Array 15 (2022) 100207

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

CE-Fed: Communication efficient multi-party computation enabled federated learning

Renuga Kanagavelu [a,*], Qingsong Wei [a], Zengxiang Li [b], Haibin Zhang [a], Juniarto Samsudin [a], Yechao Yang [a], Rick Siow Mong Goh [a], Shangguang Wang [c]

[a] *Institute of High Performance Computing, A*STAR, Singapore*
[b] *Digital Research Institute ENN Group, China*
[c] *Beijing University of Posts and Telecommunications, Beijing, China*

ARTICLE INFO

Keywords:
Federated learning
Edge computing
Multi-party computation
Committee selection

ABSTRACT

Federated learning (FL) allows a number of parties collectively train models without revealing private datasets. There is a possibility of extracting personal or confidential data from the shared models even-though sharing of raw data is prevented by federated learning. Secure Multi Party Computation (MPC) is leveraged to aggregate the locally-trained models in a privacy preserving manner. However, it results in high communication cost and poor scalability in a decentralized environment. We design a novel communication-efficient MPC enabled federated learning called CE-Fed. In particular, the proposed CE-Fed is a hierarchical mechanism which forms model aggregation committee with a small number of members and aggregates the global model only among committee members, instead of all participants. We develop a prototype and demonstrate the effectiveness of our mechanism with different datasets. Our proposed CE-Fed achieves high accuracy, communication efficiency and scalability without compromising privacy.

1. Introduction

Artificial intelligence (AI) techniques find useful applications in various domains, such as healthcare, smart building, autonomous vehicles, remote monitoring and predictive maintenance of machines in manufacturing plant. The training of such AI models requires large amount of data to achieve acceptable accuracy and throughput and thus improving the user experience. The huge amount of data generated at different organizations are aggregated at a cloud-based server to produce more-effective inference model. Though this approach is beneficial, the transfer of sheer volume of data to the centralized server burdens the back-bone network. It also introduces long latency [1], which is not acceptable for applications where real-time decisions are required, like self-driving cars and automated car assembly plant [2]. Apart from those, the legal restrictions and rising concerns about sharing private information [3] among data owners make them reluctant to send their data to the data centre for the model training [4]. As a result, huge volume of data generated by individual organizations remains in the form of fragmented data silos.

To address the above-mentioned problems, Federated learning (FL) [5] was introduced by Google's AI research team. It has emerged as the intersection of deep learning and edge computing, where the training dataset remains in the hands of data owners and there is no need to pool data into a single location. Instead, model training is brought to the edge of the network, so that data never leaves the network and only the model is sent to the central coordinator for aggregation as shown in Fig. 1. It enables the clients to learn a model without sharing the raw data and not relying on any trusted third party to hold the data. However, FL depends on the central server/coordinator for aggregating the model. All clients participating in the FL should have a complete unanimity on the role of central server/coordinator as model aggregators. The central server/coordinator may encounter single point of failure which would crash the entire system. We choose the decentralized federated learning framework [6] to overcome this issue. It is a server-less, decentralized approach, where clients communicate directly among themselves without a central server/coordinator as shown in Fig. 2.

Though federated learning keeps the training data private to the local user, still it is vulnerable to various privacy attacks during the training process and causes privacy leakage. There is also a possibility

* Corresponding author.
E-mail addresses: renuga_k@ihpc.a-star.edu.sg (R. Kanagavelu), wei_qingsong@ihpc.a-star.edu.sg (Q. Wei), zengxiang_li@outlook.com (Z. Li), zhang_haibin@ihpc.a-star.edu.sg (H. Zhang), juniarto_samsudin@ihpc.a-star.edu.sg (J. Samsudin), yang_yechao@ihpc.a-star.edu.sg (Y. Yang), gohsm@ihpc.a-star.edu.sg (R.S.M. Goh), sgwang@bupt.edu.cn (S. Wang).

https://doi.org/10.1016/j.array.2022.100207
Received 10 March 2022; Received in revised form 10 June 2022; Accepted 10 June 2022
Available online 18 June 2022

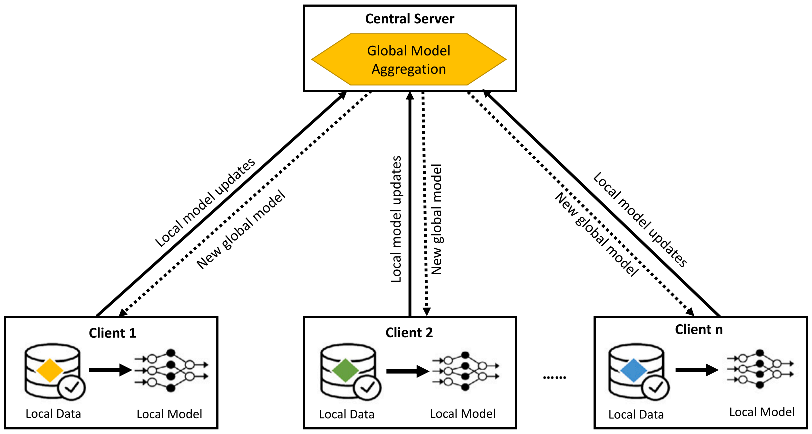


Fig. 1. Central federated learning.

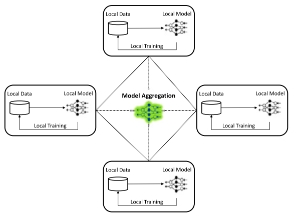


Fig. 2. Decentralized FL.

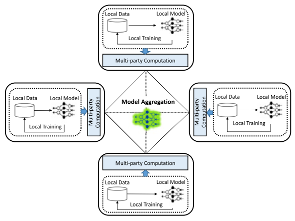


Fig. 3. MPC enabled FL.

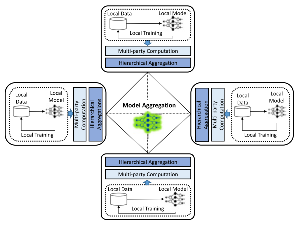


Fig. 4. MPC enabled FL-hierarchical aggregation.

to extract the local user's dataset by reverse engineering the communicated model parameters [7–9]. To ensure confidentiality and provide data privacy guarantee of the model aggregation, privacy-preserving techniques, such as Differential Privacy (DP) [10], secure Multi-party Computation (MPC) [11], and Homo-morphic Encryption (HE) [12,13] can be used together with FL [8].

Secure MPC allows for secure computation among multiple parties to compute a function jointly over their sensitive input data. All participating parties know only the output while keeping those inputs private. The parties can learn only the final output.

Though MPC has the advantage of secure data sharing, it tends to have significant communication and computation overhead when implemented on a large-scale decentralized federated learning system [14]. In the traditional MPC-enabled FL, each client exchanges its secret shares of locally trained model with all other clients in the FL as shown in Fig. 3. As a result, when the number of FL clients increases, the communication overhead also increases in an exponential manner.

These observations motivated us to propose a novel communication-efficient MPC enabled federated learning called CE-Fed, with the objective of reducing the communication overhead and scalability. The key approach of our proposed CE-Fed is to select a few clients as the committee members, who perform the aggregation of the local models of all FL clients in a hierarchical manner using the MPC service. Therefore it avoids sharing the model parameter from each client to everyone else in the FL list. Our proposed CE-Fed is executed in two phases. In the first phase, we group the FL clients that are located close to each other. The local models of all FL clients in the same group are securely aggregated using MPC to form an intra-group model. Based on the latency, one client is elected from each group to form the aggregation committee. In the second phase, the committee members work together to aggregate the inter-group models using MPC, as shown in Fig. 4.

The main contributions of our paper are summarized as follows:

- We quantitatively analysed the communication overhead when secure MPC is integrated into decentralized federated learning with experimental results.
- We propose a hierarchical model aggregation to reduce the communication cost incurred in MPC enabled Federated Learning.
- We propose latency-based committee election algorithm for MPC-based hierarchical model aggregation.
- We demonstrate the effectiveness of our decentralized communication-efficient MPC based federated learning through extensive experiments on various datasets.

The remainder of this paper is organized as follows. Section 2 describes the background and related work. The CE-Fed framework is presented in Section 3. The experimental analysis and performance evaluations are presented in Section 4, followed by conclusions in Section 5.

2. Background and related work

We present the overview of federated learning as well as the Multi-party computation in this section. We also discuss the merits and limitations of the various federated learning research works.

2.1. Federated learning

Prior to the advent of Federated Learning (FL), data has to be uploaded and consolidated at a centralized data-centre or cloud. The machine learning model will be generated from the centralized data and then be deployed for inference and recommendation. A data owner has no control over the data as well as the constructed machine learning model, since both are no longer residing at the premise of the data owner. There is a possibility that the centralized service providers can obtain extra revenue on the data as well as on model and use them for some illegal purposes. This cannot be prevented by data owners.

Federated learning (FL) has emerged as an effective approach that introduced by Google's AI research team in 2017 [15]. The central server send the baseline ML model to clients. Then, each client train its own sub-model using its local data and send the fine-tuned model back to the central server. The central server collects all trained sub-models and aggregates them. Then the aggregated models are sent back to clients out over and over again until they have learnt all there is to learn. Throughout the entire process, raw data are kept on client devices and instead the local models are transferred and shared. The federated model improves its accuracy by aggregating many local models iteratively, taking advantage of complementary data/inputs from a large number of devices.

Federated Learning (FL) has been extended to the collaborations across multiple organizations. It is categorized to horizontal FL and vertical FL [16], based on the data distribution over the sample and feature spaces. In horizontal FL, datasets of different organizations like different industrial organizations, have the same features but different sample sizes. In vertical FL, different organizations like banks, or insurance companies that are located with the same city, have collected data with different features.

Few open-source FL frameworks are available. Google's TensorFlow Federated (TFF) is a lightweight open source framework, a first attempt in the community. It is designed for android users, to predict keyboard next word on their mobile phones [17] using TFF. FATE [18], is the federated learning framework developed by Webank. It supports various federated learning methods. Pysyft [19] is a python library that only supports FedAvg algorithm. It supports MPC and differential privacy. It can either run on stand-alone or on multiple machines and uses web socket API for the communication between different clients. Clara [20] is a framework for building AI accelerated medical imaging workflows. Facebook's privacy preserving machine learning framework is CrypTen [21].

2.2. Privacy-preserving technologies

Secure MPC is a privacy preserving technique that allow for secure computation over sensitive data. It was firstly introduced by Yao in 1986 [22]. Garbled circuits and secret sharing are the two dominated MPC techniques [23] followed today. The secret sharing is a commonly used MPC protocol. It splits the sensitive data into secret shares. The original data is obtained from the combination of these secret shares. The clients cannot learn anything other than final output.

In federated learning, the model parameters and gradients are shared with the server for model aggregation. There is a possibility that the malicious user can intercept the model parameters and could perform reverse engineering to extract the sensitive data, while it is shared with the server on federated learning [24,25]. To address this issue, Multi party computation methods like additive secret sharing [26] or Shamir secret [27] sharing can be used to encrypt the gradients/model updates before performing the aggregation so that no one will be able to see the gradients.

Differential Privacy (DP) is adopted by researchers recently [28] to ensure data security and confidentiality. Data privacy is protected by adding random noise to data. The introduction of noise results in a compromise between security and accuracy. It is not suitable for FL as adding noise to model parameter data from each participating party may affect the accuracy of global model [29].

2.3. Related work

The growing demand for FL has resulted in the use of various techniques and approaches to make its deployment successful. FL can be categorized as centralized FL or Decentralized FL based on the network topology. In centralized FL, a centralized server is responsible for collecting the trained models from participating clients to build a global model and send it back to clients. Google's [30,31] research team proposes federated model averaging algorithm for the model aggregation. It uses a deep learning based on Tensor Flow and supports differential privacy method to enhance privacy guarantees to achieve a local model with minimal communication rounds. Recent research works [32,33] are focused on multi-centre FL with the objective of finding the optimal global model from multiple user groups. Federated Stochastic Expectation Maximization (FedSEM) is proposed by [32] to train multiple global ML models from a multi-cluster environment. Federate transfer learning [34] uses deep learning to address a specific situation where two participants have a part of common samples and only one participant has the label information. To protect model parameters, it uses additively homomorphic encryption. In decentralized FL, there is no need for the centralized server for model aggregation. Each participant shares their local model with their neighbours. Research works [35–37] focussed on peer-to-peer FL framework and addressing the communication delays [36]. Research work [25] focussed on privacy-aware access control mechanism, that enforce access control to privacy sensitive data.

In our earlier work [38,39], we proposed a two-phase MPC-enabled FL framework to improve scalability. We have carried out preliminary work wherein the committee members are selected randomly. Random selection may not be efficient as the parties are geographically distributed which could have impact on latency. Further, we consider MPC enabled FL with hierarchical model aggregation and carry out new performance experiments and analyse the results with different datasets in this work.

3. Motivation

MPC is used for privacy-preserving model aggregation in FL framework [40]. With MPC protocol, each client splits its locally trained model S into n secret shares, one for each of the participating clients. These secret shares are exchanged among the peer clients, with each one holding one share of the every other client's secret. Then, each client locally computes the sum of the one piece of the secret share of the models from all clients. The locally aggregated secret shares at each client are exchanged further and aggregated to reconstruct the model S. The total number of messages exchanged between n clients is proportional to $(n \times (n-1) \times 2)$ which is $O(n^2)$. MPC results in higher communication costs due to the communication and connectivity between each client to every other clients in FL framework. The model aggregation process in traditional MPC enabled FL is illustrated in Fig. 5. We consider six clients *(Node 1, Node 2, Node 3, Node 4, Node 5, and Node 6)* for the illustration. The individual client's local models *(A,B,C,D, E and F)* are privacy-sensitive. They are split into multiple secret shares *[(A* $\rightarrow A_1, A_2, A_3, A_4, A_5, A_6$*), (B* $\rightarrow B_1, B_2, B_3, B_4, B_5, B_6$*), (C* $\rightarrow C_1, C_2, C_3, C_4, C_5, C_6$*),(D* $\rightarrow D_1, D_2, D_3, D_4, D_5, D_6$*), (E* $\rightarrow E_1, E_2, E_3, E_4, E_5, E_6$*), and (F* $\rightarrow F_1, F_2, F_3, F_4, F_5, F_6$*)]*. These secret shares are exchanged among other clients in such a way that each client holds one share of every other client's secret. For example, *Node 1* holds A_1, B_1, C_1, D_1, E_1, F_1. Consequently, each client (*Node 1*) conduct a local aggregation of the secret shares (G1 $\rightarrow A_1 + B_1 + C_1 + D_1 + E_1 + F_1$). These locally aggregated secret shares *G1, G2, G3, G4, G5, and G6* at corresponding client are exchanged and globally aggregated (S $\rightarrow G_1 + G_2 + G_3 + G_4 + G_5 + G_6$) to cancel out the randomness and thus reconstruct the local models of all clients. The total number of messages exchanged between *six* clients is proportional to $(6 \times (6-1) \times 2)$ which is 60.

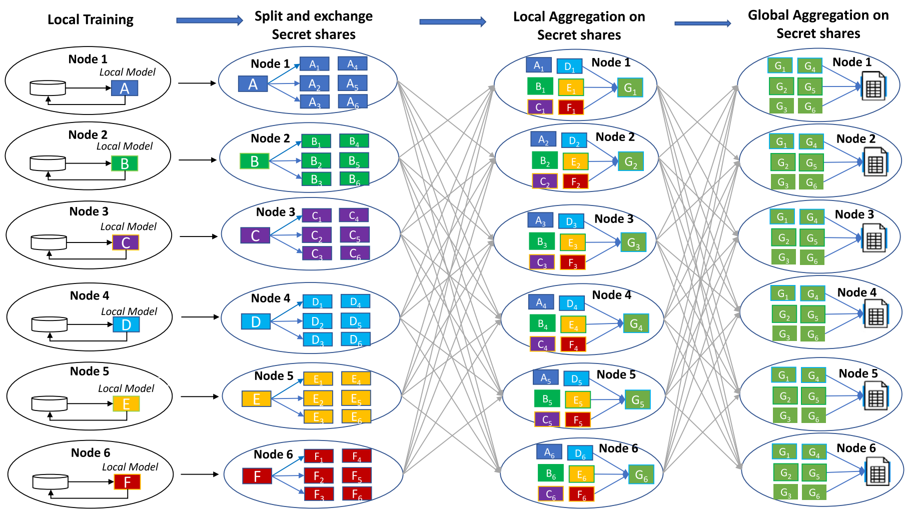

Local Training
Split and exchange Secret shares
Local Aggregation on Secret shares
Global Aggregation on Secret shares
Node 1
Node 2
Node 3
Node 4
Node 5
Node 6
Local Model

Fig. 5. MPC- enabled model aggregation.

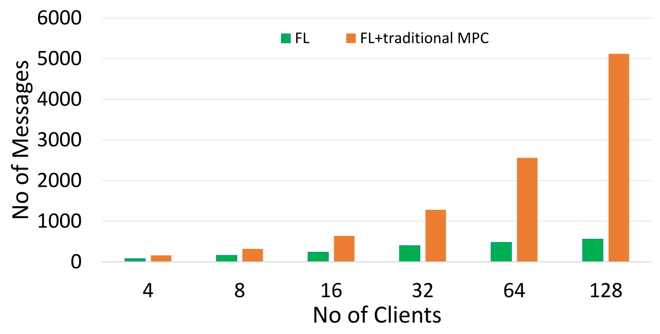


Fig. 6. Communication overhead.

We have carried out a preliminary experimental study to get insights into how the consideration of the number of clients impacts on the communication overhead in privacy preserving FL using traditional MPC. We evaluate the performance on MNIST dataset [41], using the following parameters: total number of clients participated(N) = varying from 4 to 128; Local epoch(E)=10; Learning rate(lr)=0.01; batch size=10; number of communication rounds(T)=50. The communication overhead for two scenarios: FL with traditional MPC and FL without MPC is shown in Fig. 6. It is observed from Fig. 6 that the number of communicated messages in MPC enabled FL increases with increasing number of clients, as each client exchange their secret shares of model parameters with all the other clients during each communication round. The results show that, the number of messages exchanged is 5×–10× more when compared with FL without MPC. These findings motivated us to propose a novel FL framework to reduce the communication cost incurred in traditional MPC enabled Federated Learning.

4. CE-Fed framework

In this section, we describe the proposed CE-Fed framework as shown in Fig. 7, with the objective of reducing the traditional MPC communication cost.

The proposed CE-Fed selects a few clients as the committee members who use MPC service to aggregate the local models of all FL clients in a hierarchical manner. Therefore it avoids sharing the model parameter from each client to everyone else in the FL list. Our proposed CE-Fed is executed in two phases. In the first phase, we group the FL clients that are located close to each other. The local models of all FL clients in the same group are securely aggregated using MPC to form a intra-group model. Based on the latency, one client is elected from each group to form the aggregation committee. In the second phase, the committee members work together to aggregate the inter-group models using MPC. It has three modules:

(1) **Group leader election:** This module selects the group leader based on the latencies from the leader to all the other clients in the same group.
(2) **Model aggregation committee selection:** This module selects the aggregation committee members from group leaders
(3) **Hierarchical model aggregation module:** This module uses committee based approach for global model aggregation

We first discuss the group leader selection and present the terms used in our implementation. Then, we present the aggregation committee members selection method and last we describe how committee based model aggregation helps to reduce the communication overhead. Some important variables used in our framework are described in Table 1.

4.1. *Group leader election*

There are large number of clients involve in FL. The communication between the clients in the decentralized FL poses unique challenges with an increasing number of clients. In a federated learning environment, participating clients can be either located in the same region or geographically far apart in different locations(countries). In a decentralized environment, there requires a strong co-ordination among the clients to achieve better performance. If the participating clients are located at geographically distributed locations, they have higher inter-node latencies, which in turn incurs long delay. It requires the magnitude of hundreds of milliseconds or even seconds to transfer the data across multi-geographic locations. The client with longest delay is the bottleneck which forces all other clients to wait for model aggregation. If this communication lag is not handled properly will affect the scalability of the learning algorithm. To alleviate this, we propose a clustering approach that groups the clients based on their response latency and ensure that all clients in the same group have minimum inter-node latency.

After grouping the clients, one client from the group is elected as a leader. Instead of letting everyone to share secret shares to every other one, we choose one representative from each group to communicate with other groups. This reduces communication overhead. The group

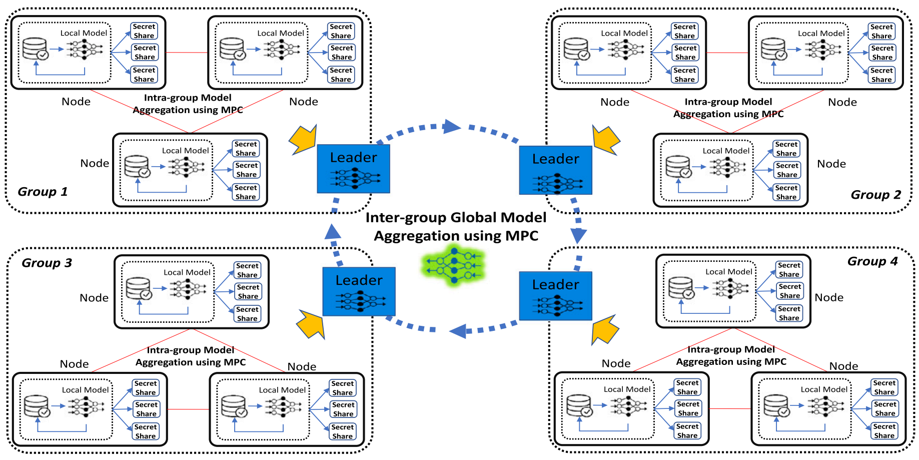

Group 1
Group 2
Group 3
Group 4
Local Model
Secret Share
Node
Intra-group Model Aggregation using MPC
Leader
Inter-group Global Model Aggregation using MPC

Fig. 7. CE-Fed framework.

Table 1
Variables in federated learning framework.

Category	Symbol	Type	Description
Federated learning	n	Int	No. of participating FL clients
	a	Int	Global model training iterations
	b	Int	Local model training rounds
	$bSize$	Int	Batch Size of each communication round
Committee	m	Int	Elected committee members
	C	Array	committee members List
CNN model	T	Tensor	Local model parameters
	W	Tensor	Parameters/weights of aggregated model
	s	Integer	The size of parameters/weights of local/aggregated models
Network	G	Int	Number of mutually exclusive Groups
	k	Int	Number of clients in group
	SG	Array	Group members List
	L	Int	inter-node latency
	Q	Array	Mutually exclusive Group Leaders List

leader is selected based on the average latency with the other clients. Each client is assumed to be at fixed position and the estimated latency between any pair of clients is known in advance. The elected group leader may have a chance to participate in the global model aggregation. There is only one leader in the group and rest of them are his followers. The leader sends *Alive* messages to its followers at frequent intervals. If there is no *Alive* message from the leader within a particular time interval, the client with minimal inter-node latency to the other followers will be elected as a new leader. The pseudo code of group formation and leader election is shown in Algorithm 1. Our algorithm considers many mutually exclusive groups with varying number of clients.

Algorithm 1 Group Leader Election

```
1: Function{Leader Selection(G,k, m,C)}
2: Group Initialization
3: Read the value
4: for $i \in [1, G]$ do
5:     $min \leftarrow Lat(SG[1])$
6:     for $j \in [1, k]$ do
7:         if $L_j < min$ then
8:             $Leader = j$
9:         end if
10:        $Q \leftarrow Leader$
11:    end for
12: end for
13: EndFunction
```

4.2. Aggregation committee selection

To further reduce the communication overhead, the subset of group leaders from different groups will be elected to form the aggregation committee. The committee members jointly working together to securely aggregate the different group local models using the MPC service. There are several ways to form committee for secure model aggregation. When all clients are placed in a close proximity to each other in a local area setting, the committee members can be selected based on the static rules [42,43]. This is not applicable when the clients are located in different geographical sites, where different links in the system have significantly difference latencies. Our committee selection is based on a latency where subset of group leaders (m out of n group leaders) are chosen as members with low latencies (inter-node latency) to other groups that increases the efficiency of the committee selection. The elected leaders continuously communicate with other group leaders and monitor their latencies with them. We consider the aggregation committee with m members (($m \geq 3$)), based on the assumption that there is no collusion among the committee members.

4.3. MPC enabled hierarchical model aggregation

The model aggregation is happening in two phases as shown in Fig. 8:

- Intra-group model aggregation and
- Inter-group aggregation.

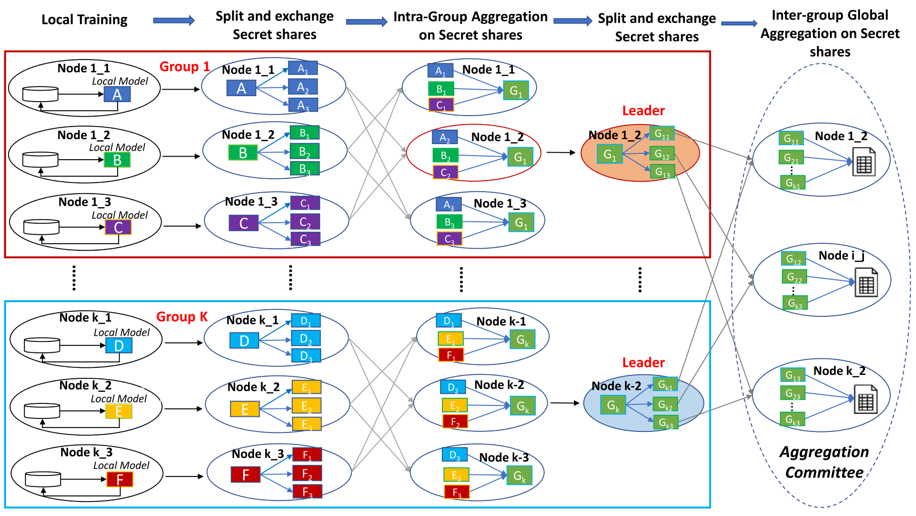

Local Training
Split and exchange Secret shares
Intra-Group Aggregation on Secret shares
Split and exchange Secret shares
Inter-group Global Aggregation on Secret shares
Group 1
Group K
Leader
Aggregation Committee
Node 1_1
Node 1_2
Node 1_3
Node k_1
Node k_2
Node k_3
Node i_j
Local Model

Fig. 8. CE-Fed Hierarchical Model Aggregation.

4.3.1. *Intra-group model aggregation*

During the training process, all the clients in the same group start training their models locally using their local datasets. Each client in a group has two main tasks:

- Local training on its local data : After extracting the features from raw data, the client's data analytics module performs local training on its local data.
- MPC Enabled intra-group Model Aggregation : The locally trained model parameters are split into secret shares and securely aggregated using MPC protocol.

Initially, each client locally trains their local dataset for few iterations, till the convergence criterion is met. After completing the local training, the clients in the group collaboratively learn the model in a peer-to-peer fashion. As there is no central server available to initiate the model aggregation process in the decentralized environment, any random client A_i in the group initiates that process. The tensor **T** of individual local model parameters/weights are privacy-sensitive. To preserve privacy, MPC secret sharing technique is used to securely aggregate the tensor **T**, without sacrificing accuracy. The tensor **T** is split into multiple tensors as the secret shares based on the number of clients in the same group. Each client holds one share and exchanges the remaining shares with all the other participating clients in the same group. Then, clients perform secure model aggregation of the secret shares using MPC to obtain their intra-group model. After learning a good intra-group model, the group leader sent their group's intra-group model to the model aggregation committee for global collaborative learning. The pseudo code of intra-group model aggregation is shown in Algorithm 2.

Algorithm 2 Intra-group Model Aggregation

```
1:  Function{Intra-Group.Model.aggregation(n, u, v, m)}
2:  for $n \in [1,u]$ do
3:      for $a \in [1,v]$ do
4:          for $b \in [1,z]$ do
5:              $T \leftarrow localtraining$
6:          end for
7:          for $t \in [1,u]$ do
8:              Split locally-trained model into $u$ secret shares)
9:              share it with members of aggregation committee
10:         end for
11:     end for
12:     $S = sumthesecretshares$
13:     for $a \in [1,u] \& a \neq x$ do
14:         Broadcast $S$ to peer clients
15:         Receive $S$ from peer clients
16:     end for
17:     $W \leftarrow Intra-groupAggregatedModel$
18: end for
```

The communication cost of intra-group model aggregation is estimated based on the number of messages exchanged between the clients. It can be calculated by adding the number of messages exchanged between clients in all groups. The variables used in our calculations are described in Table 1. The number of messages denoted as $Intra\_Grp\_Msg\_Num_i$ exchanged in the ith group during intra-group model aggregation is calculated as follows:

$$Intra\_Grp\_Msg\_Num_i = (k_i \times (k_i - 1)) \times 2 \times a \quad (1)$$

The size of messages denoted as $Intra\_Grp\_Msg\_Size_i$ exchanged in the ith group during intra-group model aggregation is given by

$$Intra\_Grp\_Msg\_Size_i = Intra\_Grp\_Msg\_Num_i \times s \quad (2)$$

The total number of messages denoted as $Total\_Intra\_Msg\_num$ exchanged in the intra-group model aggregation among all groups(G) is given by

$$Total\_Intra\_Msg\_Num = \sum_{i=1}^{G} Intra\_Grp\_Msg\_Num_i \quad (3)$$

The size of messages denoted as $Total\_Intra\_Msg\_Size$ exchanged in the intra-group model aggregation among all groups is given by

$$Total\_Intra\_Msg\_Size = \sum_{i=1}^{G} Intra\_Grp\_Msg\_size_i \quad (4)$$

4.3.2. *Inter-group model aggregation*

The optimized intra-group models from different groups are finally aggregated by the model aggregation committee members to form the inter-group global model. Each aggregation committee member splits the intra-group model received from different group leaders into m secret shares using MPC. It holds one share and exchanges the remaining shares with all the other committee members. The committee members work together to perform inter-group global model secure aggregation using MPC and broadcast the aggregated global model to all the group leaders in various groups. Then, the group leaders broadcast the inter-group model to their followers in their group. The whole process of intra-group model aggregation and inter-group model aggregation are repeated until the model converges. The pseudo code of inter-group model aggregation is shown in Algorithm 3.

The communication cost of inter-group model aggregation is estimated based on the number of messages exchanged between the model

Algorithm 3 Inter-group Model Aggregation

```
1: Function{Inter-group.Model.aggregation( Q, m, n)}
2: for $Q \in [1, m]$ do
3:     $T \leftarrow Intra-groupModel$
4: end for
5: for $Q \in [1, m]$ do
6:     Split Intra-group model into $m$ secret shares
7:     share it with members of aggregation committee
8: end for
9: for $x \in [1, m]$ do
10:     S=sum the secret shares
11:     for $n \in [1, m] \& a \neq x$ do
12:         Broadcast $S$ to peer clients
13:         Receive $S$ from peer clients
14:     end for
15:     $W \leftarrow GlobalAggregatedInter-GroupModel$
16: end for
17: return $W$
18: EndFunction
```

aggregation committee members. The number of messages denoted as $Inter\_Grp\_Msg\_Num$ exchanged in the inter-group model aggregation among m aggregation committee members can be calculated as follows:

$$Inter\_Grp\_Msg\_Num = (m \times (m-1)) \times 2 \times a \tag{5}$$

The size of messages denoted as $Inter\_Grp\_Msg\_Size$ exchanged in the intra-group model aggregation is

$$Inter\_Grp\_Msg\_Size = Inter\_Grp\_Msg\_Num \times bSize \tag{6}$$

The hierarchical model aggregation reduces the number of shares exchanged across all participating clients, because secret shares are shared only with the elected committee members m, where m ≪ n. The total number of messages exchanged between clients is denoted as Total_Msg_Num and is calculated as follows:

$$Total\_Msg\_Num = Total\_Intra\_Msg\_Num + Inter\_Grp\_Msg\_Num \tag{7}$$

The total number of messages exchanged is proportional to $O(k^2) + O(m^2)$, is very few when compared with traditional MPC enabled FL method [O(n^2)]where k, m ≪ n.

5. Experiment and results

5.1. Experimental setup

To implement and evaluate the effectiveness of our proposed CE-Fed framework, we used PyTorch 1.2.0 and Python 3.74 as machine learning library. We considered three public image datasets, MNIST [44], CIFAR-10 [45] and Fashion-MNIST dataset [41]. We used the CNN model that consists of two 5 × 5 convolution layers. The first layer has 32 channels and the second one has 64 channels. Each layer is followed with 2 × 2 max pooling, with ReLu activation, and a final softmax output layer. The model has about 600,000 trainable parameters. In our experiments, we study the performance and effectiveness in using IID (Independent and identically distributed) and non IID datasets. In the IID distribution, the data is shuffled and partitioned across all the clients, whereas in the non-IID distribution, first the data is sorted by the label and then partitioned across the clients in such a way that each party has fixed number of labels and there is no overlap between samples of different clients. To eliminate the randomness caused by client sampling, all clients are participated in each and every round of training. The learning rate is set to 0.01, local epoch number is set to 5 and batch size is set to 64. The stochastic gradient descent (SGD)is used as optimizer. We consider 4 groups and use *FedAvg* for model aggregation. We use Accuracy and communication cost as performance metrics in our experiments.

5.2. Communication cost

The communication cost is evaluated in terms of the number of the shares(messages) as well as the size of messages communicated between the clients with three different datasets: MNIST,CIFAR-10 and fashion-MNIST as shown in Fig. 9, Fig. 10 and Fig. 11. We compare the performance of our proposed CE-Fed with traditional MPC enabled FL framework (FL+traditional MPC), traditional FL (FL) and two-phase MPC enabled FL(Two-phase) [38].

5.2.1. MNIST dataset

We use two different data distributions of MNIST over clients. In Independent and Identically Distributed (IID) data settings, each client has balanced and identical data points. For IID data distribution, we first shuffle the data and equally divide among the clients. However, in practical scenarios, the datasets at clients are more likely to be non-uniform and unbalanced(non-IID). For non-IID data heterogeneous distribution, we distribute the data among the clients with different proportions.

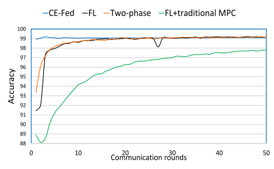
The accuracy over communication rounds for IID and non-IID are shown in Fig. 9(a) and Fig. 9(d). It takes significant more rounds (around 10×) to reach 98% test accuracy in non-IID than in IID cases, which indicates that non-IID data indeed can slow down the convergence compared with IID cases. It also observed that the accuracy is affected by non-IID data heterogeneity.

In the traditional MPC enabled FL, each client shares its secret share with every other client to reconstruct the global joint model. Consequently, the number of secret shares exchanged is directly proportional to the number of clients in the MPC enabled FL which is $0(n^2)$. It results in lots of communication overhead as analysed in Section 2.1. In the traditional FL, each client shares its model parameters with every other client for global model aggregation. The number and the size of messages increase with the increasing number of clients. In the two-phase MPC enabled FL [38], all the participating clients form a single group and committee members are selected randomly from that group for global model aggregation. The number and the size of messages are much lower than that in MPC enabled FL. However, in our proposed FL with aggregation committee method(CE-Fed), multiple groups are formed based on the client's geographical locations and one leader is elected from each group to participate in the global model aggregation. All clients share their secret shares only with the committee members. It reduces the number of messages exchanged significantly by an average of 90% in both IID and non-IID cases as shown in Fig. 9(b) and Fig. 9(e) respectively. It thus demonstrates that our proposed method has much lower communication cost and better scalability. We observe that the more messages, $\tilde{1}0\times$, are exchanged in the non-IID case than in the IID case as the non-IID takes more communication rounds to achieve maximum test accuracy than IID cases as shown in Fig. 9(a) and Fig. 9(d).

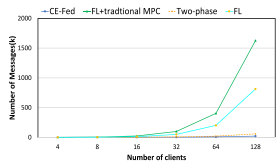
Another metric we consider to evaluate the communication cost is the size of messages exchanged between clients and committee members. Large number of messages exchanged between clients and committee members results in large amount of data transferred between them. We conduct the experiment by varying the number of clients from 4 to 128. It has been observed that our proposed CE-Fed significantly reduces the communication cost by an average of 90% in both IID and non-IID distributions as shown in Fig. 9(c) and Fig. 9(f).

5.2.2. CIFAR-10 dataset

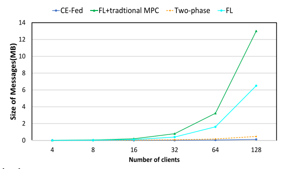
To further evaluate the effectiveness of our proposed FL, we run experiments on CIFAR-10 datasets. For IID setting, we partitioned the 50,000 training examples equally among the clients. For non-IID settings, we shared the dataset unevenly and limit the number of classes to 2 per client. The accuracy over communication rounds for IID and non-IID are shown in Fig. 10(a) and Fig. 10(d). It takes significant more



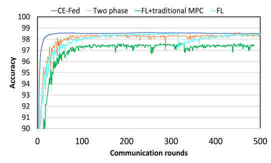
(a) Accuracy - MNIST IID



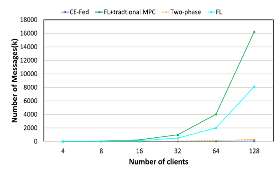
(b) No of Messages - MNIST IID



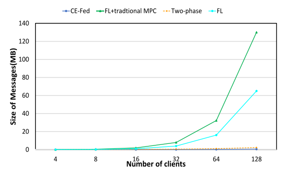
(c) Size of Messages - MNIST IID



(d) Accuracy- MNIST Non-IID

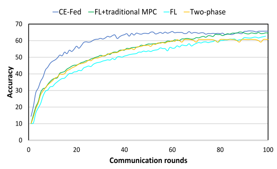


(e) No of Messages - MNIST Non-IID

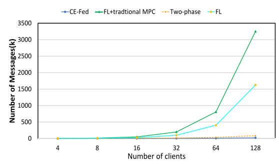


(f) Size of Messages - MNIST Non-IID

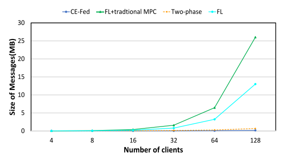
Fig. 9. MNIST dataset.



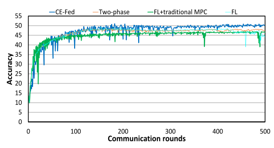
(a) Accuracy - CIFAR-10 IID



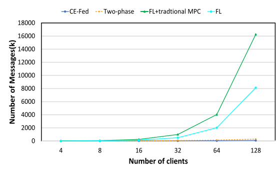
(b) No of Messages - CIFAR-10 IID



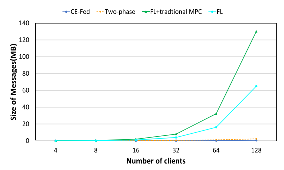
(c) Size of Messages - CIFAR-10 IID



(d) Accuracy- CIFAR-10 Non-IID



(e) No of Messages - CIFAR-10 Non-IID



(f) Size of Messages - CIFAR-10 Non-IID

Fig. 10. CIFAR-10 dataset.

rounds to converge when compared with MNIST dataset both in non-IID case, which indicates that non-IID has stronger data heterogeneity.

There is a rapid increase in the number of shares exchanged by an average of 80%–90% with the increasing number of clients in the MPC enabled FL when compared with our proposed FL with committee selection as shown in Fig. 10(b), and Fig. 10(e). This shows that the traditional MPC enabled FL is not scalable when more clients joined the FL. It proves that our method is scalable to support more clients participated in FL. Our proposed CE-Fed reduces the communication cost by an average of 80%–90% when compared with Two-phase MPC enabled FL method [38].

CIFAR-10 and MNIST datasets have the similar observations when consider the size of messages exchanged between clients. It has been observed that our CE-Fed method reduces the communication cost significantly with increasing number of clients by an average of 80%–90% in both IID and non-IID distributions as shown in Fig. 10(c) and Fig. 10(f). It proves that our proposed CE-Fed method is more efficient in incurring low communication overhead.

5.2.3. *Fashion-MNIST dataset*

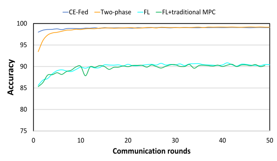
The effectiveness of our proposed FL is studied on Fashion-MNIST dataset. For IID data distribution, we first shuffle the data and equally distribute among the clients. For Non-IID data, the dataset is partitioned into 200 shards of 300 samples, and each client randomly picks two shards. The accuracy for IID and Non-IID Fashion-MNIST (FMNIST) datasets are shown in Fig. 11(a) and Fig. 11(d) and it is close to accuracy of MNIST dataset. The communication cost in terms of number of messages are shown in Fig. 11(b) and Fig. 11(e). Our proposed CE-Fed method reduces the number of messages exchanged by an average of 80%–90% in both IID and non-IID cases. It shares the similar observations with MNIST datasets.

The communication cost results in terms of size of messages are shown in Figs. 11(c) and 11(f). It has been observed that in traditional MPC enabled FL, the communication cost significantly increases with increasing number of clients in both IID and non-IID distributions. It shares the same observations with MNIST and CIFAR-10 datasets.

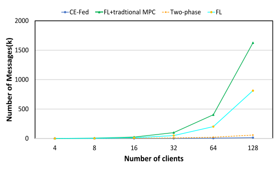
From the above experimental analysis, it has been observed that our proposed CE-Fed reduces the communication cost significantly with increasing number of clients.

5.3. *Accuracy*

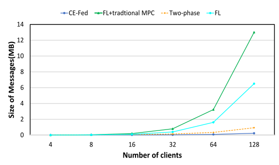
To study the effectiveness of our proposed FL, the accuracy score of the proposed CE-Fed method is compared with centralized training and local training as shown in Fig. 12. In local training, models are trained



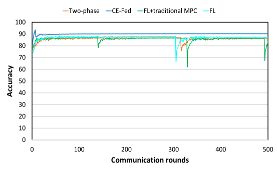
(a) Accuracy - Fashion-MNIST IID



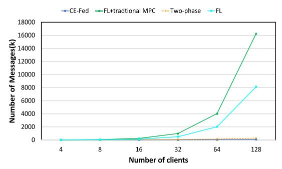
(b) No of Messages - Fashion-MNIST IID



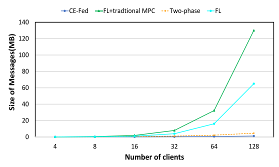
(c) Size of Messages - Fashion-MNIST IID



(d) Accuracy- Fashion-MNIST Non-IID



(e) No of Messages - Fashion-MNIST Non-IID



(f) Size of Messages - Fashion-MNIST Non-IID

Fig. 11. Fashion-MNIST dataset.

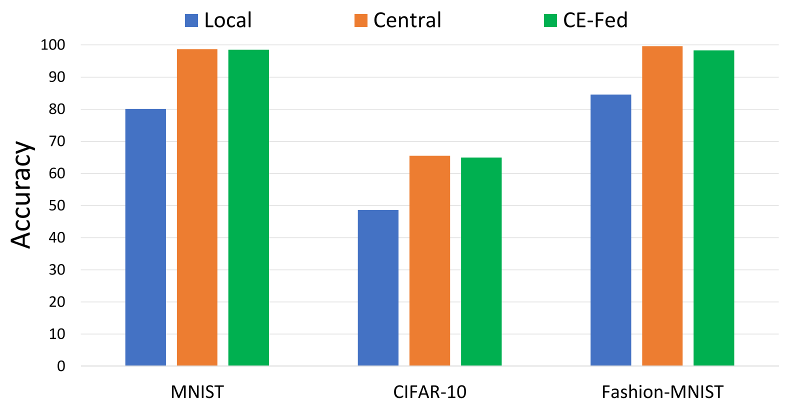


Fig. 12. Accuracy.

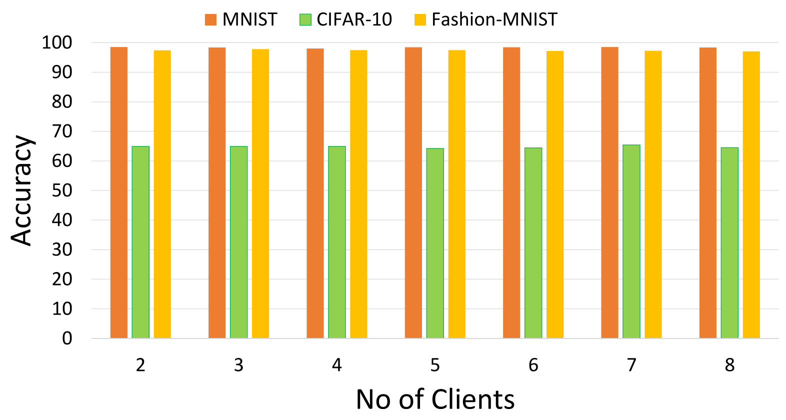


Fig. 13. Accuracy (Varying no of clients)- Comparison.

using local datasets of individual clients. Centralized training trains the model on centralized data. Experiment results shows that the test accuracy of federated learning is comparable with centralized learning and outperforms local training. It has been observed that FL training achieves an improvement in accuracy of 22%, 33%, 17% on MNIST, CIFAR-10 and Fashion-MNIST respectively when compared with local training. The key reason for the low accuracy in local training is that the size of the local dataset is not large enough.

Fig. 13 shows the accuracy score of proposed CE-Fed over MNIST, CIFAR-10 and Fashion MNIST datasets by varying the number of clients. It has been observed that the proposed CE-Fed method reaches the stable accuracy even when more and more clients joined the system. This shows that our proposed method is scalable.

6. Conclusion

In this paper, We proposed and implemented a CE-Fed framework with hierarchical MPC based model aggregation. It supports multiple parties to collectively learn a machine learning model in a privacy preserving manner. Our proposed hierarchical model aggregation reduces the communication cost incurred in MPC protocols and also aggregates the models in a privacy-preserving manner without compromising the accuracy. The effectiveness of our proposed CE-Fed framework was demonstrated on various datasets. The above experiments indicates that our proposed CE-Fed is able to reduce the communication cost significantly while achieve the similar accuracy and privacy.

CRediT authorship contribution statement

Renuga Kanagavelu: Conception and design of study, Writing – original draft. **Qingsong Wei:** Conception and design of study, Writing – original draft. **Zengxiang Li:** Conception and design of study. **Haibin Zhang:** Revising the manuscript critically for important intellectual content. **Juniarto Samsudin:** Revising the manuscript critically for important intellectual content. **Yechao Yang:** Revising the manuscript critically for important intellectual content. **Rick Siow Mong Goh:** Conception and design of study. **Shangguang Wang:** Conception and design of study.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

All authors approved the final version of the manuscript.

References

[1] Ding C, Zhou A, Liu L, Ma X, Wang S. Resource-aware feature extraction in mobile edge computing. IEEE Trans Mob Comput 2020.
[2] Lian Xiangru, Zhang Ce, Zhang Huan, Hsieh Cho-Jui, Zhang Wei, Liu Ji. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent In: NIPS. 2017.
[3] Ren H, Li H, Dai Y, Kan Y, Lin X. Querying in internet of things with privacy preserving: Challenges, solutions and opportunities. IEEE Netw 2018;32(6):144–51.
[4] Garcia Y, Kassa M, Cuevas A, Cebrian M, Moro E, Rahwan I, Cuevas R. Analyzing gender inequality through large-scale facebook advertising data. Proc Natl Acad Sci 2018;115(27):69586963.
[5] McMahan B, Ramage D. Federated learning: Collaborative machine learning without centralized training data. 2017, https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Google AI Blog.
[6] Roy bhijit Guha, Siddiqui Shayan, Polsterl Sebastian, Navab Nassir, Wachinger Christian. BrainTorrent: A peer-to-peer environment for decentralized federated learning. 2019, https://arxiv.org/pdf/1905.06731.pdf.
[7] Shokri R, Stronati M, Song C, Shmatikov V. Membership inference attacks against machine learning models. In: Proceedings of IEEE symposium on security and privacy. 2017.
[8] Nasr M, Shokri R, Houmansadr A. Comprehensive privacy analysis of deep learning: stand-alone and federated learning under passiveand active white-box inference attacks. In: Proceedings of IEEE ACM symposium on security and privacy. 2019.
[9] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K. Practical secure aggregation for privacypreserving machine learning. In: Proceedings of ACM computer and communications security. 2017.
[10] Alexandra W, Altman M, Bembenek A, Bun M, Gaboardi M, Honaker J, Nissim K, OBrien R, Steinke T, Vadhan S. Differential privacy: A primer for a non-technical audience. Vanderbilt J Entertain Technol Law 2018;21(1):209–75.
[11] Evans D, Kolesnikov V, Rosulek M. A pragmatic introduction to secure multiparty computation. NOW Publishers; 2018.
[12] Chillotti I, Georgieva M, Izabachne M. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: Advances in cryptology, Part I. Vol. 10624. Lecture notes in computer science, 2017, p. 377–408.
[13] Phong LT, Aono Y, Hayashi T, Wang L, Moriai S. Privacy preserving deep learning via additively homomorphic encryption. IEEE Trans Inf Forensics Secur 2018;13(5):1333–45.
[14] Ivan D, Ishai Y, Mikkel K. Perfectly secure multiparty computation and the computational overhead of cryptography. In: Proceedings of the 29th annual international conference on theory and applications of cryptographic techniques. 2010.
[15] McMahan B, Ramage D. Federated learning: Collaborative machine learning without centralized training data. 2017, https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Google. AI Blog.
[16] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. ACM Trans Intell Syst Technol (TIST) 2019.
[17] Tensor flow federated. 2020, Retrieved from https://www.tensorflow.org/federated/. Last [Accessed July 2020].
[18] Webank FATE (Federated AI Technology enabler). 2020, Retrieved from https://github.com/FederatedAI/FATE. Last [Accessed August 2020].
[19] OpenMined/PySyft-A library for encrypted, privacy preserving deep learning. 2020, Retrieved from https://github.com/OpenMined/PySyft. Last [Accessed August 2020].
[20] Softwareguard-extensions. 2020, Retrieved from https://en.wikipedia.org/wiki/Software_Guard_Extensions. Last [Accessed August 2020].
[21] CrypTen documentation. 2020, https://crypten.readthedocs.io/en/latest/. Last [Accessed August 2020].
[22] Yao AC. How to generate and exchange secrets. In: 27th annual symposium on foundations of computer science. IEEE; 1986, p. 162–7.
[23] Choi Joseph, Butler Kevin. Secure multiparty computation and trusted hardware Examining adoption challenges and opportunities. Secur Commun Netw 2019;1–28. http://dx.doi.org/10.1155/2019/1368905.
[24] Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic counter measures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. 2015, p. 1322–33, [Online]. Available: http://dx.doi.org/10.1145/2810103.2813677.
[25] Ni Q, Bertino E, Lobo J, Brodie C, Karat C-M, Karat J, Trombetta A. Privacy-aware role-based access control. ACM Trans Inf Syst Secur 2010;13(3).
[26] Dahl Morten. Secret Sharing, Part 1 - Distributing trust and work, https://mortendahl.github.io/2017/06/04/secret-sharing-part1/.
[27] Shamir A. How to share a secret. Commun ACM 1979;22(11):612–3.
[28] Choudhury livia, Gkoulalas-Divanis Aris, Salonidis Theodoros, Sylla Issa, Park Yoonyoung, Hsu Grace, Das Amar. Anonymizing data for privacy preserving federated learning. 2020, arXiv preprint arXiv:2002.09096.
[29] Truex Stacey, Baracaldo Nathalie, Anwar Ali, Stein-ke Thomas, Ludwig Heiko, Zhang Rui, Zhou Yi. A hybrid approach to privacy-preserving federated learning. In: Proceedings of the 12th ACM workshop on artificial intelligence and securi-ty, AISec@CCS. 2019.
[30] Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, KBonawitz, Charles Z, Cormode G, Cummings R, et al. Advances and open problems in federated learning. 2019, arXiv preprint arXiv:1912.04977.
[31] Mansour Y, Mohri M, Ro J, Suresh AT. Three approaches for personalization with applications to federated learning. 2020, arXiv pre print:2002.10619.
[32] Xie M, Long G, Shen T, Zhou T, Wang X, Jiang J. Multi-center federated learning. 2020, arXiv:2005.01026.
[33] Xie Ming, Long Guodong, Shen Tao, Zhou Tianyi, Wang Xianzhi, Jiang Jing. Multi-center federated learning, arXiv pre print: arXiv:2005.01026.
[34] Liu Yang, Chen Tianjian, Yang Qiang. Secure federated transfer learning. IEEE Intell Syst 2020;35(4):70–82.
[35] Daily J, Vishnu A, Siegel C, Warfel T, Amatya V. GossipGraD: Scalable deep learning using gossip communication based asynchronous gradient descent. 2018, arXiv:1803.05880.
[36] Wang J, Sahu AK, Yang Z, Joshi G, Kar S. MATCHA: Speeding updecentralized SGD via matching decomposition sampling. 2019, arXiv:1905.09435.
[37] Lalitha A, Kilinc OC, Javidi T, Koushanfar F. Peer-to-peer federated learning on graphs. 2019, arXiv:1901.11173.
[38] Kanagavelu Renuga, Li Zengxiang, Samsudin Juniarto, Yang Yechao, Yang Feng, Goh Rick Siow Mong, Cheah Mervyn, Wiwatphonthana Praewpiraya, Akkarajitsakul Khajonpong, Wang Shangguang. Two-phase multi-party computation enabled privacy-preserving federated learning. In: IEEE CCGRID 2020. p. 410–9.
[39] Kanagavelu Renuga, Li Zengxiang, Samsudin Juniarto, Hussain Shaista, Yang Feng, Yang Yechao, Mong Rick Siow, Cheah GohMervyn. Federated learning for advanced manufacturing based on industrial IoT data analytics, pp 143-176, Implementing Industry 4.0.
[40] Zyskind G, Nathan O, Pentland A. Enigma: Decentralized computation platform with guaranteed privacy. https://arxiv.org/abs/1506.03471.
[41] Fashion-MNIST Dataset, https://github.com/zalandoresearch/fashion-mnist.
[42] Aguilera MK, Delporte-Gallet C, Fauconnier H, Toueg S. Stable leader election. In: Proceedings of the 15th international conference on distributed computing. Springer-Verlag; 2001, p. 108–22.
[43] Aguilera MK, Delporte-Gallet C, Fauconnier H, Toueg S. Communication-efficient leader election and consensus with limited link synchrony. In: Proceeding of the 23rd annual ACM symposium on principles of distributed computing. St.John's, Newfoundland, Canada, ACM Press; 2004, p. 328–37.
[44] MNIST Dataset, https://csc.lsu.edu/saikat/n-mnist/.
[45] CIFAR-10 Dataset, https://www.cs.toronto.edu/kriz/cifar.html.

Renuga Kanagavelu received the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2015. She is a scientist at the Institute of High-Performance Computing, A*STAR, Singapore. Her research interests include Federated Learning, Privacy-preserving techniques, Industrial IoT and Edge Analytics.

Qingsong Wei received the Ph.D. degree in computer science from the University of Electronic Science and Technologies of China, in 2004. He was with Tongji University as an assistant professor from 2004 to 2005. He is a Group Manager and senior research scientist at the Institute of High-Performance Computing, A*STAR, Singapore. His research interests include decentralized computing, federated learning, high performance computing, emerging non-volatile memory and storage system. He is a senior member of the IEEE.

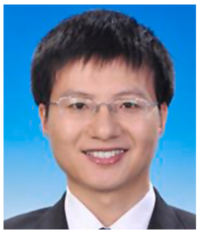
Zengxiang Li is Executive Vice President of Digital Research Institute and Director of Collaborative Intelligence Lab, ENN Group, Beijing, China. His research group is focusing on Industrial IoT, Blockchain and Federate Learning research and development for trusted ecosystem across multiple industry domains, including smart energy, healthcare, and city governance application domains. He completed his Ph.D. in Nanyang Technological University, Singapore in 2010.

Haibin Zhang received the bachelor's degrees of Computer Science and Animal Science from Shandong Agriculture University of China, in 2010. He was with NEC Corp and Alibaba Group as solution engineers to lead the team for researching and development from 2018 to 2020. He is a research engineer at the Institute of HighPerformance Computing, A*STAR Singapore. His research interests include federated learning, blockchain, high-performance computing and distributed storage system.

Juniarto Samsudin is a research engineer at the Institute of High-Performance Computing, A*STAR, Singapore. He got his master's degree in Smart Product Design from Nanyang Technological University, Singapore, 2007. His current projects include federated learning, HPC, IoT and cloud infrastructure and objectdetection using deep-learning.

Yechao Yang received the bachelor's degree in mechanical and electronic engineering from the University of Science and Technology of China, in 1998. He is a research engineer in Institute of High-Performance Computing (IHPC), A*STAR, Singapore. His research interests include Blockchain, IoT, and Federated Learning.

Rick Siow Mong Goh is the Director of the Computing Science Department at the A*STAR Institute of High-Performance Computing (IHPC). He received his Ph.D. degree in electrical and computer engineering from the National University of Singapore. At IHPC, he leads a team of more than 60 scientists in performing worldleading scientific research, developing technology to commercialization, and engaging and collaborating with industry. The research focus areas include artificial intelligence (AI), high performance computing (HPC), distributed computing, machine & deep learning, complex systems, human–machine interaction, and modelling and simulation.



Shangguang Wang is a Professor at the School of Computing, Beijing University of Posts and Telecommunications, China. He is a Vice-Director of the State Key Laboratory of Networking and Switching Technology. He has published more than 150 papers, and his research interests include service computing, cloud computing, and mobile edge computing. He served as General Chairs or TPC Chairs of IEEE EDGE 2020, IEEE CLOUD 2020, IEEE SAGC 2020, IEEE EDGE 2018, and IEEE ICFCE 2017, etc., and Vice-Chair of IEEE Technical Committee on Services Computing (2015–2018). He is currently serving as Executive Vice-Chair of IEEE Technical Committee on Services Computing (2021–), and Vice-Chair of IEEE Technical Committee.