Confined Gradient Descent: Privacy-preserving Optimization for Federated Learning

Yanjun Zhang
The University of Queensland
St Lucia, Queensland, Australia
yanjun.zhang@uq.edu.au

Guangdong Bai
The University of Queensland
St Lucia, Queensland, Australia
g.bai@uq.edu.au

Xue Li
The University of Queensland
St Lucia, Queensland, Australia
xueli@itee.uq.edu.au

Surya Nepal
Data61 CSIRO Australia
MARSFIELD, New South Wales
Australia
Surya.Nepal@data61.csiro.au

Ryan K L Ko
The University of Queensland
St Lucia, Queensland, Australia
ryan.ko@uq.edu.au

ABSTRACT
Federated learning enables multiple participants to collaboratively train a model without aggregating the training data. Although the training data are kept within each participant and the local gradients can be securely synthesized, recent studies have shown that such privacy protection is insufficient. The global model parameters that have to be shared for optimization are susceptible to leak information about training data. In this work, we propose Confined Gradient Descent (CGD) that enhances privacy of federated learning by eliminating the sharing of global model parameters. CGD exploits the fact that a gradient descent optimization can start with a set of discrete points and converges to another set at the neighborhood of the global minimum of the objective function. It lets the participants independently train on their local data, and securely share the sum of local gradients to benefit each other. We formally demonstrate CGD’s privacy enhancement over traditional FL. We prove that less information is exposed in CGD compared to traditional FL. CGD also guarantees desired model accuracy. We theoretically establish a convergence rate for CGD. We prove that the loss of the proprietary models learned for each participant against a model learned by aggregated training data is bounded. Extensive experimental results on two real-world datasets demonstrate the performance of CGD is comparable with the centralized learning, with marginal differences on validation loss (mostly within 0.05) and accuracy (mostly within 1%).

1 INTRODUCTION
The performance of machine learning largely relies on the availability of large representative datasets. To take advantage of massive data owned by multiple entities, federated learning (FL) is proposed [24, 25, 46]. It enables participants to jointly train a global model without the necessity of sharing their datasets, demonstrating the potential to address the issues of data privacy and data ownership. It has been incorporated by popular machine learning tools such as TensorFlow [1] and PyTorch [37], and increasingly spread over various industries.

The privacy preservation of FL stems from its parallelization of the gradient descent optimization, which in essence is an application of stochastic gradient descent (SGD) (or the mini-batch mode) [25]. During the training process, the participants work on the same intermediate global model via a coordinating server (in the centralized FL) [2, 4, 34] or a peer-to-peer communication scheme (in the decentralized FL) [22, 39]. Each of them obtains the current model parameters, works out a local gradient based on the local data, and disseminates it to update the global model synchronously [2, 4] or asynchronously [21]. This paradigm guarantees data locality, but has been found insufficient for data privacy: although the local gradients can be securely synthesized via a variety of techniques such as differential privacy (DP) [2, 21, 42, 48], secure multi-party communication (MPC) [4, 11, 34], and homomorphic encryption (HE) [31, 34, 40], the global model parameters that have to be shared are still susceptible to information leakage (cf. Section 5 and [35, 36]).

This work further decreases the dependency among participants by eliminating the explicit sharing of the central global model which is the root cause of the information leakage [9, 35]. We propose a new optimization algorithm named Confined Gradient Descent (CGD) that enables each participant to learn a proprietary global model. The CGD participants maintain their global models locally, which are strictly confined within themselves from the beginning of and throughout the whole training process. We refer to these localized global models as confined models, to distinguish them from the global model in traditional FL.

CGD is inspired by an observation on the surface of the typical cost function. The steepness of the first derivative decreases slower when approaching the minimum of the function, due to the small values in the Hessian (i.e., the second derivative) near the optimum [5]. This gives the function, when plotted, a flat valley bottom. As such, a gradient descent algorithm \( A \), when applied on an objective function \( F \), could start with a set of discrete points (referred to as a colony and their distance is discussed later). Iteratively descending the colony using the joint gradient of the colony would lead \( A \) to the neighborhood of \( F \)'s minimum in the “flat valley bottom”. The points in the colony would also end up with similar losses that are close to the loss of the minimum.

In Figure 1, we illustrate a holistic comparison between the workflow of CGD and that of a gradient decent in traditional FL. In traditional FL, every participant updates the same global model \( w \) using their local gradients \( g^a, g^b, g^c \). In CGD, each participant \( l \) first independently initializes the starting point of its confined model...
weights among participants differ by two orders of magnitude. For the latter, CGD is an optimization method based on the gradient decent. It is extensively used for optimizing the objective function in machine learning and deep learning. Given a cost function $F$ with the parameter $w$, SGD is defined by

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_k|} \nabla F(w_k, \xi_k),$$  

(1)

where $w_k$ are the parameter at the $k^{th}$ iteration, $\xi_k \in \xi$ is a randomly selected subset of the training samples at the $k^{th}$ iteration, and $\alpha_k$ is the learning rate. Equation 1 can generalize to the mini-batch update when $1 < |\xi_k| < |\xi|$, and to the batch update when $\xi_k = \xi$.

In FL, each local participant $l \in \mathcal{L}$ holds a subset of the training samples, denoted by $\xi_l$. To run SGD (or the mini-batch update), for each iteration, a random subset $\xi_{l,k} \subseteq \xi_l$ from a random participant $l$ is selected. The participant $l$ then computes the gradient with respect to $\xi_{l,k}$, which can be written as $\nabla F(w_k, \xi_{l,k})$, and shares the gradient with the other participants (or a parameter server). All the participants (or the server) can thus take a gradient descent step by

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_{l,k}|} \nabla F(w_k, \xi_{l,k}).$$  

(2)

2 BACKGROUND AND RELATED WORKS

CGD is an optimization method based on the gradient descent. Therefore, in this section, we review the existing techniques for gradient updates in the traditional FL.

2.1 Stochastic gradient descent

Stochastic gradient descent (SGD) [6, 38] is an efficient variant of the gradient descent algorithm. It is extensively used for optimizing the objective function in machine learning and deep learning. Given a cost function $F$ with the parameter $w$, SGD is defined by

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_k|} \nabla F(w_k, \xi_k),$$  

(1)

where $w_k$ are the parameter at the $k^{th}$ iteration, $\xi_k \in \xi$ is a randomly selected subset of the training samples at the $k^{th}$ iteration, and $\alpha_k$ is the learning rate. Equation 1 can generalize to the mini-batch update when $1 < |\xi_k| < |\xi|$, and to the batch update when $\xi_k = \xi$.

In FL, each local participant $l \in \mathcal{L}$ holds a subset of the training samples, denoted by $\xi_l$. To run SGD (or the mini-batch update), for each iteration, a random subset $\xi_{l,k} \subseteq \xi_l$ from a random participant $l$ is selected. The participant $l$ then computes the gradient with respect to $\xi_{l,k}$, which can be written as $\nabla F(w_k, \xi_{l,k})$, and shares the gradient with the other participants (or a parameter server). All the participants (or the server) can thus take a gradient descent step by

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_{l,k}|} \nabla F(w_k, \xi_{l,k}).$$  

(2)

2 BACKGROUND AND RELATED WORKS

CGD is an optimization method based on the gradient descent. Therefore, in this section, we review the existing techniques for gradient updates in the traditional FL.

2.1 Stochastic gradient descent

Stochastic gradient descent (SGD) [6, 38] is an efficient variant of the gradient descent algorithm. It is extensively used for optimizing the objective function in machine learning and deep learning. Given a cost function $F$ with the parameter $w$, SGD is defined by

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_k|} \nabla F(w_k, \xi_k),$$  

(1)

where $w_k$ are the parameter at the $k^{th}$ iteration, $\xi_k \in \xi$ is a randomly selected subset of the training samples at the $k^{th}$ iteration, and $\alpha_k$ is the learning rate. Equation 1 can generalize to the mini-batch update when $1 < |\xi_k| < |\xi|$, and to the batch update when $\xi_k = \xi$.

In FL, each local participant $l \in \mathcal{L}$ holds a subset of the training samples, denoted by $\xi_l$. To run SGD (or the mini-batch update), for each iteration, a random subset $\xi_{l,k} \subseteq \xi_l$ from a random participant $l$ is selected. The participant $l$ then computes the gradient with respect to $\xi_{l,k}$, which can be written as $\nabla F(w_k, \xi_{l,k})$, and shares the gradient with the other participants (or a parameter server). All the participants (or the server) can thus take a gradient descent step by

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_{l,k}|} \nabla F(w_k, \xi_{l,k}).$$  

(2)

2 BACKGROUND AND RELATED WORKS

CGD is an optimization method based on the gradient descent. Therefore, in this section, we review the existing techniques for gradient updates in the traditional FL.

2.1 Stochastic gradient descent

Stochastic gradient descent (SGD) [6, 38] is an efficient variant of the gradient descent algorithm. It is extensively used for optimizing the objective function in machine learning and deep learning. Given a cost function $F$ with the parameter $w$, SGD is defined by

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_k|} \nabla F(w_k, \xi_k),$$  

(1)

where $w_k$ are the parameter at the $k^{th}$ iteration, $\xi_k \in \xi$ is a randomly selected subset of the training samples at the $k^{th}$ iteration, and $\alpha_k$ is the learning rate. Equation 1 can generalize to the mini-batch update when $1 < |\xi_k| < |\xi|$, and to the batch update when $\xi_k = \xi$.

In FL, each local participant $l \in \mathcal{L}$ holds a subset of the training samples, denoted by $\xi_l$. To run SGD (or the mini-batch update), for each iteration, a random subset $\xi_{l,k} \subseteq \xi_l$ from a random participant $l$ is selected. The participant $l$ then computes the gradient with respect to $\xi_{l,k}$, which can be written as $\nabla F(w_k, \xi_{l,k})$, and shares the gradient with the other participants (or a parameter server). All the participants (or the server) can thus take a gradient descent step by

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_{l,k}|} \nabla F(w_k, \xi_{l,k}).$$  

(2)
The gradients, if shared in plain text, are subject to information leakage of the local training data. For example, model-inversion attacks [9, 32, 43] are able to restore training data from the gradients. In the immediately following sections, we summarize the existing privacy-preserving methods for synthesizing the local gradients, which fall into two broad categories, i.e., secure aggregation and learning with differential privacy.

2.2 Secure aggregation

Secure aggregation typically employs cryptographic mechanisms such as homomorphic encryption (HE) [7, 31, 40] and/or secure multiparty computation (MPC) [4, 10, 11, 30, 34, 47] to securely synthesize, by knowing $w$ and $f$, the adversary is able to derive information about the local raw data $\xi_a, \xi_b, \xi_c$.

\[
\begin{align*}
\omega^* = & w^* - \text{iter}(g(w^*, \xi^b), g(w^*, \xi^b), g(w^*, \xi^b)) \\
\omega^b = & g(w^b, \xi^b) \\
\omega^c = & g(w^c, \xi^c)
\end{align*}
\]

(a) Gradient descent in traditional federated learning. Participants jointly work on the same global model $w$ using the descent computed by $f$. Although the local gradients $g^a, g^b, g^c$ can be securely synthesized, by knowing $w$ and $f$, the adversary is able to derive information about the local raw data $\xi_a, \xi_b, \xi_c$.

\[
\begin{align*}
\omega_k = & w_k - \text{iter}(g(w_k, \xi_{l,k}), g(w_k, \xi_{l,k}), g(w_k, \xi_{l,k})) \\
\omega^a = & g(w^a, \xi^a) \\
\omega^b = & g(w^b, \xi^b) \\
\omega^c = & g(w^c, \xi^c)
\end{align*}
\]

(b) Confined Gradient Descent. Each participant strictly confines their own global models from their initialization ($w^a, w^b, w^c$) to optimal values ($w^a, w^b, w^c$). The confined models descend in the same pace, and when CGD converges, reach the bottom of the valley where the centralized model is located. Any two confined models keep the same distance throughout the training process. In other words, $w^a, w^b, w^c$ would not become closer to each other during descending, preventing any participant from predicting models of others.

Figure 1: Comparison of CGD and the gradient descent in traditional FL. This figure does not differentiate each iteration: the occurrences of traditional FL’s global model $w_k$ and confined models $w^a_k, w^b_k, w^c_k$ in all iterations are represented by $w, w^a, w^b, w^c$, and iter represents the sum-up of all iterations.

In summary, in all of the above approaches, the global model evolves from the central model to the average of the participant’s own models. The privacy loss is accumulated with repeated access to the central model.

2.3 Learning with differential privacy

Another line of studies that approaches to privacy-preserving FL is through differential privacy (DP) mechanism [2, 8, 12, 16, 21, 42, 44, 48, 49]. The common practice of achieving differential privacy is based on additive noise calibrated to $VF$’s sensitivity $S_{VF}^2$. As such, a differentially private learning framework can be achieved by updating parameters with perturbed gradients at each iteration, for example, to update parameters as

\[
w_{k+1} \leftarrow w_k - \alpha_k \frac{1}{|\xi_{l,k}|} (\nabla F(w_k, \xi_{l,k}) + N(0, S_{VF}^2 \cdot \sigma^2)),
\]

where $N(0, S_{VF}^2 \cdot \sigma^2)$ is the Gaussian distribution (a commonly used noise distribution in differentially private learning frameworks [8]) with mean 0 and standard deviation $S_{VF} \cdot \sigma$.

The privacy loss is accumulated with repeated access to the data during training epochs [2]. There is also an inherent tradeoff between privacy and utility of the trained model.

In summary, in all of the above approaches, the global model has to be shared with each participant, leading to the leakage of information. This motivates CGD’s design to eliminate the explicit sharing of the central global model.
3 CONFINED GRADIENT DESCENT

CGD optimizes an objective function in FL with multiple local datasets. It starts with a colony of discrete points, and then uses the combination of their gradients to lead the optimization to another colony of points at the neighborhood of the global optimum. In this section, we formalize this problem and present the workflow of CGD optimization.

3.1 Problem formulation

3.1.1 Optimization objective. Consider a centralized dataset $\xi = \{(x_i, y_i)\}_{i=1}^{n}$ consisting of $n$ training samples. The goal of machine learning is to find a model parameter $w$ such that the overall loss, which is measured by the distance between the model prediction $h(w, x_i)$ and the label $y_i$ for each $(x_i, y_i) \in \xi$, is minimized. This is reduced to solving the following problem

$$\arg\min_w \frac{1}{n} \sum_{i=1}^{n} F(w; \xi) + \lambda z(w), \quad (4)$$

where $F(w; \xi)$ is the loss function, and $z(w)$ is the regularizer for $w$. We use $w_*$ to denote the optimal solution of centralized training (i.e., the centralized model).

In the context of FL, we have a system of $m$ local participants, each of which holds a private dataset $\xi_l \subseteq \xi$ ($l \in [1, m]$) consisting of a part of the training dataset. The part could be a part of training samples, a part of features that have common entries, or both. Assume the training takes $T$ iterations, and let $w^l_k$ denote the confined model of participant $l$ at the $k^{th}$ iteration, where $k \in [1, T]$. Let

$$g^l_k(w^l_k, \xi_l) = \frac{1}{|\xi_l|} \nabla F(w^l_k, \xi_l) \quad (5)$$

represent the local gradient with respect to $\xi_l$. We use $w^l_0$ to denote the final confined model of participant $l$ when CGD converges. The objective of CGD is to make $w^l_k$ located at a neighborhood of the centralized model $w_*$ within a bounded gap.

3.1.2 Attacker setting. We assume an honest-but-curious white-box adversary\(^1\) who may control $t$ out of $m$ participants (where $t \leq m - 2$), including the aggregator for secure addition operation (if any).

3.2 CGD optimization

Figure 2 illustrates the architecture of CGD (Figure 2c), with a comparison to the centralized training (Figure 2a) and traditional FL (Figure 2b). In the centralized training, the datasets of all participants are gathered for training a single model. In the traditional FL, every participant owns its local training dataset, and updates the same global model $w_k$ via a parameter server using its local model/gradients. The local gradients $\nabla F(w_k, \xi_{l,k})$ can be protected via either secure aggregation [4, 30, 31, 34, 40, 47] or differential privacy mechanisms [2, 8, 21, 42, 44, 48]. This process can be decentralized by replacing the parameter server with a peer-to-peer communication mechanism [22, 39]. In CGD, each participant learns its own confined model (represented by different colors), i.e., each $w^l_k$ is different and private. The model updating in CGD synthesizes the information from all training samples by summing up the local gradients, while in federated SGD (Figure 2b), each iteration takes into account only a subset of training samples.

![Figure 2: Architectural comparison of the centralized training, traditional federated learning, and CGD.](image)

To better position CGD, we summarize the related studies in the literature in Table 1. We use federated SGD to represent the SGD or the mini-batch update in the FL, including both the plain SGD in which the local gradient/model is shared in plaintext, and privacy-preserving SGD via secure aggregation or differential privacy mechanisms. CGD guarantees that each confined model, when CGD converges, is at the neighborhood of the centralized model, retaining the model accuracy (column 5 in Table 1 and proved in Section 4). It also achieves desired privacy preservation compared to the traditional FL (column 6 in Table 1 and detailed in Section 5).

Algorithm 1 outlines CGD optimization for training with the confined model $w^l_k$. In general, the optimization process consists of the following steps.

---

\(^1\)A white-box adversary knows the internals of the training algorithms such as the neural network architecture, and can observe the intermediate computations during the training iterations.
Methodology
Desired properties

Plain SGD [6, 38]
All the participants jointly learn one and the same global model.
Theoretically guaranteed convergence [6, 38].

Privacy-preserving SGD via secure aggregation [4, 30, 31, 34, 40, 47]
The shared variance of local parameters/gradients is subject to model inversion attack [9, 32, 43].

Privacy-preserving SGD via differential privacy [2, 8, 21, 42, 44, 48]
The sharing of one and the same global model (even though the local gradients are protected) is still subject to information leakage (refer to [35] and Section 5).

Confined Gradient Descent
Each participant learns and confines a different global model.
Theoretically guaranteed convergence with bounded gap to the centralized model.

Table 1: A summary of differences between the federated SGD and CGD.

Algorithm 1 Confined Gradient Descent Optimization

1. Input: Local training data \(\xi_l \ (l \in [1, m])\), number of training iterations \(T\).
2. Output: Confined global model parameters \(w_k^l \ (l \in [1, m])\).
3. Initialize: \(k \leftarrow 1\), each participant \(l\) randomizes its own \(w_k^l\) while \(k \leq T\) do
4. for all participants \(l \in [1, m]\) do in parallel
5. Compute the local gradient: \(g_l^l (w_k^l, \xi_l)\)
6. Securely evaluate the sum: \(\sum_{l=1}^{m} g_l^l (w_k^l, \xi_l)\)
7. Choose a stepsize: \(\alpha_k\)
8. Set the new iterate as: \(w_{k+1}^l \leftarrow w_k^l - \alpha_k \sum_{l=1}^{m} g_l^l (w_k^l, \xi_l)\)
9. end for
10. end while

4 CONVERGENCE ANALYSIS

In this section, we conduct a formal convergence analysis on CGD. Convergence analysis has been extensively used in the literature [19–21] to prove the correctness of optimization algorithms. Through the analysis, we demonstrate the bound of the distance between an arbitrary \(w_k^l\) learned by CGD and the centralized model \(w^*\). The analysis is centered around a regret function \(R\), which is the difference between the CGD’s training loss and the loss of the centralized model, defined as

\[
R = \frac{1}{T} \sum_{k=1}^{T} (F(w_k^l) - F(w^*))
\] (6)

4.1 Assumptions

We make the following assumptions on the loss function \(F\). They are all common assumptions in convergence analyses of most gradient-based methods, and satisfied in a variety of widely used cost functions [6], such as mean squared error (MSE) and cross entropy.

**Assumption 1.** (Lipschitz continuity). The loss function \(F: \mathbb{R}^d \rightarrow \mathbb{R}\) is continuously differentiable and the gradient function of \(F\), namely, \(\nabla F: \mathbb{R}^d \rightarrow \mathbb{R}^d\), is Lipschitz continuous with Lipschitz constant \(L > 0\),

\[
\|\nabla F(w) - \nabla F(\hat{w})\|_2 \leq L\|w - \hat{w}\|_2 \quad \text{for all } (w, \hat{w}) \subseteq \mathbb{R}^d. 
\] (7)

Intuitively, this assumption ensures that the gradient of \(F\) does not change arbitrarily in the course of descending, such that the gradient can be a proper indicator towards the optimum [6].
We first present our main theorem below, and leave its proof to Section 4.3. It demonstrates our main result on the convergence rate of CGD.

### 4.2 Main Theorem

We first present our main theorem below, and leave its proof to Section 4.3. It demonstrates our main result on the convergence rate of CGD.

**Theorem 1.** Given a cost function satisfying Assumptions 1-3, and a learning rate of \( \alpha_k = \frac{\mu}{(k+1)^2} \) (0 < \( \mu < 1 \)), the CGD optimization gives the regret

\[
\mathcal{R} = O(\epsilon + \frac{1}{\mu T + 1} \ln |\mu T + 1|) \tag{10}
\]

where \( \epsilon = \frac{m}{T} \left| \mathbb{E} (w_t^1 - w_t^i) \right| \).

The theorem implies the following two remarks.

- **Convergence rate.** Both \( \frac{1}{T^2} \) and \( \frac{\ln |\mu T + 1|}{T} \) approach 0 as \( T \) increases, implying that CGD will converge toward the optimum. The convergence rate can be adjusted by the parameter \( \mu \) (the effect of \( \mu \) is investigated in Section 7.2.2).

- **Bounded optimality gap.** When CGD converges, the gap between the confined models and the centralized model is bounded by \( \epsilon \). CGD uses the initialization parameter \( \delta \) to determine the range of \( w_t^n \) in the way of

\[
w_t^1 = \delta \cdot \text{Rand}^d \quad \text{for all } t \in (1, m), \tag{11}
\]

where \( \text{Rand} \) is the initialization scheme. A standard \( \text{Rand} \) used in machine learning is to apply random sampling from the Gaussian distribution of mean 0 and the variance 1, and a standard \( \delta = \frac{1}{\sqrt{n}} \) (where \( n \) is the sample size) [13].

In CGD, each participant determines its own \( \text{Rand}^d \) and \( \delta \) independently to avoid leaking the average distance among the confined models. Our experiment finds that CGD keeps robust (in terms of validation accuracy) even when the participants select their \( \delta \)'s uniformly at random in a range of \( (\frac{10}{\sqrt{n}}, \frac{0.1}{\sqrt{n}}) \) (cf. Section 7.2.1).

### 4.3 Proof of Theorem 1

Our proof aims to identify an upper bound of \( \mathcal{R} \). To this end, we consider the trend of the distance from \( w_t^i \) to \( w_* \), which would shrink as \( k \) increases, if there is a bound existing. Since \( w_t^i \) is updated using \( \sum_{j=1}^{m} g^j(w_t^j, \xi_j) \) (i.e., the descent in CGD), and \( w_* \) is obtained by \( \nabla F \) (i.e., the descent in the centralized training), the trend of the distance therefore should be related to the deviation between these two. Exploring this leads to the following lemma which describes this relationship.

**Lemma 1.** Let \( S_{k+1} = \frac{1}{2} \| w_{k+1}^i - w_* \|^2_2 \) and \( S_k = \frac{1}{2} \| w_k^i - w_* \|^2_2 \).

Let \( \nabla F(\cdot) = \frac{1}{n} \sum_{t=1}^{n} \nabla F(\cdot, \xi_t) \). We have

\[
\langle w_t^i - w_* , \nabla F(w_t^i) \rangle = \frac{1}{2} \alpha_k \sum_{t=1}^{m} g^j(w_t^j, \xi_j) \| w_t^j - w_* \|^2_2 - \frac{1}{\alpha_k} (S_{k+1} - S_k)
\]

\[
- \langle w_t^i - w_* , \sum_{t=1}^{m} g^j(w_t^j, \xi_j) - \nabla F(w_t^i) \rangle.
\]

**Proof.**

\[
S_{k+1} - S_k = \frac{1}{2} \| w_{k+1}^i - w_* \|^2_2 - \| w_k^i - w_* \|^2_2
\]

\[
= \frac{1}{2} \| w_t^i - w_* \| - \alpha_k \sum_{t=1}^{m} g^j(w_t^j, \xi_j) - w_* \| - \| w_k^i - w_* \|_2^2
\]

\[
= \frac{1}{2} \| w_t^i - w_* \| - \alpha_k \sum_{t=1}^{m} g^j(w_t^j, \xi_j) - w_* \| - \| w_k^i - w_* \|_2^2
\]

\[
= \frac{1}{2} \| w_t^i - w_* \| - \alpha_k \sum_{t=1}^{m} g^j(w_t^j, \xi_j) - \nabla F(w_t^i) \| - \| w_k^i - w_* \|_2^2
\]

Dividing the above equation by \( \alpha_k \), we can prove the lemma. \( \square \)

In the following, we give the proof of Theorem 1. It calculates a function that is greater than \( \mathcal{R} \) based on the convexity of the objective function (Inequation 13). The function can be decomposed into three terms based on Lemma 1 (Equation 14). We then explore the boundedness of each term, and taking these bounds together concludes the proof.

**Proof.** By the definition of the regret function (Equation 6) and Equation 9 in Assumption 2, we have

\[
\mathcal{R} = \frac{1}{T} \sum_{k=1}^{T} (F(w_t^j) - F(w_*)) \leq \frac{1}{T} \sum_{k=1}^{T} (w_t^j - w_* , \nabla F(w_t^j)) \tag{13}
\]
Applying Lemma 1 to Inequation 13 and multiplying it by $T$, we have

$$T \cdot R \leq \sum_{k=1}^{T} \frac{1}{2} \alpha_k \| \sum_{j=1}^{m} g^j (w_k^j, \xi_j) \|^2_2$$

$$= \frac{1}{\alpha_k} (S_{k+1} - S_k) - (w_k^* - w_0, \sum_{j=1}^{m} g^j (w_k^j, \xi_j) - \nabla F(w_k^j))$$

$$\leq \sum_{k=1}^{T} \frac{1}{2} \alpha_k \| \sum_{j=1}^{m} g^j (w_k^j, \xi_j) \|^2_2 - \sum_{k=1}^{T} \frac{1}{\alpha_k} (S_{k+1} - S_k)$$

$$- \sum_{k=1}^{T} (w_k^* - w_0, \sum_{j=1}^{m} g^j (w_k^j, \xi_j) - \nabla F(w_k^j))$$

(14)

Inequation 14 can be decomposed into three terms. The first two terms $\sum_{k=1}^{T} \frac{1}{2} \alpha_k \| \sum_{j=1}^{m} g^j (w_k^j, \xi_j) \|^2_2$ and $- \sum_{k=1}^{T} \frac{1}{\alpha_k} (S_{k+1} - S_k)$ sums up the model updates throughout the training iterations. The third term $- \sum_{k=1}^{T} (w_k^* - w_0, \sum_{j=1}^{m} g^j (w_k^j, \xi_j) - \nabla F(w_k^j))$ measures the gap of the gradients between CGD and the centralized training.

Next, we explore the boundedness of each term. For the first term, we have

$$\sum_{k=1}^{T} \frac{1}{2} \alpha_k \| \sum_{j=1}^{m} g^j (w_k^j, \xi_j) \|^2_2 \leq \frac{T}{2 \alpha_k} \sum_{j=1}^{m} g^j (w_k^j, \xi_j)$$

$$= \frac{am^2G^2}{2 \alpha_k} \sum_{k=1}^{T} \frac{1}{(k + \mu T)^2} < am^2G^2.$$  

(15)

Inequation 15 is based on Assumption 3(b), and Inequation 16 is based on the solution to the Basel problem that $\sum_{k=1}^{\infty} \frac{1}{k^2} < 2$.

For the second term, we have

$$- \sum_{k=1}^{T} \frac{1}{\alpha_k} (S_{k+1} - S_k) = \sum_{k=1}^{T} \frac{1}{\alpha_k} (S_k - S_{k+1})$$

$$= \sum_{k=1}^{T} \frac{1}{\alpha_k} \left( \frac{1}{2} ||w_k^* - w_0||^2_2 - \frac{1}{2} ||w_{k+1}^* - w_0||^2_2 \right)$$

$$\leq \sum_{k=1}^{T} \frac{1}{2\alpha_k} ||w_k^* - w_0|| - (w_{k+1}^* - w_0) ||^2_2$$

(16)

(17)

(18)

(19)

(20)

Inequation 19 follows reverse triangle inequality, and Inequation 21 reuses the result of the first term (cf. Equation 16).

Determining the bound of the third term is slightly complex. We list it as the following claim, and prove it soon after the proof of Theorem 1.

**Claim 1.**

$$- \sum_{k=1}^{T} (w_k^* - w_0, \sum_{j=1}^{m} g^j (w_k^j, \xi_j) - \nabla F(w_k^j))$$

$$< DL\|T \sum_{j=1}^{m} (w_1^j - w_0^j)\| + 2m^2 GDL \left( \frac{T}{\mu T + 1} + \ln |\mu T + 1| \right)$$

(22)

Combining Inequations 14, 16, 21 and Claim 1, and dividing by $T$ we obtain

$$R < \frac{2am^2G^2}{T} + DL\|T \sum_{j=1}^{m} (w_1^j - w_0^j)\| + 2m^2 GDL \left( \frac{T}{\mu T + 1} + \ln |\mu T + 1| \right)$$

$$= O(\epsilon + \frac{1}{\mu T + 1} + \ln |\mu T + 1|),$$

concluding the proof.

\[\square\]

**Proof of Claim 1.**

$$- \sum_{k=1}^{T} (w_k^* - w_0, \sum_{j=1}^{m} g^j (w_k^j, \xi_j) - \nabla F(w_k^j))$$

$$= \langle w_1^* - w_0, \sum_{k=1}^{T} (\nabla F(w_k^j) - \sum_{j=1}^{m} g^j (w_k^j, \xi_j)) \rangle$$

$$\leq \| w_1^* - w_0 \| \cdot \| \sum_{k=1}^{T} (\nabla F(w_k^j) - \sum_{j=1}^{m} g^j (w_k^j, \xi_j)) \|$$

$$\leq \| w_1^* - w_0 \| \cdot L \| \sum_{k=1}^{T} \sum_{j=1}^{m} (w_k^j - w_0^j) \|$$

$$\leq DL \| \sum_{j=1}^{m} (w_1^j - w_0^j) + ... + \sum_{j=1}^{m} (w_T^j - w_T^j) \|$$

$$= DL \|T \sum_{j=1}^{m} (w_1^j - w_0^j)\|$$

$$+ DL \|T \sum_{j=1}^{m} (w_1^j - w_0^j)\|$$

(23)

(24)

(25)

(26)

(27)

(28)
\[\begin{align*}
\text{Inequations 24 and 28 are from triangle inequality. Inequation 25 is from the fact } & \nabla F(w_k^i) = \frac{1}{n} \sum_{i=1}^{n} \nabla F(w_k^i, \xi_i) = \sum_{i=1}^{m} g_i^j(w_k^i, \xi_j) \text{ and Assumption 1’s blockwise Lipschitz-continuity. Inequation } \\
& 26 \text{ is from Assumption 3(a) and represents } \sum_{k=1}^{T} \alpha(T-k) \text{ by a summand sequence. Equation 27 comes from the fact} \\
& w_k^i - w_k^j = (w_k^i - \alpha(T-k) \nabla F(w_k^i, \xi_i) - \alpha(T-k) \nabla F(w_k^j, \xi_j)) \\
& = (w_k^i - \alpha(T-k) \sum_{j=1}^{m} g_j(w_k^i, \xi_j) - \alpha(T-k) \sum_{j=1}^{m} g_j(w_k^j, \xi_j)) \\
& = (w_k^i - w_k^j) - \alpha(T-k) \sum_{j=1}^{m} g_j(w_k^i, \xi_j)).
\end{align*}\]

5 PRIVACY PRESERVATION

The participants in CGD have to share the sum of local gradients, i.e., \( \sum_{j=1}^{m} g_j^i(w_k^i, \xi_j) \). A straightforward way is to let each participant release its local gradient \( g_j^i(w_k^i, \xi_j) \), but it may leak information about \( \xi_j \) or \( w_k^i \) [47]. To address this, we incorporate the secure addition operation [3, 4, 29, 45] on the local gradients to calculate their sum without releasing each of them. We make use of the additive secret sharing scheme proposed by Bogdanov et al. [3], which uses additive sharing over \( \mathbb{Z}_p \) for securely evaluating addition operations in a multiparty computation environment. It guarantees the secrecy of the addends even though the majority \((m-2)\) out of \( m \) of participants are compromised. A brief introduction of the additive secret sharing scheme can be found in Appendix A.

In the rest of this section, we explore the privacy preservation of CGD. We demonstrate CGD’s privacy enhancement over traditional FL. We prove that less information is exposed in CGD compared to that of traditional FL.

5.1 Information exposed in CGD

Recall that the involved parties are a set \( L \) of \( m \) participants denoted with logical identities \( l \in [1, m] \), and \( t \) is the adversarial threshold \((t \leq m-2)\) (Section 3.1.2). Let \( c \) be any subset of \( L \) that includes the compromised and colluding parties.

We demonstrate that, during optimization in CGD, given only the sum of the local gradients which are computed on different confined models, the adversary can learn no information other than their own inputs and the sum of the local gradients from other honest parties, i.e., \( \sum_{k=1}^{m} g_k^l(w_k^i, \xi_j) \in L \setminus c \).

Our analysis is based on the simulation paradigm [14]. It compares what an adversary can do in a real protocol execution to what it can do in an ideal scenario, which is secure by definition. The adversary in the ideal scenario, is called the simulator. An indistinguishability between adversary’s view in real and ideal scenarios guarantees that it can learn nothing more than their own inputs and the information required by the simulator for the simulation. A brief introduction of the simulation paradigm is given in Appendix B.

To facilitate the understanding on our analysis, we first present the used notations. Denote \( g_k^L = (g_k^l(w_k^i, \xi_j))_{l \in L} \) as the local gradients of any subset of participants \( L' \subseteq L \) at \( k \)th iteration. Let \( \text{VIEW}_{\text{real}}(g_k^L, t, P, c) \) denote their combined views from the execution of a real protocol \( P \). Let \( \text{VIEW}_{\text{ideal}}(g_k^L, z, t, F_P, c) \) denote the views of \( c \) from an ideal execution that securely computes a function \( F_P \), where \( z \) is the information required by the simulator \( S \) in the ideal execution for simulation.

The following theorem shows that when executing CGD with the threshold \( t \), the joint view of the participants in \( c \) can be simulated by their own inputs and the sum of the local gradients from the remaining honest nodes, i.e., \( \sum g_k^l |_{l \in L \setminus c} \). Therefore, \( \sum g_k^l |_{l \in L \setminus c} \) is the only information that the adversary can learn during the execution.

**Theorem 2.** When executing CGD with the threshold \( t \), there exists a simulator \( S \) such that for \( L' \subseteq L \) and \( c \subseteq L \) with \( |c| \leq t \), the output of \( S \) from \( \text{VIEW}_{\text{ideal}}(g_k^L, z, t, F_P, c) \) is perfectly indistinguishable from the output of \( \text{VIEW}_{\text{real}} \), namely

\[\text{VIEW}_{\text{real}}(g_k^L, t, P, c) \equiv \text{VIEW}_{\text{ideal}}(g_k^L, z, t, F_P, c)\]

where \( z = \sum g_k^l |_{l \in L \setminus c} \).

**Proof.** We define \( S \) through each training iteration as:

\[S_{\text{SIM}_1} : S_{\text{SIM}_1} \text{ is the simulator for the first training iteration.}\]

Since the inputs of the parties in \( c \) do not depend on the inputs of the honest parties in \( L \setminus c \), \( S_{\text{SIM}_1} \) can produce a perfect simulation by
running $c$ on their true inputs, and $L \setminus c$ on a set of pseudorandom vectors $\eta_1^{L \setminus c} = \{\eta_i^{L \setminus c}\}_{i \in L \setminus c}$ in a way that
\[
\sum \eta_1^{L \setminus c} = \sum \eta_i^{L \setminus c} |_{i \in L \setminus c} = \sum g_i^c - \sum g_i^1 |_{i \in L \setminus c}.
\]

Since each $g_i^1(w_i^1, \xi_i) = 0$ is computed from its respective confined model $w_i^1$ which is randomized in the initialization, the pseudorandom vectors $\eta_i^{L \setminus c}$ generated by $SIM_1$ for the inputs of all parties in $L \setminus c$, and the joint view of $c$ in $VIEW_{ideal}$, will be identical to that in $VIEW_{real}$, namely
\[
\left( \sum \eta_1^{L \setminus c} + \sum g_i^1 \right) \equiv \sum g_i^L,
\]
and the information required by $SIM_1$ is $z = \sum g_i^1 |_{i \in L \setminus c}$.

$SIM_{k+1} (k \geq 1)$: $SIM_{k+1}$ is the simulator for the $(k+1)^{th}$ training iteration.

In Real execution, $\sum g_{k+1}^L$ is computed as
\[
\sum g_{k+1}^L = \sum g_i^{L\setminus c} + \sum g_i^1 |_{i \in L \setminus c} = \sum g_i^{L\setminus c} + \sum g_i^1 = \sum g_i^{L\setminus c} \sum g_i^1 |_{i \in L \setminus c}.
\]

In Ideal execution, since each $g_i^{L\setminus c} = \{g_i^{L\setminus c}\}_{i \in L \setminus c}$ is also computed from randomized $w_i^1$, $SIM_{k+1}$ can produce a perfect simulation by running the parties $L \setminus c$ on a set of pseudorandom vectors $\eta_1^{L \setminus c} = \{\eta_i^{L \setminus c}\}_{i \in L \setminus c}$ in a way that
\[
\sum \eta_1^{L \setminus c} = \sum \eta_i^{L \setminus c} |_{i \in L \setminus c} = \sum g_i^{L\setminus c} - \sum g_i^{L\setminus c} |_{i \in L \setminus c}.
\]

As such, the joint view of $c$ in $VIEW_{ideal}$, will be identical to that in $VIEW_{real}$
\[
\left( \sum \eta_1^{L \setminus c} + \sum g_i^c \right) \equiv \sum g_i^{L\setminus c},
\]
and the information required by $SIM_{k+1}$ is $z = \sum g_i^{L\setminus c} |_{i \in L \setminus c}$.

By summarizing $SIM_1$ and $SIM_{k+1}$, the output of the simulator $VIEW_{ideal}$ of each training iteration is perfectly indistinguishable from the output of $VIEW_{real}$, and knowledge of $z$ is sufficient for the simulation, completing the proof. \hfill \Box

### 5.2 Information exposed in traditional FL

In this section, we demonstrate information exposed in traditional FL, including plain federated SGD, secure aggregated federated SGD, and differentially private federated SGD.

Let $g_i^L(w_k, \xi) = \frac{\partial}{\partial \xi} \nabla F(w_k, \xi)$ be the local gradient with respect to training dataset $\xi = (x_i, y_i)$. In traditional FL, $w_k$ is the public global model shared among the participants.

For the sake of simplicity, we assume SGD is not generalized to mini-batch update, i.e., we have $\frac{1}{|x_i|} = 1$. Then, Equation 33 can be written as,
\[
\nabla F(w_k, x_i, y_i) = \frac{1}{|x_i|} \nabla F(w_k, x_i, y_i),
\]

be the local gradient with respect to training dataset $\xi = (x_i, y_i)$.

In traditional FL, $w_k$ is the global model shared among the participants.

According to the chain rule in calculus, $\nabla F(w_k, x_i, y_i)$ is computed as
\[
\frac{\partial}{\partial \xi} \left( \frac{\partial}{\partial w_k} \right) \nabla F(w_k, x_i, y_i) = \frac{\partial}{\partial w_k} \frac{\partial}{\partial \xi} \nabla F(w_k, x_i, y_i),
\]

where $x_i, w_k$ is matrix multiplication of training samples $x_i$ and $w_k$, and $h$ is the hypothesis function which is determined by the learning model. For example, in logistic regression, $h$ is usually a sigmoid function, while in neural network, $h$ is a composite function that is known as forward propagation. Let $\Delta(x_i | w_k, y_i) = \frac{\partial}{\partial \xi} \frac{\partial}{\partial w_k} \frac{\partial}{\partial \xi} \nabla F(w_k, x_i, y_i)$ equals to $x_i^T$. Then, Equation 33 can be written as
\[
g_i^L(w_k, \xi) = x_i^T \Delta(x_i | w_k, y_i).
\]

**Plain federated SGD.** In plain federated SGD, $w_k$ is updated as the following (by combining Equation 1 and 35)
\[
w_{k+1} \leftarrow w_k - \alpha_k x_i^T \Delta(x_i | w_k, y_i),
\]

in which the local gradient $x_i^T \Delta(x_i | w_k, y_i)$ is shared among the participants. As such, by knowing both $x_i^T \Delta(x_i | w_k, y_i)$ and $w_k$, the adversary is able to derive indicative information about $(x_i, y_i)$. For example, in linear regression, since $x_i^T \Delta(x_i, w_k, y_i) = x_i^T (x_i w_k - y_i)$, the adversary is able to obtain $(x_i^T x_i, x_i^T y_i)$.

**Secure aggregated federated SGD.** In this category of traditional FL \cite{4, 30, 31, 34, 40, 47}, the local gradients are protected by secure aggregation, and the global model $w_k$ is updated as
\[
w_{k+1} \leftarrow w_k - \alpha_k \sum_l g_i^L(w_k, \xi) |_{l \in D},
\]

where $D \subseteq L$. By combining Equation 37 and 35, we have
\[
w_{k+1} \leftarrow w_k - \alpha_k \sum_l x_i^T \Delta(x_i | w_k, y_i) |_{l \in D},
\]

in which the aggregated gradient, $\sum x_i^T \Delta(x_i | w_k, y_i)$, is shared among the participants.

As the global model $w_k$ is also shared, by observing the changes of the aggregated gradient during training iterations, i.e., $w_k - w_{k+1}$, the adversary is still able to obtain indicative information about $(x_i, y_i)$.

Let $x^L$, $y^L$ respectively denote the concatenated matrix of training samples $\{x_i\}_{i \in L}$, and labels $\{y_i\}_{i \in L}$ of any subset of participants $L' \subseteq L$. The following theorem shows that when executing secure aggregated federated SGD with the threshold $t$, the joint view of the participants in $c$ can be simulated by (1) the sum of the local gradients from the remaining honest nodes in $D$, that is, $\sum g_i^L(w_k, \xi) |_{l \in D} |_{c}$, and indicative information about $(x_i, y_i)$ in $D \setminus c$, that is, $x^D |_{c}^T \Delta(x^D | c, w_k, y^D | c)$. For example, in linear regression, as $x^D |_{c}^T \Delta(x^D | c, w_k, y^D | c) = x^D |_{c}^T (x^D | c w_k - y^D | c)$, the adversary is able to simulate $x^D |_{c}^T x^D |_{c}^T (x^D | c w_k - y^D | c)$.

**Theorem 3.** When executing secure aggregated federated SGD with the threshold $t$, there exists a simulator $S$ such that for $L, D, c$, with $D \subseteq L, c \subseteq L$ and $|c| \leq t$, the output of $S$ from $VIEW_{ideal}$ is perfectly indistinguishable from the output of $VIEW_{real}$, namely
\[
VIEW_{real}(\sum_l g_i^L(w_k, \xi) |_{l \in D}, t, \mathcal{P}, c) \equiv VIEW_{ideal}(\sum_l g_i^L(w_k, \xi) |_{l \in c}, z_1, z_2, t, \mathcal{P}, c)
\]
where

\[ z_1 = \sum g'_k(w_k, \xi_l) \mid l \in D, c \mid z_2 = x^D(c^T \Delta(x^D(c^T w_k, y^D(c)) \right]

Proof. SIM\(_k\) (\(k \geq 1\)): SIM\(_k\) is the simulator for the \(k^{th}\) training iteration.

In Real execution, \(\sum g_k(w_k, \xi_l) \mid l \in D\) is computed as (with Equation 37 and 38)

\[ \sum g'_k(w_k, \xi_l) \mid l \in D = \sum x_l^T \Delta(x_l w_k, y_l) \mid l \in D = \frac{1}{\alpha}(w_k - w_{k+1}) \]

(39)

In Ideal execution, since \(w_k\) is shared among the participants, by computing \(w_k - w_{k+1}\), SIM\(_k\) can produce a perfect simulation by running the parties \(D \setminus c\) on

\[ \sum g'_k(w_k, \xi_l) \mid l \in D, c \mid \text{or}, x^D(c^T \Delta(x^D(c^T w_k, y^D(c)), \right]

As such, the joint view of \(c\) in VIEW\(_{\text{ideal}}\) will be identical to that in VIEW\(_{\text{real}}\), since

\[ \sum g'_k(w_k, \xi_l) \mid l \in D, c \mid + \sum g'_k(w_k, \xi_l) \mid l \in \exists \sum g'_k(w_k, \xi_l) \mid l \in D \]

and,

\[ x^D(c^T \Delta(x^D(c^T w_k, y^D(c))) + x^T \Delta(x^T w_k, y^D(c)) \right]

\[ \equiv x^D(c^T \Delta(x^D(c^T w_k, y^D(c)))) \]

(41)

Thus the information required by SIM\(_k\) is \(z_1 = \sum g'_k(w_k, \xi_l) \mid l \in D, c \mid\) and \(z_2 = x^D(c^T \Delta(x^D(c^T w_k, y^D(c)))\).

Together, we have \(\sum g'_k(w_k, \xi_l) \mid l \in D, c \mid x^D(c^T \Delta(x^D(c^T w_k, y^D(c)))\) being the information that the adversary can learn during the execution. \(\square\)

Differentially private federated SGD. In most differentially private federated SGD, the local gradients are protected by additive noise mechanism as

\[ w_{k+1} \leftarrow w_k - \alpha_k(g'_k(w_k, \xi_l) + N_k) \]

where \(N_k\) denote the noise added at the \(k^{th}\) iteration. By combing Equation 35, it can be written as

\[ w_{k+1} \leftarrow w_k - \alpha_k(x_l^T \Delta(x_l w_k, y_l) + N_k) \]

The information exposed among participants is \(x_l^T \Delta(x_l w_k, y_l) + N_k\), and the additive noise \(N_k\) prevent one from directly deriving \(x_l^T \Delta(x_l w_k, y_l)\) by subtracting \(w_k\) and \(w_{k+1}\). However, with repeated access to the datasets during training epochs, \(\epsilon\) (the parameter of privacy loss) accumulates, i.e., privacy degrades, as the effect of added noise being canceled out [2, 8].

5.3 Enhanced privacy over traditional FL

Traditional FL requires all participants to update the same global model during the training process. Every participant thus sees the identical intermediate results, as the same aggregated gradients are shared. This is the root cause of most privacy threats against FL. CGD breaks the mode of single global model, by introducing random variation among the proprietary global models of the participants. The variation hides each global model from other participants, such that the privacy is enhanced in general.

In Table 2, we summarize the privacy enhancement from the perspective of adversary’s observation, i.e., the exposed information to the adversary during the optimization. In traditional FL, the sharing of global model \(w_k\), even if the local gradients are protected, is still subject to information leakage about the original dataset. By eliminating the sharing of \(w_k\) and letting each participant confine its own \(w'_k\), CGD achieves boosted privacy over traditional privacy-preserving FL. (1) Compared to secure aggregated FL in which indicative information about original datasets can be observed via the sharing of \(w_k\) (Theorem 3), the variation among \(w'_k\) in CGD prevents such information from disclosure, and guarantees that only the sum of local gradients is exposed during the optimization (Theorem 2). (2) Compared to differential privacy mechanism in which privacy decays with the increasing training epochs, the variation introduced to each \(w'_k\) hides the local gradients throughout the whole training process, and thus retains privacy regardless of the number of training epochs.

6 CASE STUDY: CONFINED GRADIENT DESCENT FOR A N-LAYER NEURAL NETWORK

CGD can be applicable to any machine learning algorithms that use gradient descent for optimization. In this section, we apply it to a N-layer neural network to demonstrate its usability. We assume that a centralized dataset \(\xi\) is horizontally and vertically partitioned and distributed to \(m\) participants where \(m = (m^h \times m^v)\), i.e., the number of horizontal partitions multiplied by the number of vertical partitions. The participant \(l\) owns a private part of the training dataset, denoted by \(\xi_{l(i)}\) \((i \in [1, m^h], j \in [1, m^v])\), as well as its confined model parameters, denoted by \(w'_k(l(i), \ldots, w'_k(N)(l(i))\). Since each participant owns different confined models and proportion of the dataset, the training prediction \(g'_k(l, i)\) is also different and kept confined in its own (shown in Figure 3).

Algorithm 2 presents the detailed algorithm. The participant \(l\) first randomly initializes its confined model \(w'_k(l(i), \ldots, w'_k(N)(l(i))\) (line 3). The size of the model in the first layer is \(w'_k(l(i), \ldots, w'_k(N)(l(i))\), where \(d_l\) is the number of features in \(\xi_{l(i)}\), and the size of models in the remaining layers is \(w'_k(l(i), \ldots, w'_k(N)(l(i))\). It is possible that different participants have different size of \(w'_k(l(i), \ldots, w'_k(N)(l(i))\) because the number of features \(d_l\) held by each participant may differ, while \(w'_k(l(i), \ldots, w'_k(N)(l(i))\) \((i \in [2, N])\) keep the same size in each participant.

Next, we detail the training process in each iteration. In the forward propagation, each participant separately computes the output of each layer \(a'_k(l(i), \ldots, a'_k(N)(l(i))\) based on its own confined models (line 5 to 11). In the backward propagation, each participant solely computes the local gradient of each layer \(g'_k(l(i), \ldots, g'_k(N)(l(i))\) which is computed from its own private dataset and confined model (line 12 to 18). Then, they securely evaluate the sum of local gradients from the \(N^{th}\) layer to the 2nd layer (line 19 to 21). For the first layer gradient, since the size of \(w'_k(l(i), \ldots, a'_k(N)(l(i))\) can be different, the sum
Table 2: A summary of the exposed information in traditional FL and CGD

| Techniques                        | The observation of an honest-but-curious white-box adversary during training iterations | Exposed indicative information about \((x_i, y_i)\) (in the example of linear regression) | Boosted privacy of CGD over traditional FL |
|-----------------------------------|----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|------------------------------------------|
| Plain federated SGD               | Local gradients: \(g^{(0)}_k(w_k, \xi_i)\), which equals \(x_i \Delta(x_i w_k, y_i)\) | \([x_i^T x_i^T y_i]\) | In plain federated SGD, indicative information about the local training dataset can be observed. |
| Secure aggregated federated SGD   | Sum of the local gradients from the remaining honest participants: \(\sum g^{(0)}_k(w_k, \xi_i)\mid_{i \in \mathcal{I}}\), and with a shared \(w_k\), it equals \(x^D \Delta(x^D y^D e^{T} y^D e)\) (Theorem 3). | \(\{x^D \Delta(x^D y^D e^{T} y^D e)\}\) | With the shared global \(w_k\), indicative information about the concatenated training datasets from honest participants can be observed. |
| Differentially private federated SGD (additive noise based mechanism) | Perturbed local gradients: \(g^{(r)}_k(w_k, \xi_i) + \mathcal{N}_k\), which equals \(x_i^T \Delta(x_i w_k, y_i) + \mathcal{N}_k\) | Not applicable. | The decay of privacy with increasing number of training epochs is one of the limitations of most additive-noise based differentially private learning. |
| CGD                              | Sum of the local gradients from the remaining honest participants: \(\sum g^{(r)}_k(w_k, \xi_i)\mid_{i \in \mathcal{I}}\) (Theorem 2). | Not applicable. | In CGD, each local gradient is computed from a different and private \(w_k\), and only the sum of local gradients is exposed during the training. As such, (1) the indicative information exposed in the secure aggregation cannot be derived in CGD; (2) the randomness introduced in each \(w_k\) hides the information about local gradient throughout the training, and thus privacy does not decay. |

7 EXPERIMENTS

We implement CGD and evaluate its performance of model accuracy. It is implemented using C++, and we use Eigen library [15] to handle matrix operations and ZeroMQ library [18] for distributed messaging. Our evaluation focuses on the performance of CGD in terms of validation loss and accuracy, and the influence of the factors (the parameters \(\delta\) and \(\mu\)) on its performance.

7.1 Performance Evaluation

We evaluate validation loss and accuracy on two popular benchmark datasets: MNIST [28] and CIFAR-10 [26]. We compare CGD with the training on aggregated data (referred to as the centralized training), the training on each participant’s local data only (referred to as the local training), and FL with differential privacy (referred to as the DP-FL). CGD is expected to outperform the local training and DP-FL, and approach that of the centralized training. The settings of all trainings are listed below.

- **CGD training.** The model is trained using Algorithm 1 on the dataset which is partitioned and distributed to \(m\) participants, where \(m = (m^b \times m^p)\). We observe that the performance of all confined models are proximal, and thus we report the worst performance in this section.
- **Centralized training** (the baseline). We take the performance of centralized training as the baseline. In this setting, the training is on the entire dataset using the batch update.
- **Local training.** In this setting, the data and model are distributed to \(m\) participants in the same way as in CGD. Each participant separately trains their local models from their local data, without sharing the gradients.
- **DP-FL.** We take the scheme proposed by Abadi et al. [2], one of the state-of-the-art differentially private FL schemes, in our experiments. Due to the different setting of network architecture and hyper-parameters, its centralized baseline is slightly different from ours. Therefore, we report the margin between its performance and the performance of its baseline.

7.1.1 MNIST. Our first set of experiments are conducted on the standard MNIST dataset which is a benchmark for handwritten digit recognition. It has 60,000 training samples and 10,000 test samples, each with 784 features representing 28 × 28 pixels in the image. We use a fully-connect neural network (FNN) with ReLU of 256 units and softmax of 10 classes with cross-entropy loss.

We conduct our experiments with CGD’s default settings:

- \(\mu = 0\), leading to a fixed learning rate which is in line with most machine learning algorithms,
- \(\delta = 0.1\) (cf. Equation 11), and
Algorithm 2 Confined Gradient Descent of a N-layer fully-connected neural network for \( m = (m^h \times m^p) \) participants

1. **Input:** Local training data \( \xi_{(i,j)} \) (\( i \in m^h, j \in m^p \)), activation functions of \( N \) layers: \( \sigma^{(1)} \) \( ... \) \( \sigma^{(N)} \), cost function \( J \), the number of training iterations \( T \)
2. **Output:** Confined global model parameters of \( N \) layers:
   \[ w_k^{(1)}(i,j), ..., w_k^{(N)}(i,j) \] (\( i \in m^h, j \in m^p \))
3. **Initialize:** \( k \leftarrow 1 \), each participant \( l_{(i,j)} \) randomizes its own \( w_k^{(1)}(i,j), ..., w_k^{(N)}(i,j) \)
4. **while** \( k \leq T \) **do**
   
   **Forward propagation**
   
   for all participants \( l_{(i,j)} \) **do in parallel**
   
   \[
   z_k = \sum_{l=1}^{(i,j)} \xi(l_{(i,j)}) \]
   
   \[
   z_k = \sigma^{(1)}(z_k) \]
   
   \[
   z_k = \sigma^{(2)}(z_k) \]
   
   **end for**

   **Backward propagation**

   Calculate the local gradients of each layer
   
   for all participants \( l_{(i,j)} \) **do in parallel**
   
   \[
   \delta_k^{(N)}(z_{(i,j)}) = \frac{\partial J}{\partial z_k^{(N)}} \]
   
   \[
   g_k^{(N)}(z_{(i,j)}) = \frac{\delta_k^{(N)}(z_{(i,j)})}{\partial z_k^{(N)}} \]
   
   **end for**

   Securely evaluate the sum of local gradients
   
   for all participants \( l_{(i,j)} \) **do in parallel**
   
   \[
   g_k^{(r)} = \sum_{i,j=1}^{m^h \times m^p} g_k^{(r)(i,j)} \quad (r \in [2, N]) \]
   
   **end for**

   for all groups \( l_j = \{l_{(i,j)}, ..., l_{(m^h,j)}\} \) **do in parallel**

   Evaluate the sum of first layer:
   
   \[
   g_k^{(1)}(l_{(i,j)}) = \sum_{i,j=1}^{m^h \times m^p} g_k^{(1)(i,j)} \]
   
   **end for**

   Descend
   
   Choose a stepsize: \( \alpha_k \)

   for all groups \( l_j = \{l_{(i,j)}, ..., l_{(m^h,j)}\} \) **do in parallel**

   \[
   w_k^{(1)}(i,j) - \alpha_k g_k^{(1)(i,j)} \]
   
   **end for**

   for all participants \( l_{(i,j)} \) **do in parallel**

   \[
   w_k^{(r)}(i,j) = w_k^{(r)(i,j)} - \alpha_k g_k^{(r)}(i,j) \quad (r \in [2, N]) \]
   
   **end for**

   \[
   k \leftarrow k + 1 \]
   
   **end while**

---

**Figure 4:** Results on the validation loss and accuracy for different number of participants on the MNIST dataset.

- the standard weight initialization scheme based on the Gaussian distribution of mean 0 and variance 1.

Figure 4 summarizes the performance with varying number of participants. In general, CGD stays close to the centralized baseline in both validation loss and accuracy, and as expected, significantly outperforms the local training. In the worst case of \( m = (100 \times 112) \) when CGD contains the greatest number of participants, i.e., each participant owns a small proportion consisting of 60 samples with 7 features, CGD still achieves 0.209 and 93.74% in validation loss and accuracy respectively, close to 0.081 and 97.54% of the centralized baseline. In contrast, the performance of local training declines to 2.348 and 11.64%. Table 3 lists the detailed performance comparison with both centralized and local trainings, and Table 4 lists the comparison with DP-FL. More results are given with varying number of participants in Appendix C (Figure 9 and Table 8).

---

7.1.2 CIFAR-10. Our second set of experiments are conducted on the CIFAR-10 dataset. It consists of 60000 32 \( \times \) 32 colour images in 10 classes (e.g., airplane, bird, and cat), with 6000 images per class. The images are divided into 50000 training images and 10000 test images. We use the ResNet-56 architecture proposed by He et al. [17]. It takes as input images of size 32 \( \times \) 32, with the per-pixel mean subtracted. Its first layer is 3 \( \times \) 3 convolutions, and then
Table 3: Performance comparison on MNIST with default settings.

|                  | Validation loss | Validation accuracy |
|------------------|-----------------|---------------------|
| Centralized      | 0.081           | 97.54%              |
| CGD              |                 |                     |
| m=(10 × 7)       | 0.143           | 96.5%               |
| m=(100 × 49)     | 0.189           | 94.21%              |
| m=(1000 × 112)   | 0.209           | 93.74%              |

Table 4: Comparison between CGD and traditional FL via differential privacy in terms of the difference of validation accuracy to the centralized baseline on MNIST.

|                  | Difference to the baseline |
|------------------|-----------------------------|
| Participants     |                             |
| m=(10 × 7)       | 1.0%                        |
| m=(100 × 49)     | 3.3%                        |
| m=(1000 × 112)   | 3.8%                        |
| Noise levels     |                             |
| $\epsilon = 8$  | (small noise)               |
| $\epsilon = 2$  | (medium noise)              |
| $\epsilon = 0.5$| (large noise)               |

1 The centralized baseline of validation accuracy on the MNIST in [2] is 98.3%.

a stack of 3 × 18 layers is used, with 3 × 3 convolutions of filter sizes {32, 16, 8}, respectively, and 18 layers for each filter size. The numbers of filters are {16, 32, 64} in each stack. The network ends with an average pooling layer, a fully-connected (FC) layer with 1024 units, and softmax. Our training data augmentation follows the setting in [17]. For each training image, we generate a new distorted image by randomly flipping the image horizontally with probability 0.5, and adding 4 amounts of padding to all sides before cropping the image to the size of 32 × 32.

Since our focus is on evaluating the proposed CGD, rather than enhancing the state-of-the-art analysis on CIFAR-10, we utilize the transferability2 of convolutional layers to save the computational cost of computing per-example gradients. We follow the experiment setting of Abadi et al. [2], which treats the CIFAR-10 as the private dataset and CIFAR-100 as a public dataset. CIFAR-100 has the same image types as CIFAR-10, and it has 100 classes containing 600 images each. We use CIFAR-100 to train a network with the aforementioned architecture, and freeze the parameters of the convolutional layers and retrain only the last FC layer on CIFAR-10.

Training on the entire dataset with batch update reaches the validation accuracy of 75.75%, which is taken as our centralized training baseline3. For the CGD training, each participant feeds the pre-trained convolutional layers with their private data partitions, generates the input features to the FC layer, and randomly initializes the confined model parameters of the FC layer. They then take the input to the FC layer as the private local training data, and train a model using Algorithm 1 with $\delta = 0.01$ and $T = 6000$.

2Transfer learning allows the analyst to take a model trained on one dataset and transfer it to another without retraining [41].
3We note that by making the network deeper or using other advanced techniques, better accuracy can be obtained, with the state-of-the-art being about 99.37% [23].

Figure 5: Results on the validation loss and accuracy for different number of participants on the CIFAR-10 dataset.

Table 5: Performance comparison on CIFAR-10.

|                  | Validation loss | Validation accuracy |
|------------------|-----------------|---------------------|
| Centralized      | 0.675           | 75.72%              |
| CGD              |                 |                     |
| m=(10 × 2)       | 0.720           | 74.79%              |
| m=(100 × 16)     | 0.723           | 74.42%              |
| m=(1000 × 32)    | 0.724           | 74.29%              |

Figure 5 and Table 5 summarize our experimental results against centralized training and local training. The results on the validation loss and accuracy are generally in line with that on the MNIST dataset. In the worst case of $m = (1000 \times 32)$, CGD achieves 0.724 and 74.29% in validation loss and accuracy respectively, which are relatively near to the centralized baseline (0.675 and 75.72%). Table 6 summarize the results against DP-FL. In line with the results on MNIST, the accuracy difference to the centralized baseline in CGD is smaller than that in DP-FL.
### 7.2 Influencing Factors of CGD

In this section, we study the influence of initialization and the parameter $\mu$ on CGD performance. This study is conducted with the MNIST dataset.

#### 7.2.1 Role of the initialization

According to Theorem 1, the gap between CGD solution and the centralized model is bounded by $e = m \| \frac{1}{E} \sum_{j \in m} (w_1^j - w_0^j) \|$, in which each $w_1^j$ is decided by $\delta$ in Equation 11. Therefore, we investigate how the initialization setting affects CGD’s performance.

To this end, we conduct two experiments in which we mutate parameter $\delta$ while keeping other settings unchanged (i.e., $\mu = 0$ and $T = 2000$). In the first experiment, we let all participants use the same $\delta$ in $\{0.1, 0.06, 0.01, 0.001\}$, as they are around $\frac{1}{\sqrt{60000}}$ (recall that 60000 is the sample size of MNIST). In the other experiment, we let each participant randomly select its own $\delta$ based on the uniform distribution within the range from 0.001 to 0.1.

The results of our first experiment are shown in Figure 6 and the first five columns in Table 7. In general, as $\delta$ decreases, CGD achieves better performance. This confirms our expectation: decreasing $\delta$ would reduce the value of $\| \frac{1}{E} \sum_{j \in m} (w_1^j - w_0^j) \|$, such that the confined models are closer to the centralized optimum. We have not observed significant difference from $\delta = 0.1$, 0.06 and 0.01. In the case of $m = (1000 \times 112)$, the validation accuracy and loss achieve 97.08% and 0.096 with $\delta = 0.01$ which outperform 93.74% and 0.209 in the setting of $\delta = 0.1$, but both are close to the centralized model whose validation accuracy and loss are 97.61% and 0.078 respectively. However, $\delta$ cannot be set too small, in order to maintain numerical stability in neural network [13]. For example, when we lower $\delta$ to 0.001, the performance starts decreasing.

The results of our second experiment are shown in Figure 7 and the last column in Table 7. When the participants uniformly randomize their $\delta$s from 0.001 to 0.1, the performance of CGD is still comparable with the centralized model. For example, with $m = (1000 \times 112)$ participants, CGD achieves 95.82% and 0.136 in the validation accuracy and loss. This suggests that CGD keeps robust when the $\delta$s of its participants differ by two orders of magnitude.

#### 7.2.2 Effect of the parameter $\mu$

According to Theorem 1, the convergence speed of CGD is affected by the parameter $\mu$. We thus conduct an experiment to investigate this relation. In this experiment, we tune $\mu$ while keeping $\delta$ fixed as 0.1. For the stepsize ($\alpha_k$), we run CGD with a fixed $\alpha$ till it (approximately) reaches a preferred point, and then continue our experiment with $\alpha_k = \alpha$. This is to keep CGD practical, as each time the stepsize is diminished, more iterations are required. Therefore, the first 6000 iterations is run with $\alpha = 0.01$, and then diminish $\alpha_k$ in each of the following 8000 iterations.

Figure 8 demonstrates the results with varying $\mu$ for $m = (100 \times 49)$ and $m = (1000 \times 112)$. For the case of $m = (100 \times 49)$, $\mu \geq 0.01$...
gives a slightly faster speed. CGD reaches the validation accuracy of 95.9% at 11401^{th} iteration, 2340 iterations (16.7% less) faster than training with $\mu < 0.01$. For the case of $m = 1000 \times 112$, $\mu > 0.05$ gives a faster speed. CGD reaches the validation accuracy 95.68% at 11821^{th} iteration, 2140 iterations (15.28% less) faster than training with $\mu \leq 0.05$. Even though such slight difference is observed, our experiment suggests that the effect of tuning $\mu$ is relatively limited. The gain of the validation accuracy is only within 0.2% in the cost of around 2000 iterations.

8 CONCLUSION

We have presented CGD, a novel optimization algorithm for learning confined models to enhance privacy for federated learning. Privacy preservation is achieved against an honest-but-curious adversary even though the majority ($74.4\%$ with ($0<\mu<0.01$), $97.01\%$ with ($0.01=\mu<1$) of the participants on the MNIST dataset. Both are comparable to the performance of centralized training.

A number of future work directions are of interest. In particular, we see new research opportunities in applying our techniques to asynchronous FL, allowing different participants to be at different iterations of model updates up to a bounded delay. We are also considering other types of deep networks, for example, unsupervised neural networks such as autoencoders.

REFERENCES

[1] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016).
[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 308–318.
[3] Dan Bogdanov, Sven Laur, and Jan Willemson. 2008. Sharemind: A framework for fast privacy-preserving computations. In European Symposium on Research in Computer Security. Springer, 192–206.
[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 1175–1191.
[5] Léon Bottou. 2012. Stochastic gradient descent tricks. In Neural networks: Tricks of the trade. Springer, 421–436.
[6] Leon Bottou, Frank E. Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. Siam Review 60, 2 (2018), 233–311.
[7] Yi-Ruei Chen, Amar Rezapour, and Wen-Guyu Tseng. 2018. Privacy-preserving ridge regression on distributed data. Information Sciences 451 (2018), 34–49.
[8] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science 9, 3–4 (2014), 211–467.
[9] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 1322–1333.
[10] Adrágio Gascon, Philipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. 2016. Secure Linear Regression on Vertically Partitioned Datasets. IACR Cryptology ePrint Archive 2016 (2016), 892.
[11] Adrágio Gascon, Philipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. 2017. Privacy-preserving distributed linear regression on high-dimensional data. Proceedings on Privacy Enhancing Technologies 2017, 4 (2017), 345–364.
[12] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557 (2017).
[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics. 249–256.
[14] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1990. How to play any mental game, or a completeness theorem for protocols with honest majority. In Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali. 307–328.
[15] Gaël Guennébaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.
[16] Inken Hagestedt, Yang Zhang, Matthias Humbert, Pascal Berrang, Haixu Tang, XiaoFeng Wang, and Michael Backes. 2019. MBeacon: Privacy-Preserving Beacons for DNA Methylation Data. In NDSS.
[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.
[18] Pieter Hintjens. 2013. ZeroMQ: messaging for many applications. “O’Reilly Media, Inc.”
[19] Qirong Ho, James Cigar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Philip B Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. 2013. More effective distributed ml via a stale synchronous parallel parameter server. In Advances in neural information processing systems. 1223–1231.
[20] Kevin Huse, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomidou, Gregory R Ganger, Philip B Gibbons, and Onur Mutlu. 2017. Gaa: Geo-distributed machine learning approaching (LAN) speeds. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI’17). 629–647.
[21] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. 2019. FDML: A collaborative machine learning framework for distributed features. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2232–2240.
[22] Hyesung Kim, Jihong Park, Mehlid Bennis, and Seong-Lyun Kim. 2019. Blockchaained on-device federated learning. IEEE Communications Letters 24, 6 (2019), 1279–1283.
[23] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. 2019. Big transfer (bit): General visual representation learning. arXiv preprint arXiv:1912.11796 v. 2 (2019), 8.
[24] Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.05257 (2016).
[25] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016).
[26] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. Commun. ACM 60, 6 (2017), 84–90.
[28] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/.
[29] Hsiao-Ying Lin and Wen-Guy Tseng. 2005. An efficient solution to the millionaires’ problem based on homomorphic encryption. In International Conference on Applied Cryptography and Network Security. Springer, 456–466.
[30] Jian Liu, Mika Juuti, Yao Lu, and Nadarajah Asokan. 2017. Oblivious neural network predictions via minionn transformations. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 1322–1333.
Appendix A ADDITIVE SECRET SHARING SCHEME

Secret sharing schemes aim to securely distribute secret values amongst a group of participants. CGD employs the secret sharing scheme proposed by [3], which uses additive sharing over $\mathbb{Z}_{2^2}$.

In this scheme, a secret value $srt$ is split to $s$ shares $E_{srt}^1, ..., E_{srt}^s \in \mathbb{Z}_{2^2}$ such that

$$E_{srt}^1 + E_{srt}^2 + ... + E_{srt}^s \equiv srt \mod 2^2,$$  \hspace{1cm} (42)

and any $s - 1$ elements $E_{srt}^i, ..., E_{srt}^{s-1}$ are uniformly distributed. This prevents any participant who has part of the shares from deriving the value of $srt$, unless all participants join their shares.

In addition, the scheme has a homomorphic property that allows efficient and secure addition on a set of secret values $srt_1, ..., srt_s$ held by corresponding participants $S_1, ..., S_s$. To do this, each participant $S_i$ executes a randomised sharing algorithm $Shr(srt_i, S)$ to split its secret $srt_i$ into shares $E_{srt_i}^1, ..., E_{srt_i}^s$, and distributes each $E_{srt_i}^j$ to the participant $S_j$. Then, each $S_i$ locally adds the shares it holds $E_{srt_i}^j, ..., E_{srt_i}^j$, and produces $\Sigma_{j=1}^s E_{srt_i}^j$ (denoted by $E^i$ for brevity). After that, a reconstruction algorithm $Rec((E^i, S_i)_i \in S)$, which takes $E^i$ from each participant and add them together, can be executed by an aggregator to reconstruct the $\Sigma_{i=1}^s srt_i$ without revealing any secret addends $srt_i$.

Appendix B SIMULATION PARADIGM

In simulation paradigm (a.k.a., the real/ideal model) [14], the security of a protocol is proved by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario, which is secure by definition. Formally, a protocol $P$ securely computes a functionality $F_p$, if for every adversary $A$ in the real model, there exists an adversary $S$ in the ideal model, such that the view of the adversary from a real execution $VIEW_{real}$ is indistinguishable from the view of the adversary from an ideal execution $VIEW_{ideal}$. The adversary $S$ in the ideal model, is called the simulator. An indistinguishability between $VIEW_{real}$ and $VIEW_{ideal}$ guarantees that the adversary can learn nothing more than their own inputs and the information required by $S$ for the simulation. In other words, the information required by $S$ for the simulation is the only information that can leak to adversary $A$ from the real execution.

Let $c$ denote the set of corrupted parties. The simulator performs the following operations:

- Generate dummy inputs $\{\eta\}$ for each honest party $l \notin c$ and receives the actual inputs $\{x\}$ of corrupted parties $l \in c$;
- Run $P$ over $\{\eta\} (l \notin c)$ and $\{x\} (l \in c)$ and add all messages sent/received by corrupted parties to $VIEW_{ideal}$;
- Send the inputs of corrupted parties $\{x\} (l \in c)$ to the trusted third party;
- Receive the outputs of corrupted parties $\{y\} (l \in c)$ from the trusted third party and add them to $VIEW_{ideal}$.

Meanwhile, a real instance of $P$ is executed with actual inputs for all parties, and $VIEW_{real}$ is created by gathering inputs of corrupted parties, messages sent/received by corrupted parties during the protocol and their final outputs. Once the simulation is finished, the security is proved by showing that $VIEW_{ideal}$ is indistinguishable from $VIEW_{real}$.
Appendix C  SUPPLEMENTARY EXPERIMENTAL RESULTS

Figure 9: Supplementary experimental results on the validation loss and accuracy for different number of participants on the MNIST dataset in the default setting.

Table 8: Supplementary experimental results on the validation loss and accuracy for different number of participants on the MNIST dataset in the default setting.

|                | Validation loss | Validation accuracy |
|----------------|-----------------|---------------------|
| Centralized    | 0.081           | 97.54%              |
| m=(10 x 1)     | 0.143           | 95.62%              |
| m=(100 x 1)    | 0.188           | 94.22%              |
| m=(1000 x 1)   | 0.208           | 93.76%              |

a) $m = (10 \times 1)$

b) $m = (10 \times 1)$

c) $m = (100 \times 1)$

d) $m = (100 \times 1)$

e) $m = (1000 \times 1)$

f) $m = (1000 \times 1)$