Efficient and Privacy-preserving Distributed Learning in Cloud-Edge Computing Systems

Yili Jiang
University of Nebraska-Lincoln
Lincoln, NE, USA
yilijiang@huskers.unl.edu

Kuan Zhang
University of Nebraska-Lincoln
Lincoln, NE, USA
kuan.zhang@unl.edu

Yi Qian
University of Nebraska-Lincoln
Lincoln, NE, USA
yi.qian@unl.edu

Rose Qingyang Hu
Utah State University
Logan, UT, USA
rose.hu@usu.edu

ABSTRACT

Machine learning and cloud computing have been integrated in diverse applications to provide intelligent services. With powerful computational ability, the cloud server can execute machine learning algorithm efficiently. However, since accurate machine learning highly depends on training the model with sufficient data. Transmitting massive raw data from distributed devices to the cloud leads to heavy communication overhead and privacy leakage. Distributed learning is a promising technique to reduce data transmission by allowing the distributed devices to participant in model training locally. Thus a global learning task can be performed in a distributed way. Although it avoids to disclose the participants' raw data to the cloud directly, the cloud can infer partial private information by analyzing their local models. To tackle this challenge, the state-of-the-art solutions mainly rely on encryption and differential privacy. In this paper, we propose to implement the distributed learning in a three-layer cloud-edge computing system. By applying the mini-batch gradient decent, we can decompose a learning task to distributed edge nodes and participants hierarchically. To improve the communication efficiency while preserving privacy, we employ secure aggregation protocol in small groups by utilizing the social network of participants. Simulation results are presented to show the effectiveness of our proposed scheme in terms of learning accuracy and efficiency.

CCS CONCEPTS

• **Security and privacy** → *Mobile and wireless security.*

KEYWORDS

privacy, efficiency, distributed learning, Cloud-Edge computing

WiseML '21, June 28-July 2, 2021, Abu Dhabi, United Arab Emirates

ACM ISBN 978-1-4503-8561-9/21/06…$15.00
https://doi.org/10.1145/3468218.3469044

ACM Reference Format:
Yili Jiang, Kuan Zhang, Yi Qian, and Rose Qingyang Hu. 2021. Efficient and Privacy-preserving Distributed Learning in Cloud-Edge Computing Systems. In *3rd ACM Workshop on Wireless Security and Machine Learning (WiseML '21), June 28-July 2, 2021, Abu Dhabi, United Arab Emirates.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3468218.3469044

1 INTRODUCTION

The past decade has witnessed the development of machine learning and cloud computing in providing intelligence for smart applications [10, 13]. By collecting raw data from the Internet-of-Things (IoT) devices, cloud server executes machine learning algorithms to train the data and performs data prediction. However, the growing number of IoT devices has generated tremendous amount of data. Transmitting massive data to centralized machine learning has significant communication overhead and computational complexity. Therefore, distributed learning is proposed as a promising approach to provide reliable data processing while reducing communication and computational costs [2, 6, 14]. Specifically, the cloud server assigns a global machine learning model to the distributed participants. Those participants perform data training locally using their IoT devices (e.g., smart phone, laptop, tablet, and so on.) and upload their updated models/parameters to the cloud server. The cloud server then aggregates the updated information and corrects the global model. The same procedure is repeated until an accurate global model is obtained. Since the participants are unnecessary to upload their raw data to the centralized server, the risks of privacy leakage are reduced [5, 7].

However, in conventional distributed learning, attackers can launch white-box attack to infer the raw data by analyzing the updated machine learning models from the participants [3]. To tackle this challenge, the state-of-the-art solutions mainly depend on differential privacy (DP) and encryption techniques. On the one hand, differential privacy [16] is deployed to add random noise to the updated models when they are transmitted from the distributed participants to the centralized cloud server. Random noise can also be introduced to the raw data via local differential privacy (LDP) [8, 17] to achieve privacy preservation. Nevertheless, adding noise makes it difficult to train machine learning model accurately [12]. On the other hand, cryptographic tools, such as secure multiparty computation (SMC), secret sharing (SS), and homomorphic encryption (HE), are employed to design secure aggregation protocols for

guaranteeing model security. For instance, in SMC and HE based protocols [9], the participants upload the encrypted models and the cloud server can aggregate models over the ciphertext. In SS-based protocols [4], the participants can securely aggregate their models by exchanging random values with all the others. Whereas, since these protocols rely on heavily cryptographic primitives and require frequent information exchange, the computation and communication overheads are high, resulting in latency and inefficiency in learning procedure [11].

Motivated by these challenges, we propose a privacy-preserving distributed learning scheme to improve computation and communication efficiency. We integrate edge computing and develop a three-lay distributed learning architecture, involving the cloud layer, edge layer, and participant layer. When decomposing a global learning task to the distributed participants, the edge nodes assist in data transmission and aggregation, reducing direct communications between cloud and participants. In our work, we avoid introducing randomness to the original data or the updated models. In addition of secure aggregation protocols, we utilize the social relationships to divide participants into groups. The participants are able to apply secure aggregation protocols in small groups, which improves the communication efficiency while guaranteeing privacy preservation. The main contributions of this paper are summarized as follows.

- We propose a three-layer distributed learning scheme which integrates cloud-edge computing into the distributed learning. By utilizing mini-batch gradient decent (MBGD), a global learning task can be decomposed to edge nodes and participants hierarchically.
- To preserve privacy and improve efficiency in the learning procedure, we employ the social relationships between participants to implement SS-based aggregation protocols in small groups. In this way, each participant is unnecessary to communicate with all the others in the system, the overall communication overhead is reduced.
- We conduct extensive simulations to evaluate the effectiveness of our scheme. The results indicate that the proposed scheme can train a machine learning model in a distributed way with sufficient accuracy and efficiency.

The rest of this paper is organized as follows. In Section 2, the system model and problem statement are provided. In Section 3, the proposed approaches and the corresponding privacy analysis are described. After that, simulation results are analyzed in Section 4 and conclusions are provided in Section 5.

2 SYSTEM MODEL AND PROBLEM STATEMENT

In this section, we introduce the system model, problem statement, privacy model and design goals. All notations used in this part are listed in Table 1.

2.1 System Overview

We consider a three-layer system that supports efficient and privacy-preserving distributed learning. As shown in Fig. 1, the system model contains three types of entities: participants, edge nodes, and cloud server. Theses three entities are specifically described as follows.

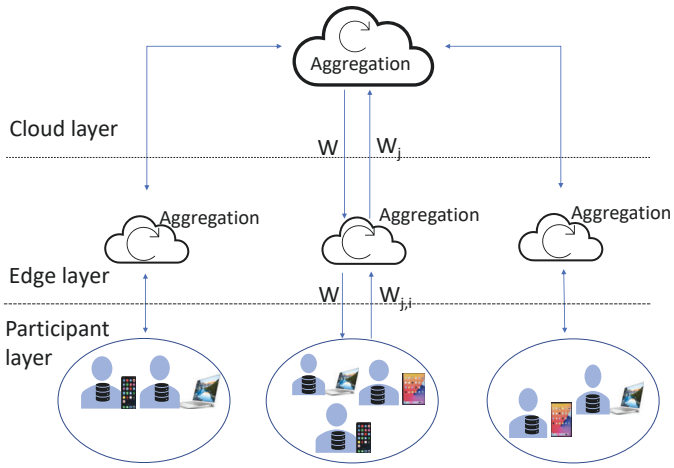

Figure 1: System Model

- *Participants:* Participants refer to the distributed data owners. They use their raw data to train a machine learning model locally. The obtained model is defined as *local machine learning model (LMLM)*. The LMLMs are uploaded to edge nodes.
- *Edge nodes:* Every edge node is in charge of a local area that consists of a certain number of participants. When an edge node receives the LMLMs from the participants, it aggregates those models into a *regional machine learning model (RMLM)* and uploads to the cloud server.
- *Cloud server:* After receiving the RMLMs from edge nodes, the cloud server aggregates them and updates to a *global machine learning model (GMLM)*. The cloud server distributes the updated GMLM to the edge nodes. The edge nodes then assist in delivering GMLM to the participants to perform iterative distributed learning.

2.2 Problem Statement

In general, most of machine learning models can be formulated as an optimization problem. For simplicity, we take the support vector machine (SVM) as an example to introduce how to decompose the optimization problem in distributed learning systems. The goal of SVM is typically to find a hyperplane for binary classification problems. We train it by feeding labeled examples (x_m, y_m) from dataset D, where x_m is the n-dimensional feature vector and y_m is the labeled class. Then the SVM model is defined as

$$f(x_m) = sign(\boldsymbol{w}^\star \cdot x_m + \boldsymbol{b}^\star), \forall x_m \in D, \tag{1}$$

where $\boldsymbol{w}^\star$ and $\boldsymbol{b}^\star$ are the optimal weights and intercepts that define the hyperplane. The corresponding cost function of Eq. (1) is

$$h(\boldsymbol{w}) = \frac{1}{|D|} \sum_{m=1}^{|D|} \left[\frac{1}{2} \|\boldsymbol{w}\|^2 + C \max(0, 1 - y_m(\boldsymbol{w} \cdot x_m)) \right]. \tag{2}$$

To identify $\boldsymbol{w}^\star$, we train the model by minimizing the cost function as follows.

$$\mathbf{P1}: \min \quad h(\boldsymbol{w}) \tag{3}$$

where $|D|$ is the size of the dataset and C is the soft margin parameter.

In a fully distributed learning system, since each participant i trains the SVM model locally with its dataset D_i, the optimization problem **P1** can be decomposed as follows:

$$\mathbf{P2}: \min_{\boldsymbol{w}_i} \quad h_i(\boldsymbol{w}_i), \tag{4}$$

$$\text{s.t.} \quad \boldsymbol{w}_i = z, \tag{5}$$

where h_i is the cost function and $\boldsymbol{w}_i$ is the weight values for each participant i. z is an introduced parameter which forces all $\boldsymbol{w}_i$ to be the same, so that we can obtain an certain global SVM model at the end of the optimization.

As shown in Fig. 1, we consider a three-layer system which involves one cloud server and a set of edge nodes that is denoted as $\mathcal{F} = \{f_1, f_2, ..., f_J\}$. Each edge node f_j connects with a set of participants $\mathcal{P}_j = \{p_1^j, p_2^j, ..., p_{N_j}^j\}$. In this system, the SVM model is trained hierarchically as follows.

- *Cloud-Edge:* When considering the distributed learning between cloud layer and edge layer, we have the cost minimization as follows:

$$\mathbf{P3}: \min_{\boldsymbol{w}_j} \quad h_j(\boldsymbol{w}_j) \tag{6}$$

$$\text{s.t.} \quad \boldsymbol{w}_j = z', \tag{7}$$

  where h_j is the regional cost function and $\boldsymbol{w}_j$ is the regional weight values for each edge node f_j. The weights update between cloud and f_j following the MBGD approach,

$$\begin{aligned} \boldsymbol{w}_j^{t+1} &= \boldsymbol{w}_j^t - \alpha' \nabla h_j \\ &= \boldsymbol{w}_j^t - \alpha' \frac{\partial h_j}{\partial \boldsymbol{w}_j}, \end{aligned} \tag{8}$$

$$z' = \frac{1}{J} \sum_{j=1}^{J} \boldsymbol{w}_j^{t+1}, \tag{9}$$

  where α' is the learning rate.
- *Edge-Participants:* When considering the distributed learning between edge layer and participant layer, we have the cost minimization as follows:

$$\mathbf{P4}: \min_{\boldsymbol{w}_{j,i}} \quad h_{j,i}(\boldsymbol{w}_{j,i}) \tag{10}$$

$$\text{s.t.} \quad \boldsymbol{w}_{j,i} = z_j'', \tag{11}$$

  where $h_{j,i}$ is the local cost function and $\boldsymbol{w}_{j,i}$ is the local weight values for each participant $p_i^j (i \in [1, N_j], j \in [1, J])$. The weights update between f_j and p_i^j following the MBGD approach as follows,

$$\begin{aligned} \boldsymbol{w}_{j,i}^{t+1} &= \boldsymbol{w}_{j,i}^t - \alpha'' \nabla h_{j,i} \\ &= \boldsymbol{w}_{j,i}^t - \alpha \frac{\partial h_{j,i}}{\partial \boldsymbol{w}_{j,i}}, \end{aligned} \tag{12}$$

$$z_j'' = \frac{1}{N_j} \sum_{i=1}^{N_j} \boldsymbol{w}_{j,i}^{t+1}, \tag{13}$$

Table 1: Notation Definitions

Variable	Definition
$h(\boldsymbol{w})$	Global cost function
$\boldsymbol{w}$	Global weight values
f_j	The edge node j
J	The number of edge nodes
N_j	The number of participants under f_j
p_i^j	The participant i under f_j
$h_j(\boldsymbol{w})$	Regional cost function
$h_{j,i}(\boldsymbol{w})$	Local cost function
$\boldsymbol{w}_j$	Regional weight values
$\boldsymbol{w}_{j,i}$	Local weight values
a, a', a''	Learning rate
$\boldsymbol{w}_j'$	Perturbed regional weight values
$\boldsymbol{w}_{j,i}'$	Perturbed local weight values
q	Percentage of participants who are involved in a group

where α'' is the learning rate.

In this way, **P1** is decentralized to **P3** and **P4**. As shown in Fig. 1, each participant uploads its $\boldsymbol{w}_{j,i}$ to the edge node f_j. After aggregation, f_j uploads its $\boldsymbol{w}_j$ to the cloud. Since $\boldsymbol{w}_{j,i}$ and $\boldsymbol{w}_j$ reflect some basic knowledge of participant's data, attackers can use them to reconstruct partial original data. How to protect $\boldsymbol{w}_{j,i}$ and $\boldsymbol{w}_j$ from being disclosed during the learning process is the problem targeted in this work.

2.3 Privacy Model and Design Goals

All entities considered in the system are assumed to be semi-trusted. Specifically, the participants can honestly train the model and update local weight values to edge nodes, but one participant may be curious to infer the other participants' private data via their local weight values. The cloud and edge nodes are honest in information aggregation and model update, but they are curious to derive the participants' raw data by analyzing the received weight values.

The goals of our work are to 1) protect the participant's local weight values from the other participants, the edge nodes, and the cloud; 2) protect the edge node's regional weight values from the other edge nodes and the cloud; 3) achieve accurate global machine learning model; 4) guarantee communication and computation efficiency in the learning procedure.

3 PROPOSED APPROACHES

In this section, we propose the approaches for protecting the local weight values and regional weight values. Since weight values are uploaded from bottom to up in the three-layer system, we first describe the proposed approach between participant layer and edge layer. Then we introduce the privacy preservation between edge layer and cloud layer.

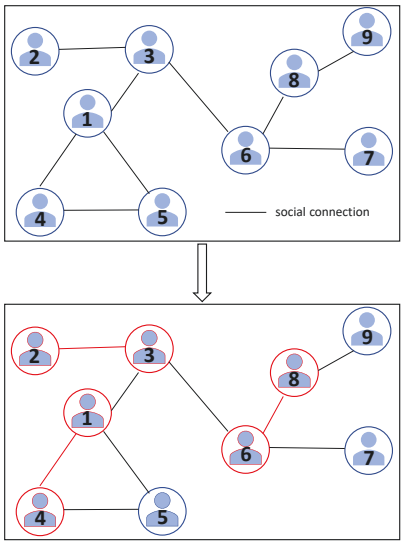

Figure 2: Undirected Graph of A Social Network

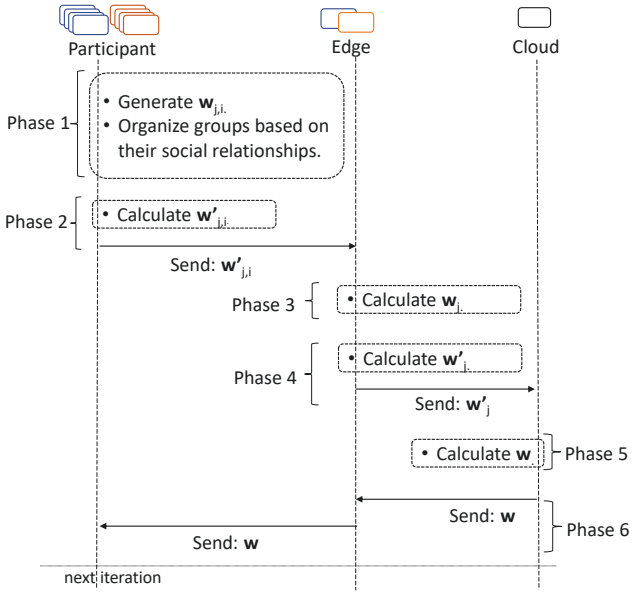

Figure 3: Overview of the Proposed Approaches

3.1 Privacy-preserving learning in edge/participant layers

As described in **P4**, each edge node f_j needs to aggregate the local weight values following Eq. (13). To calculate the summation of local weight values privately, the SS-based secure summation protocol [4] is a popular solution. Basically, assume there are N users and each user i has a value v_i. The protocol has the following steps:

- Each user i randomly generates a number r_{ik} and sends r_{ik} to user k with $\forall k \in [1, N], k \neq i$. Thus, each user i generates $N-1$ random numbers and receives $N-1$ random numbers from other users in total.
- Each user i calculates the summation of its generated numbers as $A_i = \sum_{k=1,k\neq i}^{N} r_{ik}$ and the summation of its received numbers as $B_i = \sum_{k=1,k\neq i}^{N} r_{ki}$.
- Each user i sends $v_i + A_i - B_i$ to the agent.
- The agent can calculate the summation of v_i by $\sum_{i=1}^{N}(v_i + A_i - B_i) = \sum_{i=1}^{N}(v_i + \sum_{k=1,k\neq i}^{N} r_{ik} - \sum_{k=1,k\neq i}^{N} r_{ki}) = \sum_{i=1}^{N} v_i$.

However, since each user needs to communicate with all the other users, the communication overhead is extremely heavy for large-scale systems. In our work, in addition to the secure summation protocol, we utilize the social relationship of participants to reduce the communication overhead while preserving secure aggregation. As shown in Fig. 2, we can use an undirected graph to illustrate the social relationship in a social network. Each user is represented as a node and two nodes are connected by an edge if there exists social relationship between them. In our work, we allow the participants to self-organize groups based on their social relationships. Each group has only two participants and each participant can only be involved in one group. For instance, in Fig. 2, node 1 and node 4 form a group. Although node 5 has social connection with node 1 and node 4, it cannot form groups with them. Similarly, there exist other two groups. One consists of node 2 and node 3 while the other one has node 6 and node 8. For each group, the two nodes perturb their private data following four steps.

- Node i sends a random number r to node k. Node k sends a random number d to node i.
- Node i calculates $A = v_i + r - d$. Node k calculates $B = v_k + d - r$.
- Node i and node k send A, B to the agent respectively.
- The agent can calculate $v_i + v_k = A + B$.

It is noted that nodes 5,7,9 are isolated nodes without being involved in any groups. These nodes utilize the secure summation protocol to protect the information.

Fig. 3 shows the details of the learning procedure between edge layer and participant layer. Specifically, it involves the following phases.

Phase 1: Each participant p_i^j trains machine learning model locally and obtains $\boldsymbol{w}_{j,i}$. Meanwhile, each participant attempts to form a group with other participants based on their social relationships. Note that one group only includes two participants and one participant cannot be involved in more than one group.

Phase 2: When being involved in a group, each participant generates $\boldsymbol{w}'_{j,i}$ by following the steps:

- p_i^j sends a random number r to p_k^j. p_k^j sends a random number d to p_i^j.
- p_i^j calculates $\boldsymbol{w}'_{j,i} = \boldsymbol{w}_{j,i} + r - d$. p_k^j calculates $\boldsymbol{w}'_{k,i} = \boldsymbol{w}_{k,i} + d - r$.

When being isolated, all isolated participants communicate with each other to generate $\boldsymbol{w}'_{j,i}$ by following the steps:

- p_i^j randomly generates a number r_{ik} and sends r_{ik} to p_k^j with $\forall k \in$ other isolated participants.
- p_i^j calculates $A_i = \sum_k r_{ik}$, $B_i = \sum_k r_{ki}$, $\boldsymbol{w}'_{j,i} = \boldsymbol{w}_{j,i} + A_i - B_i$.

Then p_i^j sends $\boldsymbol{w}'_{j,i}$ to edge node f_j.

Phase 3: Each edge node f_j performs data aggregation and calculates $\boldsymbol{w}_j = \frac{1}{N_j} \sum_{i=1}^{N_j} \boldsymbol{w}'_{j,i}$.

Following these three phases, the participants can upload their local learning models to the edge nodes without exposing their local weight values. Meanwhile, the edge nodes can aggregate these models following Eq. (13) without information loss.

3.2 Privacy-preserving learning in cloud/edge layers

Fig. 3 also shows the details of the learning procedure between cloud layer and edge layer. Specifically, it involves the following phases.

Phase 4: Each edge node f_j communicates with other edge nodes to generate $\boldsymbol{w}'_j$ by following the steps:

- f_j randomly generates a number r_{jk} and sends r_{jk} to f_k with $\forall k \in [1, J], k \neq j$.
- f_j calculates $A_j = \sum_k r_{jk}$, $B_j = \sum_k r_{kj}$, $\boldsymbol{w}'_j = \boldsymbol{w}_j + A_j - B_j$.

Then f_j sends $\boldsymbol{w}'_j$ to cloud.

Phase 5: Cloud performs data aggregation and calculates $\boldsymbol{w} = \frac{1}{J} \sum_{j=1}^{J} \boldsymbol{w}'_j$.

Phase 6: Cloud sends $\boldsymbol{w}$ to the edge nodes and the edge nodes continue to send $\boldsymbol{w}$ to the participants for the learning of next iteration.

3.3 Privacy Analysis

In the proposed scheme, the global learning task is decentralized to edge nodes and participants. Eqs. (8), (9), (12), (13) provide the iterative decomposition at edge layer and participant layer. For each participant, since it generates and updates the local weight values $\boldsymbol{w}_{j,i}$ following Eq. (12) without exposing its own data, there is no privacy leakage for this step which is the phase 1 in Fig. 3. The computation of z''_j from Eq. (13) is possible to leak $\boldsymbol{w}_{j,i}$ to f_j and other participants. As discussed in the phase 2 of the proposed approaches, the participants exchange random numbers with each other and perturb the $\boldsymbol{w}_{j,i}$ with those random numbers. Since the secure summation protocol guarantees the privacy of the shared information, $\boldsymbol{w}_{j,i}$ is protected from the edge nodes and other participants. Specifically, for each participant, it only sends random numbers to other participants while keeping $\boldsymbol{w}_{j,i}$ as private along the procedure. Thus, the other participants cannot infer $\boldsymbol{w}_{j,i}$ via the random numbers. For each edge node f_j, since it only receives $\boldsymbol{w}'_{j,i}$ which is a perturbed value. Without knowing the random values of A_i and B_i in phase 2, the advantage of f_j recovers $\boldsymbol{w}_{j,i}$ from $\boldsymbol{w}'_{j,i}$ is negligible. Therefor, $\boldsymbol{w}_{j,i}$ is protected from edge nodes and the other participants.

Similarly, the computation of z' from Eq. (9) is possible to leak $\boldsymbol{w}_j$ to cloud and other edge nodes. As discussed in the phase 4 of the proposed approaches, since secure summation protocol is utilized to aggregate $\boldsymbol{w}_j$ without exposure, the privacy of $\boldsymbol{w}_j$ is preserved.

4 PERFORMANCE EVALUATIONS

In this section, we evaluate the performance of the proposed scheme by implement SVM over the dataset BCD [1]. BCD is breast cancer dataset that involves 570 instances and 32 feature attributes, describing the characteristics of the cell nuclei of breast mass. Based on these features, we train the SVM model to classify if the mass is harmless or cancerous. When implementing the SVM in the three-layer scheme, we consider one cloud server, two edge nodes, and five participants under each edge node. We first evaluate the learning accuracy by comparing with centralized learning [15] and DP-based distributed learning [3]. Then we discuss the communication efficiency by comparing with the SS-based scheme [4].

Table 2: Classification Accuracy

	Accuracy	Recall	Precision
Centralized	98.1%	96.6%	96.6%
DP-based	84.2%	81.3%	81.3%
Proposed	97.5%	94.8%	94.8%

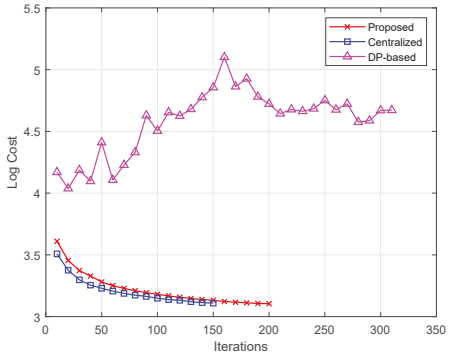

Figure 4: Convergence iteration.

Table 2 shows the comparison of the classification results in terms of accuracy, recall, and precision. The centralized learning performs best and achieves extremely high accuracy, which is more than 98%. Our proposed scheme obtains significant accuracy, although not as high as the centralized learning. In the centralized learning, all instances of the dataset are fed into the training procedure. However, in the proposed scheme, since we apply MBGD to decompose the global learning task, each participant only pick a fixed size of instances to train the model. This could be a reason why our performance is slightly degraded compared with the centralized learning. The DP-based distributed learning has much lower accuracy due to the introduced noise data.

Fig. 4 describes the performance in terms of convergence iteration. For centralized learning and our proposed scheme, with the increase of iterations, the value of $\log h(\boldsymbol{w})$ (denoted as log cost) tends to be minimum and stable. Specifically, the centralized learning converges at 150 iterations while our proposed scheme converges at around 200 iterations. The model decomposition could be a possible reason that leads to such a lower convergence speed. The log cost of DP-based distributed learning fluctuates irregularly. This is because in each iteration, the introduced noise is random, which perturbs the weight values significantly. Then it is difficulty for DP-based scheme to converge smoothly.

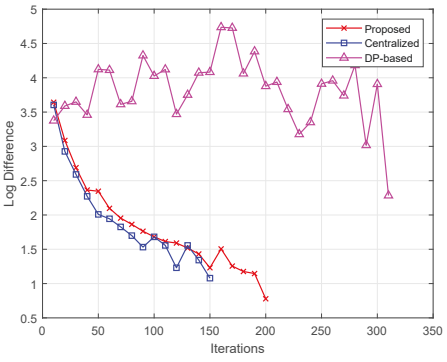

Figure 5: Convergence vibration.

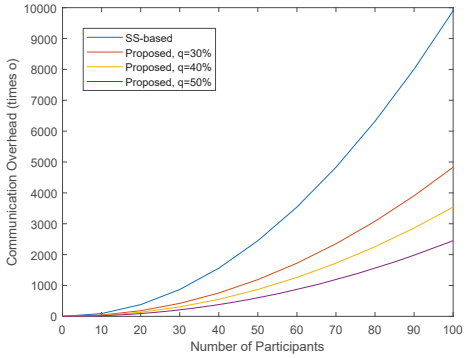

Figure 6: Efficiency.

Fig. 5 evaluates the performance in terms of convergence vibration. Log difference represents the difference of costs in two consecutive iterations, which is defined as $\log|h_t(\boldsymbol{w}) - h_{t+1}(\boldsymbol{w})|$. Similarly, the centralized learning and our proposed scheme can converge smoothly within 200 iterations. It means our proposed scheme is time efficient in training the machine learning model and can achieve sufficient accuracy when comparing with the centralized learning. For DP-based distributed learning scheme, it cannot converge within 300 iterations and may take much more iterations to achieve a satisfying convergence. Thus, DP-based scheme may also lead to longer latency in the training procedure due to lower convergence speed.

We compare the proposed scheme with SS-based scheme in term of communication efficiency in Fig. 6. Assume the communication overhead between two users to exchange the random numbers is o, then the communication overhead grows exponentially with the increased number of participants. For our proposed scheme, since we utilize the social relationships between participants, the participants who are not involved in a group are isolated. Let the percentage of participants who are involved in a group in the system be q. As shown in Fig. 6, our proposed scheme has consumed less communication resource than the SS-based scheme. The grater value of q is, the better our performance would be.

5 CONCLUSION AND FUTURE WORK

In this paper, we have proposed a three-layer distributed learning scheme, involving the cloud layer, edge layer, and participant layer. Different from the fully distributed system, a global learning task can be decomposed from cloud to participants hierarchically through edge layer. To achieve efficient learning while guaranteeing privacy preservation during the training procedure, we have proposed to employ the secure summation protocols to protect the weight values and utilize the social networks to reduce the communication overhead. By implementing SVM in the proposed scheme, we have evaluated the effectiveness of our work. The simulation results have demonstrated that our proposed scheme can achieve high learning accuracy with significant communication efficiency.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation under grants CNS-2007995 and CNS-2008145.

REFERENCES

[1] [n.d.]. *Breast Cancer Wisconsin (Diagnostic) Data Set.* Retrieved March 20, 2021 from https://www.kaggle.com/uciml/breast-cancer-wisconsin-data
[2] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani. 2021. A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet of Things Journal* 8, 7 (2021), 5476–5497.
[3] Z. He, T. Zhang, and R. B. Lee. 2020. Attacking and Protecting Data Privacy in Edge-Cloud Collaborative Inference Systems. *IEEE Internet of Things Journal* (2020), 1–1. https://doi.org/10.1109/JIOT.2020.3022358
[4] Q. Jia, L. Guo, Y. Fang, and G. Wang. 2019. Efficient Privacy-Preserving Machine Learning in Hierarchical Distributed System. *IEEE Transactions on Network Science and Engineering* 6, 4 (2019), 599–612.
[5] Q. Jia, L. Guo, Z. Jin, and Y. Fang. 2018. Preserving Model Privacy for Machine Learning in Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems* 29, 8 (2018), 1808–1822.
[6] M. Langer, Z. He, W. Rahayu, and Y. Xue. 2020. Distributed Training of Deep Learning Models: A Taxonomic Perspective. *IEEE Transactions on Parallel and Distributed Systems* 31, 12 (2020), 2802–2818.
[7] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. C. Liang, Q. Yang, D. Niyato, and C. Miao. 2020. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Communications Surveys Tutorials* 22, 3 (2020), 2031–2063.
[8] P. C. Mahawaga Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman. 2020. Local Differential Privacy for Deep Learning. *IEEE Internet of Things Journal* 7, 7 (2020), 5827–5842.
[9] S. Shen, T. Zhu, D. Wu, W. Wang, and year=2020 publisher=Wiley Online Library W. Zhou, journal=Concurrency and Computation: Practice and Experience. [n.d.]. From distributed machine learning to federated learning: In the view of data privacy and security. ([n. d.]).
[10] J. Siryani, B. Tanju, and T. J. Eveleigh. 2017. A Machine Learning Decision-Support System Improves the Internet of Things' Smart Meter Operations. *IEEE Internet of Things Journal* 4, 4 (2017), 1056–1066.
[11] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim. 2020. Privacy-Preserving Deep Learning on Machine Learning as a Service—a Comprehensive Survey. *IEEE Access* 8 (2020), 167425–167447.
[12] X. Wang, J. He, P. Cheng, and J. Chen. 2019. Privacy Preserving Collaborative Computing: Heterogeneous Privacy Guarantee and Efficient Incentive Mechanism. *IEEE Transactions on Signal Processing* 67, 1 (2019), 221–233.
[13] Y. Wang, Q. Chen, D. Gan, J. Yang, D. S. Kirschen, and C. Kang. 2019. Deep Learning-Based Socio-Demographic Information Identification From Smart Meter Data. *IEEE Transactions on Smart Grid* 10, 3 (2019), 2593–2602.
[14] Z. Wang, L. Sui, J. Xin, L. Qu, and Y. Yao. 2020. A Survey of Distributed and Parallel Extreme Learning Machine for Big Data. *IEEE Access* 8 (2020), 201247–201258.
[15] S. Yu, X. Li, X. Zhang, and H. Wang. 2019. The OCS-SVM: An Objective-Cost-Sensitive SVM With Sample-Based Misclassification Cost Invariance. *IEEE Access* 7 (2019), 118931–118942.
[16] T. Zhang and Q. Zhu. 2017. Dynamic Differential Privacy for ADMM-Based Distributed Classification Learning. *IEEE Transactions on Information Forensics and Security* 12, 1 (2017), 172–187.
[17] H. Zheng, H. Hu, and Z. Han. 2020. Preserving User Privacy for Machine Learning: Local Differential Privacy or Federated Machine Learning? *IEEE Intelligent Systems* 35, 4 (2020), 5–14.