcal{D}_s; \mathbf{w}_{-i}^t) - L(\mathcal{D}_s; \mathbf{w}^t)$, where $\mathbf{w}_{-i}^t$ denotes the global model without aggregating the local model update from client i at round t, and $\mathcal{D}_s$ is the server's validation dataset. LOO uses FedAvg [42] as the aggregation rule.

- **CFFL** [41]: This is an individual performance based contribution evaluation. The contribution of client i is measured by the accuracy of local model from client i using the validation dataset $\mathcal{D}_s$, i.e., $\mathcal{E}(\mathbf{g}_i^t) = \frac{\text{vacc}_i}{\sum_{j \in \Gamma} \text{vacc}_j}$, where vacc_i is the validation accuracy of client i's local model $\mathbf{w}_i^{t+1}$. The aggregation rule of CFFL is a variant of FedAvg [42], which considers both the data size and the number of classes of a client. Specifically, when the data size is imbalanced, the aggregation rule follows FedAvg. When the class number is imbalanced, the aggregation rule is $\mathcal{A}(\mathbf{g}_1^t, \mathbf{g}_2^t, \cdots, \mathbf{g}_N^t) = \sum_{i \in \Gamma} \frac{|class_i|}{\sum_j |class_j|} \mathbf{g}_i^t$, where $|class|_i$ denotes the number of classes in $\mathcal{D}_i$.

- **GDR** [73]: This is an individual performance based contribution evaluation. It leverages the cosine distance between the aggregated global model updates and local model updates to estimate SV. It then uses the so-called cosine gradient Shapley value as the client contribution, i.e., $\mathcal{E}(\mathbf{g}_i^t) = S_c(\mathbf{u}^t, \mathbf{u}_i^t)$, where $\mathbf{u}_i^t := \varepsilon \mathbf{g}_i^t / \|\mathbf{g}_i^t\|$ represents the local model update of client i normalized using a coefficient ε, and $\mathbf{u}^t$ is the global model update by aggregating the normalized local model updates. The aggregation rule is $\mathcal{A}(\mathbf{g}_1^t, \mathbf{g}_2^t, \cdots, \mathbf{g}_N^t) = \sum_{i \in \Gamma} r_i^t \mathbf{g}_i^t$, where r_i^t is a normalized weight coefficient calculated as the rolling mean of $\mathcal{E}(\mathbf{g}_i^t)$, i.e., $r_i^t = \alpha r_i^{t-1} + (1-\alpha)\mathcal{E}(\mathbf{g}_i^t)$, where the relative weight $\alpha \in (0, 1)$.

- **RFFL** [72]: This is an individual performance based method. The contribution is quantified by the cosine similarity between the local model update $\mathbf{g}_i^t$ and the aggregated global model update $\mathbf{g}^t$, i.e., $\mathcal{E}(\mathbf{g}_i^t) = S_c(\mathbf{g}^t, \mathbf{g}_i^t)$, where $S_c(\mathbf{a}, \mathbf{b})$ denotes the cosine similarity between vectors $\mathbf{a}$ and $\mathbf{b}$. The aggregation rule of RFFL is similar to GDR, which uses r_i^t as the weight of $\mathbf{g}_i^t$.

Appendix C Algorithm Details

This appendix presents the L-BFGS algorithm and the complete algorithm of ACE. We summarize the L-BFGS algorithm in Algorithm 1, which takes the buffers $\Delta\mathbf{W}$ and $\Delta\mathbf{G}$ as well as $\mathbf{v} = \mathbf{w}^t - \mathbf{w}^{t-1}$ as inputs, and outputs the Hessian-vector product $H^t(\mathbf{w}^t - \mathbf{w}^{t-1})$ for global model update prediction. The complete algorithm of ACE is given in Algorithm 2.

Algorithm 1: L-BFGS Algorithm

Input: $\Delta\mathbf{W} = [\Delta\mathbf{w}_0, \Delta\mathbf{w}_1, \cdots, \Delta\mathbf{w}_{m-1}]$, $\Delta\mathbf{G} = [\Delta\mathbf{g}_0, \Delta\mathbf{g}_1, \cdots, \Delta\mathbf{g}_{m-1}]$, and a vector $\mathbf{v}$.

Output: Approximation of Hessian-vector product $\hat{H}\mathbf{v}$.

$\mathbf{A} = \Delta\mathbf{W}^T\Delta\mathbf{G}$

$\mathbf{D} = \text{diag}(\mathbf{A})$ {Diagonal matrix of $\mathbf{A}$}

$\boldsymbol{L} = \text{tril}(\boldsymbol{A})$ {Lower triangular matrix of $\mathbf{A}$}

$\sigma = \left(\Delta\mathbf{g}_{m-1}^T\Delta\mathbf{w}_{m-1}\right) / \left(\Delta\mathbf{w}_{m-1}^T\Delta\mathbf{w}_{m-1}\right)$

$$\mathbf{p} = \begin{bmatrix} -\mathbf{D} & \mathbf{L}^T \\ \mathbf{L} & \sigma\Delta\mathbf{W}^T\Delta\mathbf{W} \end{bmatrix}^{-1} \begin{bmatrix} \Delta\mathbf{G}^T\mathbf{v} \\ \sigma\Delta\mathbf{W}^T\mathbf{v} \end{bmatrix}$$

return $\sigma\mathbf{v} - [\Delta\mathbf{G} \quad \sigma\Delta\mathbf{W}]\mathbf{p}$

Appendix D Proofs of Section 5

Proof of Proposition 1. Since $\cos(\mathbf{g}', c\cdot\hat{\mathbf{g}}_i) = \cos(\mathbf{g}', \hat{\mathbf{g}}_i)$, we have

$$\begin{aligned}
&\cos(\mathbf{g}, \hat{\mathbf{g}}_i) - \cos(\mathbf{g}', c\cdot\hat{\mathbf{g}}_i) = \cos(\mathbf{g}, \hat{\mathbf{g}}_i) - \cos(\mathbf{g}', \hat{\mathbf{g}}_i) \\
&= (1 - S_c(\mathbf{g}, \hat{\mathbf{g}}_i)) - (1 - S_c(\mathbf{g}', \hat{\mathbf{g}}_i)) = S_c(\mathbf{g}', \hat{\mathbf{g}}_i) - S_c(\mathbf{g}, \hat{\mathbf{g}}_i) \\
&= \frac{\mathbf{g}'\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|} - \frac{\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} = \frac{\|\mathbf{g}\|\mathbf{g}'\cdot\hat{\mathbf{g}}_i - \|\mathbf{g}'\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|}.
\end{aligned}$$

We denote by $\mathbf{g}' = \mathbf{g} + (c-1)\alpha_i\hat{\mathbf{g}}_i$. For $\mathbf{g}\cdot\hat{\mathbf{g}}_i \geq 0$, we have:

$$\begin{aligned}
&\cos(\mathbf{g}, \hat{\mathbf{g}}_i) - \cos(\mathbf{g}', c\cdot\hat{\mathbf{g}}_i) \\
&\geq \frac{\|\mathbf{g}\|(\mathbf{g} + (c-1)\alpha_i\hat{\mathbf{g}}_i)\cdot\hat{\mathbf{g}}_i - (\|\mathbf{g}\| + \|(c-1)\alpha_i\hat{\mathbf{g}}_i\|)\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \\
&= \frac{\|\mathbf{g}\|\mathbf{g}\cdot\hat{\mathbf{g}}_i + (c-1)\alpha_i\|\mathbf{g}\|\hat{\mathbf{g}}_i\cdot\hat{\mathbf{g}}_i - \|\mathbf{g}\|\mathbf{g}\cdot\hat{\mathbf{g}}_i - \|(c-1)\alpha_i\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \\
&= \frac{(c-1)\alpha_i\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|^2 - (c-1)\alpha_i\|\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \\
&= \frac{(c-1)\alpha_i\|\hat{\mathbf{g}}_i\|(\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\| - \mathbf{g}\cdot\hat{\mathbf{g}}_i)}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \geq 0.
\end{aligned}$$

The first inequality uses Triangle Inequality, i.e., $\|\mathbf{a}+\mathbf{b}\| \leq \|\mathbf{a}\| + \|\mathbf{b}\|$. The second inequality holds because $c > 1$ and $\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\| \geq \|\mathbf{g}\cdot\hat{\mathbf{g}}_i\| \geq \mathbf{g}\cdot\hat{\mathbf{g}}_i$.

Similarly, for $\mathbf{g}\cdot\hat{\mathbf{g}}_i \leq 0$, we have:

$$\cos(\mathbf{g}, \hat{\mathbf{g}}_i) - \cos(\mathbf{g}', c\cdot\hat{\mathbf{g}}_i) = \frac{\|\mathbf{g}\|\mathbf{g}'\cdot\hat{\mathbf{g}}_i - \|\mathbf{g}'\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|}$$

Algorithm 2: Complete Algorithm of ACE

for each round t **do**
  Receive the current global model $\mathbf{w}^t$ from the server.
  Update $\Delta\mathbf{W}^t$ and $\Delta\mathbf{G}^t$.
  if in *attack rounds* **then**
    if $t \leq m$ **then**
      Perform strategy for preliminary iteration to get $\hat{\mathbf{g}}_i^t$.
    else
      for t' in local evolution rounds **do**
        $\mathbf{v} = \mathbf{w}^{t+t'} - \mathbf{w}^{t+t'-1}$
        $H^{t+t'}\mathbf{v} = \text{L-BFGS}(\Delta\mathbf{W}^{t+t'}, \Delta\mathbf{G}^{t+t'}, \mathbf{v})$
        if $\|H^{t+t'}\mathbf{v}\| \leq l\|\mathbf{v}\|$ **then**
          if $t' = 0$ **then**
            Perform strategy for threshold based filtering to get $\hat{\mathbf{w}}^{t+1}$.
          else if $t' > 0$ **then**
            $t' = t' - 1$
          end if
          Break
        end if
        $\mathbf{g}^{t+t'} = \mathbf{g}^{t+t'-1} + H^{t+t'}\mathbf{v}$
        $\hat{\mathbf{w}}^{t+t'+1} = \hat{\mathbf{w}}^{t+t'} - \mathbf{g}^{t+t'}$
        Update $\Delta\mathbf{W}^{t+t'}$ and $\Delta\mathbf{G}^{t+t'}$.
      end for
      $\hat{\mathbf{g}}_i^t = \mathbf{w}^t - \hat{\mathbf{w}}^{t+t'+1}$
      $\hat{\mathbf{g}}_i^t = c\cdot\hat{\mathbf{g}}_i^t$
    end if
  else
    $\mathbf{g}_i^t \leftarrow \eta_i\nabla L(\mathcal{D}_i, \mathbf{w}^t)$ {Normal training}
  end if
end for

$$\begin{aligned}
&= \frac{\|\mathbf{g}\|(\mathbf{g} + (c-1)\alpha_i\hat{\mathbf{g}}_i)\cdot\hat{\mathbf{g}}_i - \|\mathbf{g}'\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \\
&= \frac{\|\mathbf{g}\|\mathbf{g}\cdot\hat{\mathbf{g}}_i + (c-1)\alpha_i\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|^2 - \|\mathbf{g}'\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \\
&= \frac{\|\mathbf{g}' - (c-1)\alpha_i\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\hat{\mathbf{g}}_i + (c-1)\alpha_i\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|^2 - \|\mathbf{g}'\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \\
&\geq \frac{(\|\mathbf{g}'\| + \|(c-1)\alpha_i\hat{\mathbf{g}}_i\|)\mathbf{g}\cdot\hat{\mathbf{g}}_i + (c-1)\alpha_i\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|^2 - \|\mathbf{g}'\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \\
&= \frac{(c-1)\alpha_i\|\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\hat{\mathbf{g}}_i + (c-1)\alpha_i\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|^2}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \\
&= \frac{(c-1)\alpha_i\|\hat{\mathbf{g}}_i\|(\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\| + \mathbf{g}\cdot\hat{\mathbf{g}}_i)}{\|\mathbf{g}'\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \geq 0.
\end{aligned}$$

The first inequality holds uses Triangle Inequality, i.e., $\|\mathbf{a} - \mathbf{b}\| \leq \|\mathbf{a}\| + \|\mathbf{b}\|$. The second inequality holds because $c > 1$ and $\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\| \geq \|\mathbf{g}\cdot\hat{\mathbf{g}}_i\| \geq -\mathbf{g}\cdot\hat{\mathbf{g}}_i$.

Combine the above two cases, we have:

$$\cos(\mathbf{g}, \hat{\mathbf{g}}_i) - \cos(\mathbf{g}', c\cdot\hat{\mathbf{g}}_i)$$

$$\geq \frac{(c-1)\alpha_i\|\hat{\mathbf{g}}_i\|(\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\| - |\mathbf{g}\cdot\hat{\mathbf{g}}_i|)}{\|\mathbf{g}+(c-1)\alpha_i\hat{\mathbf{g}}_i\|\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} \geq 0.$$

Therefore, $\cos(\mathbf{g}', c\cdot\hat{\mathbf{g}}_i) \leq \cos(\mathbf{g}, \hat{\mathbf{g}}_i)$.

Proof of Corollary 1. Since $\cos(\mathbf{g}, \hat{\mathbf{g}}_i) \leq \cos(\mathbf{g}, \mathbf{g}_j)$, we have:

$$\begin{aligned}
&\cos(\mathbf{g}, \mathbf{g}_j) - \cos(\mathbf{g}, \hat{\mathbf{g}}_i) = S_c(\mathbf{g}, \hat{\mathbf{g}}_i) - S_c(\mathbf{g}, \mathbf{g}_j) \\
&= \frac{\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|} - \frac{\mathbf{g}\cdot\mathbf{g}_j}{\|\mathbf{g}\|\|\mathbf{g}_j\|} \\
&= \frac{\|\mathbf{g}_j\|\mathbf{g}\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\mathbf{g}_j}{\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\|} \geq 0.
\end{aligned}$$

Since the denumerator $\|\mathbf{g}\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\| \geq 0$, the numerator $\|\mathbf{g}_j\|\mathbf{g}\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\mathbf{g}_j \geq 0$. Denote $\mathbf{g}' = \mathbf{g} + (c-1)\alpha_i\hat{\mathbf{g}}_i$, we have:

$$\begin{aligned}
&\cos(\mathbf{g}', \mathbf{g}_j) - \cos(\mathbf{g}', c\hat{\mathbf{g}}_i) = \cos(\mathbf{g}', \mathbf{g}_j) - \cos(\mathbf{g}', \hat{\mathbf{g}}_i) \\
&= S_c(\mathbf{g}', \hat{\mathbf{g}}_i) - S_c(\mathbf{g}', \mathbf{g}_j) \\
&= \frac{\mathbf{g}'\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|} - \frac{\mathbf{g}'\cdot\mathbf{g}_j}{\|\mathbf{g}'\|\|\mathbf{g}_j\|} = \frac{\|\mathbf{g}_j\|\mathbf{g}'\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\mathbf{g}'\cdot\mathbf{g}_j}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\|} \\
&= \frac{\|\mathbf{g}_j\|(\mathbf{g}+(c-1)\alpha_i\hat{\mathbf{g}}_i)\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|(\mathbf{g}+(c-1)\alpha_i\hat{\mathbf{g}}_i)\cdot\mathbf{g}_j}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\|} \\
&= \frac{\|\mathbf{g}_j\|\mathbf{g}\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\mathbf{g}_j + (c-1)\alpha_i(\|\mathbf{g}_j\|\hat{\mathbf{g}}_i\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\hat{\mathbf{g}}_i\cdot\mathbf{g}_j)}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\|} \\
&\geq \frac{(c-1)\alpha_i(\|\mathbf{g}_j\|\hat{\mathbf{g}}_i\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\hat{\mathbf{g}}_i\cdot\mathbf{g}_j)}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\|} \\
&= \frac{(c-1)\alpha_i(\|\mathbf{g}_j\|\|\hat{\mathbf{g}}_i\| - \hat{\mathbf{g}}_i\cdot\mathbf{g}_j)}{\|\mathbf{g}'\|\|\mathbf{g}_j\|} \geq 0.
\end{aligned}$$

The inequality holds because $\|\mathbf{g}_j\|\mathbf{g}\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\mathbf{g}_j \geq 0$. Therefore, we have $\cos(\mathbf{g}', \mathbf{g}_j) \geq \cos(\mathbf{g}', c\hat{\mathbf{g}}_i)$.

Proof of Proposition 2. By the definition of cosine distance, we have

$$\begin{aligned}
&\cos(\mathbf{g}', \mathbf{g}_j) - \cos(\mathbf{g}', c\hat{\mathbf{g}}_i) = \cos(\mathbf{g}', \mathbf{g}_j) - \cos(\mathbf{g}', \hat{\mathbf{g}}_i) \\
&= \frac{\mathbf{g}'\cdot\hat{\mathbf{g}}_i}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|} - \frac{\mathbf{g}'\cdot\mathbf{g}_j}{\|\mathbf{g}'\|\|\mathbf{g}_j\|} = \frac{\|\mathbf{g}_j\|\mathbf{g}'\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\mathbf{g}'\cdot\mathbf{g}_j}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\|} \\
&= \frac{\|\mathbf{g}_j\|(\mathbf{g}+(c-1)\alpha_i\hat{\mathbf{g}}_i)\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|(\mathbf{g}+(c-1)\alpha_i\hat{\mathbf{g}}_i)\cdot\mathbf{g}_j}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\|} \\
&= \frac{\|\mathbf{g}_j\|\mathbf{g}\cdot\hat{\mathbf{g}}_i - \|\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\mathbf{g}_j + (c-1)\alpha_i\|\hat{\mathbf{g}}_i\|(\|\mathbf{g}_j\|\|\hat{\mathbf{g}}_i\| - \hat{\mathbf{g}}_i\cdot\mathbf{g}_j)}{\|\mathbf{g}'\|\|\hat{\mathbf{g}}_i\|\|\mathbf{g}_j\|} \\
&\geq 0.
\end{aligned}$$

Note that since $\cos(\mathbf{g}, \hat{\mathbf{g}}_i) > \cos(\mathbf{g}, \mathbf{g}_j)$, we have $\|\mathbf{g}_j\|\|\hat{\mathbf{g}}_i\| - \hat{\mathbf{g}}_i\cdot\mathbf{g}_j > 0$. Thus we have:

$$c \geq \frac{\|\hat{\mathbf{g}}_i\|\mathbf{g}\cdot\mathbf{g}_j - \|\mathbf{g}_j\|\mathbf{g}\cdot\hat{\mathbf{g}}_i}{\alpha_i\|\hat{\mathbf{g}}_i\|(\|\mathbf{g}_j\|\|\hat{\mathbf{g}}_i\| - \hat{\mathbf{g}}_i\cdot\mathbf{g}_j)} + 1.$$