Dynamic Sampling and Selective Masking for Communication-Efficient Federated Learning

Shaoxiong Ji, Department of Computer Science, Aalto University, 02150 Helsinki, Finland
Wenqi Jiang, Computer Science, Columbia University, New York, NY, 10027, USA
Anwar Walid, Mathematics of Network Systems, Nokia Bell Labs, Murray Hill, NJ, 07069, USA
Xue Li, The University of Queensland, Brisbane, QLD, 4072, Australia

Federated learning (FL) is a novel machine learning setting that enables on-device intelligence via decentralized training and federated optimization. Deep neural networks’ rapid development facilitates the learning techniques for modeling complex problems and emerges into federated deep learning under the federated setting. However, the tremendous amount of model parameters burdens the communication network with a high load of transportation. This article introduces two approaches for improving communication efficiency by dynamic sampling and top-$k$ selective masking. The former controls the fraction of selected client models dynamically, while the latter selects parameters with top-$k$ largest values of difference for federated updating. Experiments on convolutional image classification and recurrent language modeling are conducted on three public datasets to show our proposed methods’ effectiveness.

The widespread applications of mobile communication technology and personal mobile devices are turning machine learning to edge devices and making distributed agents more intelligent. Federated learning (FL) is a decentralized machine learning method, which uses distributed training on local users without sending the data to a central server. Unlike centralized learning, FL trains the model without directly accessing private user data. Local computing and secure parameter transport can solve the security and privacy issues in traditional centralized training. In this new setting, users’ data are stored securely on their own devices.

Federated deep learning takes deep neural networks as learning models on local devices and uses averaged model aggregation over sampled clients under the control of a central server. However, the humongous amount of transport cost in federated deep learning is one bottleneck for real-world applications because deep learning models have many parameters. Modern devices use wired communication or wireless communication with limited bandwidth. Thus, it is hard for current network transportation to handle such a large amount of transportation, making communication-efficient FL a critical mission.

There are two strategies to increase communication efficiency, i.e., to reduce the number of communication rounds between server and clients during federated training, and to transport fewer parameters for each federated iteration. To save the transportation cost, current work uses static sampling to select a fraction of client models, while other improvements use compression algorithms for more efficient communication. This article improves the vanilla federated averaging methods by dynamic sampling for client models and selective masking on client models’ neural parameters. The former algorithm aims to reduce communication rounds, while the latter aims to sample a fraction of the parameters for transportation.
Our proposed methods can also be combined with cutting-edge compression algorithms for furthering communication efficiency during model downloads and uploads.

Our contributions are summarized in the following.

- We propose a dynamic sampling strategy for federated averaging with exponential annealing of the sampling rate.
- We propose selective masking on client models’ neural parameters to save the amount of data during model transportation.
- Through experiments on convolutional image classification tasks on two popular image datasets and recurrent language modeling tasks on a typical text dataset, our proposed methods outperform their baseline methods in most cases.

**RELATED WORK**

FL performs distributed training and learns efficiently from decentralized data using iterative averaging. It has a wide range of application in wireless communication.\(^5\)\(^,\)\(^13\) Several improvements to the federated framework have been proposed, including per-user domain adaptation\(^9\) and attentive aggregation.\(^4\)\(^,\)\(^5\) Communication efficiency is considered a vital evaluation metric of federated learning. To enable efficient communication, Konečný et al.\(^5\) proposed sketched updating methods for reducing transport costs in FL and use random subsampling in their sketched updating method. Bonawitz et al.\(^1\) designed a scalable FL system architecture. Recently, several works proposed different strategies to improve the communication efficiency, including the sparse ternary compression;\(^12\) Count Sketch-based compression;\(^11\) parameter quantization;\(^10\) and federated ensembles.\(^3\)

**PRELIMINARIES**

**Federated Averaging**

There is a central server for model aggregation in the federated setting and a set of client devices for local training. One typical algorithm is federated averaging, where the central server averages client models to obtain a global model that can well generalize distributed clients. For deep neural networks, the global model at \(t\) is denoted as \(\Theta_t = \{W_t^1, \ldots, W_t^L\}\) with multiple layers, where a real-value matrix \(W_t^i \in \mathbb{R}^{d_{i-1} \times d_i}\) represents parameters in the \(i\)th layer. The parameters of the global model are then distributed to clients via network transportation. For each selected client, the downloaded model is trained on its physical device using its data, with the trained local model in the \(i\)th client at \(t\)th communication round denoted as \(\Theta_t^i\). Then, clients upload trained models to the central server for model aggregation by computing the weighted average. The aggregated global model is computed as

\[
\Theta_{t+1} = \frac{1}{m} \sum_{i \in S} \Theta_t^i
\]

where \(m\) is the total number of selected clients as a set \(S\) in the distributed environment, For each layer in the neural network model, pair-wise matrix summation is calculated. To consider data imbalance, federated averaging performs weighted averaging by taking number of training samples as weights. The aggregation of client models is calculated as

\[
\Theta_{t+1} = \frac{1}{m} \sum_{i \in S} \frac{n_i}{n} \Theta_t^i
\]

where \(n_i\) is the number of training instances in the \(i\)th device and \(n = \sum_{i=0}^{m} n_i\) is the total number of training samples of all selected clients.

**Static Sampling**

FL applies static sampling by selecting a random fraction of clients for federated averaging. The server initially sets a sampling rate of \(C\), and then it waits for updates from clients. Once there are enough updates to meet the sampling rate, the server will stop receiving updates and turn to federated averaging. During this procedure, the sampling rate of \(C\) remains unchanged, which is why it is a so-called static sampling.

**Random Masking**

In a distributed computing environment, the bottleneck of FL is the vast amount of transport cost. The bandwidth of the central server is fixed when receiving updates from distributed clients. A straightforward strategy is to randomly select a part of updated model parameters for transportation to save the transportation cost. We term this as random masking because a random mask is applied to the parameters of each neural layer. Random masking is randomized by a random function given a random seed in each client, which generates a matrix of \(A \in \mathbb{R}^{a \times b}\) with \(y\) of ones. A certain proportion of parameters is masked via the pair-wise product of \(A\) matrix and parameters in each layer. Then, the masked model is compressed when uploaded to the central server.
**PROPOSED METHODS**

We propose two strategies to save the communication cost. They are dynamic sampling and selective masking. The iterative model aggregation method uses federated averaging. Dynamic sampling is proposed to control the sampling rate of client models. At the same time, selective masking chooses parameters according to the absolute difference value of parameters with a preset proportion of $\gamma$ and saves communication costs.

**Dynamic Sampling**

Static subsampling uses a fixed subsample rate throughout the training process, no matter the epochs and training steps. This method is secure for implementation by evenly choosing the number of clients to enable model aggregation. We propose dynamic sampling with a high sampling rate first, and then decrease the sampling rate during each communication. Our motivation is to accelerate convergence at the beginning of FL by involving more clients for model aggregation at the very beginning. Once a more generalized federated model is trained based on the initialization, our method dynamically decreases the number of clients for model aggregation to save communication costs. Even though it costs more at the beginning of federated training, its selected number of clients’ model declines rapidly after several rounds of training. The declining rate of sampling rate can be chosen accordingly to ensure that the total amount of parameter transportation of dynamic sampling is fewer than its counterpart of static sampling after individual rounds of communication.

Our proposed dynamic subsampling method uses an exponential decay rate to anneal the sampling rate in the training process, where the subsample rate $R$ is a function of current epoch $t$ and decay coefficient $\beta$ as shown in

$$R(t, \beta) = \frac{1}{\exp(\beta t)}.$$  

(3)

With the decreasing rate multiplied with a preset initial sample rate $C$, we get the dynamic sampling rate as $c = \frac{C}{\exp(\beta)}$ at the $t$th training round. With the increase of communication rounds, the sampling rate becomes very small, making less than one client selected for model aggregation. In practice, the minimum number of selected client models is set to two. Integrated with federated averaging, the algorithm of dynamic sampling is written in Algorithm 1. The core difference between static sampling is the dynamically changed sampling rate.

Algorithm 1. Federated Averaging With Dynamic Sampling

1: $C \in \mathbb{R}$ is a constant of initial sampling rate; $c \in \mathbb{R}$ is a variable of the sampling fraction of clients; $R \in \mathbb{N}^+$ is the value of preset communication rounds; $k \in \mathbb{R}$ is a constant of the decay coefficient in (3).

2: **Input:** a set of $M$ registered clients $S = \{s_1, \ldots, s_M\}$

3: **Output:** updated global parameters $\Theta_{t+1}$

4: **procedure** Federated Averaging $\triangleright$ Run on a central server

5: Initialize model $\Theta_0$, set the decay efficiency of sampling $\beta$.

6: for $t = 1 : R$ do

7: Initialize an empty list $L$

8: Calculate sample rate $c = \frac{C}{\exp(\beta t)}$

9: Number of sampled clients $m = \max(c \times M, 1)$

10: while $\text{len}(L) < m$ do $\triangleright$ Dynamic sampling

11: Send connection request to clients

12: if ACK from $i$-th client then

13: call ClientUpdate($i$, $\Theta_t$)

14: $\Theta_i^t \leftarrow \text{receiveParam}(i)$

15: $L.\text{add}(\Theta_i^t)$

16: $\Theta_{t+1} = \frac{1}{m} \sum_{i \in L} n_i \Theta_i^t$

Selecting Masking

This section turns to another improvement—the selective masking technique. Model parameters stored in client devices take up most of the transportation cost. When transporting data from user devices to central servers, the random masking algorithm randomly selects some computed updates and discards the rest. However, this method is less heuristic because it is unable to select prior updates. Consequently, some important updates can be discarded when it randomly selects parameters.

We propose selective masking to consider the importance of model parameters in each local training. Given a static masking rate on the proportion of model parameters as the selective criteria, only model parameters with the largest absolute difference are selected proportionally for federated aggregation and model updating. First, the difference of current model parameters of the $i$th layer and updated counterpart in the next time step is calculated as

$$D_i^t = |W_i^{t+1} - W_i^t|.$$  

(4)

Given the masking proportion of $\gamma$, top-$k$ largest values are selected together with their indices, where $k$ equals $\gamma$ multiplied with the number of elements of the weight matrix. A mask matrix is generated with the selected indices as $M$, which contains $1 - \gamma$ zeros and $\gamma$ ones. Then, the masked weight matrix is
calculated by the pair-wise product of mask matrix and full weight matrix as

$$W_{t+1}^{ij} = M \otimes W_{t+1}^{ij}. \quad (5)$$

Finally, the masked parameters are compressed and ready for transportation via client–server communication.

The algorithm of on-device training with selective masking is written in Algorithm 2, where the genMask function generates the mask matrix according to the top-k indices.

**Algorithm 2. On-Device Training With Selective Masking**

- $\gamma$ is the proportion of masked parameters; $B$ is the local mini-batch size; $E$ is the number of local epochs; $\eta$ is the learning rate.
- **Input**: ordinal of user $k$, user data $X$.
- **Output**: updated user parameters $\Theta_{t+1}$ at $t + 1$.
- **procedure** Client Update($k, \Theta$) $\triangleright$ Run on the $k$-th client

```plaintext
1: for each layer $i$ from 1 to $l$ do
2:     for each local epoch $i$ from 1 to $E$ do
3:         for each batch $b$ of data $X$ into batches do
4:             $\Theta_{t+1} \leftarrow \Theta_t - \eta \nabla L(\Theta_t)$
5:             $M = \text{genMask}(\{0, 1\}, a, b, \text{indices})$
6:             $W_{t+1}^{ij} = M \otimes W_{t+1}^{ij}$
7:         end for
8:     end for
9:     end for
```

**EXPERIMENTS**

**Datasets and Settings**

**Datasets**

We perform several neural architectures on two typical tasks—image classification and language modeling—on two image datasets of MNIST and CIFAR-10, and a natural language dataset of WikiText-2. They are typical public datasets for training machine learning algorithms. MNIST\(^a\) and CIFAR-10\(^b\) are two widely used datasets with handwritten digits and objects, respectively. WikiText-2\(^c\) is a dataset for word-level language modeling, which contains more than two million tokens extracted from Wikipedia. Our experiments are performed on these three widely used datasets for validation. Statistics of these three datasets are summarized as in Table 1.

**Table 1. Summary of datasets.**

| Dataset       | Type    | # train | # test |
|---------------|---------|---------|--------|
| MNIST         | Image   | 60,000  | 10,000 |
| CIFAR-10      | Image   | 50,000  | 10,000 |
| WikiText-2    | Token   | 2,088,628 | 245,569 |

**Data Partitioning**

Those three datasets are designed for training a centralized machine learning model. Data partitioning is adopted for generating decentralized datasets by sampling the whole dataset under independent and identical distribution (IID). We followed the data partitioning rule of MNIST and CIFAR-10 proposed by McMahan et al.\(^d\) Each partitioned subset of the whole dataset is regarded as a private dataset in a client device. The same partitioning rule is then applied to WikiText-2 to construct the IID federated dataset.

**Settings**

We conduct both convolutional and recurrent neural networks according to specific tasks. For the image classification task, we use LeNet as a client model, and for furthering the effect of a large-scale model, VGG is implemented as a client model on the CIFAR-10 dataset. The language modeling task deals with sequential text data. Thus, we use long short-term memory (LSTM) networks to capture sequential dependency with tied and untied word embedding. All neural models are implemented by the PyTorch framework and accelerated by Nvidia K40m GPU.

Our aim is to enable communication-efficient FL. Thus, transportation cost is considered for evaluation, which is related to sampling rate, masking rate, and communication round. Taking a single communication between one client and the server with full model parameters as unit, it is calculated as

$$f(\beta, \gamma) = \frac{\gamma}{R} \sum_{r=1}^{R} \frac{C}{\exp(\beta r)} \quad (6)$$

for $R$ rounds of client–server communication. In the following section, we take transportation costs together with prediction accuracy as evaluation metrics.

**Convolutional Image Classification**

We first begin with the task of convolutional image classification in the federated setting. Two groups of comparisons are conducted on static versus dynamic

\(^a\)Available at http://yann.lecun.com/exdb/mnist/

\(^b\)Available at https://www.cs.toronto.edu/~kriz/cifar.html

\(^c\)Available at https://s3.amazonaws.com/research.metamind.io/wikitext/wikitext-2-v1.zip

---

IEEE Intelligent Systems March/April 2022
sampling, and random versus selective masking on MNIST digits classification using LeNet. Then, an analogical comparison is further performed using a large-scale VGG model on CIFAR-10 objects classification. Due to the sample rate decay, the dynamic sampling method can train more federated rounds than the static method, given the same transport cost and the same initial sampling rate. For example, with a decay coefficient of 0.1 and the same amount of transporta-
tion cost, the dynamic method can update 31 epochs. In comparison, the static method can only train 10 epochs of updates. In the following experimental analysis, a convolutional image classification task is performed under separate comparison first, and then with two methods combined for federated training.

Static Versus Dynamic Sampling
Sampling strategies are conducted in static and dynamic manners. We take the whole cohort of clients as initial training. The static sampling method keeps this rate during the whole process of federated training. At the same time, the dynamic counterpart takes an exponential decay on the sampling rate with the decay coefficient in (3) of 0.01 and 0.1. Results of prediction accuracy and communication cost are reported in Figure 1 after 10, 50, and 100 rounds of federated training. According to Figure 1(a), with the increase of training epochs, prediction accuracy grows steadily for all three settings. For fewer federated training rounds (that is, 10 and 50 rounds), the testing accuracy of static sampling is more insufficient than dynamic sampling with 0.01 as the decay coefficient. However, when the sampling rate drops faster (i.e., a higher decay coefficient of 0.1), testing accuracy of dynamic sampling is imparted. After 50 and 100 rounds of training, static sampling gains a better prediction performance than the dynamic sampling method. As for the transportation cost during federated communication, the static method takes 100% of transportation, while the dynamic sampling method can efficiently save communication costs. With the increase of training epochs and the decay coefficient, many more rounds of client–server communication have been saved, as illustrated in Figure 1(b).

Random Versus Selective Masking
After that, we evaluate the performance of the selective method using top- \( k \) masking. This section fixes the sampling rate to be 0.1 and conducts experiments with random masking and selective masking. For a fair comparison, these two methods use the same hyperparameter setting for 10 rounds of training, and the learning rate is set to 0.01. Experimental results are reported in Figure 2, where the masking rate varies from 0.1 to 0.9. With a relatively higher masking rate, the testing accuracy of random masking and selective masking is close. Selective masking performs a bit higher testing accuracy for a masking rate of 0.8 and 0.9. When a large number of parameters are discarded with a rate of 0.1 and 0.2, the performance of random masking drops dramatically. According to this result, our proposed top- \( k \) selective masking method can maintain a stable performance to save communication costs even with a high proportion of parameters ignored.

Combined Experiment
After an independent comparison of two proposed methods, we combine them into federated training for evaluation. In this section, four initial sampling rates of 0.3, 0.5, 0.7, and 1.0 are included in the dynamic sampling method. As for the selective masking, two decay coefficients of 0.01 and 0.1 are used. Experimental results after 50 training rounds are shown in the two bar charts of Figure 3. Selective masking outperforms random masking in the dynamic sampling setting of these two cases except when the initial sampling rate equals 1 with the decay coefficient of 0.01.
Experiments on CIFAR-10

Then, we further experiment on the CIFAR-10 dataset using the VGG-16 model to evaluate the performance on a large-scale model. The aggregated prediction accuracy of random and selective masking after 100 federated training rounds is shown in Figure 4, where static sampling with a 100% sampling rate is applied. Notice that we aim to compare the performance of masking methods but not the image classification model’s performance. Thus, we do not tune client learners hard. All the experiments have not achieved the state-of-the-art results of centralized training as the comparison is conducted in the federated setting with limited communication rounds. However, our comparison is reported with the same set of client learners to ensure a fair comparison of the two proposed methods and their counterparts. As we can see from the table, the top-k selective masking method outperforms random masking for a masking rate from 0.1 to 0.6. When the masking rate is high, these two methods gain similar testing accuracy. Our proposed top-k selective masking can maintain satisfactory performance with a large proportion of parameters saved in federated training.

Besides, we conduct experiments with both sampling and masking strategies adopted for deeper analysis. This comparison aims to evaluate the effect of the decay coefficient in dynamic sampling with masked updating applied. The results of using masking rates of 0.3, 0.5, 0.7, and 0.9 are reported in Figure 5 where the x-axis is log-scaled. These figures show that when the masking rate is 0.3, selective masking outperforms random masking with all settings of the decay coefficient. For the masking rate being 0.5 and 0.7, selective masking gains better testing accuracy in most cases. With a higher masking rate of 0.9, the performance gap between these two methods is narrow. Generally, with a larger decay coefficient (more communication-efficient), the performance experiences a fluctuation and decreases to a relatively low level when the decay coefficient is set as 0.5.

Recurrent Language Modeling

Mobile keyboard suggestion with private neural language modeling is a typical application of FL, which interacts with users to provide supervised labels. This section models the next word prediction in the mobile keyboard as private RNN-based language modeling. Specifically, we adopted the gated recurrent unit (GRU) as the client learner. GRU is a simplified variant of the LSTM network, which suits saving communication costs with fewer parameters. The natural language corpus usually has a vast vocabulary. To further communication-efficient FL, tying word embedding and word classifier are introduced by using shared parameters. In this section’s experiments, tried embedding is applied. For the evaluation metric, we use the aggregated perplexity. Perplexity is a standard
metric for language modeling tasks. According to its definition, lower perplexity means better performance.

We first compared the effect of sampling strategies. Figure 6 takes 50 rounds of client–server communication with different rates of masking to compare the performance of static and dynamic sampling. This bar chart shows that dynamic sampling achieves a lower perplexity in most cases, excluding $\beta = 0.5$ with a masking rate of 0.5 and 0.7, and $\beta = 0.1$ with a masking rate of 0.8 and 0.9.

Second, we compared random masking with selective masking. Results using different masking rates are reported in Figure 7. Our proposed selective masking is better for larger masking rates. Surprisingly, random masking gains better performance when the masking rate is low. It is hard to interpret why random masking is better, even when many parameters are discarded for updating. One possible guess is that the randomness improves the generalization of an aggregated recurrent model, making the testing perplexity decrease.

Discussion
Comprehensive experiments on a single computing node are conducted to mimic the federated setting in this section. Two tasks of convolutional image classification and recurrent language modeling are performed with comparative analysis. Our proposed methods achieve competitive performance in most experimental settings, which provides empirical approaches for saving communication costs in the federated setting. For simplicity, we ignore the loss of network transmission. However, admittedly, the federated setting in the real-world environment is more complicated, which requires further simulation experiments. Deep neural models have a large number of parameters. Taking VGG-16 as an example, its total number of parameters is more than one hundred million. High-speed network techniques are also required. Due to the lack of computing resources, we leave experimental simulations on multiple computing nodes for future work.

**CONCLUSION**

FL decouples modeling training and data accessing, which protects data privacy. However, it incurs a communication cost issue when combined with large-scale deep neural networks. This article proposes two empirical approaches—dynamic sampling and selective masking to save communication costs while ensuring satisfying prediction performance. The proposed two strategies can save server–client communication and save the number of model parameters for each transmission. Experiments on convolutional image classification and recurrent language modeling show that our proposed methods gain competitive results.

**REFERENCES**

1. K. Bonawitz et al., "Towards federated learning at scale: System design,” in Proc. Mach. Learn. Syst., 2019, vol. 1, 2019, pp. 374–388. [Online]. Available: https://proceedings.mlsys.org/paper/2019/file/bd686fd640be98efaae0091fa301e613-Paper.pdf

2. M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, “A joint learning and communications framework for federated learning over wireless networks,” IEEE Trans. Wireless Commun., vol. 20, no. 1, pp. 269–283, Jan. 2021, doi: 10.1109/TWC.2020.3024629.

3. J. Hamer, M. Mohri, and A. T. Suresh, “FedBoost: A communication-efficient algorithm for federated learning,” in Proc. 37th Int. Conf. Mach. Learn., 2020, pp. 3973–3983. [Online]. Available: https://proceedings.mlr.press/v119/hamer20a.html

4. S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, “Learning private neural language modeling with attentive aggregation,” in Proc. Int. Joint Conf. Neural Netw., 2019, pp. 1–8, doi: 10.1109/IJCNN.2019.8852464.
5. J. Jiang, S. Ji, and G. Long, “Decentralized knowledge acquisition for mobile internet applications,” World Wide Web, vol. 23, pp. 2653–2669, 2020, doi: 10.1007/s11280-019-00775-w.

6. J. Končný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” 2016.

7. Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, “A secure federated transfer learning framework,” IEEE Intell. Syst., vol. 35, no. 4, pp. 70–82, Jul./Aug. 2020, doi: 10.1109/MIS.2020.2988525.

8. H. B. McMahan, E. et al. “Communication-efficient learning of deep networks from decentralized data,” in Proc. 20th Int. Conf. Artif. Intell. Statist., 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

9. D. Peterson, P. Kanani, and V. J. Marathe, “Private federated learning with domain adaptation,” Presented at the Workshop on Federated Learning for Data Privacy and Confidentiality (in Conjunction with NeurIPS), 2019.

10. A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, “FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization,” in Proc. Int. Conf. Artif. Intell. Statist., vol. 108, 2020, pp. 2021–2031. [Online]. Available: https://proceedings.mlr.press/v108/reisizadeh20a.html

11. D. Rothchild et al., “FetchSGD: Communication-efficient federated learning with sketching,” in Proc. Int. Conf. Mach. Learn., vol. 119, 2020, pp. 8253–8265. [Online]. Available: https://proceedings.mlr.press/v119/rothchild20a.html

12. F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, “Robust and communication-efficient federated learning from non-iid data,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 9, pp. 3400–3413, 2020, doi: 10.1109/TNNLS.2019.2944481.

13. Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, “Energy efficient federated learning over wireless communication networks,” IEEE Trans. Wireless Commun., vol. 20, no. 3, pp. 1935–1949, Mar. 2021, doi: 10.1109/TWC.2020.3037554.

14. H. Zheng, H. Hu, and Z. Han, “Preserving user privacy for machine learning: Local differential privacy or federated machine learning,” IEEE Intell. Syst., vol. 35, no. 4, pp. 5–14, Jul./Aug. 2020, doi: 10.1109/MIS.2020.3010335.

WENQI JIANG received the bachelor’s degree from the Huazhong University of Science and Technology, Wuhan, China. He is currently a postgraduate student at Columbia University, New York, NY, USA. He is also an adjunct professor at Columbia University, New York, NY, USA. His research interests include heterogeneous parallel computing, distributed system, and machine learning. Contact him at wenqi.j@columbia.edu.

ANWAR WALID is the director of the Network Intelligence and Distributed Systems Research Group, Nokia Bell Labs, Murray Hill, NJ, USA. He is also an adjunct professor at Columbia University, New York, NY, USA. His research interests include control and optimization of distributed systems; statistical learning models and algorithms with applications to network automation and slicing, machine learning, Internet of Things (IoT), cloud computing, digital health, and smart transportation. He is an IEEE Fellow. He is the Senior Editor for the IEEE Journal on Selected Areas in Communications, and an Associate Editor for the IEEE Transactions on Network Science and Engineering and IEEE Network Magazine. Contact him at anwar.walid@nokia-bell-labs.com.

XUE LI received the Ph.D. degree from the Queensland University of Technology, Brisbane, QLD, Australia, in 1997. He is currently a professor with the School of Information Technology and Electrical Engineering, University of Queensland (UQ), Brisbane, QLD, Australia. He also holds a professor title with the School of Medicine, Griffith University, Gold Coast, QLD, Australia. His major areas of research interests and expertise include machine learning, health data analytics, data mining, social computing, and intelligent web information systems. Contact him at xueli@itee.uq.edu.au.