applied sciences

MDPI

Article

FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning

Muhammad Asad [1,2,*], **Ahmed Moustafa** [1,3] and **Takayuki Ito** [1,2]

[1] Department of Computer Science, Nagoya Institute of Technology, Nagoya 466-8555, Aichi, Japan
[2] Nagoya Institute of Technology (NITech) Frontier Institute, Nagoya 466-8555, Aichi, Japan
[3] Faculty of Informatics, Zagazig University, Zagazig 44519, Egypt
* Correspondence: m.asad@itolab.nitech.ac.jp

Received: 12 March 2020; Accepted: 14 April 2020; Published: 21 April 2020

Abstract: Artificial Intelligence (AI) has been applied to solve various challenges of real-world problems in recent years. However, the emergence of new AI technologies has brought several problems, especially with regard to communication efficiency, security threats and privacy violations. Towards this end, Federated Learning (FL) has received widespread attention due to its ability to facilitate the collaborative training of local learning models without compromising the privacy of data. However, recent studies have shown that FL still consumes considerable amounts of communication resources. These communication resources are vital for updating the learning models. In addition, the privacy of data could still be compromised once sharing the parameters of the local learning models in order to update the global model. Towards this end, we propose a new approach, namely, Federated Optimisation (FedOpt) in order to promote communication efficiency and privacy preservation in FL. In order to implement FedOpt, we design a novel compression algorithm, namely, Sparse Compression Algorithm (SCA) for efficient communication, and then integrate the additively homomorphic encryption with differential privacy to prevent data from being leaked. Thus, the proposed FedOpt smoothly trade-offs communication efficiency and privacy preservation in order to adopt the learning task. The experimental results demonstrate that FedOpt outperforms the state-of-the-art FL approaches. In particular, we consider three different evaluation criteria; model accuracy, communication efficiency and computation overhead. Then, we compare the proposed FedOpt with the baseline configurations and the state-of-the-art approaches, i.e., Federated Averaging (FedAvg) and the paillier-encryption based privacy-preserving deep learning (PPDL) on all these three evaluation criteria. The experimental results show that FedOpt is able to converge within fewer training epochs and a smaller privacy budget.

Keywords: Federated Learning; Artificial Intelligence; privacy preserving; communication efficiency

1. Introduction

Artificial Intelligence (AI) has been employed in a plethora of application fields in recent years [1]. In this context, as a notable branch of AI, Deep Learning (DL) has been broadly used to empower plenty of data-driven real-world applications, such as facial recognition, autonomous driving and smart grid systems [2–4]. These DL-based applications usually demand the gathering of large quantities of data from various IoT edge-devices for training high-quality learning models. However, the traditionally centralised DL models require the local edge-devices to upload their private data to a central cloud server, which may cause serious privacy threats [5]. These privacy threats can be mitigated through distributing the local training among multiple edge-devices, which has led to the emergence of Federated Learning (FL) [6]. Federated Learning (FL) resolves this problem by allowing the edge-devices to collaboratively train a DL model on their individually gathered data, without revealing their private-sensitive data to a centralised server. This privacy-preserving collaborative learning technique is achieved by following three simple steps as illustrated in Figure 1. In the first step, all the users download the global model and the learning parameters from the cloud server. In the second step, the users train the local learning models based on their local data using Distributed Stochastic Gradient Descent (DSGD) [7]. Finally, in the third step, all the users upload the parameters of their locally trained models to the server, where they are aggregated to generate a new global model. These three steps are continuously repeated until the desired convergence level is achieved. However, despite this efficient training scheme, a major issue in FL is the massive communication overhead that generally evolves from the model updates [8]. In specific, following the above described FL protocol, each user has to communicate its full gradient update during each epoch. This update is normally the same size as the fully trained model, where the trained model could be in the size of gigabytes based on the DL architecture and its millions of parameters [9]. This size can easily reach petabytes when the training is conducted on large-scale datasets that require thousands of training epochs. As a result, communication cost with the limited bandwidth makes FL completely infeasible and unproductive. In addition, FL still faces various privacy concerns as shown in recent studies [10]. For example, a malicious user in a network of shared parameters can access the personal images of users from surveillance systems through various attacks. In a different context, several adversaries can similarly violate the emergency responses of autonomous vehicles or change the health records of several patients from their wearable devices [11]. These threats not only result in serious privacy leakages but also bring in unpredictable life losses. As a result, privacy preservation in FL has become an important factor that spurs the further advancements and developments of efficient FL approaches. To the best of our knowledge, none of the existing approaches supports communication efficiency and privacy preservation in FL at the same time [12].

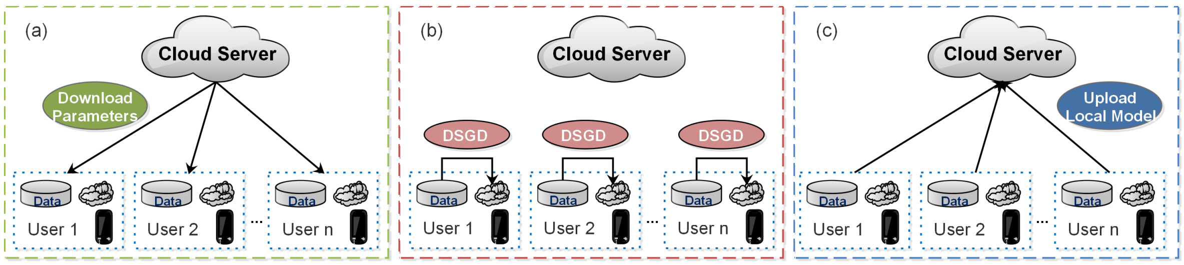

Figure 1. Federated learning process in one communication round of distributed stochastic gradient descent (DSGD): (**a**) Users downloads the parameters and synchronise with the cloud server. (**b**) Users compute the local model based on individual's data in a distributed manner. (**c**) Finally, users upload the computed local models to the cloud server, where they are averaged to generate a global model.

To this end, in this paper, we propose a novel approach, namely, Federated Optimisation (FedOpt), based on Distributed Stochastic Gradient Descent (DSGD) optimisation. The major contributions in this approach are summarised as follows:

1. FedOpt utilises the novel Sparse Compression Algorithm (SCA) in order to reduce the communication overhead. In particular, SCA extends the existing top-k gradient compression technique and enables downstream compression with a novel mechanism.
2. FedOpt adopts a lightweight homomorphic encryption for efficient and secure aggregation of the gradients. In particular, FedOpt provides a concrete abstract, where additively homomorphic encryption is completely utilised in order to eliminate the key-switching operation and to increase the space for plain-text.
3. To further ensure the privacy of local users from the collusion of adversaries, FedOpt uses a differential-privacy scheme based on Laplace mechanism in order to keep the originality of local gradients.

4. FedOpt tolerates user drops during the training process with negligible amounts of accuracy losses. Furthermore, the performance evaluation demonstrates the training accuracy of FedOpt in real-life scenarios as well as its efficient communication and low computation overhead.

The remainder of this paper is organised as follows: The system model and the problem statement are presented in Section 2. Federated learning and the primary techniques of cryptography are briefly explained in Section 3. Afterwards, we introduce FedOpt in Section 4 and conduct the experimental evaluations in Section 5. We discuss the related work and a comprehensive comparison to the exiting approaches in Section 6. Finally, Section 7 concludes the paper with future directions.

2. System Model and Problem Statement

Below, we first describe the system model and then define the problem statement of the proposed approach.

2.1. System Model

In the proposed FL environment, two main entities constitute the basic parts of the whole system: users and the cloud server. The major objective of the proposed approach is to minimise the communication cost and to secure the privacy of individual users from the honest-but-curious adversaries during the training process. In particular, the cloud server honestly executes all the data aggregation process but it is also curious to infer private data from the inputs of users. Therefore, the proposed approach is designed in a way that it can prevent the collusion between the users and the cloud server. For this, we demand that the cloud server receives only the encrypted aggregated result from the local gradients in order to avoid the harmful use of private information. To this regard, in this model, we assume that all the users agree on the same leaning task with the same objectives and parameters as shown in Figure 2. In specific, these users are required to compute the local gradients from their private training datasets and then upload it to the cloud server. Afterwards, these users receive the aggregated global gradient from the cloud server. To ensure privacy, each local gradient is encrypted before being uploaded to the cloud server. Meanwhile, the cloud server is assigned the primary task, that is to compute the global gradient based-on the encrypted local gradients. After computing the global gradient, the cloud server broadcasts this global gradient to all the users, and then the training begins on the proposed model. Finally, the proposed approach works by following this iterative collaboration between the cloud server and the users.

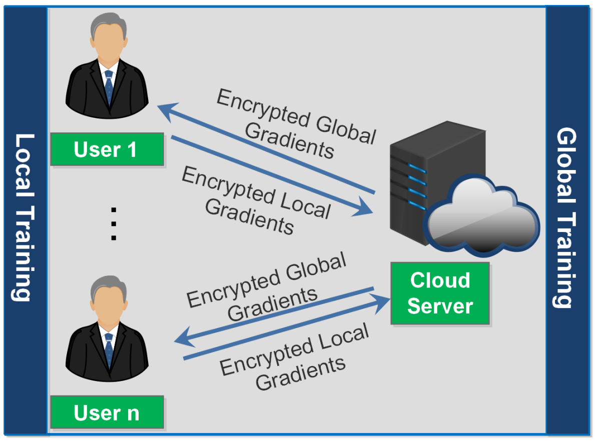
Local Training
User 1
Encrypted Global Gradients
Encrypted Local Gradients
Encrypted Global Gradients
Encrypted Local Gradients
User n
Cloud Server
Global Training

Figure 2. System Model.

2.2. Problem Statement

As mentioned in Section 1, massive communication overhead and malicious users can make the FL infeasible. In this context, we consider the typical environment of FL, where local users collaboratively learn a global parametric neural network. Thus, we propose an approach that use data compression technique for efficient communication and integrates additively homomorphic encryption with differential privacy to prevent data from being compromised. The major objective in this approach is to obtain a parameter vector ν in Deep Neural Network (DNN) that is required in order to minimise the expected loss ϱ:

$$\ell(\nu) = \sum_{D_i \in D} \sum_{(\mathbf{x},\mathbf{y}) \in D_i} \varrho(f_{\amalg}(\mathbf{x}, \nu), \mathbf{y}) \tag{1}$$

As described in the system model, the users learn their local models on their personal datasets and then upload their gradients which are calculated using this loss function to the cloud server. Meanwhile, $\varrho(x, y)$ denotes the loss function and each user $\amalg$ computes the local gradient using gradient function $f_{\amalg}$ on its private dataset D_i. In order to further ensure the privacy, we apply differential privacy with additively homomorphic encryption on the uploaded gradients during the training process to achieve the highest accuracy.

3. Preliminaries

In this section, we first briefly explain FL and then discuss the primary cryptographic techniques that serve as a foundation of the proposed FedOpt.

3.1. Federated Learning

Federated Learning (FL) is an emerging privacy-protecting and decentralised learning scheme that enables edge-devices (local users) to learn a shared global model without disclosing their personal and private data to the cloud server. In FL, user download a shared global model from the cloud server, train this global model over individuals' local data, and then send the updated gradients back to the cloud server. Afterwards, the cloud server aggregates these updated gradients in order to compute a new global model. The following are some unique features of FL compared to traditional centralised learning.

1. The learned model is shared between the users and the cloud server. However, the training data which is distributed on each user is not available to the cloud server.
2. Instead of the cloud server, the training of learning model occurs on each user. The cloud server receives the local gradients and aggregates these gradients to obtain a global gradient and then send this global gradient back to all the users.

In this paper, we consider the standard settings of FL, where large-scale of local users train the global learning model in a distributed and collaborative manner.

3.2. Additively Homomorphic Encryption

The homomorphic encryption performs a set of mathematical computations on plain-text and derives a new cipher-text which presumably same as the plain text after decryption. Meanwhile, additively homomorphic encryption performs the additivity on multiple cipher-texts and decrypts the encrypted result at the same time [13]. Therefore, local users can send this encrypted data for processing on the cloud server without revealing the private information. For instance, consider two plain-texts ξ_1, ξ_2, such that

$$\begin{aligned} E_\delta(\xi_1 + \xi_2) &= \tau_1 \oplus \tau_2 \\ E_\delta(\alpha \times \xi_1) &= \alpha \otimes \tau_2 \end{aligned} \tag{2}$$

where E_δ represents the encrypted-secret text, τ_1, τ_2 denotes the cipher-text of ξ_1, ξ_2, respectively, and α is a constant for any encrypted text.

3.3. Differential Privacy

Differential privacy is a privacy preserving technique that ensures the overall statistics of a dataset will remain same, regardless of change in a single tuple. For example, any algorithm Λ satisfies ϵ-differential privacy (ϵ-DP), if it satisfies the following:

$$P[\Lambda(D) \in T] \leq e^{\epsilon} P[\Lambda(\overline{D}) \in T] \tag{3}$$

where P indicates privacy, D and $\overline{D}$ represent any two neighbouring datasets that have only a single different element, T denotes a set of tuples, and ϵ represents the privacy budget. Whereas, the privacy budget ϵ is an important factor in differential privacy which ranges from 0 (minimum-ϵ) to 1 (maximum-ϵ) [14].

3.4. Laplace Mechanism

Any gradient function f_{II} satisfies the ϵ-DP, if it satisfies the following:

$$f'_{\text{II}}(D) = f_{\text{II}}(D) + Lap(\frac{\Delta f_{\text{II}}}{\epsilon}) \tag{4}$$

where $Lap(\frac{\Delta f_{\text{II}}}{\epsilon})$ is generated from Laplace distribution which satisfies the $P[Lap(\frac{\Delta f_{\text{II}}}{\epsilon}) = x] = \frac{\epsilon}{2\Delta f_{\text{II}}} e^{\frac{-|x|\epsilon}{\Delta f_{\text{II}}}}$ and function f_{II} determines the gradients for each user during the epoch [15].

4. Federated Optimisation (FedOpt)

In this section, we propose a new FedOpt approach based on DSGD optimisation in order to promote communication efficiency and privacy preservation in FL.

4.1. Sparse Compression Algorithm (SCA)

In the existing literature, sparse top-k, a compression algorithm prove the significant performance in distributed training of data [16–18]. Therefore, we use this observation as a starting point to construct a communication efficient protocol in FL. To this end, we design a Sparse Compression Algorithm (SCA) for FedOpt, to reduce the number of communication bits during the models training. In particular, in SCA algorithm, we introduce temporal sparsity into DSGD, which is inspired by [6] to reduce the communication delay. SCA allow each user to perform multiple epochs of SGD, to compute more informative updates. These updates are given by

$$\Delta \nu = SGD_n(\nu, G_{\text{II}}) - \nu \tag{5}$$

where $SGD_n(\nu, G_{\text{II}})$ refers to the set of gradient updates after n epochs of SGD on DNN parameters ν during the sampling of mini-batches from local data G_{II}. Based on the experiments in Sections 5.1–5.3, we conclude that communication delay reduces drastically, with marginal degradation of accuracy. For details about the impact of existing compression techniques on communication delay, we refer the reader to [18].

SCA Technique

We use the proportion of each user gradient into a full gradient update. To implement this, we set the biggest and smallest fraction q of gradient updates to zero. Then, we compute the mean Ψ of all the remaining negative and positive gradient updates, separately. Afterwards, if the absolute negative mean Ψ^- is smaller than the positive mean Ψ^+, then we set all the positive values to the positive mean Ψ^+ and all the negative values to zero. Otherwise, if the absolute negative mean Ψ^- is bigger than positive mean Ψ^+, then we set all the positive values to zero and all the negative values to the negative mean Ψ^-. The detailed technique is formalised in Algorithm 1. In order to find the values of biggest and smallest fraction q in a parameter vector ν, SCA requires the number $\mathcal{O}(|\nu_n|)$ operations, where ν_n refers as the total number of parameters in ν. Following the above technique, SCA reduces the required number of bits $\mathscr{b}_{num}$ from 32 to 0 through computing the non-zero values of sparse gradient update to the mean Ψ. This results in the reduction of communication cost of up to $\times 3$.

Algorithm 1: SCA: Communication Efficiency in FedOpt

```
    Input   : temporal vector $\Delta\nu$, Sparsity Fraction $q$
    Output  : sparse temporal $\Delta\nu^*$
1   Initialisation:
2   $num^+ \leftarrow top_{q\%}\ (\Delta\nu)$; $num^- \leftarrow top_{q\%}\ (-\Delta\nu)$
3   $\Psi^+ \leftarrow$ mean $(num^+)$; $\Psi^- \leftarrow$ mean $(num^-)$
4   if $\Psi^+ \geq \Psi^-$
5     then
6   return $(\Delta\nu^* \leftarrow \Psi^+\ (\nu \geq \min(num^+)))$
7     else return $(\Delta\nu^* \leftarrow -\Psi^-\ (\nu \leq \min(-num^-)))$
8   end
```

4.2. Gradient Aggregation in FedOpt

Secure gradient aggregation in the form of cipher-text can be achieved through homomorphic encryption. However, the large amounts of required communication resources and the computation overhead on public-key encryption might delay and disturb the accuracy of data [19,20]. Towards this end, we utilise the additively homomorphic encryption in FedOpt in order to achieve efficiency throughout the learning process. Furthermore, differential privacy is used in order to tolerate the local users' dropouts and to add calibrated noises before encryption in each gradient. In this context, each user II uses a small-size batch from the local dataset D_i and learns the model to compute the local gradient G_{II} in each epoch. In order to protect their local gradients, the local users use Laplace mechanism to encrypt their local gradients using $E_{\text{II}} = E_\delta(G_{\text{II}} + Lap(\frac{\Delta f_{\text{II}}}{\epsilon}))$. Once the cloud server receives all the encrypted gradients, it conducts the aggregation operation where the noises are nearly eliminated due to the symmetry of the Laplace mechanism. This aggregation operation is processed by the following equation:

$$E_{add} = \tau_1 \oplus \tau_2 \oplus \ldots \oplus \tau_n = E_\delta(\sum_{\text{II}=1}^{n} G_{\text{II}}) \tag{6}$$

In the end, all the users decrypt the encrypted global gradient E_{add} that is received from the cloud server using the following equation:

$$D_\delta(E_{add}) = \sum_{\text{II}=1}^{n} G_{\text{II}} \tag{7}$$

The detailed pseudocode of privacy preservation technique using differential privacy which is integrated with additively homomorphic encryption is formalised in Algorithm 2.

Algorithm 2: Pseudocode of Privacy Preserving

```
Input   : Users for local datasets $D_i$, the cloud server to initialise global parameters $\varpi_o$
Output: New global parameters $\varpi$
1  Initialisation:
2  while Cloud server initialise global parameters $\varpi_o$ do
3      Aggregate global parameters $\varpi_o$ to users
4      while Users obtain local gradients $G_{\text{II}}$ by training local models $D_i$ do
5          Add noise $\epsilon$-DP $\leftarrow G_{\text{II}}$
6          Encrypt $G_{\text{II}} \leftarrow E_\delta(G_{\text{II}} + Lap(\frac{\Delta f_{\text{II}}}{\epsilon}))$
7          Generate encrypted local gradients $E_{\text{II}}$
8          Aggregate $E_\delta(\sum_{\text{II}=1}^{n} G_{\text{II}})$
9      end
10     while Cloud server aggregates encrypted local gradients to users II do
11         $E_{add} \leftarrow E_\delta(\sum_{\text{II}=1}^{n} G_{\text{II}})$
12         Generate cipher-text from $E_{\text{II}}$
13         Generate encrypted global gradients $E_{add}$
14     end
15     while Users decrypts $E_{add}$ to get global gradients $B_{\text{II}}$ do
16         $D_\delta(E_{add}) \leftarrow \sum_{\text{II}=1}^{n} G_{\text{II}}$
17         Update existing parameters $\varpi$
18         Aggregate new parameters $\varpi$ to the cloud server
19     end
20 end
```

4.3. Efficiency and Privacy in FedOpt

The efficiency and privacy preservation of FedOpt are set in each epoch and the complete process of each epoch is divided into multiple phases as follows:

4.3.1. Initialisation Phase

In the beginning, the global parameters ϖ_o and the learning rate $\wp$ are initialised by the cloud server. Then, all the users copy the global training model to the private devices. Apart from having the security parameter σ, a secret key δ is assigned to each user which is comprised of two big prime numbers $j, k(|j| = |k| = \sigma)$ where, these prime numbers are given as public parameters M.

4.3.2. Encryption Phase

In this phase, all the users jointly choose the same level of privacy budget ϵ in order to maintain the differential privacy. Specifically, in each epoch, the set of users II derives their initial parameters ϖ and obtains their local gradients G_{II} through their individual datasets. Afterwards, the set of users II utilise a privacy measure by randomly choosing the noises from the Laplace distribution $Lap(\frac{\Delta f_{\text{II}}}{\epsilon})$ and adds it to the local gradients.

$$
\begin{aligned}
G_{\text{II},j} &\equiv (G_{\text{II}} + Lap(\tfrac{\Delta f_{\text{II}}}{\epsilon})) \mod j \\
G_{\text{II},k} &\equiv (G_{\text{II}} + Lap(\tfrac{\Delta f_{\text{II}}}{\epsilon})) \mod k
\end{aligned}
\tag{8}
$$

In the equation above, both the privacy budget ϵ and the sensitivity Δf_{II} of Laplace distribution play important roles in differential privacy. Meanwhile, Δf_{II} can be set to 1 and each gradient is assumed to set at $0 \leq G_{\text{II}} \leq 1$ by utilising the min-max normalisation [21].

Subsequently, the users encrypt their local gradients using the secret key δ from j, k as given below:

$$E_{\text{II}} = j^{-1}jG^{k}_{\text{II},k} + k^{-1}kG^{j}_{\text{II},j} \mod M \tag{9}$$

where, M is the public parameter $M = jk$ and j^{-1}, k^{-1} denote the inverses of j, k respectively. In the end, these encrypted local gradients E_{II} from all the users are sent to the cloud server.

4.3.3. Aggregation Phase

Once all the gradients G_{II} are received by the cloud server, it initialises the secure aggregation process as given below:

$$\begin{aligned} &E_{add} = \textstyle\sum_{\text{II}=1}^{n} E_{\text{II}} \\ &= j^{-1}j(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},k})^{k} + k^{-1}k(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},j})^{j} \mod M \end{aligned} \tag{10}$$

Afterwards, the cloud server begins communication with all the local users and broadcasts the encrypted global gradient E_{add}, in order to avoid collusion from adversaries.

4.3.4. Decryption Phase

Once the local users receive the global encrypted gradient E_{add}, each user begins the decryption process as follows:

$$\begin{aligned} &E_{add} \mod k \\ &= j^{-1}j(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},k})^{k} + k^{-1}k(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},j})^{j} \\ &= j^{-1}j(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},k})^{k-1}(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},k}) \mod k \\ &= \textstyle\sum_{\text{II}=1}^{n} E_{\text{II},k} \mod k \\ &= E_{add,k} \mod k \end{aligned} \tag{11}$$

In similar fashion,

$$\begin{aligned} &E_{add} \mod j \\ &= k^{-1}k(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},j})^{j} + j^{-1}j(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},k})^{k} \\ &= k^{-1}k(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},j})^{j-1}(\textstyle\sum_{\text{II}=1}^{n} E_{\text{II},j}) \mod j \\ &= \textstyle\sum_{\text{II}=1}^{n} E_{\text{II},j} \mod j \\ &= E_{add,j} \mod j \end{aligned} \tag{12}$$

Following the above procedure, the local users utilise the Chinese Remainder Theorem (CRT) in order to obtain the final decrypted global gradients B_{II} [22]:

$$E_{add} \equiv \begin{cases} E_{add,j} \mod j \\ E_{add,k} \mod k \end{cases} \tag{13}$$

Since the number of users is sufficient in real-world scenarios, therefore, FedOpt tolerates the users which might drop at any instance of time. Therefore, there is nearly zero effect on eliminating the noises. In the end, each user updates the parameters ϖ according to $\varpi \leftarrow \varpi - \frac{\text{II}}{N}E_{add}$, where N is received from the cloud server. Afterwards, the whole operation is performed repeatedly until the loss function ϱ is achieved.

The complete FedOpt approach that features two-way (upstream and downstream) compression via SCA and performs optimal encryption through differential privacy is shown in Algorithm 3.

Algorithm 3: FedOpt: Communication-Efficiency and Privacy-Preserving

```
Input  : Initial parameters $\varpi$
Output : Global model with improved parameters $\varpi_o$
1  Initialisation: all users $\amalg_i, i = 1, \ldots,$[Total number of users] are initialised with the same
     parameters $\nu_i \leftarrow \nu$. Those users who carry different private datasets $D_i$ with |{c : (x, y) $\in$
     $D_i\}$| = [total classes per user]. The remaining $\amalg$ are initialised to zero $\Delta\nu, \mathcal{R}_i, \mathcal{R} \leftarrow 0$.
2  for epoch e = 1,..., E | E = Total number of Epochs | do
3      for $\amalg_i \in \amalg \subseteq$ {1,..., [Number of users]} do
4          User $\amalg_i$ execute:
5          Plain-text = $\xi \leftarrow downloads_{CS \rightarrow \amalg_i}\ (\xi)$
6          $\Delta\nu \leftarrow$ decrypt ($\xi$)
7          $\nu_i \leftarrow \nu_i + \Delta\nu$
8          $\Delta\nu_i \leftarrow \mathcal{R}_i + SGD(\nu_i, D_i) - \nu_i$
9          $\Delta\overline{\nu_i} \leftarrow SCA_{upload}(\Delta\nu_i)$
10         $\mathcal{R}_i \leftarrow \Delta\nu_i - \Delta\overline{\nu_i}$
11         $\xi_i \leftarrow$ encrypt $\Delta\overline{\nu_i}$
12         $upload_{\amalg_i \rightarrow CS}(\xi_i)$
13     end
14     Cloud Server CS execute:
15     $collect_{\amalg_i \rightarrow CS}(\Delta\overline{\nu_i}), e \in \amalg$
16     $\Delta\nu \leftarrow \mathcal{R} + \frac{1}{\amalg} \sum_{e \in \amalg} \Delta\overline{\nu_i}$
17     $\Delta\overline{\nu} \leftarrow SCA_{download}(\Delta\nu)$
18     $\mathcal{R} \leftarrow \Delta\nu - \Delta\overline{\nu}$
19     $\nu \leftarrow \nu + \Delta\overline{\nu_i}$
20     $\xi \leftarrow$ encrypt $\Delta\overline{\nu_i}$
21     $Aggregate_{CS \rightarrow \amalg_i}(\xi), i = 1, \ldots, GlobalModel$
22 end
23 return $\varpi_o$
```

5. FedOpt Evaluation

In this section, we conduct the experimental evaluation of the proposed FedOpt in terms of model accuracy, communication efficiency and computational overhead. We conduct our experiments on the server with an Intel(R) Core(TM) CPU i7-4980HQ (2.80 GHz) and 16 GB of RAM. The compression and privacy-preserving algorithms are simulated by TensorFlow in Python. For evaluation, we consider baseline configuration of FL, Federated Averaging (FedAvg) [23] and Privacy Preserving Deep Learning (PPDL) [24]. In particular, we evaluate the performance of FedOpt on MNIST dataset where the gradient consists of 60,000 training examples and each example consists of 28 $\times$ 28 size images. Then, similar to MNIST dataset, we assess the performance of FedOpt on CIFAR-10 dataset where the gradient consists of 50,000 training examples and 10,000 testing examples and each example consists of 32 $\times$ 32 size images with three different RGB channels. The baseline configuration setup is given in Table 1.

Table 1. Baseline Configuration.

Parameters	Number of Users	Participation Ratio	Mini-Batch Size	Classes per User	Gradient Size	Number of Epochs	Privacy Budget
Value	Various	10%	20	10	32-bits	Various	0.5

5.1. Accuracy Test

Accuracy is an important factor to measure the performance of any model in DL. In this regard, the proposed FedOpt is able to achieve the accuracy of 99.6% and 98.4% after 500 epochs on MNIST and CIFAR-10 datasets, respectively. As shown in Figures 3a and 4a, we conduct the experiments on various numbers of privacy budgets ϵ, i.e., $0.2, 0.4, 0.6, 0.8$ and 1.0, in order to test the accuracy of FedOpt on MNIST and CIFAR-10 datasets, respectively. Compared with FedAvg and PPDL, FedOpt is able to achieve 92.3% on 0.2 (lowest-ϵ) and 99.6% on 1.0 (highest-ϵ) of accuracy on MNIST dataset. Similarly, FedOpt is able to achieve 91.2% on 0.2 (lowest-ϵ) and 98.7% on 1.0 (highest-ϵ) of accuracy on CIFAR-10 dataset. The above results demonstrate that the number of privacy budgets ϵ has a huge impact on the prediction accuracy. Therefore, we conclude that, higher levels of privacy budget ϵ produce higher accuracy, but provide lower levels of privacy. Furthermore, we also conduct the accuracy tests with regard to the impact of various number of users $\amalg$, e.g., $200, 400, 600, 800$ and 1000 on the constant privacy budget at 0.5-ϵ. For example, in Figures 3b and 4b, the accuracy increases with the increasing number of users on MNIST and CIFAR-10 datasets, respectively. In specific, in Figure 3b, FedOpt achieves 97.1% on 200 users (minimum-$\amalg$) and 99.7% on 1000 users (maximum-$\amalg$) of accuracy on MNIST dataset. Similarly, as shown in Figure 4b, FedOpt achieves 93.4% on 200 users (minimum-$\amalg$) and 98.6% on 1000 users (maximum-$\amalg$) of accuracy on CIFAR-10 dataset. As shown in Figures 3 and 4, the proposed FedOpt is compared with FedAvg and PPDL, where it is able to achieve the highest level of accuracy. This is attributed to the fact that a huge part of the noises is eliminated through the symmetry of Laplace mechanism and the complete utilisation of SCA. Furthermore, differential privacy provides protection to gradients during the training process.

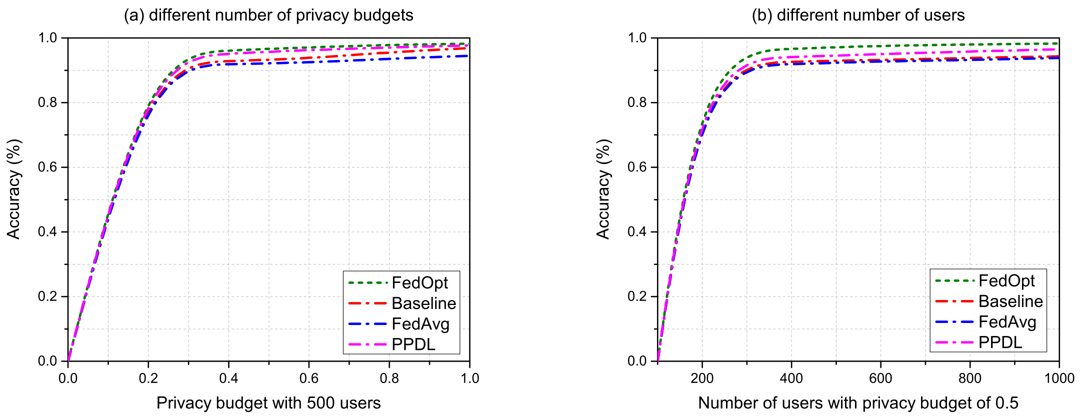

Figure 3. Achieved accuracy of model on MNIST dataset.

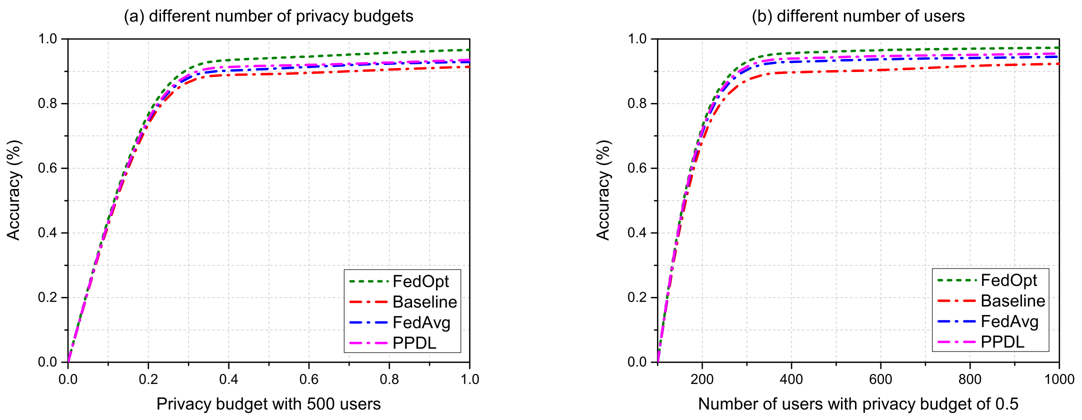

Figure 4. Achieved accuracy of model on CIFAR-10 Dataset.

5.2. Communication Efficiency

In our experiments, we consider the communication efficiency among the cloud server and the users Π as they are the main entities of the whole system. In specific, during the aggregation phase, we assume there are n epochs in the whole training process and each user Π has a single thread with the security parameter σ is set to 512 and the size of each local gradient G_{Π} is 32 bits. In each epoch, the users Π aggregate the encrypted local gradients E_{Π} to the cloud server and receives the shared parameters E_{add} from the cloud server. Figures 5 and 6 show the comparison result of communication efficiency between FedOpt, FedAvg and PPDL on MNIST and CIFAR-10 datasets, respectively. In specific, we consider different numbers of gradients and different numbers of users for evaluation in Figures 5a,b and 6a,b respectively. Clearly, it can be demonstrated that the increasing numbers of gradients with the maximum numbers of users has the maximum communication efficiency. Compared to the FedAvg and PPDL, FedOPT has 56% and 38% more communication efficiency, respectively, on MNIST dataset. Similarly, FedOpt outperforms on CIFAR-10 dataset with 54% and 32% more communication efficiency as compare to the FedAvg and PPDL. The major reason behind this higher communication efficiency is that, FedOpt completely utilises pallier encryption [25] which helps in the rapid growth of cipher-text volume. In addition, SCA algorithm helps FedOpt in faster convergence in terms of training epochs with significant compression rate.

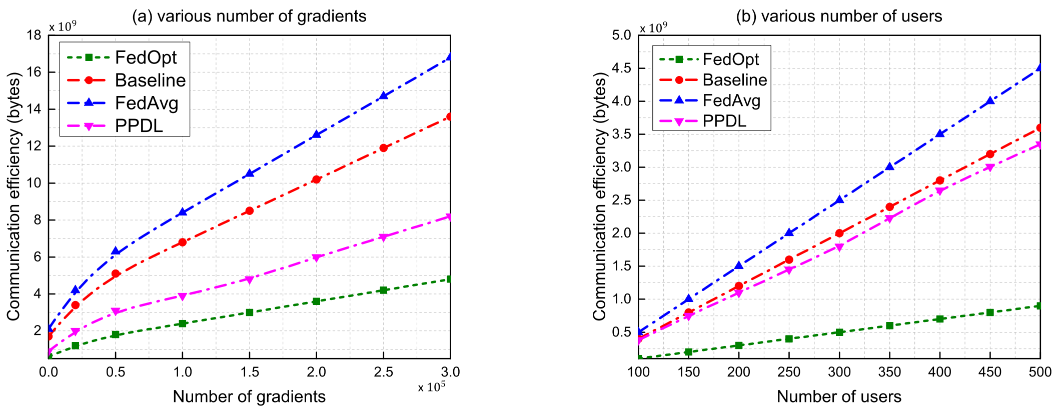

Figure 5. FedOpt communication efficiency on MNIST dataset.

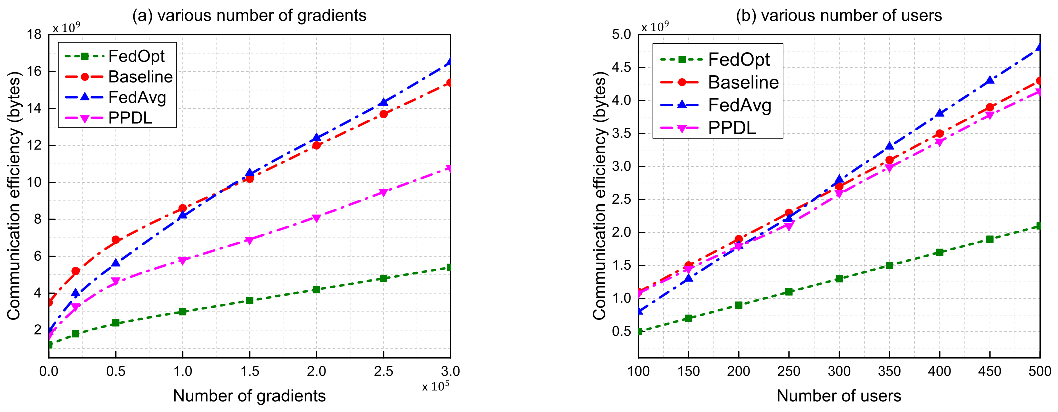

Figure 6. FedOpt communication efficiency on CIFAR-10 dataset.

5.3. Analysis of Communication Efficiency w.r.t Accuracy

In this subsection, we compare the proposed compression algorithm SCA with respect to the number of epochs and the communicated bits that are required to achieve the targeted accuracy on a FL task. In the above subsections, FedOpt performed significantly better than FedAvg and PPDL. In order to have a meaningful comparison, we choose 100 users for 50 and 100 epochs, where every user holds 10 different classes and uses a batch-size of 20 during training. This setup of less number of users and epochs favours the FedAvg and PPDL. The rest of the parameters of the learning environment is the same as given in Table 1. We train the datasets until the targeted accuracy is achieved in the given number of epochs and measure the total communicated bits both for upload and download. The required amounts of upstream and downstream communication bits to achieve the targeted accuracy is given in megabytes (MB) in Table 2.

Table 2. Communication bits required for upload and download to achieve the targeted accuracy.

	MNIST (Accuracy = 91.3)	**CIFAR-10 (Accuracy = 87.6)**
Baseline	2218/2218 MB	35653 MB/35653 MB
FedAvg $epochs = 50$	119.65 MB/119.65 MB	2589.5 MB/2589.5 MB
FedAvg $epochs = 100$	84.73 MB/84.73 MB	1665.7 MB/1665.7 MB
PPDL $epochs = 50$	98.63 MB/311.6 MB	1472.2 MB/4739.2 MB
PPDL $epochs = 100$	63.74 MB/432.2 MB	958.3 MB/6342.4 MB
FedOpt $epochs = 50$	10.2 MB/102 MB	109.23 MB/1090.3 MB
FedOpt $epochs = 100$	14.6 MB/146 MB	172.3 MB/1723 MB

In Table 2, FedOpt communicates 14.6 MB and 172.3 MB of data on MNIST and CIFAR-10 datasets, which is a reduction in communication by a factor of $\times$152 and $\times$207 as compared to baseline configurations. Meanwhile, FedAvg and PPDL ($epochs = 100$) requires 84.73 and 63.74 MB of data on MNIST dataset and 1665.7 and 958.3 MB of data on CIFAR-10 dataset which proves that proposed FedOpt have a minimum delay period in order to achieve the targeted accuracy within a given number of training epochs.

5.4. Computation Overhead

In the end, we discuss the computation cost of FedOpt on MNIST and CIFAR-10 datasets as shown in Figures 7 and 8, respectively. We only consider the running time of the cipher-text operation to prove our main contribution. By considering the security requirement, we select plain-text $\xi_1 = 2^{16}$ with the security parameter of $\sigma = 128$ bits, and analyse the computational cost per each user Π and the cloud server on each phase as mentioned in Section 4.3. In specific, in each subfigure, as demonstrated in Figures 7 and 8, the computational cost increases linearly with the increasing number of gradients because FedOpt encrypts every single packet in each aggregation. Therefore, the computational overhead over the encryption process is related to the total number of gradients regardless of number of users. Furthermore, increased security (higher security parameter σ) leads to the inefficiency. In this regard, as shown in Figure 7, FedOpt achieves 74% and 53% at the encryption phase, 72% and 45% at the aggregation phase, and 86% and 31% at the decryption phase, less computational overhead than FedAvg and PPDL, respectively, on MNIST dataset. Similarly, Figure 8 shows the computation overhead on CIFAR-10 dataset where FedOpt achieves 61% and 52% at the encryption phase, 43% and 31% at the aggregation phase and 72% and 48% at the decryption phase, less computational overhead than FedAvg and PPDL. The overall computational overhead for users at the encryption phase with the security parameter of $\sigma = 128$ bits is about $\times$2.8 slower than the baseline configurations because FedOpt requires fewer addition and multiplication operations. Similarly, the overall computational overhead for the cloud server with the security parameter of $\sigma = 128$ bits is about $\times$9.3 slower than the baseline configurations. This less computational overhead at the server-end is because FedOpt decrypts every single packet in each aggregation, where the number of decryption process linearly increases with the increasing number of gradients. Therefore, the proposed FedOpt is able to support the learning scenarios with large numbers of users.

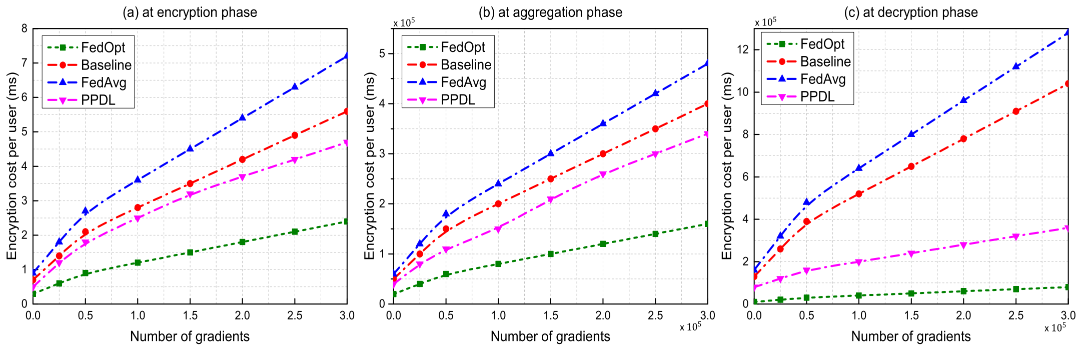

Figure 7. Computational cost per user with different number of gradients on MNIST dataset.

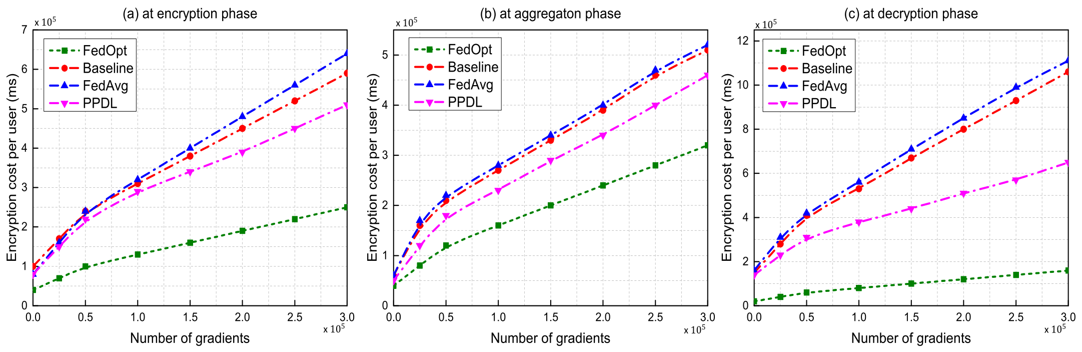

Figure 8. Computational cost per user with different number of gradients on CIFAR-10 dataset.

6. Related Work and Discussions

Stochastic Gradient Descent (SGD) is very popular optimisation training technique that supports various DL applications in DNN models. In particular, on one end of the spectrum, SGD can be used to reduce the convergence time in large-scale applications of DL models by using the device-level parallelism [26–28]. On the other end of the spectrum, SGD can be used to enable and enhance the privacy preserving in DL algorithms [29]. Since the users are require to share the gradient updates, where SGD helps in training of model from the combined data of all the users without revealing the individual's local data to a centralised cloud server [30]. However, despite the tremendous advantages and the extensive applications of SGD based DL, existing research show that, learning the model updates suffers from a massive communication overhead [31,32]. In order to reduce this communication overhead, a wide variety of methods had been proposed in the past. For example, in [23], the authors proposed a novel approach, i.e., FedAvg, where each user computes the gradient updates by performing multiple epochs of SGD which results into increasing the number of gradient evaluations that causes delay in communication. In order to minimise this communication delay, the authors in [33], use probabilistic-quantisation and random-sparsification. In particular, the authors force the random-sparsity on the users or restrict them in order to learn random-sparse gradient updates (structured and sketched updates) and combine probabilistic quantisation with this sparsification. This method however is not suitable for SGD epochs as it slows down the convergence speed significantly. To overcome this convergence issue, the authors in [34], propose a compression technique; namely, SignSGD that theoretically guarantees the convergence over iid data. This SignSGD quantises each gradient from each user to a binary sign and reduces the bit-size per gradient update by $\times 32$. This compression on gradients is done by means of a majority vote which may result into the loss of important updates. In addition, the compression rate and empirical performance does not reach up-to the convergence requirement in FL.

On the other hand, several adversaries may violate the private information of users from DL networks [35,36]. As a result, several privacy preserving DL schemes are proposed in order to eliminate those privacy threats. For example, in [19], the authors proposed the first privacy preserving approach for collaborative DL where, the users share their selective partial gradients with the cloud server in order to ensure the privacy of the training data. However, even sharing a small part of the gradient, the privacy of users can still be compromised [37]. Additionally, in [38], the authors proposed a DL model using additively homomorphic encryption where, all the users share the same decryption key, which may lead to collusion among the users and the cloud server. However, all these approaches suffer from serious privacy threats because of the shared parameters.

On the contrary, unlike the proposed FedOpt, various privacy preserving centralised training approaches are proposed to overcome the privacy issues. For example, the authors in [39], adopt a two-party computation scheme that utilises two-party computation training model, where the local users separate the sensitive data into two parts before sending it to the two non-collusive central servers. In order to eliminate privacy leakage, [40] used a Gaussian mechanism to update the gradients and proposed a rigid method (e.g., moment accountant) in order to keep record of the entire privacy loss.

Functional Comparison

Several communication efficient and privacy preservation distributed approaches have been proposed recently including Practical Secure Aggregation (PSA) [41], Federated Extreme Boosting (XGB) [42], Efficient and Privacy-Preserving Federated Deep Learning (EPFDL) [43] and Privacy-preserving collaborative learning (PPCL) [44] as listed in Table 3. Specifically, PSA and XGB utilise collaborative training in order to resist collusion among adversaries, but both approaches do not guarantee communication efficiency. Compared with EPFDL, the proposed FedOpt is not only communication efficient but also serves for collaborative training. Meanwhile, PPCL serves most of the security features by sharing the same decryption key among all the users, which is not desirable in real-time applications. On the contrary, FedOpt considers these real-time applications, where adversaries may act as honest parties but colludes due to the shared parameters. Table 3 shows that, none of the state-of-the-art approaches completely address the challenges of privacy preservation and communication efficiency. On the other hand, the proposed FedOpt not only mitigates the above attacks by utilising differential privacy, but also provides communication efficiency via compression algorithm.

Table 3. Functionality comparison with existing FL approaches.

Functionality	**PSA**	**XGB**	**EPFDL**	**PPCL**	**FedOpt**
Communication Efficient			✓		✓
Collaborative Training	✓	✓		✓	✓
Non-IID Support			✓		
Gradient Confidentiality	✓		✓	✓	✓
Attack Resilience		✓			✓
Post-Quantum Security		✓	✓		
Collusion Resistance	✓		✓	✓	✓
Fast Convergence Speed	✓		✓		✓
Application Aware		✓		✓	✓
Algorithm Complexity	✓	✓		✓	

7. Conclusions

This paper proposes a novel approach, namely, Federated Optimisation (FedOpt) that is able to simultaneously decrease the communication cost and increase the privacy in federated learning settings. In particular, we design a Sparse Compression Algorithm (SCA) for communication efficiency and integrates the additively homomorphic encryption with differential privacy in order to prevent data from being leaked. Compared to the existing approaches, the proposed FedOpt compresses the upstream and downstream communication and reduces the communication overhead. In general, FedOpt is advantageous especially in the network where communication is costly or bandwidth is constrained as it achieve the targeted accuracy within fewer amounts of communication bits. Furthermore, the proposed FedOpt is able to mitigate the security threats for both the local users and the cloud server. In addition, the proposed FedOpt is completely non-interactive which provides higher levels of privacy at the aggregation phase, even when the adversaries collude with honest-users. The experimental evaluation on both MNIST and CIFAR-10 datasets proves that the proposed FedOpt outperforms the state-of-the-art approaches in terms of accuracy, efficiency and privacy. In the future, we will consider the virtualisation of this work through docker to make it useful in real-life environments. In addition, we plan to investigate further approaches for communication efficiency and privacy preservation while maintaining robustness in federated learning, especially with complex neural networks and high-dimensional datasets for diverse learning tasks and their models.

Author Contributions: All the authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially supported by NITech Frontier Institute.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations and notations are widely used in this manuscript, while the arithmetic operations and their notations are to be understood element-wise:

FedOpt	Federated Optimisation
AI	Artificial Intelligence
FL	Federated Learning
DL	Deep Learning
SGD	Stochastic Gradient Descent
DNN	Deep Neural Network
SCA	Sparse Compression Algorithm
ϵ-DP	Privacy Budget on Differential Privacy
II	users
δ	Secret Key
ϱ	Expected Loss
ϖ	Initial Parameters
ϖ_o	Global Parameters
ν	Parameter Vector

References

1. Müller, V.C.; Bostrom, N. Future progress in artificial intelligence: A survey of expert opinion. In *Fundamental Issues of Artificial Intelligence*; Springer: Cham, Switzerland, 2016; pp. 555–572.
2. Cheng, E.J.; Chou, K.P.; Rajora, S.; Jin, B.H.; Tanveer, M.; Lin, C.T.; Young, K.Y.; Lin, W.C.; Prasad, M. Deep sparse representation classifier for facial recognition and detection system. *Pattern Recognit. Lett.* **2019**, *125*, 71–77. [CrossRef]
3. Li, D.; Zhao, D.; Zhang, Q.; Chen, Y. Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]. *IEEE Comput. Intell. Mag.* **2019**, *14*, 83–98. [CrossRef]

4. Yang, Y.; Li, W.; Gulliver, T.A.; Li, S. Bayesian Deep Learning Based Probabilistic Load Forecasting in Smart Grids. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4703–4713. [CrossRef]
5. Kumar, D.S.; Thilagam, P.S. Approaches and challenges of privacy preserving search over encrypted data. *Inf. Syst.* **2019**, *81*, 63–81. [CrossRef]
6. McMahan, B.; Ramage, D. Federated learning: Collaborative machine learning without centralized training data. *Google Res. Blog* **2017**, *3*.
7. Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; Le, Q.V.; et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2012; pp. 1223–1231.
8. Sattler, F.; Wiedemann, S.; Müller, K.R.; Samek, W. Robust and communication-efficient federated learning from non-iid data. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**. [CrossRef]
9. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S. Analyzing federated learning through an adversarial lens. *arXiv* **2018**, arXiv:1811.12470.
10. Nasr, M.; Shokri, R.; Houmansadr, A. Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks. *arXiv* **2018**, arXiv:1812.00910.
11. Hao, M.; Li, H.; Luo, X.; Xu, G.; Yang, H.; Liu, S. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans. Ind. Inform.* **2019**. [CrossRef]
12. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; d'Oliveira, R.G.; et al. Advances and open problems in federated learning. *arXiv* **2019**, arXiv:1912.04977.
13. Zhang, X.; Chen, X.; Liu, J.; Xiang, Y. DeepPAR and DeepDPA: Privacy-Preserving and Asynchronous Deep Learning for Industrial IoT. *IEEE Trans. Ind. Inform.* **2019**, *16*, 2081–2090. [CrossRef]
14. Tripathy, A.; Wang, Y.; Ishwar, P. Privacy-preserving adversarial networks. In Proceedings of the 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 24–27 September 2019; pp. 495–505.
15. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confid.* **2016**, *7*, 17–51.
16. Aji, A.F.; Heafield, K. Sparse communication for distributed gradient descent. *arXiv* **2017**, arXiv:1704.05021.
17. Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, W.J. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv* **2017**, arXiv:1712.01887.
18. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. A survey of model compression and acceleration for deep neural networks. *arXiv* **2017**, arXiv:1710.09282.
19. Shokri, R.; Shmatikov, V. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1310–1321.
20. Peralta, G.; Cid-Fuentes, R.G.; Bilbao, J.; Crespo, P.M. Homomorphic Encryption and Network Coding in IoT Architectures: Advantages and Future Challenges. *Electronics* **2019**, *8*, 827. [CrossRef]
21. Vadhan, S. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*; Springer: Cham, Switzerland, 2017; pp. 347–450.
22. Roy, P. A homomorphism based zero knowledge proof of authentication for chinese remainder theorem based secret sharing. In *Annual Convention of the Computer Society of India*; Springer: Singapore, 2018; pp. 562–572.
23. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S. Communication-efficient learning of deep networks from decentralized data. *arXiv* **2016**, arXiv:1602.05629.
24. Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1333–1345.
25. Fazio, N.; Gennaro, R.; Jafarikhah, T.; Skeith, W.E. Homomorphic secret sharing from paillier encryption. In *International Conference on Provable Security*; Springer: Cham, Switzerland, 2017; pp. 381–399.
26. Chilimbi, T.; Suzue, Y.; Apacible, J.; Kalyanaraman, K. Project adam: Building an efficient and scalable deep learning training system. In Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), Broomfield, CO, USA, 6–8 October 2014; pp. 571–582.
27. Zinkevich, M.; Weimer, M.; Li, L.; Smola, A.J. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2010; pp. 2595–2603.

28. Li, M.; Andersen, D.G.; Smola, A.J.; Yu, K. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2014; pp. 19–27.
29. Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-preserving deep learning: Revisited and enhanced. In *International Conference on Applications and Techniques in Information Security*; Springer: Singapore, 2017; pp. 100–110.
30. Agarwal, N.; Suresh, A.T.; Yu, F.X.X.; Kumar, S.; McMahan, B. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems*; Neural Information Processing Systems Conference: Montréal, QC, Canada, 2018; pp. 7564–7575.
31. Dörner, S.; Cammerer, S.; Hoydis, J.; ten Brink, S. Deep learning based communication over the air. *IEEE J. Sel. Top. Signal Process.* **2017**, *12*, 132–143. [CrossRef]
32. Nishio, T.; Yonetani, R. Client selection for federated learning with heterogeneous resources in mobile edge. In Proceedings of the 2019 IEEE International Conference on Communications (ICC 2019), Shanghai, China, 20–24 May 2019; pp. 1–7.
33. Konečný, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv* **2016**, arXiv:1610.05492.
34. Bernstein, J.; Wang, Y.X.; Azizzadenesheli, K.; Anandkumar, A. signSGD: Compressed optimisation for non-convex problems. *arXiv* **2018**, arXiv:1802.04434.
35. Wang, Z.; Song, M.; Zhang, Z.; Song, Y.; Wang, Q.; Qi, H. Beyond inferring class representatives: User-level privacy leakage from federated learning. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2019), Paris, France, 29 April–2 May 2019; pp. 2512–2520.
36. Chabanne, H.; de Wargny, A.; Milgram, J.; Morel, C.; Prouff, E. Privacy-preserving classification on deep neural network. *IACR Cryptol. ePrint Arch.* **2017**, *2017*, 35.
37. Ma, X.; Ma, J.; Li, H.; Jiang, Q.; Gao, S. PDLM: Privacy-preserving deep learning model on cloud with multiple keys. *IEEE Trans. Serv. Comput.* **2018**. [CrossRef]
38. Moriai, S. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. In Proceedings of the 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), Kyoto, Japan, 10–12 June 2019; p. 198.
39. Mohassel, P.; Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 19–38.
40. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 308–318.
41. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191.
42. Liu, Y.; Ma, Z.; Liu, X.; Ma, S.; Nepal, S.; Deng, R. Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing. *arXiv* **2019**, arXiv:1907.10218.
43. Hao, M.; Li, H.; Xu, G.; Liu, S.; Yang, H. Towards Efficient and Privacy-Preserving Federated Deep Learning. In Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
44. Zhao, L.; Wang, Q.; Zou, Q.; Zhang, Y.; Chen, Y. Privacy-preserving collaborative deep learning with unreliable participants. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1486–1500. [CrossRef]