DAFL: Delay Efficient Federated Learning over Mobile Devices via Device-to-Device Transmissions

Huai-an Su*, Pavana Prakash*, Rui Chen*, Yanmin Gong†, Rong Yu‡, Xin Fu* and Miao Pan*
*Department of Electrical and Computer Engineering, University of Houston, Houston, TX
‡School of Automation, Guangdong University of Technology, Guangzhou, China
†Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10437066

***Abstract*—Federated learning (FL) over mobile devices is an emerging distributed learning paradigm for numerous delay sensitive applications. In FL, the training delay is composed of the computing and communication delay. Some of the participating mobile devices may have slow local computing or wireless communications, which results in high FL training delay. Intuitively, if fast devices help slow ones, the FL training delay can potentially be reduced. However, helping each other among devices requires frequent transmissions and may cause additional delay. Fortunately, we observe that Device-to-Device (D2D) transmission, a fast and direct transmission, may be applied between device pairs to mitigate the additional delay from frequent transmissions. Inspired by those observations, we develop the D2D transmission assisted FL (DAFL), a novel FL scheme to improve the training delay over mobile devices. Briefly, we first put the eligible mobile devices into pairs, each pair consisting of a fast and a slow device. Then, we apply D2D transmission between each device pair to: (1) improve the transmission delay of each pair, by letting the fast device help with the model parameters transmission to the server, and (2) improve the computing delay by splitting learning task between paired devices. The emulation results demonstrate that DAFL surpasses existing peer designs in terms of reducing training delay by more than 20%.**

***Index Terms*—Federated learning, Delay efficiency, D2D transmissions, Split learning.**

I. Introduction

Nowadays, federated learning (FL) is considered a promising distributed deep learning (DL) solution, which can yield efficient and privacy-preserving collaborative training over many devices [1]. Meanwhile, with ever-increasing computing capability, mobile devices can conduct the on-device training of more and more deep neural network (DNN) models. FL over mobile devices has prompted numerous applications, such as keyboard predictions [2], physical hazard detection in smart homes [3], health event detection [4], etc. A key obstacle to unleash the full potential of FL over mobile devices is the long training delay, especially for delay-sensitive mobile applications. One of the possible reasons behind this is the device heterogeneity among FL clients, which can be defined as diverse computing capabilities and communication conditions [5]. Among those mobile devices, "slow" performing ones are considered stragglers for the FL training. In FL, the server and other devices may need to wait for all the stragglers to finish their local updates. Therefore, the FL training delay is bottlenecked by such slow devices.

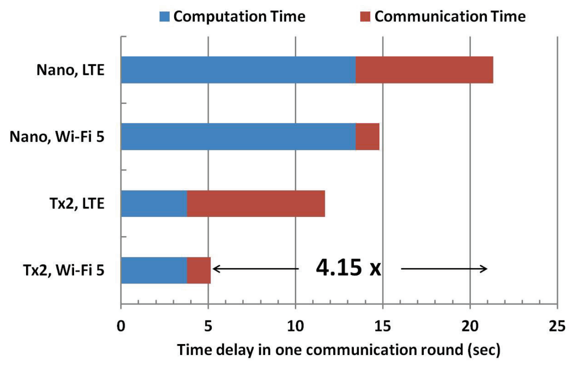


Fig. 1. Time delay of a single-round local update on different hardware platforms with varying communication conditions.

For illustrative purpose, we conducted an empirical study on the training delay by performing one round of FL on ResNet20 model over independent and identically distributed (IID) CIFAR10 dataset. We evaluated the training delay of local training on two NVIDIA Jetson family platforms (i.e., Nano and TX2), under different wireless accessing technologies, i.e., LTE and Wi-Fi 5, respectively. As shown in Fig. 1, the training delay per round on Nano using LTE communication is 4.15 times more than that of TX2 using Wi-Fi 5. This huge delay gap stems from the device heterogeneity, i.e., the differences of computing and communication capabilities. In terms of computing, TX2 (1.33 TFLOPs) has a higher computing capacity than Nano (472 GFLOPs). In terms of communications, Wi-Fi 5 (100 Mbps) has a faster transmission rate than LTE (20 Mbps).

In the literature, there are many existing approaches to address straggler issues due to device heterogeneity [6]–[10]. For example, the participation selection approaches in [8]–[10] allow FL server to select the fast devices for performing model updates and abandon the stragglers. However, since only the fast devices are chosen for partial participation, the training may become biased over slow devices, which can lead to accuracy drop [11], [12]. Asynchronous FL [13] is another possible solution, which allows for asynchronous local model updates for FL aggregation. Nevertheless, the slow devices may become stale to the global model, which may cause the accuracy drop of FL training [8].

To reduce training delay while keeping learning accuracy, an intuitive idea is to let the fast devices (i.e., high computing and communication capability) help slow devices (i.e., low computing and communication capability) to conduct the training. However, such an attempt is challenging because the frequent transmissions among devices may incur extra large transmission delays. Fortunately, we find that Device-to-Device (D2D) transmission (e.g., Wi-Fi Direct) [14] offers a direct and fast communication option among nearby mobile devices, which can serve as an enabling technology to facilitate fast devices to help slow ones in FL training.

Specifically, to reduce the communication delay, the slow device, which has poor wireless connections with FL server, can send its model parameters to the fast device, which has good wireless connections with FL server, via D2D communications. The fast device then aggregates the two model updates from both the slow device and fast device, and uploads the aggregated update to the FL server. In this way, the transmission delay of the slow device and thus that of FL training can be potentially reduced. Furthermore, to reduce the computing delay, we propose to apply split learning (SL) [15] to collaboratively train the slow device's local model between the fast device and slow device. The SL activation exchange between the slow device and the fast device will be transmitted via high-speed D2D communications. In this way, SL splits the training workload between slow and fast devices, which is promising to reduce the computing delay of the slow device.

Despite the benefits above, there are two major challenges to fulfill the idea of ”fast device helping slow device”. First, how to pair slow and fast devices given their location constraints to effectively reduce the training delay in FL? Second, how to optimally determine the model layer to split between the paired slow and fast devices? Note that the delay of SL is jointly determined by the activation transmission and the split model training between slow and fast devices [16].

To address those challenges, in this paper, we propose a D2D transmission assisted FL (DAFL) scheme to improve delay efficiency of FL over mobile devices. DAFL design spans over multi-lateral knowledge of D2D communications, matching theory and FL, and the main contributions of DAFL are summarized as follows.

- We introduce DAFL framework which leverages D2D communications and SL to reduce the communication and computing delay in FL training.
- In DAFL, we exploit matching theory to pair the slow and fast devices considering their geo-location constraints, where the preferences are based on the computing and communication latency of the mobile device per FL round.
- To reduce computing delay, we employ SL to split the local computing workload between the paired slow and fast devices, and identify the optimal model layer to split, through in-depth analysis.
- We perform emulations with a group of mobile devices in a given area and show that DAFL outperforms its peer designs and is effective in improving delay efficiency in FL training.

II. System Model

We consider a set of $\mathcal{N}$ devices as clients and a server in a federated learning system. For computation, given any client device $n \in \mathcal{N}$, we let p_n be the AI performance (FLOPs) of device n. For communication, we let $r_n = r(n, S)$ denote the transmission rate between device n and the server S. We let $r_{n,m} = r(n, m)$ denote the transmission rate between client devices n and m, known as the D2D transmission.

To determine whether a device has fast or slow computing, we use common computing devices in the real world as reference. The following devices are Jetson Nano (472 GFLOPs), Jetson TX2 (1.33 TFLOPs), and Jetson Xavier NX (21 TFLOPs), indicating their AI performances for computing. We determine 1 TFLOPs as the threshold between fast and slow computing. To determine whether a device has fast or slow transmission, we use real world transmission rate of Wi-Fi 5 (50 Mbps) and LTE (10 Mbps) as references to define 10 Mbps as the threshold between fast and slow transmission devices. In addition, given any two devices n and m, the D2D transmission rate $r_{n,m}$ is always higher than the transmission rate between individual device and the server (r_n and r_m). Such condition can be indicated by the following relation,

$$\max_{\pi \in \{n,m\}} \{r_\pi\} < r_{n,m}. \tag{1}$$

We consider a DNN of $\mathcal{L} = \{1, 2, ..., L\}$ layers. Let M_i^m, M_i^n denote the i-th layer of device m, n model. Hence, the full model for device n and m are $\sum_{i=1}^{L} M_i^n$ and $\sum_{i=1}^{L} M_i^m$. Let $T^{c_n}(\cdot)$ and $T^{c_m}(\cdot)$ be delay functions for device n and device m respectively, in terms of AI performance. We denote the computing delay of the i-th layer of DNN on device n and device m by $T_i^{c_n} = T^{c_n}(p_n, M_i^n)$ and $T_i^{c_m} = T^{c_n}(p_m, M_i^m)$. Next, we define a function $T^{tr}(\cdot)$ to represent parameter transmission delay. Then, the transmission delay for the i-th layer between device n and server is denoted by $T_i^{tr_n} = T^{tr}(r_n)$, and the transmission delay for the i-th layer between client device n and m is denoted by $T_i^{tr_{n,m}} = T^{tr}(r_{n,m})$. The transmission delay in this paper consists of both uplink and downlink delay. Consequently, the goal of our work is to find the minimum training delay $T_{n,m}$ for each pair of devices$(n, m) \in \mathcal{N}$.

III. D2D Assisted FL over Mobile Devices

We divide DAFL into three sections: pre-processing, training process, and post-processing sections. In the pre-processing section, we pair up devices with each pair consisting of one fast and one slow device. In the training section, we run each pair of devices under a designed process, where model upload assisting improves the transmission delay and SL helps with the computing delay. In the post-processing section, we apply partial client participation by selecting the devices with better performance, in terms of model convergence contribution.

A. Pre-processing: matching based device pairing

We consider a set of heterogeneous devices distributed in a given area. Each individual device has its own computation and transmission speed. Our goal is to pair up these devices, with each pair consisting of one slow training device and one fast training device (computation + transmission delay).

We pair the devices based on the concept of stable marriage matching, a bipartite one-to-one matching problem with two-sided preferences [17]. Originally in stable marriage matching, clients find their own partner to pair with based on their preference list. In our work, however, we use the central server to determine the pair arrangement. We first have clients report their training delay (computation + transmission) to the server. Then, server splits the devices into two sides, $\mathcal{S}$ consisting of slow training devices and $\mathcal{F}$ consisting of fast training devices. T_k and T_l denote the training delay of fast training device k, l for one round, where $k, l \in \mathcal{F}$. For a slow training device u where $u \in \mathcal{S}$, we can construct a preference relation $\succ_u$ over all fast devices as follows,

$$\begin{aligned} &k \succ_u l \Leftrightarrow \frac{1}{T_k} \times v_{u,k} \succ_u \frac{1}{T_l} \times v_{u,l},\ k, l \in \mathcal{F}, \\ &v_{u,k} = \begin{cases} 1, & \text{if } d_{u,k} < d_{th}, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{2}$$

Here $d_{u,k}$ denotes the distance between u and k, and d_{th} denotes the D2D transmission distance threshold. T_u and T_v denote the training delay of slow training device u, v for one round, where $u, v \in \mathcal{S}$ For a fast training device k where $k \in \mathcal{F}$, we can construct a preference relation $\succ_k$ over all slow devices as follows,

$$u \succ_k v \Leftrightarrow T_u \times v_{u,k} \succ_k T_v \times v_{v,k},\ u, v \in \mathcal{S}. \tag{3}$$

After constructing the preference for both sides, the server pairs up clients based on the preference list. The outcome of our matching consists of slow-fast device pairs.

B. Training process: D2D assisted training for paired devices

For each pair of devices, there is one fast device and one slow device. We take device pair (n, m) as an example, where device n is the fast device and device m is the slow device. The relation of transmission and delay of the two devices and the server are defined as follows.

$$T_i^{tr_n} = \alpha \times T_i^{tr_m}, \tag{4}$$

$$T_i^{tr_{n,m}} = \beta \times T_i^{tr_m}, \tag{5}$$

where $i \in \mathcal{L}$. Let h denote the number of iterations for local training per transmission round. The training delay of this pair under traditional FL for one communication round is,

$$\begin{aligned} T_{n,m} &= \max_{\pi \in \{n,m\}} \left\{ h \times \sum_{i=1}^{L} T_i^{c_\pi} + \sum_{i=1}^{L} T_i^{tr_\pi} \right\} \\ &= h \times \sum_{i=1}^{L} T_i^{c_m} + \sum_{i=1}^{L} T_i^{tr_m}. \end{aligned} \tag{6}$$

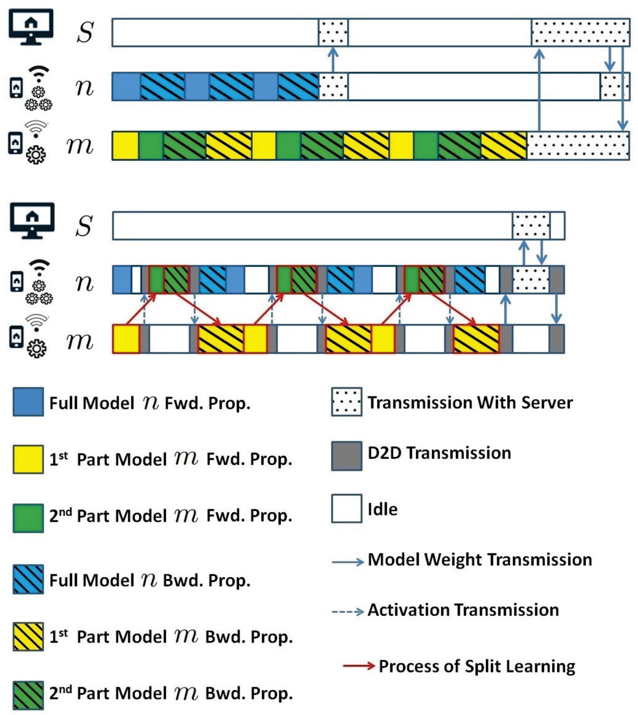

S
n
m
S
n
m
Full Model n Fwd. Prop.
1st Part Model m Fwd. Prop.
2nd Part Model m Fwd. Prop.
Full Model n Bwd. Prop.
1st Part Model m Bwd. Prop.
2nd Part Model m Bwd. Prop.
Transmission With Server
D2D Transmission
Idle
Model Weight Transmission
Activation Transmission
Process of Split Learning

Fig. 2. The sketches of FL training per round for the paired slow and fast devices, where the upper one is traditional FL and the lower one is DAFL.

Specifically, the training assisting process consists of two primary components, namely, SL and model upload assisting. For SL, since device m is the slow device, it is only necessary to perform SL on device m. Hence, the model on device m is split into two parts, with l being the split layer. The first part of the model $\sum_{i=1}^{l} M_i^m$ is on the device m side and the second part $\sum_{i=l+1}^{L} M_i^m$ is on the device n side. At the start, device m performs forward propagation on the first part of the model $\sum_{i=1}^{l} M_i^m$, while device n performs forward propagation on its own full model $\sum_{i=1}^{L} M_i^n$. Later on, device m transmits its activation to device n to perform forward and backward propagation of the second part of the model $\sum_{i=l+1}^{L} M_i^m$. Device n then sends back the gradients at the split layer to device m to finish rest of the backward propagation for $\sum_{i=1}^{l} M_i^m$, meanwhile device n finishes its own full model $\sum_{i=1}^{L} M_i^n$ of backward propagation. After both the devices complete their local training, we perform model upload assisting to aid the slower device m. To achieve this, device m transmits its model parameters to the faster device n through D2D transmission. Device n then aggregates the model parameters from both the devices and uploads to the server.

To measure the training delay of our training assisting method for one communication round, we provide the following relation,

$$\begin{aligned} T_{n,m}^{DAFL} &= \min_{l \in \mathcal{L}} \{ h \times (\sum_{i=1}^{l} T_i^{c_m} + \sum_{i=l+1}^{L} T_i^{c_n} + T_l^{tr_{n,m}}) \} \\ &+ \sum_{i=1}^{L} T_i^{tr_{n,m}} + \sum_{i=1}^{L} T_i^{tr_n} \\ &= h \times (\sum_{i=1}^{l^*} T_i^{c_m} + \sum_{i=l^*+1}^{L} T_i^{c_n} + T_{l^*}^{tr_{n,m}}) \\ &+ \sum_{i=1}^{L} T_i^{tr_{n,m}} + \sum_{i=1}^{L} T_i^{tr_m}. \end{aligned} \tag{7}$$

We subtract Eqn. (6) from Eqn. (7) to show the difference as,

$$\begin{aligned} &T_{n,m}^{DAFL} - T_{n,m} \\ &= h \times (\sum_{i=l^*+1}^{L} (T_i^{c_{n_\mathcal{D}}} - T_i^{c_m}) + T_{l^*}^{tr_{n,m}}) \\ &+ \sum_{i=1}^{L} T_i^{tr_{n,m}} + \sum_{i=1}^{L} T_i^{tr_n} - \sum_{i=1}^{L} T_i^{tr_m} \\ &= h \times (\sum_{i=l^*+1}^{L} (T_i^{c_n} - T_i^{c_m}) + T_{l^*}^{tr_{n,m}}) \\ &+ (\alpha + \beta - 1) \times \sum_{i=1}^{L} T_i^{tr_m}. \end{aligned} \tag{8}$$

To make Eqn. (8) less than 0, our work must satisfy the following condition: $\beta < \alpha < 0.5$. This implies that device n has to be at least twice as fast as device m in terms of transmission speed, and the D2D transmission rate needs to be higher than device n's transmission rate. Moreover, the SL process must meet the potential splitting points condition in $\mathbb{P}$ given by,

$$\mathbb{P} = \{ l \in \mathcal{L} | \sum_{i=l+1}^{L} T_i^{c_n} + T_l^{tr_{n,m}} < \sum_{i=l+1}^{L} T_i^{c_m} \}. \tag{9}$$

The detailed processes per round for device pair (n, m) can be seen in Fig. 2, where the upper scheme shows the process of traditional FL and the lower scheme shows the training assisting method under DAFL.

C. Post-processing: partial client participation

For every round of local training and updating, straggler issues occur when training delay is bottleknecked by slow devices. Despite applying our method, there is still a chance of straggler issues if some slow devices are too far away to be paired up, or if the devices around them are also slow and thus cannot help with the training. Hence, to consider this scenario, at the post-processing section after the training process, we apply partial client participation to evaluate our performance. Overall, for a given round in post-training section, we choose devices with better model convergence to participate [8], [9]. By doing so, a portion of devices that could potentially be the bottleneck of FL will not be selected. Consequently, we address the straggler issue and training delay will be reduced.

IV. Performance Evaluation

A. Experimental setup

We use image classification as the FL task, and choose VGG19 and ResNet20 as three local DNN models. To meet the splitting points condition in Eqn. (9) in Sec. III-B, we select the split layer for SL as the first max-pooling layer for VGG19, and the first batch normalization layer for ResNet20. The open dataset we use is CIFAR10. Each device contains 5000 training data out of a total of $50,000$ training examples for CIFAR10. We use the SGD optimizer with a learning rate of 0.01 for all approaches. The SGD weight decay is set to 0.0001 and the SGD momentum is set to 0.9. The number of local iterations is set to 10 for all approaches. The batch size is set to 64, and the number of communication rounds R is set to 700.

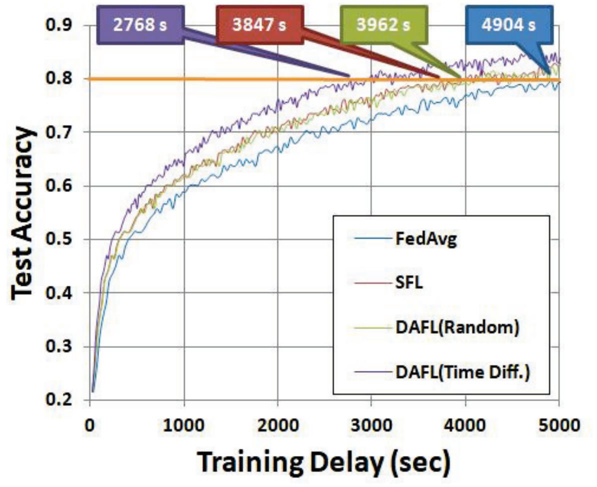
To simulate the computationally limited mobile devices, we use NVIDIA Jetson TX2 and Jetson Nano as our devices. We use NVIDIA RTX 8000 that has 16.31 TFLOPs peak performance and 48GB memory as the server in FL with much higher computation performance. We employ LTE for slow devices' transmissions, and Wi-Fi 5 for fast devices' transmissions to the FL server. Wi-Fi Direct is used for D2D transmissions among paired devices. The transmission rates for Wi-Fi 5, LTE and Wi-Fi Direct are 100 Mbps, 20 Mbps and 150 Mbps, respectively.

We conduct an emulation involving 20 mobile devices in a 30000 m^2 circular area participating in FL process. We use Jetson Nano with LTE and Jetson TX2 with Wi-Fi 5 to represent slow and fast training devices respectively. The devices are randomly assigned in the area, each with the radius of $60m$ representing the transmission range for D2D transmission. After pairing, we apply D2D transmission between the device pairs and SL for the slower device of the pair. After obtaining the training delay, we apply partial client participation to select the devices. We then perform training with the set number of communication rounds and obtain the average training delay.

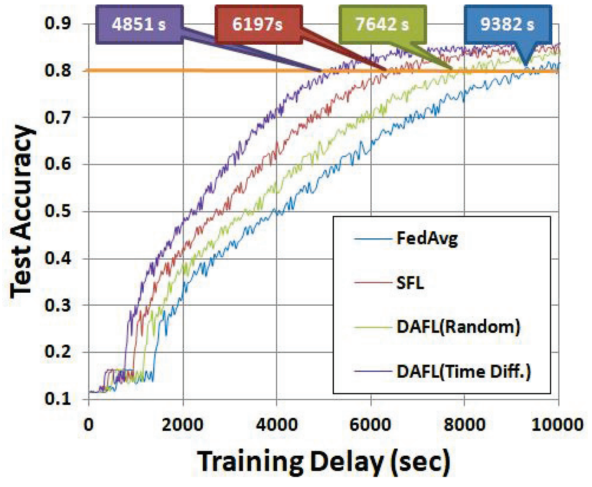
B. Result analysis

We compare DAFL with FedAvg, a general federated learning algorithm with partial client participation to deal with straggler issues due to device heterogeneity. DAFL has the benefit of reducing computation delay with SL and improving transmission delay with D2D transmission. In addition, we take SFL [18] as our other comparison method. Both DAFL and SFL use the concept of SL for computation delay reduction. DAFL, however, applies SL between clients, while SFL performs SL between clients and server. Furthermore, our method holds the benefit of less computing and communication burden on the server side. DAFL also introduces D2D transmission between devices, which can further save transmission time during activation transmission in comparison with SFL.

To analyze the training delay performance, we compare DAFL with other baselines on ResNet20 and VGG19 models

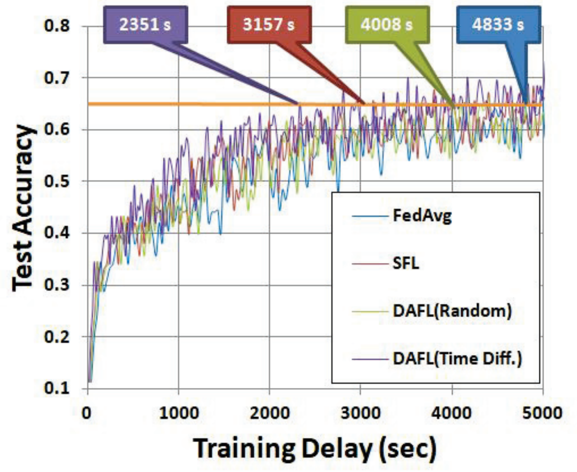


(a) ResNet20

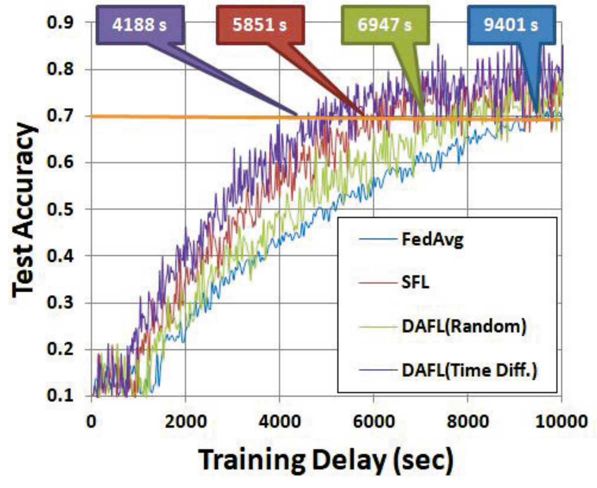


(b) VGG19

Fig. 3. Test accuracy of models ResNet20 and VGG19 under CIFAR10 IID dataset with participation rate $p = 0.8$.



(a) ResNet20



(b) VGG19

Fig. 4. Test accuracy of models ResNet20 and VGG19 under CIFAR10 non-IID dataset with participation rate $p = 0.8$.

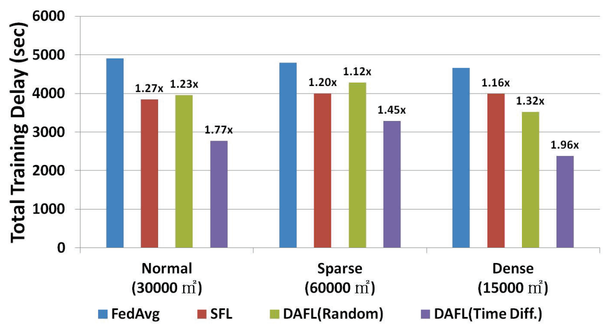
under CIFAR10 dataset, with partial client participation rate $p = 0.8$. From Fig. 3, we can conclude that our method reaches the target accuracy faster with high partial client participation rate. Moreover, our work is viable with non-IID dataset as well. As shown in Fig. 4, our method outperforms the baselines in terms of training delay.

To analyze the impact of device density on the training delay, we aim to alter the size of the area in which the devices are located. In our default set up of device sparsity and distribution, we have 20 client devices randomly distributed in a $30,000m^2$ area and the range for D2D transmission is $30m$. We first determine a sparse scenario, in which 20 devices are randomly distributed in a $60,000m^2$ area. With wider space, devices are more distant to one other and thus, the chance of devices being paired up is lowered. As a result, our work improves less under sparse areas. On the other hand, a dense scenario with 20 devices randomly distributed in $15,000m^2$ area, where devices are closer, the chance of pairing up increases. Therefore, our work improves even more in dense areas. This influence of different space sparsity ratio on training delay is shown in Fig. 5 for IID and non-IID data.

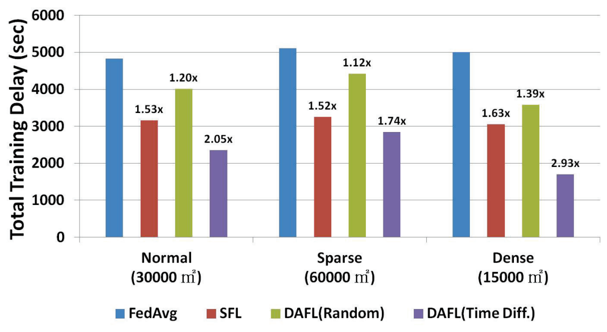
To assess the impact of device location distribution, we choose normal distribution and grid distribution for comparison. This is plotted in Fig. 6 for both IID and non-IID datasets. As seen in the figure, when devices are distributed normally, a majority of them are close to each other, increasing the chance of pairing up. This can be further improved in dense areas as our method works even better.

V. Conclusion

In this paper, we have proposed DAFL, a FL framework that addressed straggler issues due to device heterogeneity and provided delay efficient FL process over mobile devices. We have designed a matching scheme for pairing devices into pairs of fast and slow devices (straggler) in terms of transmission rate and computing capability. For each pair of devices, we have applied a training assisting method to reduce the training delay. The method included D2D transmission for reducing transmission delay, and SL for reducing computing delay of slow devices. We have also applied partial client

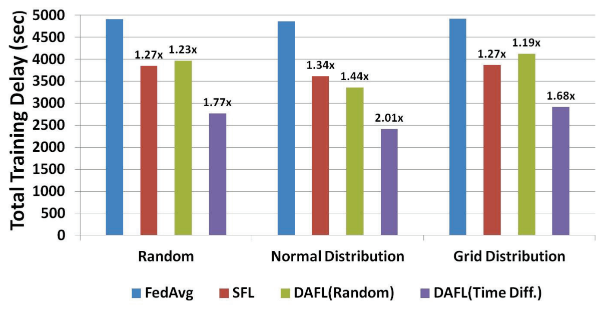


(a) IID, target accuracy: 0.8

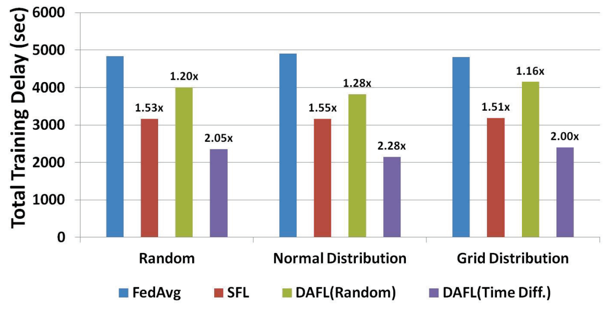


(b) non-IID, target accuracy: 0.65

Fig. 5. Total training delay of model updates for different space sparsity of different methods, with partial client participation rate $p = 0.8$ on ResNet20.



(a) IID, target accuracy: 0.8



(b) non-IID, target accuracy: 0.65

Fig. 6. Total training delay for different device location distributions of different methods, with partial client participation rate $p = 0.8$ on ResNet20.

participation at the end of each round to further reduce the training delay. We have compared DAFL with other schemes and shown a considerable training delay reduction by 0.21x to 0.28x.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of AISTATS 2017*, 20–22 Apr. 2017, pp. 1273–1282.

[2] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *ArXiv*, vol. abs/1811.03604, 2018.

[3] T. Yu, T. Li, Y. Sun, S. Nanda, V. Smith, V. Sekar, and S. Seshan, "Learning context-aware policies from multiple smart homes via federated multi-task learning," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2020, pp. 104–115.

[4] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International Journal of Medical Informatics*, vol. 112, pp. 59–67, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138650561830008X

[5] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. USENIX Association, Jul. 2021, pp. 19–35. [Online]. Available: https://www.usenix.org/conference/osdi21/presentation/lai

[6] R. Chen, D. Shi, X. Qin, D. Liu, M. Pan, and S. Cui, "Service delay minimization for federated learning over mobile devices," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 990–1006, 2023.

[7] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[8] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[9] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," 2021. [Online]. Available: https://openreview.net/forum?id=PYAFKBc8GL4

[10] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23 920–23 935, 2020.

[11] P. J. Bickel, E. A. Hammel, and J. W. O'Connell, "Sex bias in graduate admissions: Data from berkeley," *Science*, vol. 187, no. 4175, pp. 398–404, 1975. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.187.4175.398

[12] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," *CoRR*, vol. abs/1610.02413, 2016. [Online]. Available: http://arxiv.org/abs/1610.02413

[13] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices," *CoRR*, vol. abs/1911.02134, 2019. [Online]. Available: http://arxiv.org/abs/1911.02134

[14] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 4, pp. 1801–1819, 2014.

[15] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804518301590

[16] A. Banitalebi-Dehkordi, N. Vedula, J. Pei, F. Xia, L. Wang, and Y. Zhang, "Auto-split: A general framework of collaborative edge-cloud ai," in *KDD '21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 08 2021, pp. 2543–2553.

[17] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962. [Online]. Available: http://www.jstor.org/stable/2312726

[18] C. Thapa, P. C. M. Arachchige, and S. A. Çamtepe, "Splitfed: When federated learning meets split learning," in *AAAI Conference on Artificial Intelligence*, 2020.