Communication-Efficient Federated Learning in Channel Constrained Internet of Things

Tao Hu*, Xinran Zhang*, Zheng Chang*‡, Fengye Hu †, and Timo Hämäläinen‡
*School of Computer Science and Engineering, University of Electronic Science and Technology of China, 611731 Chengdu, China
†College of Communication Engineering, Jilin University, Changchun, China
‡Faculty of Information Technology, University of Jyväskylä, P. O. Box 35, FIN-40014 Jyväskylä, Finland

***Abstract*—Federated learning (FL) is able to utilize the computing capability and maintain the privacy of the end devices by collecting and aggregating the locally trained learning model parameters while keeping the local personal data. As the most widely-used FL framework, *federated averaging* (FedAvg) suffers an expensive communication cost especially when there are large amounts of devices involving the FL process. Moreover, when considering asynchronous FL, the slowest device becomes the bottleneck for the cask effect and determines the overall latency. In this work, we propose a communication-efficient federated learning framework with partial model aggregation (CE-FedPA) algorithm to utilize compression strategy and weighted device selection, which can significantly reduce the size of uploaded data and decrease the communication time. We perform a series of experiments on the MNIST/CIFAR-10 datasets, in both IID and non-IID data settings. We compare the communication time of different aggregation schemes, in terms of iteration rounds and target accuracy. Simulation results demonstrate that the uploading time of the proposed scheme is up to 4.3 times shorter than other existing ones. Experiments on an end-to-end FL framework also verify the communication efficiency of CE-FedPA in a real-world setting.**

***Index Terms*—Compression, Internet of Things, device selection strategy, dynamic communication environment.**

I. Introduction

During the past few years, the applications of Internet of Things (IoT) has gained growing popularity [1], such as smartphones, wearable devices, and autonomous vehicles. These IoT devices generate a vast amount of data which has not been efficiently utilized. As the raw material for machine learning (ML), the data has attracted attention from a variety of organizations for different purposes [2], such as image classification [3] and speech recognition [4]. However, personal data privacy has become an increasing concern [5]. Some papers have revealed the data leakage concerns and pointed out gathering user data in the centralized ML is risky for the end-users [6]. As the local computation capability is getting stronger and data privacy is becoming more concerned, there is a potential to store data locally and train models at the network edge.

In traditional distributed ML schemes, all local devices contribute to training a global model cooperatively. The data in each device is independent and identically distributed (IID) and the load on each device is roughly balanced. However, this is not practical in most IoT scenarios, because the data distribution may vary significantly [7]. Devices generate non-independent identically distributed (non-IID) data, which may affect the ML performance.

Therefore, it is crucial to investigate how to train an ML model by utilizing the massive number of unbalanced and non-IID data stored locally. FL [8] addresses this problem by collaboratively training an ML model. Multiple devices perform *stochastic gradient descent* (SGD) locally and upload the updated model parameters to the central server without revealing their local data to others. In this context, one typical FL method is so all *federated averaging* (FedAvg) [9], which is based on averaging the uploaded local parameters for aggregation, and has been shown to work effectively by Google with their GBoard [10].

However, some practical problems arise in FedAvg in the typiscal IoT scenario. The most crucial one is the expensive communication cost. FL process typically comprises a large number of devices, e.g., hundreds of smartphones or autonomous vehicles, so there are long periods of uplink and downlink communication time. Then, FL is 50 to 115 orders of magnitude slower than traditional centralized ML schemes [11]. Besides, if the transmission rate of some devices is plodding, the central server and other participating devices have to wait for a period. Moreover, due to the asymmetric property of wireless network, the uplink data rate is typical much slower than the downlink data rate [12] and the uplink data rates of different end-user also vary a lot, which inherently create the communication bottleneck for the FL process. In order to promote the application of FL in practice, the communication latency needs to be reduced.

Many researchers have considered the communication cost, especially the wireless channel effect, as one of the key problems in FL [7], and proposed some methods for reducing communication costs, which can be divided into three categories. The first one is the transmitted parameter model compression schemes [13], such as sparsification, subsampling, and quantization [14]. Although this method reduces the size of the transmitted message, it cannot significantly reduce the communication consumption due to the different communication environments of the participating devices. The second approach, federated learning with partial model aggregation (FedPA), selects only a subset of all devices to participate in aggregation to reduce the communication consumption [15].

A common practice is to randomly select devices with the same probability, but this method treats all devices equally and does not consider the differences and priorities of all devices. The third group is to reduce the number of training rounds by optimizing the model's convergence speed [16]. Moreover, existing schemes are generally primarily developed with the assumption that the communication of devices is in a static network. However, this assumption does not apply to real scenarios.

Bearing in mind aforementioned works, in this article, we propose a communication-efficient FL partial model aggregation framework (CE-FedPA), which is able to significantly reduce the total communication time compared with previous works. Our major contributions are as follows:

- We propose the CE-FedPA algorithm, composed of two parts: a) model compression; b) device selection. This model can reduce the uploaded model size in each round and the total communication time by taking into channel constraints compared with FedPA and FedAvg.
- In the device selection algorithm, we propose to use a new indicator for evaluating device contribution: *contribution index*. Using this indicator, we can determine the priority of the selected device. Moreover, we consider that all devices are in a dynamically changing network, making our simulation results closer to the real channel constrained IoT scenario.
- We use typical data sets MNIST and CIFAR-10 to perform extensive simulations and experiments on different ML models (MLP, CNN). Simulation results show that CE-FedPA significantly reduces the communication time. Then we perform CE-FedPA on an end-to-end federated learning framework composed of several Android cell phones and a server. The experiment results show that our model can significantly reduce the communication time in the real world.

The rest of this paper is organized as follows. In section II, we first introduce a concept of partial model aggregation, and then we demonstrate the structure of CE-FedPA. In section III, we provide the simulation results compared to FedAvg. Moreover, we present the end-to-end federated learning framework results, which show that our proposed model performs well in the practical IoT scenario.

II. Preliminary

A. System Model

We consider a general FL model, as shown in Fig. 1, in which there are a central server and a number of IoT devices. Assume that we have N IoT devices $\{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_i, ..., \mathcal{I}_N\}$ with local data sets $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_i, ..., \mathcal{D}_n\}$, denote $\mathcal{D} \triangleq \sum_{i=1}^{N} \mathcal{D}_i$. At each communication round, the selected IoT devices perform local training on their local data. Then the local parameters are uploaded to the central server for global aggregation. After aggregating all the local parameters, the server broadcasts the updated global model back to all participating IoT devices.

B. Partial Model Aggregation

There is an expensive communication consumption in *FedAvg* because the server aggregates the uploaded parameters from all the devices. Assume some devices in the model have an impoverished communication environment, other models in good communication condition and the central server have to wait for a long time, which causes the bottleneck problem for the federated learning. Therefore, FedPA is proposed to mitigate the communication bottleneck.

In FedPA, the loss function at device i is $F(w; x_i, y_i)$, where w is the parameter matrix, (x_i, y_i) is the data sample. The objective function is as follows:

$$w^* = \arg\min \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} F(w; x_i, y_i). \tag{1}$$

As shown in Fig. 1, the FedPA consists of three phases:

- **Phase I:Initialization.**
  The central server firstly initializes a global ML model, the parameter matrix W_0 and the fraction of selected devices θ. Then the server sends w^0 to the randomly selected device subset C^0. The number of selected devices is determined by $N \times \theta$.
- **Phase II:Local training.** At the t-th round, the devices i in C^t run the Stochastic Gradient Descent(SGD) algorithm with the local data sets to calculate the local parameter matrix w_i^t. The mathematical expression is formulated as follows:
  $$w_i^t = W^{t-1} - \eta \frac{\sum_{(x_i, y_i) \in \mathcal{D}_i} \nabla F(W^{t-1}; x_i, y_i)}{|\mathcal{D}_i|} \tag{2}$$
  where η is the learning rate. Moreover, the calculation process of Δw_i^t is as follows:
  $$\Delta w_i^t = w_i^t - w_i^{t-1} \tag{3}$$
- **Phase III:Aggregation.**
  At the t-th round, the server determines the device subset C^t with the device selection algorithm, and then the selected devices in C^t participate in calculating the global parameter matrix W^t. The mathematical form is as follows:
  $$W^t = \mathcal{AVG}([w_i^t \forall i \in C^t]) \tag{4}$$
  where $\mathcal{AVG}()$ is the weighted averaging method. In order to reduce the communication consumption, typically, the device i uploads the update of the parameter matrix Δw_i^t. Thus, the above equation is rewritten as follows:
  $$W^t = W^{t-1} + \mathcal{AVG}([\Delta w_i^t \forall i \in C^t]), \tag{5}$$

Repeat the above phases until the model converges.

III. Proposed CE-FedPA Scheme

In this section, with the introduction of model compression, we present the procedure of CE-FedPA, and then explore the device selection algorithm.

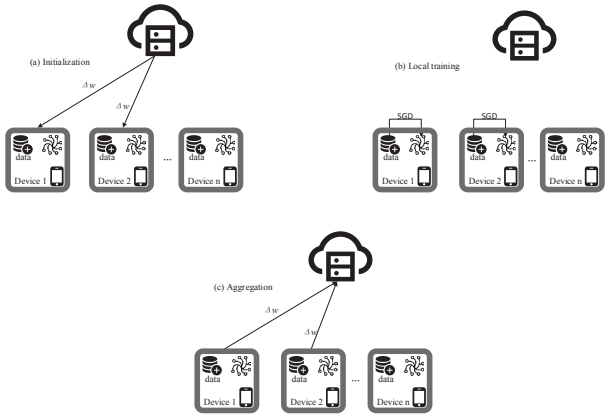

Fig. 1. Federated learning with a parameter server. (a) The server sends global parameters to a subset of devices. (b) Devices train the local model on their own data. (c) The local parameter updates are uploaded to the server, and then aggregated to generate the new global model.

A. Model Compression

The model compression method is used by many researchers to reduce communication consumption, and how to find an appropriate compression method is a key problem. As known in the data structure field, if a matrix is compressed losslessly, it must be in a particular form, such as a symmetric matrix, a diagonal matrix, a sparse matrix, etc. However, there are no strict form requirements for lossy matrix compression.

In the aggregation phase, the parameter matrix uploaded by each device is trivial. Therefore, in theory, compressing the uploaded parameter matrix is lossy, and the relationship between the information loss and the compression ratio is linear. In FedPA, the matrix updates Δw^t are uploaded in the aggregation phase, which has a *pre-specified structure*—**low rank**.

In FedPA, the number of bits that each device uploads to the central server in the aggregation phase is given by

$$b \in \mathcal{O}(|W| \times (H(\Delta W) + \eta))) \tag{6}$$

where $|W|$ is the size of the parameter matrix, $H(\Delta W)$ is the entropy of the weight updates exchange during the upload phase, η is the encoding redundancy [17].

$|W|$ cannot always decrease in order to reduce the amount of data uploaded by devices. There are two directions to achieve this targit: 1) reduce $H(\Delta W)$ through lossy compression schemes; 2) reduce η by using a more efficient encoding scheme. The following is a detailed introduction to the compression scheme in our proposed model.

We assume the updated parameter matrix $w_i^t \in \mathbf{R}^{m \times n}$, as mentioned above, is a low-rank matrix. For simplicity of description, we define its rank as k. Therefore, w_i^t as the product of two matrices: $w_i^t = R_i^t \times P_i^t$, where $R_i^t \in \mathbf{R}^{m \times k}$ is the reconstruction matrix, and $P_i^t \in \mathbf{R}^{k \times n}$ is the projection matrix. R_i^t is generated randomly in each round via pseudo-random generation algorithm $pse()$, and then optimized and updated during the training procedure [12].

The procedures for compressing the model are divided into the following steps in CE-FedPA: 1) at round t, the central server generates a random seed s_i^t and shares it with the device i. Then the pseudo-random generation algorithm computes $R_i^t = pse(s_i^t)$ on the central server and device i. For security reasons, random seeds are generated afresh in each round; 2) the device i trains local parameter matrix w_i^t by SGD with local data. Then it computes P_i^t by low-rank matrix factorization(MF), and sends it to the central server; 3) The central server gets the P_i^t, and computes the $w_i^t = R_i^t \times P_i^t$. With this compression scheme, we save a factor of d_1/k in the aggregation phase.

B. Device Selection Algorithm

1) Contribution Index: In the general implementation of FedPA, the devices are selected randomly, which ignores the differences between devices. Therefore, we define an indicator to evaluate the difference between devices.

Different devices have different data and CPUs with different computing capabilities in the practical case. Assume that the device's contribution is affected by two indicators: the amount of local data D_i^t and the computing capability λ_i. We define the contribution index as:

$$\pi_i^t = \frac{D_i^t}{(1-\alpha)^{\lambda_i}} \tag{7}$$

where α is the control parameter used to balance the influence of the local data and computing capabilities on the contribution index. For a device, the larger the size of the data sets is, the stronger the computing capabilities are, and the higher its contribution index is. A device with a higher contribution factor is more possible to be selected by the central server.

2) Problem Formulation: We propose an algorithm for selecting devices to participate in the aggregation phase. $\mathcal{T}$ is a pre-defined time, representing the time window to collect device updates at each round. In real-world scenarios, The communication environment of each device is dynamic. We use C_i^t to represent the channel size allocated to the device i at round t, representing its communication environment. According to the Shannon theory, the communication time T_i^t consumed by device i communication is shown below.

$$T_i^t = \frac{k \times |\mathcal{W}| \times (H(\triangle \mathcal{W}) + \eta)}{C_i^t \times log_2(1 + SNR)}. \tag{8}$$

We define $\mathcal{C}$ as the total channel in our model. Our goal is to obtain a bigger contribution index under the constraints of $\mathcal{T}$ and $\mathcal{C}$. Thus, the device selection problem at round t is formulated as:

$$\begin{aligned} & \max \sum_{i=1}^{N} \pi_i^t S_i^t, \\ \text{s.t.} \quad & \sum_{i=1}^{N} \frac{k \times |\mathcal{W}| \times (H(\triangle \mathcal{W}) + \eta)}{c_i^t log_2(1+SNR)} S_i^t \leq \mathcal{T}, \\ & \sum_{i=1}^{N} c_i^t S_i^t \leq \mathcal{C}, \\ & S_i^t \in \{0,1\}, \end{aligned} \tag{9}$$

where k is the compression ratio. (9) is a 0-1 knapsack problem with two constraints. Next, we will explain how to solve this problem with dynamic programming in detail.

1) Firstly, we define $I_{\{i\}}$ as a device set that contains the first i devices, and define the state $\Pi(I_{\{i\}}, \mathcal{C}, \mathcal{T})$ as the optimal choice under the constraints of $\mathcal{C}$ and $\mathcal{T}$.
2) Then, we provide an explanation on the state transition equation. If $C_i > \mathcal{C}$ or $T_i > \mathcal{T}$, $\Pi(I_{\{i\}}, \mathcal{C}, \mathcal{T}) = \Pi(I_{\{i-1\}}, \mathcal{C}, \mathcal{T})$, otherwise, $\Pi(I_{\{i\}}, \mathcal{C}, \mathcal{T}) = \max(\Pi(I_{\{i-1\}}, \mathcal{C}, \mathcal{T}), \Pi(I_{\{i-1\}}, \mathcal{C} - C_i, \mathcal{T} - T_i) + \pi_i)$.
3) Finally, we obtain the optimal device selection subset at each round according to the device selection algorithm, which is shown in Algorithm 1.

Note that I_{select} is an N-bit integer representing the selection vector, i.e. when N = 3, 000 indicates that no device is selected, 111 means that all devices are selected. In the algorithm, we have two hash tables S and D. S holds the state and D holds the device selection vector. According to the state transition equation given before, new state and device selection vectors are added continuously. If device i is selected, the I_{select}'s ith bit will be set to 1. Then the selection vector is updated bitwisely with the previous selection vector. Finally the optimal device selection vector is obtained in $D(\Pi(I_{\{N\}}, \mathcal{C}, \mathcal{T})$.

The algorithm consists of three loops. Thus the time complexity is $\mathcal{O}(N \times |T| \times |C|)$, where N represents the number of devices, $|T|$ represents the number of time intervals divided, $|C|$ represents the number of divided channel intervals.

To this end, we analyze the optimality of the device selection algorithm. If the time and channels can be divided infinitely small, we can obtain the optimal solution, at the cost of increased time complexity. In the actual scenario, even with lower time complexity, i.e. the time and channel partition is not so fine, we can still obtain a satisfying result. Let S^* be the optimal device selection set, and selected device set in our algorithm is S, then the difference between optimal solution and ours is $S^* - S \in \mathcal{O}(\frac{1}{|T| \times |C|})$.

Algorithm 1 Device selection algorithm

Input: I, $\mathcal{T}$, $\mathcal{C}$
Output: Device select set

Initialization $S \leftarrow \{\}$ //State set
Initialization $D \leftarrow \{\}$ //Devices selection set
Initialization $I_{select} \leftarrow 0_{[N]}$ // type N-bit int
for $c \leftarrow 0 : \mathcal{C}$ **do**
  for $t \leftarrow 0 : \mathcal{T}$ **do**
    for each $i \in I$ **do**
      if $T_i > t$ or $C_i > c$ **then**
        $S(\Pi(I_{\{i\}}, c, t)) \leftarrow S(\Pi(I_{\{i-1\}}, c, t))$
        $D(\Pi(I_{\{i\}}, c, t)) \leftarrow D(\Pi(I_{\{i-1\}}, c, t))$
      else
        $\pi_1 \leftarrow S(\Pi(I_{\{i-1\}}, c, t))$
        $\pi_2 \leftarrow S(\Pi(I_{\{i-1\}}, c - C_i, t - T_i)) + \pi_i$
        if $\pi_1 \geq \pi_2$ **then**
          $S(\Pi(I_{\{i\}}, c, t)) \leftarrow \pi_1$
          $D(I_{\{i\}}, c, t) \leftarrow D(\Pi(I_{\{i-1\}}, c, t))$
        else
          $S(\Pi(I_{\{i\}}, c, t)) \leftarrow \pi_2$
          $I_{select}[i] \leftarrow 1$
          $I_{select} \leftarrow D(\Pi(I_{\{i-1\}},$
            $c - C_i, t - T_i)) \& I_{select}$
          $D(I_{\{i\}}, c, t) \leftarrow I_{select}$
        end if
      end if
    end for
  end for
end for

IV. Performance Evaluations

In this section, we firstly conduct simulations and experiments to evaluate the performance of the proposed CE-FedPA method and compare it with the FedPA approach. Then, we apply this scheme in a practical scenario, where several Android phones are located around a server.

A. Simulation

1) Simulation Setup: We consider two ML-based models for image classification, multilayer perceptron (MLP) and convolutional neural networks (CNN). The task is conducted on the datasets MNIST and CIFAR-10, which cover handwritten digit images and 10 different class colored figures.

To verify the effectiveness of CE-FedPA, we compare the training performances under different numbers of devices, compression ratios, and the amounts of data on each device. Besides, both IID data and non-IID data are leveraged in the simulations. The IID data owned by each device is assigned randomly. To generate non-IID data, all data are separated into ten categories and then divided into fragments, and each device occupies one fragment. Thus each device has only one class of data. Furthermore, the test set comes from official test data. The channel size of each device in each round of simulation is represented by the random numbers from 1 to 3. The compression ratio in CE-FedPA is set to be 0.7. For the purposes of comparison, we apply two baselines:1) FedPA (θ = 0.3) and 2) FedAvg.

2) Simulation Results: Fig. 2 shows the difference among the training loss of CE-FedPA, FedAvg and FedPA on the

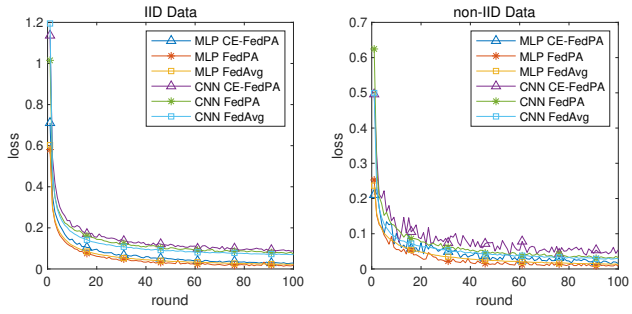

Fig. 2. Training loss per round for CE-FedPA ($k = 0.7$), FedPA ($\theta = 0.3$) and FedAvg.

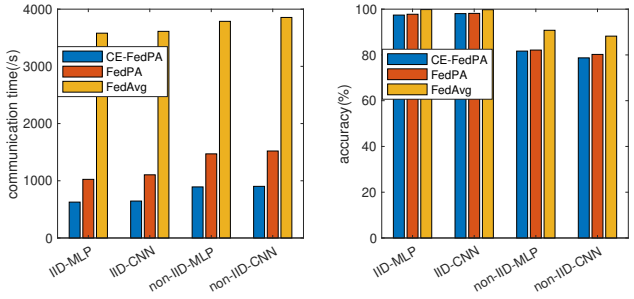

Fig. 3. Left is the total upload time for CE-FedPA($k = 0.7$), FedPA($\theta = 0.3$) and FedAvg. Right is the test accuracy for CE-FedPA ($k = 0.7$), FedPA ($\theta = 0.3$) and FedAvg.

dataset MINIST with the same learning round. For the IID setting, the training losses in all three algorithms first decrease and eventually converged. While for the non-IID settings, the training loss of FedPA and CE-FedPA's oscillate in a larger region. Compared to FedAvg, fewer devices are involved in the aggregation phase in CE-FedPA, so only several classes are allowed of data are used in the local training.

Fig. 3 presents the communication time and accuracy of CE-FedPA, FedPA and FedAvg. It is obvious that there is less communication time in CE-FedPA than that in FedPA ($1.8\times$) and that in FedAvg ($5.8\times$). Furthermore, the accuracy of CE-FedPA and FedPA are similar, but smaller than that in FedAvg.

The above results indicate that CE-FedPA highly reduces the communication time with only a small cost of model accuracy.

In Fig. 4, to achieve the same accuracy, the communication time of FedAvg, FedPA, and CE-FedPA is displayed. The target accuracy on the dataset MNIST is 97.5% for the IID setting and 85% for the non-IID setting, and 50% on the dataset CIFAR-10 data sets for both data distributions. In the IID setting, the accuracy profiles of all three schemes are stable compared to those in the non-IID setting. The curve for FedAvg is the most stable in the IID setting, while FedPA oscillates violently in the non-IID setting. In both settings, the CE-FedPA has the least communication time compared with FedPA and FedAvg.

Similarly, Fig. 5 shows the accuracy and communication time on the dataset CIFAR-10. Compared to MNIST, it is more difficult to improve the training accuracy in CIFAR-10.

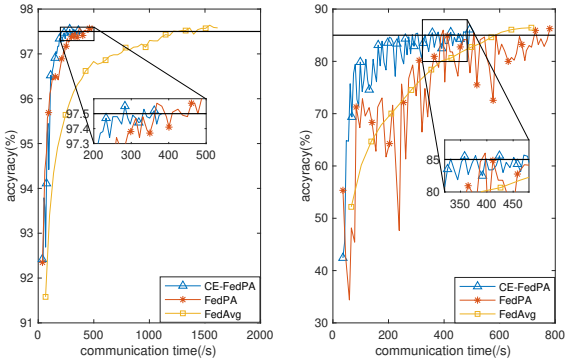

Fig. 4. Accuracy during training on MNIST (left: IID, right: non-IID)

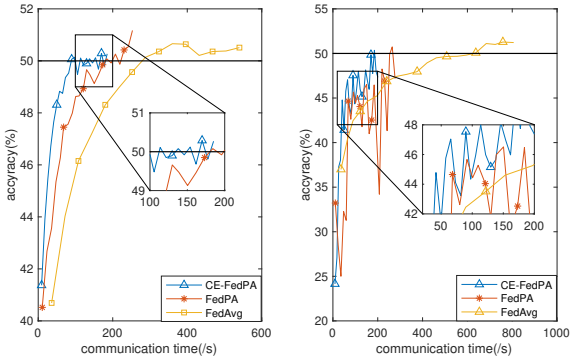

Fig. 5. Accuracy during training on CIFAR-10 (left: IID, right: non-IID)

In the IID setting, the profile in FedAvg is the smoothest, and the communication time in CE-FedPA is the shortest compared with FedPA ($2.3\times$) and FedAvg ($3.2\times$). In the non-IID setting, the curves of the three schemes are oscillating, and the curve for FedPA oscillates more violently. The CE-FedPA has the least communication time over FedPA ($1.4\times$) and FedAvg($3.4\times$).

B. Practical tests

1) Testbed Setup: To further evaluate the performances of CE-FedPA, we consider a practical channel constrained IoT scenario consisting of 4 Android devices and a server, and we leverage an end-to-end FL framework. All devices are using the same wireless network. To simulate the constrained networks, the wireless network setup tools are utilized to limit the bandwidth for all devices. The server selects the device for FL based on the device selection algorithm. The devices upload not only local models to the server, but also the timestamp of the uploading time, and then the server can calculate the communication time.

The client application were installed in the Android devices for local training and data upload purposes. A laptop acted as a server. The software is supported by Python using Tensorflow and cherrypy. We train the Multi-class Neural Networks (MNN) training model on the dataset MNIST. For comparison

TABLE I
SPECIFIC PARAMETERS OF EXPERIMENTAL EQUIPMENT

Device model	CPU	ROM	RAM	Battery capacity
M2007J22C	Dimensity 800U	256GB	8GB	5000mAh
M2010J19SC	Qualcomm Snapdragon 662	128GB	4GB	6000mAh
CHL-AN00	MediaTek MT6833	128GB	8GB	4000mAh
V2072A	Dimensity 1100 Octacore	128GB	8GB	4000mAh

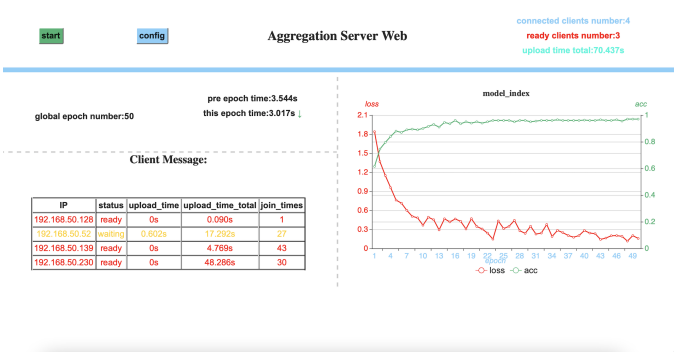

Fig. 6. CE-FedPA on end-to-end federated learning framework

perposes, FedPA($\theta = 0.5$) is set as the baseline. The system parameters of the Android devices are shown in Table I.

2) Testbed results: Fig. 6 shows the result of CE-FedPA on end-to-end federated learning framework. Table II displays the performances of CE-FedPA and FedPA with the same learning iterations. Total upload time includes the time to compress the data and upload the model. The result demonstrates that CE-FedPA can reach a similar high accuracy but take much less uploading time than FedPA (up to 2× less).

V. CONCLUSION

In this paper, we develop a communication-efficient FL framework in a channel constrained IoT, CE-FedPA, which largely reduces the transmission latency while guaranteeing a high learning performance. In CE-FedPA, we design model compression scheme and a device selection algorithm. Accordingly, the amount of uploaded data is reduced, and thus the uploading time significantly decreases. To verify the performance advantages, we perform a series of experiments on the datasets MNIST and CIFAR-10. In the simulations, CE-FedPA takes less communication time than FedPA(1.8×) and FedAvg(5.8×) to achieve the same learning accuracy. Further experiments are conducted where an end-to-end FL framework composed of several Android devices is applied. In this experiment, CE-FedPA takes 2× less uploading time to reach the target accuracy. To investigate how to better evaluate the quality of local data, future works can take the data quality into account for device selection priorities and select a better subset of devices in each round of training.

TABLE II
THE PERFORMANCE OF CE-FEDPA OVER FEDPA

Model	Total upload time(s)	Training accuracy	Testing accuracy
CE-FedPA	70.437	97.4%	98.1%
FedPA	156.606	98.7%	98.2%

REFERENCES

[1] D.-R. Berte, "Defining the iot," in *Proceedings of the International Conference on Business Excellence*, vol. 12, no. 1, 2018, pp. 118–128.

[2] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] W. J. Long and M. P. Quek, "Personal data privacy protection in an age of globalization: the us-eu safe harbor compromise," *Journal of European Public Policy*, vol. 9, no. 3, pp. 325–344, 2002.

[6] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "Mlbase: A distributed machine-learning system." in *Cidr*, vol. 1, 2013, pp. 2–1.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[8] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[9] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *arXiv preprint arXiv:1602.05629*, 2016.

[10] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[11] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication-efficient federated learning," *Proceedings of the National Academy of Sciences*, vol. 118, no. 17, 2021.

[12] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[13] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2350–2358.

[14] A. Li, J. Sun, X. Zeng, M. Zhang, H. Li, and Y. Chen, "Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, pp. 42–55.

[15] J. Jiang and L. Hu, "Decentralised federated learning with adaptive partial gradient aggregation," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 230–236, 2020.

[16] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, 2019.

[17] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.