IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 May 2024, accepted 15 June 2024, date of publication 24 June 2024, date of current version 1 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3418016

RESEARCH ARTICLE

Enhancing Efficiency in Privacy-Preserving Federated Learning for Healthcare: Adaptive Gaussian Clipping With DFT Aggregator

MUHAMMAD AYAT HIDAYAT, (Graduate Student Member, IEEE), YUGO NAKAMURA, (Member, IEEE), AND YUTAKA ARAKAWA, (Member, IEEE)
Department of Information Science and Technology, ISEE, Kyushu University, Fukuoka 819-0385, Japan
Corresponding author: Muhammad Ayat Hidayat (muhammad.971@s.kyushu-u.ac.jp)

This work was supported by Japan Science and Technology Agency (JST), PRESTO, Japan, under Grant JPMJPR21P7.

ABSTRACT Machine learning's exponential growth has transformed healthcare, with Federated Learning (FL) playing a pivotal role. Despite its significance, FL is vulnerable to privacy attacks. In response, researchers have integrated differential privacy (DP) into FL. Nevertheless, incorporating DP introduces challenges such as increased total communication costs and computational overheads due to the introduction of noise. This drawback renders FL with DP less viable for healthcare systems, characterized by numerous low-resource devices and network bandwidth constraints. To overcome this limitation, we propose integrating a Discrete Fourier Transform (DFT) aggregator post-noise addition to transform the gradient generated by local training before sending it to the central server. This process reduces the gradient size and provides rudimentary encryption. The evaluation results reveal the superior performance of our proposed method, demonstrating an enhanced accuracy ranging from 0.2% to 2% compared to existing differential privacy techniques, including RDP, DP-SGD, ZcDP, LDP-Fed, and DP-AdapClip. Our approach substantially reduces the total communication costs (ranging from 6% to 43% across different privacy budgets) with faster training times in healthcare datasets such as the PIMA Indian database and Breast Cancer Histopathology Images.

INDEX TERMS Federated learning, differential privacy, adaptive Gaussian clipping, healthcare.

I. INTRODUCTION

The field of machine learning (ML) has undergone a remarkable surge in growth in the past decade, resulting in substantial progress across a multitude of technologies and applications. Particularly in healthcare systems, ML has played a pivotal role in transforming the landscape of patient care and medical research. Within the realm of ML, one noteworthy approach is Federated Learning (FL), which has emerged as a significant paradigm. FL empowers the training of machine learning models across numerous distributed devices without centralizing sensitive data. This innovation is particularly crucial in healthcare, where the privacy and security of medical data are paramount. In the FL framework, a collaborative network of clinicians develops their own local learning model using their respective medical data. These individual models generate incremental updates before being subsequently transmitted to a central coordinating node. This decentralized approach ensures that each clinician retains control over their data governance and privacy requirements, aligning with the stringent regulations governing healthcare information [1]. While FL advocates for data privacy, it remains susceptible to attacks that can diminish or destabilize model accuracy. A prevalent attack

The associate editor coordinating the review of this manuscript and approving it for publication was Ganesh Naik.

is known as a ''poisoning attack'' to distort the model by introducing corrupted training data. This action renders the server incapable of distinguishing data sources from authorized clients [2]. Furthermore, other types of attacks can be executed within the FL framework. The ''model reconstruction attack'' [3] involves monitoring and recording all communication between clients and the server, which is then used to reconstruct the client's local model. The ''GAN-based inference attack'' identifies user inputs on the server and selectively sends global updates to an isolated client [4]. Another attack, the ''inferring membership attack,'' seeks to exploit gradients from the SGD algorithm to access information related to the training process [5]. Given the multitude and diversity of these attacks, FL alone may not be sufficient to safeguard the data used in model training under certain circumstances. In these cases, many researchers have applied the principle of differential privacy to improve privacy measures in Federated Learning. This involves introducing random noise into the model gradient to modify the original value before transmitting it to a central server for aggregation.

Simply adding noise to a model without optimization inevitably reduces its accuracy. The Adaptive Gaussian Clipping Differential Privacy (AGC-DP) method, discussed in our previous research [6], offers a solution to improve the trade-off between privacy and model effectiveness. By incorporating Privacy Loss Distribution (PLD), dynamically adjusting privacy parameters based on data complexity, and optimizing data sampling through Poisson sub-sampling, AGC-DP outperforms other DP algorithms in terms of model accuracy. However, we noted that AGC-DP incurs elevated total communication costs, which often lead to higher latency. This latency can be detrimental in critical healthcare scenarios where timely data processing is essential.

Lowering communication costs is particularly beneficial for healthcare facilities, as it enables real-time synchronization of patient records across different locations. This ensures that healthcare providers have the most current information for better decision-making. Therefore, in this paper, we extend our previous work on AGC-DP by introducing a Discrete Fourier Transform (DFT) to encrypt and reduce the size of the gradient generated from local training before adding noise to it. This compressed noisy gradient is then sent to a central server for aggregation, reducing the gradient size sent to the central server and thereby lowering total communication costs. We also evaluate the system using healthcare datasets, such as the PIMA Indian diabetes dataset and Breast Cancer Histopathology CT Images, which contain sensitive medical information. This evaluation aligns with the system's purpose of safeguarding such sensitive data. Our key contributions can be summarized as follows:

1) **Enhanced Gradient Compression**: We introduced a Discrete Fourier Transform (DFT) aggregator to reduce the gradient size and implement basic encryption on AGC-DP. We applied DFT after local training and then added noise to the compressed gradient before sending it to the central server. This approach significantly decreases the total communication cost while enhancing security by protecting the gradient before transmission to the central server.
2) **Comprehensive Evaluation**: We conducted in-depth simulations of Enhanced AGC-DP using two distinct healthcare datasets. Through rigorous comparison with other DP methodologies, we assessed accuracy, total communication cost, and CPU time to demonstrate the effectiveness of our enhancements.

The structure of this article is outlined as follows. Section II introduces the related work on implementing secure FL in healthcare. In Section III, we describe the features related to our proposed method, including a system overview and threat model. Section IV presents our proposed method, including the description of the detailed process sequentially. Section V provides a detailed description of the datasets used for the learning process. Section VI describes the model detail used, including layers and activation type. Section VII explains the simulation parameters, specifically the environment and simulation scenario. Section VIII presents the simulation results and discussion. In Section IX, conclusions and future work are presented.

II. RELATED WORK

A. PRIVACY-PRESERVING FEDERATED LEARNING

In FL, client devices refrain from sending raw data to the central server, opting instead to exchange only the model's gradients. This protocol protects the raw data, ensuring it remains exclusively stored on local devices, thus bolstering the security and privacy of FL. However, despite clients transmitting only model gradients to the central server, research, such as [7], has demonstrated that these gradients can still be utilized to reconstruct the local model and deduce information about the training dataset. To tackle this issue, various strategies have emerged.

One approach, Homomorphic Encryption (HE) [8], utilizes the ElGamal algorithm to encrypt local gradients before sending them to the central server. However, the application of HE may face problems, especially with the complex models often encountered in deep learning, as encrypting many layers used in such models can introduce significant computational overhead. Conversely, another strategy employs blockchain technology [9], leveraging blockchain and secure global aggregation methodologies to safeguard against potential attacks from malicious edge devices and servers. Nonetheless, storing data continuously on the blockchain could lead to increased storage demands, while integrating complex blockchain technology could increase computational overhead.

Differential Privacy (DP) offers another solution by injecting random noise into the released information to distort the original values. In the realm of FL, several studies have explored DP implementation methods, including [10], which

utilizes Renyi differential privacy to prevent privacy leaks from the FL-MAC as discussed in the paper. DP-SGD [11], which adds noise based on the proportion of the clipping norm to the gradient updates, and ZcDP [12], which employs zero-concentrated differential privacy (ZcDP) to achieve a tighter privacy guarantee, thereby reducing the amount of noise added to the model compared to other methods with the same DP assurance.

There are other approaches, such as LDP−Fed [13], which use local differential privacy (LDP) based on the local device privacy budget and using utility-aware privacy perturbation to prevent uncontrolled noise from overwhelming the FL training algorithm in the presence of large, complex model parameter updates. The evaluation of the work shows that it achieved improved accuracy but did not show how efficient the method was in terms of communication and computational overhead, and DP-AdapClip [14] which is an approach that simplifies the training of neural networks in federated learning through user-level differential privacy. Instead of using a fixed clipping norm for each user's model update, this work proposes setting it dynamically based on a quantile of the update norm distribution, adapting to different tasks. This eliminates the need to fine-tune the fixed clipping hyperparameters, making the process much more efficient.

Another approach is presented in [15], which is also an extension of [6], as well as our previous work. This approach utilizes DCT-pruned weights to compress gradients using a dynamic compression ratio. The compression ratio is determined based on the device's resource availability, including processor and memory capacity. The primary distinction between our proposed method and [15] lies in how to manipulate the model weights. In the DCT-pruned weight approach, insignificant coefficients produced from the Discrete Cosine Transform (DCT) of the model weights are first pruned or removed and then masked before adding noise. This process involves identifying and discarding coefficients that contribute minimally to the representation of the model, thus reducing the overall size of the weight matrix. On the other hand, in this work, we employ Discrete Fourier Transform (DFT) and rotation techniques to manipulate the model weights. Specifically, we utilize the DFT to transform the model weights into the frequency domain, where rotations are applied before noise is added to enhance the privacy and security of the gradients. Another difference is that we implement [15] in a cloud environment and resource-constrained device (edge device) like Raspberry Pi. So, we specifically designed the method to fit the lower resource in an edge device, different from the work presented in this article, which focuses on healthcare applications that may use high image resolution, which is not really the kind of data used in edge devices.

B. PRIVACY-PRESEVING FEDERATED LEARNING IN HEALTHCARE

The implementation of privacy-preserving federated learning in the healthcare system is commonly utilized, given the sensitive nature of healthcare data, which includes confidential information such as patient records and medical diagnoses. Data protection represents one of the main challenges in the healthcare system. Several studies have implemented privacy-preserving federated learning, such as the work by [16], which employs traditional homomorphic encryption to encrypt model updates from local IoT devices. This data is then collected in a fog node and sent to the central server for aggregation and decryption. However, the use of encryption, as demonstrated in works like [17] and [18], incurs a high computational overhead. Particularly when using longer encryption keys, even if encryption is only performed once during the learning process.

Another work is [19], which combines differential privacy to ensure the privacy of the data and blockchain to record all user activity in the system to ensure transparency. Even though the evaluation results show that the method has better resource consumption (memory and CPU), it is unclear how much total communication cost is generated from this method. Even though network latency is mentioned, it solely depends on the network speed and environment. There is also [20], which combines differential privacy, Secure Multi-Party Computation (SMPC), and Homomorphic Encryption (HE) for model updates to provide maximum privacy, mitigating potential data leakage risks. However, the combination of these features can have a very high impact on the computational overhead and total communication costs. Even though the method is 10% more efficient in terms of computational overhead than the state-of-the-art method, it is not clear how much of the total communication costs is affected by this method, which is one of the weaknesses of differential privacy in this context. The summary of related work and the feature comparison of our proposed method can be seen in Table 1.

TABLE 1. Feature comparison between related work and proposed method.

Method	HA	Low CO	Low TCC	Secure
Homorphic Encryption [8]	✓	✗	✗	✓
Blockchain [8]	✓	✓	✗	✓
RDP [10]	✓	✗	✗	✓
DP-SGD [11]	✓	✗	✗	✓
ZcDP [12]	✓	✗	✗	✓
LDP-Fed [13]	✓	✗	✗	✓
DP-AdapClip [14]	✓	✗	✗	✓
HE-Med [16]	✓	✗	✗	✓
PPFLHE [17]	✓	✗	✗	✓
HE-Covid19 [18]	✓	✗	✗	✓
Block-DP [19]	✓	✓	✗	✓
DP-HE-SMPC [20]	✓	✓	✗	✓
Proposed Method	✓	✓	✓	✓

* **HA** : High Accuracy, **TCC** : Total Communication Cost, **CO** :Computational Overhead

III. PREELIMINARIES

In this section, we describe the general features that constitute our proposed method. We begin by discussing the features of

our previous work (AGC-DP) and the infrastructure of our system, including an overview of the system and the threat model.

A. ADAPTIVE GAUSSIAN CLIPPING DP(AGC-DP)

AGC-DP is a strategy designed to enhance the security of Federated Learning by incorporating varying levels of noise into the model's gradient by perturbing its original values. The variable noise levels in AGC-DP are created through the following components:

1) PRIVACY LOSS DISTRIBUTION (PLD)

In AGC-DP, the Privacy Loss Distribution (PLD) establishes a stringent privacy guarantee. The PLD measures the logarithmic ratio between the likelihood of observing an outcome given different inputs. For two discrete distributions, μ_{up} and μ_{lo}, the privacy loss at an outcome $o \in \sup(\mu_{up})$ (meaning o is within the support of μ_{up}, the set of all possible values with non-zero probability) is defined as follows:

Definition 1: The PLD of involving μ_{up} and μ_{lo}, referred to as PLD μ_{up}/μ_{lo}, represents a distribution on $RU\infty$, where $y \sim PLD_{\mu_{up}/\mu_{lo}}$ is obtained by sampling $o \sim \mu_{up}$ and setting $y = \mathcal{L}_{\mu_{up}/\mu_{lo}}(o)$. Mathematically, this is expressed as:

$$\mathcal{L}_{\mu_{up}/\mu_{lo}}(o) := \ln\left(\frac{\mu_{up}(o)}{\mu_{lo}(o)}\right) \tag{1}$$

AGC-DP utilizes the Gaussian mechanism to generate noise. This mechanism is characterized by a Gaussian (or normal) random variable with a mean (μ) and a standard deviation (σ). The likelihood of a Gaussian random variable assuming a specific value is described by its probability density function (PDF). The PDF for a Gaussian random variable at a point x is given by: $x = \frac{1}{\sigma\sqrt{2\pi}} \times e^{\frac{(x-\mu)^2}{2\sigma^2}}$.

Where μ denotes the mean of the distribution, indicating its central tendency, and σ represents the standard deviation, which measures the spread or dispersion of the distribution. The function exp denotes the exponential function. In the context of the Gaussian mechanism used in differential privacy (DP), noise is added to the output of a function to ensure the privacy of individuals in the dataset is preserved. The amount of noise is typically scaled according to the sensitivity of the function. To introduce the concept of scaled sensitivity, a new variable, $\tilde{\Delta}$, is defined as $\tilde{\Delta} = \delta(f)/\sigma$. Here, $\Delta(f)$ represents the sensitivity of the function f, indicating the maximum change in f due to the modification of a single data point. σ represents the standard deviation of the Gaussian noise being added. The noise generated by the Gaussian mechanism follows a normal distribution, often standardized as $N(0, 1)$, signifying it has a mean of 0 and a standard deviation of 1. This standardization is critical as any Gaussian distribution can be transformed into the standard normal distribution through normalization. Mathematically, this is represented as:

$$\ln\left(\frac{\frac{1}{\sigma\sqrt{2\pi}} \times e^{-x^2/2}}{\frac{1}{\sigma\sqrt{2\pi}} \times e^{-(x-\tilde{\Delta})^2/2}}\right) = \frac{\tilde{\Delta}}{2}.(\tilde{\Delta} - 2(x)) \tag{2}$$

Here, the natural logarithm of a ratio compares the probability density function (PDF) of a standard normal distribution $N(0, 1)$ at point x with and without an additional noise term ($\tilde{\Delta}$). ($\tilde{\Delta} - 2(x)$) represents half of the scaled sensitivity ($\tilde{\Delta}$), indicating how much the function's sensitivity contributes to the noise level. ($\tilde{\Delta} - 2(x)$) represents the difference between the scaled sensitivity ($\tilde{\Delta}$) and twice the observed value ($2(x)$). This difference captures the discrepancy between the scaled sensitivity and the observed value and their influence on the distribution. By utilizing the above formula, AGC-DP can analyze the impact of noise addition on the Privacy Loss Distribution, which is crucial for determining the amount of noise to be added to the gradient.

2) SUB-SAMPLING

AGC-DP also incorporates a subsampling method to enhance privacy protection, employing the Poisson sub-sampled technique with a sampling probability of q. This method randomly selects individual data points to be included in a sub-sampled dataset independently, each chosen with a probability of q. The mechanism's output is subsequently generated based on this sub-sampled dataset. In this process, the Privacy Loss Distribution (PLD) is computed for each mechanism (including noise addition through Gaussian distribution and data subsampling) employed at AGC-DP, and their convolutions are analyzed alongside an evaluation of the divergence of the ϵ-hockey stick. This divergence metric indicates how closely the mechanism aligns with the desired privacy level, measuring the discrepancy between the actual and ideal privacy loss distributions in DP.

In the context of DP, mechanisms often operate within datasets where data points can be added or removed, such as during data collection or when individuals request data deletion. The addition of a data point to the dataset can potentially increase privacy loss by introducing additional sensitive information, while the removal of a data point may also impact privacy loss by altering the dataset's composition. Taking into account both addition $\mathcal{L}_{\mu/\nu'}$ and removal $\mathcal{L}_{\mu'/\nu}$ adjacency, the privacy loss functions for the Poisson sub-sampled mechanism are represented as:

$$\mathcal{L}_{\mu/\nu'} = \log(1 - q + q \cdot e^{-\mathcal{L}_{\mu/\nu(o)}}) \tag{3}$$

$$\mathcal{L}_{\mu'/\nu} = -\log(1 - q + q \cdot e^{\mathcal{L}_{\mu/\nu(o)}}) \tag{4}$$

3) ADAPTIVE CLIPPING

AGC-DP integrates adaptive clipping as part of its operations. In this regard, AGC-DP employs the initial clipping threshold, referred to as C^0, based on the target unclipped quantile UC. This threshold establishes the limit beyond which noisy data points are clipped to safeguard privacy

while upholding data utility. During the training phase, UC gradually decreases over time. This decay in UC affects C by influencing its value. Specifically, as UC decreases, indicating a lower tolerance for unclipped data points, C is adjusted accordingly to maintain the balance between privacy protection and model utility. This process can be defined as:

$$C_j = UC_{t-1} \cdot (1 - r)^t \tag{5}$$

(C_j) represents a clipping threshold, where $(1 - r)$ represents the decay factor determining how quickly the threshold decreases over time t. If the threshold is initially high, meaning r is small, then less noise is added, leading to a slower decrease in the threshold. Conversely, if the threshold is initially low, meaning r is large, more noise is added, leading to a faster decrease in the threshold. The decay of the target unclipped quantile every 20 iterations corresponds to the term UC_{t-1}, where the $t - 1$ indicates that the value of the unclipped quantile decays over time.

B. SYSTEM OVERVIEW

For the system's architecture, we consider a general Federated Learning (FL) system, illustrated in Fig.1, comprising a server and K clients. Each client is denoted by k_i, where i ranges from 1 to K, and each client possesses a local database $\mathcal{D}_{ki}$. The server aims to iteratively train a model using data from K-associated clients over T steps.

In our proposed system, the process starts with the central server's program setting up the global model as w_0. The next step involves determining the number of participating clients and establishing the noise level (S) to be injected into the gradient. Each client's program then updates its local model (w_t) and preprocesses the dataset for training purposes. Once ready, the client's program initiates the training process to generate its local model. Following this, the system applies the Discrete Fourier Transform (DFT) aggregator to the model gradient, converting it into a 1D rank tensor and rotating it to increase randomness. This produces c_t^k, which is subsequently added with noise determined by the previously calculated value of S. Following this, the client's program transmits the gradients with added noise, represented as $\hat{c}_t^k$, to the server. The server's program aggregates these gradients, which are subsequently subjected to an inverse Discrete Fourier Transform (DFT) to create an updated parameter for adjusting the global model. After completing the update process, the program shares the global model that has been updated with the clients, initiating the subsequent iteration round, which continues until $t = T$.

C. THREAT MODEL

For the threat model, we assume that the central server operates honestly. It accurately aggregates all uploaded gradients of the local model, ensuring no intentional omissions. Furthermore, clients show no interest in the private data of other clients and refrain from seeking further insights from the collaborative models. Our threat model can be defined as follows:

1) **Privacy leakage**. Local clients and central servers cannot conduct inference attacks directed towards individual clients for exchanging gradients in order to reconstruct their local model or training datasets.
2) **External adversaries**. External adversaries cannot manipulate data within the local client to introduce false information that could disrupt the model's learning process. However, they can eavesdrop on the transmitted data or gradients sent from the client to the central server, potentially enabling them to reconstruct training datasets.

IV. PROPOSED METHOD

In this section, we explain the steps of our proposed method in detail. As shown in Fig.1, our proposed method consists of nine steps: 1) Initialization, 2) Preparation for local training, 3) Privacy loss calculation, 4) Local training, 5) Implementation of the Discrete Fourier Transform (DFT) aggregator, 6) Noise addition, 7) Aggregation, 8) Inverse DFT aggregator, and 9) Global model update. In this section, we provide a detailed explanation of each process.

A. INITIALIZATION

In this phase, the server program will set up a global model, referred to as w_0. The central server and all clients collaborate to determine an initial weight range based on prior knowledge, which helps improve the model's convergence during training. Using this range, the server initializes w_0. The central server then randomly distributes w_0 to the participating clients for local training. This random distribution is crucial to avoid bias or favoritism towards specific clients by the central server during the training process.

B. PREPARATION FOR LOCAL TRAINING

During this phase, we set up all the necessary parameters for the training process. The first parameter defined was the base noise multiplier (bnm), which influences the amount of noise added to the local model on each client. Additionally, we established the number of base clients per round ($bcpr$). Furthermore, we determined the values for target epsilon ϵ_t and target delta δ_t. These values have a significant impact on the privacy parameters and play a role in controlling the level of privacy protection in the subsequent stages of the process.

C. PRIVACY LOSS CALCULATION

Like [15], we also used the Privacy Loss Distribution (PLD) defined in AGC-DP [6] to compute the privacy loss (ϵ), as outlined in Equations (3) and (4). The PLD generates PL(ϵ), clients per round (cpr), and noise multiplier (nm), as shown in lines 1 to 5 of Algorithm 1. The computed ϵ from PLD is then compared to the predefined target privacy (ϵ_t). This target privacy determines the desired level of privacy protection for the system (lower values indicate higher privacy, while higher values imply lower privacy). The values of cpr and nm will be adjusted based on ϵ until it

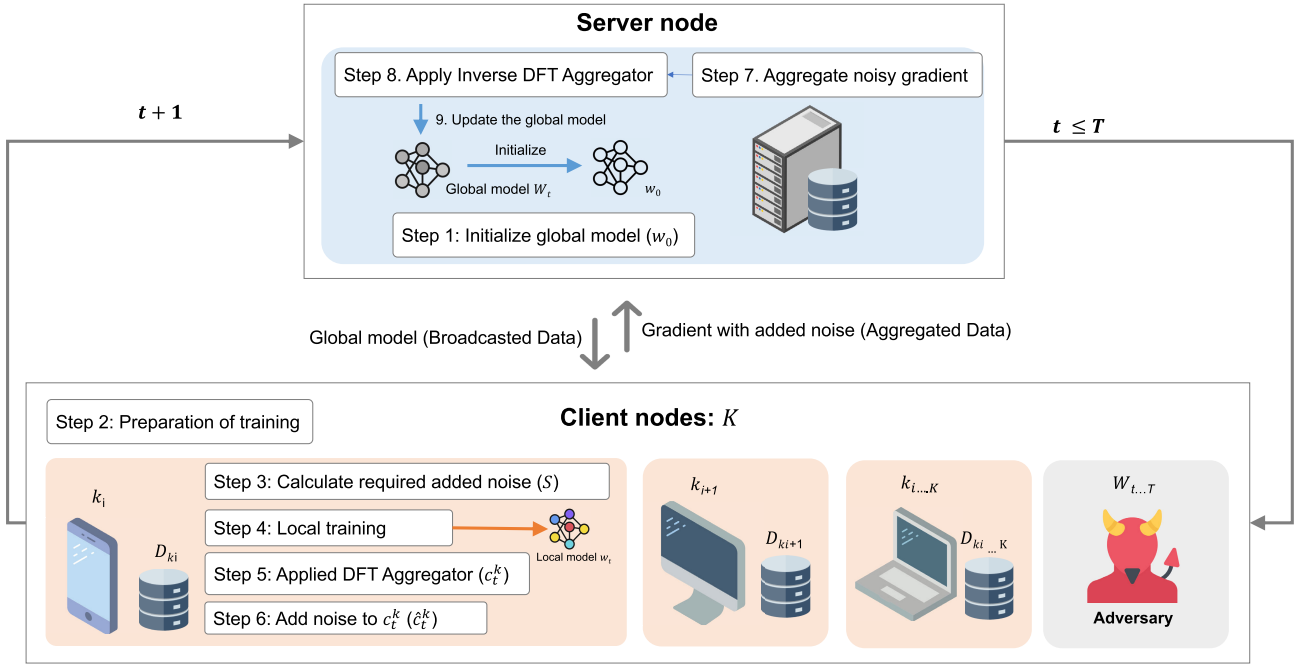


FIGURE 1. Overview of the proposed system.

Algorithm 1 Privacy Loss Calculation

Data: ϵ_t, δ_t, $bcpr = 50$, $bnm = 0.5$
Result: nm
1 **Function** `GetEps(`cpr`)`**:**
2 $nm \leftarrow bnm$
3 $nm\times = cpr/bcpr$
  `// Calculate privacy loss`
4 $\epsilon = PLD_{accnt}(nm)$
5 **return** cpr, ϵ, nm
6 **Function** `FindClientNeeded()`**:**
7 $ep \leftarrow$ `GetEps(`$bcpr$`)`
8 **if** $ep[1] < \epsilon_t$ **then**
9 **return** ep;
10 **end**
11 **while** True **do**
12 $ls \leftarrow ep$
13 $ep \leftarrow GetEps(2 * ls[0])$
14 **end**
15 **return** ep

matches ϵ_t. Once achieved, the appropriate cpr and nm for ϵ_t are determined as a reference in lines 7 to 14 of Algorithm 1. These parameters are then utilized in local training, where cpr dictates the number of client participation in one iteration, and nm governs the amount of random noise added to the gradient after local training.

D. LOCAL TRAINING

In the initial stage of local training, the clipping process was initiated. During this phase, the target unclipped quantile gradually diminished, as specified in lines 2 of Algorithm 3. Alterations to the target unclipped quantile have a direct impact on the threshold value employed for gradient clipping. This threshold, in turn, indirectly affects the introduction of noise into the model gradient at regular intervals, typically every 20 iterations. Subsequently, the local clients commenced their training process, utilizing the parameters generated in stages 1-3. Each client conducted local model training, denoted as w_{kt}, for a single local epoch to ensure comprehensive local training. During each local epoch, the client meticulously processed their entire local dataset. To streamline and expedite the local training procedure while avoiding the substantial time delays associated with multiple local epochs, the client divided the local dataset into smaller batches. Following this segmentation, the client applied training to one randomly selected mini-batch from the collection of batches, one at a time.

E. APPLY DFT AGGREGATOR

In this phase, the gradient from local training was applied to the randomized discrete Fourier transform or DFT. The details of this process are described below.

1) **Flattens the gradients**: This process converts a multi-dimensional gradient w_t^k into a one-dimensional vector. This operation essentially "unwraps" the gradient, transforming it into a sequence of elements in a specific order, as seen in Algorithm 2 line 2-3.
2) **Pads the flattened gradients**: This process involves adding zero values to the flattened gradient flt so then the resulting gradient has an even number of elements.

Algorithm 2 DFT Aggregator

Data: w_t^k
Result: c_t^k
1 **Function** `ClientTrans` (w_t^k) **:**
  Data: $flt, padW, cw_t^k, rw_t^k, dw_t^k, uw_t^k$
  Result: c_t^k
2 **for** *each element x in* w_t^k **do**
3 $flt \leftarrow \text{Reshape}(x, [-1])$
4 **end**
5 **foreach** *element y in flt* **do**
6 **if** $size(y)\%2 = 0$ **then**
7 $num\_zeros \leftarrow 0$
8 **end**
9 **else**
10 $num\_zeros \leftarrow 1$
11 **end**
12 $padW \leftarrow \text{Concat}([y, num\_zeros])$
13 **end**
14 $cw_t^k \leftarrow \text{Complex}(rl = padW[0], im = padW[1])$
15 $rw_t^k \leftarrow \text{Rotate}(cw_t^k)$
16 $dw_t^k \leftarrow \text{FFT}(rw_t^k)$
17 $uw_t^k \leftarrow \text{Concat}(real(dw_t^k), imag(dw_t^k))$
18 **foreach** *element j in* uw_t^k **do**
19 $c_t^k \leftarrow \sqrt{\frac{\text{size}(j)}{2}}$
20 **end**
21 **return** c_t^k;

3) **Packs the gradient**: This operation transforms the gradient padded with zero $padW$ into a complex-valued gradient by creating complex numbers. The resulting complex gradient cw_t^k will have half as many elements as the original real gradient. The transformation is achieved by pairing the real rl and imaginary im values from the original gradient to form complex numbers collectively. The complex tensor will have half as many elements as the original gradient, indicated by $d/2$. In other words, if the original gradient had d elements, the resulting complex gradient would have $d/2$ complex numbers. This reduces the size of the gradient. This process can be seen in Algorithm 2 line 14.
4) **Rotating complex gradient coordinates**: This process applies a random phase shift to the individual complex numbers within the gradients cw_t^k as defined in Algorithm 2 line 15. Each complex number consists of a real part and an imaginary part, and an angle effectively rotates these two components randomly.
5) **Applies the discrete Fourier transform**: This process applies DFT to the cw_t^k to convert it from the spatial domain into the frequency domain as seen in Algorithm 2 line 16. In this context, the DFT is applied to the complex tensor. It reveals the frequency components present in the data after the previous transformations.
6) **Unpacks the complex gradient**: Unpacks the complex gradient: This step reverses the process of step 3. The complex gradient dw_t^k is converted back into a real gradient by separating the real and imaginary components of the complex tensor and concatenating them back into a real tensor with the length of d. This process can be seen in Algorithm 2 line 19.
7) **Normalizes the gradient**: This process divides the un- pack gradients uw_t^k as defined in Algorithm 2 line 18-20. This process ensures that the gradient remains within a desired range or that the magnitude of the gradient is consistent for the inverse process on the server. This process will generate the transform gradient c_t^k that will be added noise on the next process.

F. NOISE ADDITION

By incorporating adaptive Gaussian clipping, as detailed in [6], we determined a clipping threshold C by referencing a target unclipped quantile denoted as UC. This process involved gradually reducing UC every 20 iterations. Adjusting UC directly impacted the clipping threshold value, consequently affecting the amount of noise introduced to the gradient (S) as illustrated in Algorithm 3 in line 2.

Algorithm 3 Noise Applied to the Gradient Locally

Data: c_t^k
Result: $\hat{c}_t^k$
1 – *Decrease the desired unclipped quantile*;
2 $C_j = UC_{t-1} \cdot (1 - r)^t$;
3 – *Injecting noise into the models gradient*;
4 $\hat{c}_t^k = c_t^k / max(1, \frac{||c_t^k||2}{S})$;
5 –*Send the noisy gradient to the central server.*;

Subsequently, we introduced noise to the compressed gradient denoted as c_t^k, using the computed noise factor S. To implement the noise mechanism, we utilized a Gaussian mechanism. This procedure yielded a compressed gradient with the incorporated noise, represented as $\hat{c}_t^k$, as illustrated in Algorithm 3 in line 4. Following this, the gradient was transmitted to the server for further steps, including aggregation, inversion, and the updating of the global model.

G. AGGREGATION

During this phase, the server received the compressed noisy gradient $\hat{c}_t^k$. Subsequently, the server conducted gradient aggregation, following the procedure outlined in lines 14-16 of Algorithm 4. The aggregation process produces the updated parameter, denoted as w_t, representing the parameter values at a specific iteration or time step during the optimization process. These updated parameters are obtained by aggregating $\hat{c}_t^k$ from participating clients. The parameter w_t will be employed for the inverse Discrete Fourier Transform (DFT) aggregator.

H. INVERSE DFT AGGREGATOR

This process was the inverse of the DFT aggregator; the step aimed to restore the original gradient transformed in the DFT aggregator process. The detailed steps of this process can be described as follows:

Algorithm 4 Inverse DFT and Model Update

Data: w_t
Result: o_w
1 **Function** `ServerTrans` (w_t) **:**
  Data: o_c, o_i, o_{ir}
  Result: o_w
2 **foreach** *element* o *in* w_t **do**
3 $o_s \leftarrow o * \sqrt{\frac{2}{d}}$
4 **end**
5 **foreach** *element* g *in* o_s **do**
6 $o_r \leftarrow$ Reshape$(g, [-1])$
7 **end**
8 $o_c \leftarrow$ Complex$(real = o_r[0], imag = o_r[1])$
9 $o_i \leftarrow$ IFFT(o_c)
10 $o_{ir} \leftarrow$ InverseR(o_i)
11 $o_w \leftarrow$ Concat$(real(o_{ir}), imag(o_{ir}))$ **return** o_w;
12
13 – Server Aggregation
14 **for** $k \in k_1, k_2, \ldots, k_n, K$ **do**
15 $w_t \leftarrow \sum_{t=1}^{T} \frac{n_k}{n} c_t^k$;
16 **end**
17 – Inverse transformation
18 $o_w \leftarrow$ `ServerTrans` (w_t) ;
19 – Global model update
20 $W_t \leftarrow W_{t-1} - n\nabla g(o_w)$;

1) **Scaling**: This step starts by reversing the scaling step, which involves multiplying each element in the gradient w_t by a scaling factor. In this case, it multiplies the tensor by $\frac{1}{\sqrt{\frac{d}{2}}}$ to normalize it. This process can be seen in the Algorithm 4 line 2-4.
2) **Reshaping**: This step reshapes the scaled gradient o_s into a two-row matrix with a flexible number of columns as seen in Algorithm 4 line 5-7. This process effectively reverses the flattening operation in the DFT aggregator process.
3) **Complex Packing**: this step repacks the real-valued gradient o_r into a complex data type, reversing the operation of unpacking it into real and imaginary components.
4) **Inverse Fourier Transform (IFFT)**: This applies the inverse Fourier transform (IFFT) to convert the complex gradient o_c from the frequency domain back to the time or spatial domain, effectively reversing the forward Fourier transform. This process can be seen in Algorithm 4 line 9.
5) **Random Inverse Rotation**: This process reverses the random rotations introduced in the DFT aggregator process in the client by applying the inverse rotation operation to o_i. This random inverse rotation has the same parameter as the previous one used in random rotation. To reverse the random rotations, it typically needs access to the same random rotation parameters used in the forward transformation. Without knowledge of these parameters, accurately inverting the rotations is challenging or impossible. Implementing random rotation ensures that sensitive information remains protected. This process can be seen in Algorithm 4 line 10.
6) **Real-Imaginary Concatenation**: This process concatenates the real and imaginary components into a complex gradient o_w as defined in Algorithm 4 line 11, effectively reversing the separation of real and imaginary parts.

I. UPDATE GLOBAL MODEL

This phase commences following the inverse operation, which calculates the reverse gradient of o_w. This inverted gradient is then utilized to update the global model W, as indicated in Algorithm 4 at line 20. This process generated an updated global model W_t. Subsequently, this updated global model W_t is distributed to all participating clients and will be used for the next iteration in the learning process. The detailed algorithm of our proposed method can be found in Algorithm 5.

Algorithm 5 Overall Algorithm

1 **Program** *SERVER*
2 - Initialization of global model w_0
3 **for** $t|t_1|t_2|t_n \ldots..T$ **do**
4 - Inverse Transform and Model Update
5 $W_t \leftarrow$ **Algorithm 4**;
6 **end**
7 **return** cpr, nm, w_0, W_t;
8 **end**
9 **Program** *CLIENT*
10 – Privacy Loss Calculation
11 $S \leftarrow$ **Algorithm 1**
12 $w_t^k \leftarrow$ Local Training
13 – Gradient Transformation with DFT Aggregator
14 $c_t^k \leftarrow$ **Algorithm 2**
15 – Adaptive Gaussian Clipping DP
16 $\hat{c}_t^k \leftarrow$ **Algorithm 3**
17 **return** $\hat{c}_t^k$
18 **end**

V. DATASETS

We tested our proposed framework on two datasets: the PIMA Indian Diabetes dataset and Breast Cancer Histopathology images. These two datasets are of different sizes and have distinct characteristics, which are described below:

1) **PIMA Indian Diabetes Dataset**[1]. The Pima Indian Diabetes Dataset, originally obtained from the National Institute of Diabetes and Digestive and Kidney Diseases, provides information about 768 women living near Phoenix, Arizona, USA. This dataset includes data about eight factors believed to influence diabetes, along with the corresponding classifications. It is organized into 9 columns and 768 rows, with 500 entries representing non-diabetic individuals and 268 representing diabetic individuals. The classification outcome variable is binary, using 0 (indicating a negative diabetes test) and 1 (indicating a positive diabetes test).
2) **Breast Cancer Histopathology (BCH)**[2]. The dataset comprises 162 whole-mount slide images of Breast Cancer (BCa) specimens scanned at 40x magnification. From these images, a total of 277,524 patches measuring 50 × 50 were extracted, with 198,738 being IDC negative and 78,786 being IDC positive.

VI. MODELS

For each client, we used a local convolutional neural network model. We also applied a convolutional neural network for the Global model. Different models are used in the PIMA Indian Diabetes Dataset and Breast Cancer Histopathology. Below is the description of the model used in this paper.

1) **Model for PIMA Indian Diabetes Dataset**: The model used for this type of dataset was a feedforward neural network comprised of an input layer and two hidden layers with 6 and 3 neurons, respectively, all using sigmoid activation functions, as well as an output layer with a single neuron and sigmoid activation. The Glorot Normal weight initialization method was employed for all layers. Notably, the output layer incorporates L1 and L2 regularization with specific strengths ($l1 = 0.0001$ and $l2 = 0.01$).
2) **Model for Breast Cancer Histopathology Dataset**: The model for this dataset consisted of four convolutional layers with 32, 64, 128, and 128 filters of size 3×3, respectively, each followed by a rectified linear unit (ReLU) activation function and max-pooling with a 2×2 pool size. Dropout layers with a rate of 0.25 were inserted after each max-pooling layer. Subsequently, a flattening layer converted the 2D feature maps into a 1D vector, followed by a fully connected dense layer with 128 units and ReLU activation. Another dropout layer with a rate of 0.5 was added to reduce overfitting further, and the final output layer consisted of two neurons with sigmoid activation.

We used the above model for the different evaluation scenarios and implemented it for all the state-of-the-art methods used as comparators, including the proposed method.

[1]https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
[2]https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images

TABLE 2. Detail of simulation hyperparameter.

Scenario	Parameter	Value
1	Client learning rate	0.1
	Server learning rate	1
	Server momentum	0.9
	bcpr	20
	Total Client	538
	bnm	1
2	Client learning rate	0.005
	Server learning rate	0.5
	Server momentum	0.5
	bcpr	50
	Total Client	2500
	bnm	0.5

VII. SIMULATIONS

A. SIMULATION ENVIRONMENT

The simulation was conducted on a computer equipped with an 11th Generation Intel Core i9-11900K CPU, 128 GB of RAM, and an NVIDIA GTX 3070 graphics card with 8 GB of VRAM. The computer ran the Ubuntu 20.04 LTS operating system. The simulation platform included Python 3.8 and Tensorflow 2.8.3, with Tensorflow Federated 0.22.0 serving as the framework for Federated Learning (FL), and Tensorflow Privacy 0.8.0 being utilized as the privacy library.

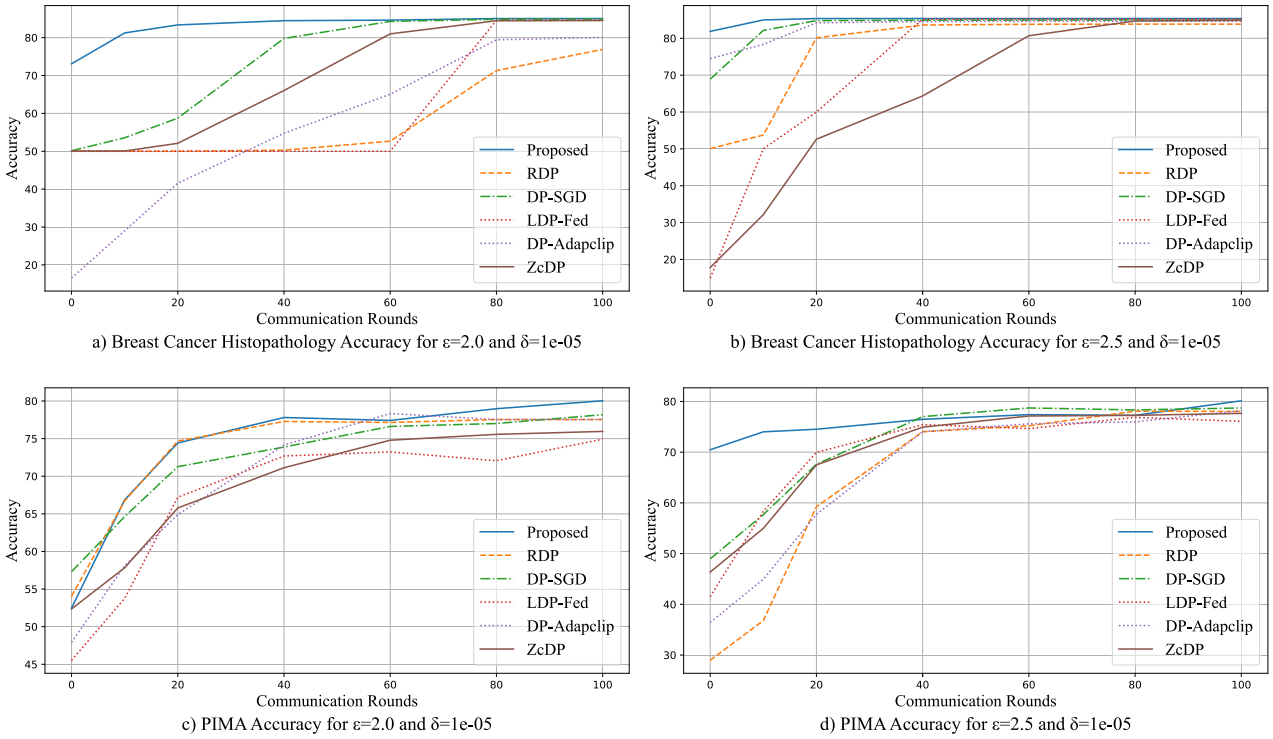
B. SIMULATION SCENARIOS

In our evaluation, we utilized a simulation to compare our proposed method with the other algorithms previously employed in Federated Learning (FL), including DP-SGD, RDP, ZcDP, LDP-Fed, and DP-AdapClip. We assessed the differences among the five algorithms and our proposed approach in terms of model accuracy, total communication costs, and computational complexity. Our simulation process encompassed two distinct scenarios to provide a comprehensive analysis.

- **Scenario 1**. We evaluate our proposed method and five other algorithms using different datasets. For the privacy budget, we set $\epsilon = 2.0$ and $\delta =$ 1e-05. For the dataset, we used the PIMA Indian Diabetes dataset and Breast Cancer Histopathology datasets. We iterated the scenario for 100 communication rounds.
- **Scenario 2**. We evaluate our proposed method and five other algorithms using the same datasets as scenario 1. The difference lay in the privacy budget used. In this scenario, we used $\epsilon = 2.5$ and $\delta =$ 1e-05. We also iterated this scenario using 100 communication rounds. The detailed simulation hyperparameters are listed in Table.2.

VIII. RESULTS AND DISCUSSION

In this section, we present and discuss the results obtained from the simulation process. In particular, we have engaged in an analysis of the results in terms of accuracy, total communication costs, and computational complexity.



a) Breast Cancer Histopathology Accuracy for ε=2.0 and δ=1e-05

b) Breast Cancer Histopathology Accuracy for ε=2.5 and δ=1e-05

c) PIMA Accuracy for ε=2.0 and δ=1e-05

d) PIMA Accuracy for ε=2.5 and δ=1e-05

FIGURE 2. Global model accuracy of proposed method compare to other DP algorithm.

TABLE 3. Detail of simulation results on model's accuracy.

No	Dataset	Eps	Delta	Proposed	RDP	ZcDP	LDP-Fed	DP-AdapClip	DP-SGD
1	PIMA Indian Diabetes	2	1e-05	**80.03**	77.54	75.96	74.93	77.54	78.18
		2.5	1e-05	**80.13**	78.03	77.65	76.08	78.2	78.72
2	Breast Cancer Histophatology	2	1e-05	**85.04**	76.89	84.52	84.56	80.07	84.9
		2.5	1e-05	**85.36**	83.84	84.88	85.2	84.64	84.96

A. MODEL ACCURACY

In Fig.2, we can see that our method achieves higher model accuracy compared to other methods in both scenarios 1 and 2. While the accuracy improvement in the Breast Cancer Histopathology datasets may seem modest, it is notable that our approach outperforms all state-of-the-art methods in the PIMA Indian Diabetes datasets. Furthermore, the findings indicate that our method achieves faster model convergence than all state-of-the-art methods in the Breast Cancer Histopathology datasets, regardless of whether the value of ϵ is higher or lower. Specifically, our method reaches convergence within 20 communication rounds. In contrast, the other state-of-the-art methods require 40 communication rounds to converge, except for DP-SGD at $\epsilon = 2.5$, which converges in the same time frame as our proposed method.

B. TOTAL COMMUNICATION COSTS

We also evaluated the total communication costs incurred by each method. To assess this, we introduced the Aggregated Data (AD) variable, which represents the tensors returned by each client to the server (upstream data) after local training. The size of this parameter was measured in Gigabytes (GB) for the Breast Cancer Histopathology dataset and Kilobytes (KB) for the PIMA Indian Diabetes dataset. Fig.4 shows that our proposed method results in significantly lower total communication costs compared to other state-of-the-art methods in both scenarios for the Breast Cancer Histopathology and PIMA Indian Diabetes datasets. Additionally, for the PIMA Indian Diabetes datasets, our method's total communication costs do not increase substantially beyond 80 communication rounds. In contrast, methods like RDP, DP-SGD, and ZcDP consistently show increasing communication costs with each round. Although LDP-Fed and DP-AdapClip initially have lower total communication costs than our method, they exhibit exponential increases beyond 80 rounds, eventually surpassing our proposed method.

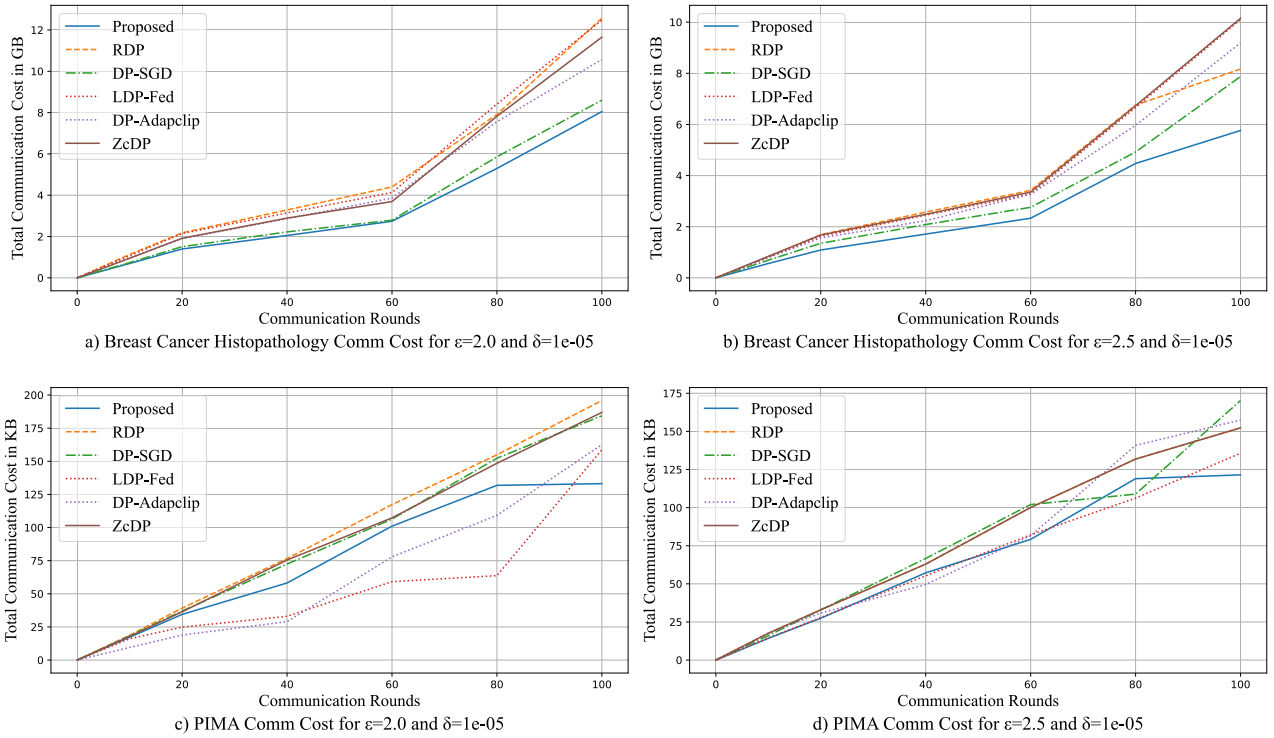
Based on Table 4, our proposed method has a communication efficiency advantage compared to another state-of-the-art method, even when evaluated using more extensive datasets and complex models in additional relation to different privacy parameters. We also observed an increase in total

TABLE 4. Detail of simulation results on total communication cost.

No	Dataset	Eps	Delta	Proposed	RDP	ZcDP	LDP-Fed	DP-AdapClip	DP-SGD
1	PIMA Indian Diabetes (In KB)	2	1e-05	**133.12**	195.84	186.88	158.72	162.56	184.32
		2.5	1e-05	**121.49**	152.32	152.32	135.68	157.44	170.24
2	Breast Cancer Histopathology (In GB)	2	1e-05	**8.05**	12.57	11.65	12.48	10.57	8.60
		2.5	1e-05	**5.76**	8.17	10.15	10.10	9.19	7.87

TABLE 5. Detail of simulation results on total training time in (Seconds).

No	Dataset	Eps	Delta	Proposed	RDP	ZcDP	LDP-Fed	DP-AdapClip	DP-SGD
1	PIMA Indian Diabetes	2	1e-05	**72**	452	475	310	509	468
		2.5	1e-05	**67**	364	305	240	413	420
2	Breast Cancer Histophatology	2	1e-05	**1971**	4683	4080	4045	3936	3803
		2.5	1e-05	**1889**	4353	3573	3515	3173	2739



a) Breast Cancer Histopathology Comm Cost for ε=2.0 and δ=1e-05

b) Breast Cancer Histopathology Comm Cost for ε=2.5 and δ=1e-05

c) PIMA Comm Cost for ε=2.0 and δ=1e-05

d) PIMA Comm Cost for ε=2.5 and δ=1e-05

FIGURE 3. Total communication cost of proposed method compared to other DP algorithms.

communication costs when there was a decrease in ϵ. This increase occurred because as ϵ decreases, more noise is added to the model gradient, resulting in a more biased model. To address this challenge, as noted in [6], each algorithm increases the number of clients participating in the learning process, which in turn raises total communication costs. Notably, DP-SGD had the lowest increase in total communication costs, indicating that it did not significantly increase the number of participating clients in the learning process in order to meet a lower privacy budget of ϵ and δ, unlike other methods.

C. COMPUTATIONAL COMPLEXITY

We measured computational complexity based on the observed training time for all methods, with lower training times indicating lower computational complexity and faster convergence. This has significant real-world implications for resource efficiency, responsiveness, and feasibility in privacy-preserving federated learning applications. Our proposed method demonstrated consistently lower training times than other state-of-the-art methods for both the Breast Cancer Histopathology and PIMA Indian Diabetes datasets in Scenarios 1 and 2, as shown in Fig.3. For the PIMA Diabetes dataset at $\epsilon = 2.0$, our method achieved substantial reductions in training time compared to other methods. Even at $\epsilon = 2.5$, our method maintained its efficiency advantage. Similarly, for the Breast Cancer Histopathology dataset, our method exhibited significant training time efficiency for both ϵ values. Our proposed method achieved this efficiency, even with additional time added, because of the DFT transform

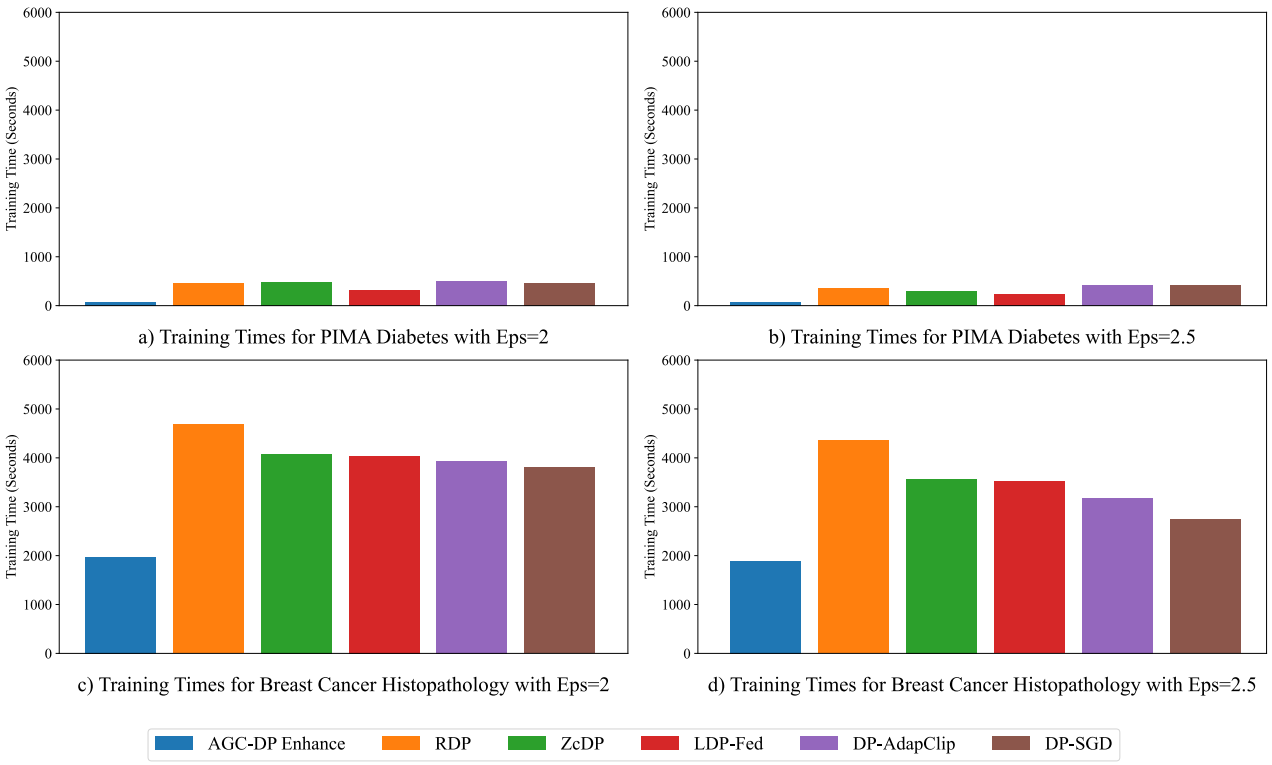


FIGURE 4. Total training time of proposed method compared to other DP algorithms.

aggregator in the client and inverse process in the server. The detailed results of the total training time can be seen in Table 5.

D. DISCUSSION

The simulation results show that accuracy decreases as the epsilon value decreases from 2.5 to 2.0 across both datasets due to increased noise making the model more biased. Despite this, our proposed method demonstrates remarkable resilience to stricter privacy constraints, exhibiting only minimal decreases in accuracy. In contrast, other methods show more pronounced accuracy decreases. For example, the RDP method shows a substantial drop in accuracy in both datasets. Similarly, ZcDP, LDP-Fed, and DP-AdapClip also experience significant decreases, indicating varying vulnerability to stricter privacy budgets. Our proposed method and DP-SGD exhibit minimal decreases in accuracy as epsilon decreases. While DP-SGD performs slightly better in one dataset, our method performs better in the other. Overall, DP-SGD demonstrates greater resilience to accuracy decreases with lower epsilon values in larger and more complex datasets, such as CT scan images, compared to all other methods, including our proposed method.

In our study, we carefully selected two privacy budgets, $\epsilon = 2.0$ and $\epsilon = 2.5$, with a fixed $\delta =$ 1e-05, as the foundation for our simulations. This deliberate choice allowed us to assess the performance of each algorithm under different privacy budgets, demonstrating the adaptability and flexibility of our proposed method across various settings and environments. Furthermore, we extended our evaluation to include practical healthcare datasets. This step was essential to showcase the real-world applicability of our proposed system, particularly within the healthcare sector. Given the prevalence of resource-constrained devices in healthcare applications, in terms of both computational power and communication capabilities, we recognized the need to evaluate the total communication costs generated by our method and previous approaches. Our evaluation results provide valuable insights into the communication burdens imposed by our proposed method and its impact on network bandwidth in real-world environments compared to prior methods. This insight is invaluable for understanding the practical implications and scalability of our approach within the broader context of privacy-preserving federated learning in healthcare.

IX. CONCLUSION AND FUTURE WORKS

This paper outlines a novel technique aimed at enhancing the privacy measures of Differential Privacy (DP) within the context of Federated Learning (FL) for healthcare. Our approach integrates a DFT Aggregator and leverages adaptive Gaussian clipping. To gauge the effectiveness of our proposed system, we conducted evaluations using two healthcare datasets: PIMA Indian Diabetes and Breast Cancer Histopathology, each operating under different privacy budget constraints.

Our simulation results demonstrate that our method surpasses the traditional DP algorithms typically employed in FL, achieving heightened accuracy. Notably, this enhanced accuracy is accomplished while reducing total communication costs and training time. Through our proposed method, we enhance patient confidentiality and data security, which are crucial in modern healthcare. Additionally, our method's increased efficiency can streamline processes, improving patient outcomes and resource allocation. In future work, we aim to optimize hyperparameters and refine our algorithm to enable implementation on real devices and for use with more complex and larger datasets.

REFERENCES

[1] Z. A. E. Houda, A. S. Hafid, L. Khoukhi, and B. Brik, "When collaborative federated learning meets blockchain to preserve privacy in healthcare," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2455–2465, Nov. 2023.

[2] R. Gosselin, L. Vieu, F. Loukil, and A. Benoit, "Privacy and security in federated learning: A survey," *Appl. Sci.*, vol. 12, no. 19, p. 9901, Oct. 2022.

[3] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*. San Francisco, CA, USA: IEEE, May 2019, pp. 691–706.

[4] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[5] O. Zari, C. Xu, and G. Neglia, "Efficient passive membership inference attack in federated learning," 2021, *arXiv:2111.00430*.

[6] M. A. Hidayat, Y. Nakamura, B. Dawton, and Y. Arakawa, "AGC-DP: Differential privacy with adaptive Gaussian clipping for federated learning," in *Proc. 24th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jul. 2023, pp. 199–208.

[7] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.

[8] L. Zhang, J. Xu, P. Vijayakumar, P. K. Sharma, and U. Ghosh, "Homomorphic encryption-based privacy-preserving federated learning in IoT-enabled healthcare system," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 1–17, 2022.

[9] Z. Yang, Y. Shi, Y. Zhou, Z. Wang, and K. Yang, "Trustworthy federated learning via blockchain," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 92–109, Jan. 2023.

[10] S. Wu, M. Yu, M. A. M. Ahmed, Y. Qian, and Y. Tao, "FL-MAC-RDP: Federated learning over multiple access channels with Rényi differential privacy," *Int. J. Theor. Phys.*, vol. 60, no. 7, pp. 2668–2682, Jul. 2021.

[11] M. Abadi, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Vienna Austria, 2016, pp. 308–318.

[12] R. Hu, Y. Guo, and Y. Gong, "Concentrated differentially private federated learning with performance analysis," *IEEE Open J. Comput. Soc.*, vol. 2, pp. 276–289, 2021.

[13] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-fed: Federated learning with local differential privacy," in *Proc. 3rd ACM Int. Workshop Edge Syst., Analytics Netw.* New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 61–66.

[14] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy, "Differentially private learning with adaptive clipping," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 17455–17466.

[15] M. A. Hidayat, Y. Nakamura, and Y. Arakawa, "Privacy-preserving federated learning with resource-adaptive compression for edge devices," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13180–13198, Apr. 2024.

[16] H. Ku, W. Susilo, Y. Zhang, W. Liu, and M. Zhang, "Privacy-preserving federated learning in medical diagnosis with homomorphic re-encryption," *Comput. Standards Interfaces*, vol. 80, Mar. 2022, Art. no. 103583.

[17] B. Wang, H. Li, Y. Guo, and J. Wang, "PPFLHE: A privacy-preserving federated learning scheme with homomorphic encryption for healthcare data," *Appl. Soft Comput.*, vol. 146, Oct. 2023, Art. no. 110677.

[18] F. Wibawa, F. O. Catak, M. Kuzlu, S. Sarp, and U. Cali, "Homomorphic encryption and federated learning based privacy-preserving CNN training: COVID-19 detection use-case," in *Proc. Eur. Interdiscipl. Cybersecurity Conf.* New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 85–90.

[19] L. Ngan Van, A. H. Tuan, D. Phan The, T.-K. Vo, and V.-H. Pham, "A privacy-preserving approach for building learning models in smart healthcare using blockchain and federated learning," in *Proc. 11th Int. Symp. Inf. Commun. Technol.* New York, NY, USA: Association for Computing Machinery, Dec. 2022, pp. 435–441.

[20] M. Abaoud, M. A. Almuqrin, and M. F. Khan, "Advancing federated learning through novel mechanism for privacy preservation in healthcare applications," *IEEE Access*, vol. 11, pp. 83562–83579, 2023.

MUHAMMAD AYAT HIDAYAT (Graduate Student Member, IEEE) was born in 1991. He received the bachelor's degree in informatics engineering from Pasundan University, in 2015, and the M.Eng. degree in computer engineering from Bandung Institute of Technology, in 2018. He is currently pursuing the Ph.D. degree with the Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan. His research interests include security and privacy in distributed learning and sensors and embedded systems

YUGO NAKAMURA (Member, IEEE) was born in 1992. He received the B.E. degree from the Advanced Course of Production System Engineering, National Institute of Technology, Hakodate College, Japan, in 2015, and the M.E. and Ph.D. degrees from the Graduate School of Information Science, Nara Institute of Science and Technology, Japan, in 2017 and 2020, respectively. He is currently an Assistant Professor with the Graduate School and the Faculty of Information Science and Electrical Engineering, Kyushu University. His current research interests include the Internet of Things, ubiquitous computing, and human–computer interaction. He is currently a member of ACM and IPSJ.

YUTAKA ARAKAWA (Member, IEEE) received the B.E., M.E., and Ph.D. degrees from Keio University, Japan, in 2001, 2003, and 2006, respectively. He is currently a Professor with the Graduate School and the Faculty of Information Science and Electrical Engineering, Kyushu University. He also an Invited Professor with Osaka University. His current research interests include human activity recognition, behavior change support systems, and location-based information systems. He has received the 2023 Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology, PerCom 2019 Best Demonstration Award, the IPSJ/IEEE-CS Young Scientist Award, the Ubicomp/ISWC 2016 Best Demo Award, and more than 35 other awards.