A2FL: Availability-Aware Selection for Machine Learning on Clients with Federated Big Data

Ahmed M. Abdelmoniem[1,2,*] and Yomna M. Abdelmoniem[2,#], and Ahmed Elzanaty[3]
[1]School of EECS, Queen Mary University of London, UK
[2]Faculty of Computers and Information, Assiut University, Egypt
[3]Institute of Communication Systems, 5G & 6G Innovation Centre, University of Surrey, UK
ahmed.sayed@qmul.ac.uk, yomna.m.a.sayed@gmail.com, a.elzanaty@surrey.ac.uk

***Abstract*—Recent advances in Big Data Analytics are primarily driven by innovations in Artificial Intelligence and Machine Learning Methods. Due to the richness of data sources at the edge and with the increasing privacy concerns, Distributed privacy-preserving machine learning (ML) methods are increasingly becoming the norm for training ML models on federated big data. In a popular approach known as Federated learning (FL), service providers leverage end-user data to train ML models to improve services such as text auto-completion, virtual keyboards, and item recommendations. FL is expected to grow in importance with the increasing focus on big data, privacy and 5G/6G technologies. However, FL faces significant challenges such as heterogeneity, communication overheads, and privacy preservation. In practice, training models via FL is time-intensive and worse its dependent on client participation who may not always be available to join the training. Our empirical analysis shows that client availability can significantly impact the model quality which motivates the design of an availability-aware selection scheme. We propose *A2FL* to mitigate the quality degradation caused by the under-representation of the global client population by prioritizing the least available clients. Our results show that, compared to state-of-the-art methods, *A2FL* can improve the client diversity during the training and hence boost the trained model quality.**

***Index Terms*—Federated Learning, Heterogeneity, Selection**

I. Introduction

Recently, Big Data analytic has seen a paradigm shift towards Edge AI (i.e., pushing intelligence towards the edge). This is mainly driven by the need to move computation toward data sources in an effort to reduce communication needs as well as enhance privacy and security [1], [2]. These efforts led to the formation of the Federated Learning (FL) paradigm which transformed traditional distributed machine learning (ML) training methods. Many service providers such as Google, Facebook, and Apple use FL to train global models for natural language processing (NLP) and computer vision (CV) tasks to server applications such as virtual keyboards, object detection, image classification, and recommendation systems [3]–[7]. FL is commonly used with distributed medical imaging data [8]; smart camera images [9]. In FL, the central server managing the models ships them to the clients' end-device on which the training is performed locally to preserve the privacy and security of user data. Due to the lack of control over client devices, FL environments are highly heterogeneous which presents a variety of challenges. In FL, the process is participatory and relies on the availability of the clients and their data. These clients produce and store the application data used to locally train the central ML model and contribute their model updates to the central server for incorporation into the global model.
Time-to-accuracy is a vital performance measure for training quality and is the focus of much work in this area [2], [10]–[13]. Generally, the objective is to reduce the time-to-accuracy by reducing the training time and improving the statistical efficiency. Reducing training time requires hardware acceleration, and time-efficient training algorithms, and reducing the training time requires time-efficient training algorithms, hardware acceleration, and bandwidth-efficient communication methods on the devices [2], [13], [14].
On the other hand, enhancing statistical efficiency depends on the number of participating clients and their data as well as learning-specific hyper-parameters such as minibatch size, learning rate, and the number of local training epochs.[1]. Improving statistical efficiency is challenging, especially for heterogeneous big datasets. Therefore, many efforts are dedicated to addressing the data heterogeneity problem [15]–[18].
Another factor that can impact the training quality is the selection method used to pick a subset of clients from a large population of clients affecting the data sample distribution [13], [19], [20]. During the selection stage, the server samples from the available clients to participate in training the global model on their local datasets in this round.Most methods focused on the heterogeneity of client devices and data. For instance, FedCS [21] favours fast clients over slow ones; InclusiveFL [19] ships models of different sizes to accommodate various clients' computational capabilities; DivFL [20] mitigates the data heterogeneity by selecting a sample of clients to approximate the majority of data distributions.
In heterogeneous environments, clients tend to be battery-powered mobile devices (e.g., smartphones, smart wear, or IoT devices) and so the availability for participation is typically dependent on one or more factors such as the charging state of the device, whether the device is connected to power, WIFI, and/or idle. Existing solutions do not take into account the dynamics of clients' availability who possess the data samples.

*Corresponding author. #Contributed during an internship at QMUL, UK

[1]Hyper-parameters require gradual tuning for different FL jobs and settings

Therefore, the term behavioural heterogeneity has been coined to represent the availability dynamics of the clients [2], [11], [22]. Specifically, clients exhibit variable availability patterns at different times during the training rounds which makes learning tasks more challenging on heterogeneous (or non-IID) data distributions [1], [2], [11], [22]. Although the previous works tried to analyze behavioural heterogeneity and its huge impact on training performance, they did not provide a solution to the problem. Therefore, any practical FL framework should take into account the behavioural heterogeneity of the clients to boost the trained model quality.

In this work, we aim to focus on mitigating the impact of behavioural heterogeneity in FL training. We first dissect the impact on time-to-accuracy caused by training models on non-diverse sets of big data samples. This phenomenon arises as some clients (with unique data samples) are unavailable for participation during the selection stage. Note that the server determines the availability criteria for the devices, e.g., power, network, and idle status. Then, we propose a behavioural heterogeneity-aware selection method to improve the system's robustness to the dynamic and variable data distributions of the clients that usually occur in practice. To this end, we introduce *A2FL*, a participant selection method which accounts for clients' availability and maximizes clients' diversity during FL training. *A2FL* intelligently selects the available clients that have lower availability probability in the future, ensuring their data distributions are accounted for in the trained model. *A2FL* is compatible with and is a plug-in component for existing practical FL systems [1], [23]. Our contributions can be summarized as follows:

1) We conduct an extensive empirical study which demonstrates that the limited availability of participants in FL can significantly impact both the quality and time.
2) Motivated by the lack of schemes with availability consideration, we propose *A2FL* to maximize the diversity of the clients throughout the training process.
3) We conduct experiments using a common FL benchmark to evaluate *A2FL* and show its benefits in enhancing the model quality with data distributions.

II. Related Work

Federated Learning (FL): is an ML paradigm where a server distributes the training task on a sampled set of decentralized clients. The clients train a global model on their local and private data without sharing it [24]. FL is used to improve the quality of many user-facing applications (e.g., virtual keyboard applications [1], [3]). FL frameworks enable researchers to conduct experiments with new designs [11], [25], [26]. As an example, the Leaf framework [25] was used to study heterogeneity [11] and the FedScale framework [26] covers a diverse set of real benchmark datasets.

Participant Selection Strategies: In each round, the server selects among a subset of the available clients to train the global model. Several works designed better selection strategies other than Random selection. For instance, [21] select clients with fast hardware and network speed. Other works [27], [28] select clients whose updates result in higher loss values (i.e., statistical gain). Oort [13] combines both goals (i.e., system and statistical efficiency). These approaches result in models trained on a limited subset of the large population compromising on an inclusive client representation.

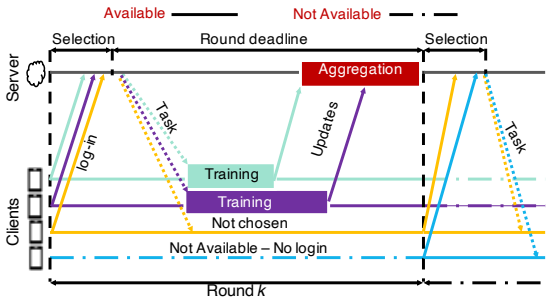


Fig. 1: A FL training round. Clients are sampled to run an FL training task and submit their updates in a given round.

Heterogeneity in FL: Wide dynamics in system behaviour due to client, system, and data heterogeneity is a great challenge for FL systems. clients' compute speed can introduce stragglers and increase round duration [16]. Architectural and algorithmic solutions have been proposed [10], [13], [16]. FL Heterogeneity is quite challenging because clients have variable data, device capabilities, and availability which are hard to tune [1]–[3].

FL proposals: FL systems have improved over time. Some proposals reduce communication time [1], [14], improve system privacy techniques [1], [29], handle stragglers [10], [16], [30], [31], minimize power usage [32], [33], and produce personal models from global models [34]. Many works have addressed data [15] and device [12], [13], [21], [30], [31] heterogeneity.

In this work, we address a unique problem impacting the quality of models arising from the non-inclusive selection. To address this, we treat clients as a diverse set of data sources and leverage their availability for improved diversity.

III. Background

We first introduce the FL training procedure while focusing on system design aspects. Then, we highlight the major challenges in existing FL systems based on empirical evidence from real FL benchmarks. Then, we motivate our work by highlighting the main drawbacks of existing solutions.

A. Federated Learning

In this work, we build upon the common FedAvg aggregation method [1], [24] where the training requires a (logically) centralized server and a large set of decentralized devices (e.g., sensors, smartphones, and/or IoT devices). These devices possess private training data in their local storage and are called clients. For privacy and security reasons, data are not shared with the FL server or other clients. The FL server manages the training process and invokes clients to start a collaborative task of training a global model on their distributed data.

We show the FL stages for training a common model in Fig. 1. At the start of each round, the FL server initiates the

selection phase and waits for a sufficient number of clients to become available for training where a client is available if it is: charged, idle, and/or on WIFI [1]. The server chooses a sample from a large number of online clients to participate in updating the global model in the current round. It sends each participant the task which consists of the current version of the global model and any task-specific settings.

After receiving the task, each participant runs a local optimization process to optimize the model over local data for several epochs. The updated model is sent to the aggregation server during the reporting stage. The FL server either waits until a deadline expires or when the target number of updates is received. The server then aggregates the updates and checkpoints a new version of the model. This completes the round and several rounds are repeated until the training objective is fulfilled (e.g., the target accuracy is achieved or the training cost exceeds a threshold).

The main distinction between conventional distributed data-parallel training and federated learning is that the clients are independent and not under a single entity's management. Therefore the following types of heterogeneity are common in FL: 1) ***data heterogeneity:*** the participants may have varying data samples that are different in size, number of classes, and/or distribution; 2) ***device heterogeneity:*** the participants may use devices of different computational and communication capabilities owing to variable hardware and network settings; 3) ***behavioural heterogeneity:*** the participants' availability for training changes over training rounds which is mainly driven by users' device usage patterns.

Several works have tried to address the challenges posed by different types of heterogeneity in FL settings. Though, most of these works focus on tackling data or device heterogeneity [10], [12]–[14]. Their main objective is to improve model quality and/or training speed to boost time-to-accuracy. In this work, we focus on behavioural heterogeneity which creates unique challenges for FL systems. This is because this type of heterogeneity is harder to control and can have a large influence on the trained model quality.

IV. Motivation

We note that many existing efforts attempt to address data heterogeneity [10] and/or device heterogeneity [24], [29]. Unfortunately, existing system designs do not take into account behavioural heterogeneity which may limit the inclusion of the larger population of the data sources (i.e., some clients may never get selected due to their unavailability). This becomes especially important when data are non-IID which is the typical setting in FL environments where each client has a unique data distribution [15], [35]. Among recent designs, Oort is a state-of-the-art (SOTA) FL selection method that favours clients with high statistical and system utility [13]. Oort proposes a participant selection algorithm that favours higher utility clients to improve time-to-accuracy. The utility is composed of a statistical term which relates to the convergence speed and a system term which relates to the training time. Systematically, Oort's selection method prefers fast clients to reduce round time and can trade-off training time to include slow clients whenever statistical efficiency is not improving.

We highlight the trade-offs between two objectives **system efficiency** as targeted by Oort and **clients' representation** as targeted by *A2FL*. These are conflicting optimization goals in FL. Exploring the extremes of these two objectives, we show empirically how the existing SOTA systems such as Oort fail to perform as designed on common FL benchmarks.

A. Training Speed vs. Client Representation

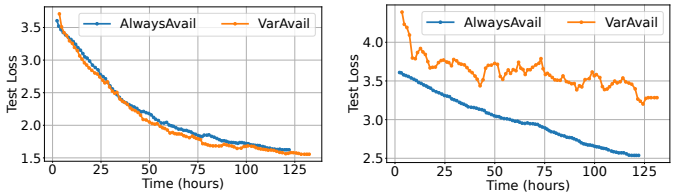
Many FL systems aim to decrease the time taken to reach a target accuracy by prioritizing the selection of fast clients (i.e., system efficiency) [13] or increasing convergence speed by preferring clients with higher data quality (i.e., statistical efficiency) [10], [15]. This results in models that are biased towards clients which possess faster devices and/or produce data more frequently. Though these approaches improve time-to-accuracy, they do not provide good coverage of the large pool of clients' data and fail to have good client representation. This can be addressed with a selection strategy that is more inclusive over the large population though inevitably results in increased time due to the potential inclusion of stragglers [12].

It is evident that the two goals present two extremes of the design space which FL designers need to explore and balance. One extreme is represented by Oort which aggressively reduces training time by exploiting fast clients while ignoring the diverse set of clients' data distributions. As a result, the trained models are less robust to variations in data distributions when deployed in practice, where non-IID data is the norm rather than the exception. It is intuitive to expect unfair selection to produce a global model that does not cover the majority of clients' data [36]. At the other extreme, the designer could skip the selection phase and invoke all available clients for training to maximize client representation but this also comes at the expense of increased resource wastage [12].

To balance the two goals, an FL system can perform client selection in a manner that ensures high levels of client representation. This would result in some reduction in time to accuracy. In this work, we aim to leverage this concept and address this gap in existing system designs with a novel approach that places clients' representation at the forefront of FL system design. In the following, we empirically demonstrate that SOTA systems fail to produce satisfactory models for realistic FL scenarios and present our design of *A2FL* which aims to enhance model quality and fairness through maximizing clients' representation in FL training.

B. Selection Phase and Client Representation

Commonly uniform random sampling is used by existing FL systems to sample clients during the selection phase [1], [3], [25]. The authors in [13] highlight that this naive method can lead to selecting clients with random computation and communication configurations which results in a large variation in completion time. Consequently, this increases the training time as the server must wait for stragglers to submit their updates. On the contrary, Oort advocates biasing selection



(a) Uniform IID (b) Label-Limited (non-IID)

Fig. 2: Influence of dynamic clients' availability.

towards clients with fast devices to decrease the training time in each round. This is undesirable because the trained model may be biased towards a specific group of clients because the model is trained mainly on this group's data.

To empirically study the impact of the selection algorithm on the final accuracy vs time-to-accuracy, we conduct experiments on the state-of-the-art selection, Oort, using the Google Speech dataset and train a speech recognition model (i.e., ResNet32) in FL setting for 1,000 rounds. We use two partitions of the data: (1) distributed uniformly the samples among clients from all labels in the dataset (i.e., IID); (2) each client is constrained only to a random ≈10% of the labels but data points for the allocated labels are sampled uniformly (i.e., non-IID).

Impact of Clients' Availability: In this work, we are interested in the impact of clients' availability on FL performance (or time-to-accuracy). Hence we set clients' availability based on profiles from a real-world trace of mobile users. To this end, we analyze and extract profiles from a large-scale user behaviour trace [11] involving more than 136K users of an FL application over a week. Availability is set based on devices being connected to a charge.We anticipate the availability to have a significant impact on the data distributions of the clients on which the model is trained that varies per round depending on the currently online clients [11], [36].

To this end, we compare the execution of the Oort selection algorithm in two different conditions: i) all clients are always available (**AlwaysAvail**); ii) the availability of the clients varies over time depending on the dynamics of users' availability profile (**VarAvail**). Fig. 2 shows the results for the average test loss (i.e., averaged over the held-out test dataset of the sampled test clients) vs the training wall-clock time. We observe that in the IID case, the availability of the clients has almost no impact on the average test loss obtained in the two different approaches. This is because clients with IID datasets possess data samples with similar distributions. On the other hand, in the non-IID case, we observe that the variable availability of the clients has a detrimental impact on the achieved final test loss and hence the quality of the obtained global model. In the following, we detail the design of our proposed selection technique to mitigate this impact.

V. Availability-Aware Federated Learning

We observe that in federated learning settings, clients' data distributions play a critical role in the quality of the model. Therefore, especially in realistic non-IID cases, the model must be exposed to most of the clients' data samples (or distributions) to increase its generalization abilities. The state-of-the-art selection method (Oort) has a principled selection method, however, by biasing it towards a certain group of clients (e.g., faster ones) it does not fully address the selection problem. We propose that the selection method should maximize the exposure of the model to clients' data while not sacrificing the system efficiency of the training (i.e., time-to-accuracy). This can be achieved by prioritizing clients whose availability for training in the system is limited while they possess valuable data. In essence, this takes into account the variable availability of clients as a key factor which perturbs the global data distribution over time. In the following, we present *A2FL*, an inclusive selection method for federated learning which gives selection priority to the clients with limited availability than the ones that can complete the training faster (i.e., faster computation and/or network).

A2FL aims to mitigate the under-representation of various clients' data distributions during the FL training process without adversely increasing training time. *A2FL* is a novel selection method which prioritizes the least available clients in the future. The goal is to train the model on a larger base of data samples covering the diverse set of data distributions represented by each unique client.

Algorithm 1: Mixed Selection Algorithm

Input : N_t-Target Number of clients
Input : C_{as}-clients selected by alternative method
Output: C_s-Selected clients
Initialize $C_s = \Phi$, $P_t = \Phi$
Let S: the server and c: a client
On_Event ***Client_Join_Task***
  S: send an estimation of the period for the next round;
  c: use FAP and send availability probability p_l
  S: append p_l to the probability list $P_t = P_t \cup p_c$;
On_Event ***End_Selection_Timer***
  S: add c in C_s all unavailable clients $P_l < 0.5$;
  if *len*(C_s) < N_t **then**
    S: select a random c_a from C_{as}
    S: add C_a to C_s if $c \notin C_s$
  end

A. High-Level Design of A2FL

We present the design of *A2FL* by contrasting it with that of Oort. At the high level, from a design perspective, both selection methods introduce a client selection plug-in module which is responsible for the decision logic on which clients participate in the training rounds. *A2FL* consists of two parts:

1) the Future Availability Prediction (FAP) model which produces the probability of client's availability;
2) the Availability-based Prioritization (AP) module which selects the clients based on clients' availability.

In principle, Oort avoids the selection of slow clients leaving them under-represented. On the other hand, *A2FL* improves the representation of the full client population by taking into account the clients' availability in the future regardless of their computational and/or communication capabilities. *A2FL*

selects ones that are highly unlikely to be present in future rounds and then fills the remaining slots randomly with clients selected by another method (e.g., Random or Oort). We next describe the FAP module which uses a time-series model to produce availability probability for each client and then communicate the probability to the AP module.

B. Future Availability Prediction (FAP)

The FAP model needs to be of low computational overhead on the clients' devices. Therefore, a linear forecasting model is used [37]. The model is trained locally on devices' changes in status (e.g., charging). To evaluate such a model, we use Facebook's Prophet tool [38] which trains a time-series forecasting model using linear regression. We leverage the Stunner dataset [39], which contains event traces collected worldwide from a large-scale number of mobile devices. We use a total of 135 devices after processing a total of one million trace events in May 2018 by filtering out any device with less than 1000 trace samples. For predicting availability, we use the plugged and charging state to train a forecasting model for each device. We use the first half of the devices' samples for training and the second half for testing. The results show that the models predict future states with high accuracy with the values of the mean absolute error (**MAE**), mean square error (**MSE**), and coefficient of determination (**R2Score**) averaged across devices, which are **0.027848**, **0.011563**, and **0.928258**, respectively. Notably, *A2FL* does not compromise users' privacy and integrates with the existing techniques for secure aggregation or differential privacy.

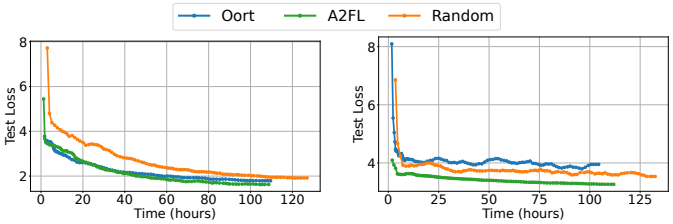
C. Availability-based Prioritization (AP) module

The main objective of *A2FL* is to improve all clients' representation in the training process by exposing the model to the wide-spectrum distribution(s) of clients' Non-IID data. Algorithm 1 describes how the AP module selects clients in each training round. Each client maintains and trains the FAP model periodically on their local device state. For each client c that joins the FL training task, the AP module sends the client a future round time slot which can be estimated from the historical round duration. Each client uses its FAP model to infer its availability probability in the queried time slot. When the selection phase comes to end, the AP module sorts the online clients based on their availability probabilities P (breaking ties with a random shuffle) and selects the least available clients to start the round. Selected participants hold off participating for a few rounds (e.g., 5 rounds) if they successfully submit their updates, similar to [1].

VI. Evaluation

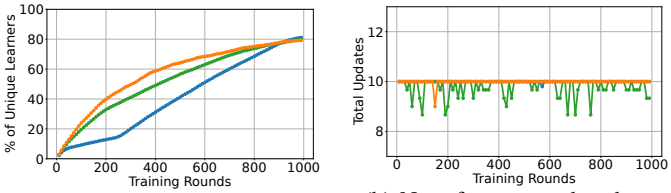
In this section, we evaluate *A2FL* against the state-of-the-art methods and show its benefits in heterogeneous settings.

Experimental Setting: We run a common speech recognition FL benchmark using Google Speech dataset [40] to be trained on ResNet model [41]. The clients are assigned realistic device and network profiles collected from AI Benchmark [42] and MobiPerf Trace [43], respectively. The experiments are



(a) Oort's Data Mapping (b) Non-IID Label-Limited

Fig. 3: Perf. of selection methods in IID vs non-IID cases



(a) Percent. of unique clients (b) No. of aggregated updates

Fig. 4: Percentage of unique clients and number of aggregated updates in the non-IID (Label-limited) case

event-driven in which the time is advanced based on the computational and communication profiles of the devices. That is the completion of training and upload/download of the model are based on devices' computation and networks speeds, respectively. To obtain the quality metrics of training the model, we use a cluster of 4 GPU servers and run batches of 4 clients in parallel. We use PyTorch v1.8.0 as the training backend [44]. Similar to Oort, YoGi [45] is used for aggregation. The number of epochs, batch size, and learning rate hyper-parameters are set to 1, 20, 0.005, respectively. The per-round target number of clients is 10.

Data Partitioning: We use the default data mappings in Oort [13]. And, we use more realistic Non-IID partition in which clients can have only a random 10% of the labels (i.e., 4 out of 35). Then, clients are assigned data samples from each client's pre-chosen labels uniformly at random.

Devices and Availability: clients' devices are assigned a random profile for inference from AI [42]. It is assumed that the training cost is 3× of the inference [13]. Network latency is profiled from MobiPerf [43] traces. For availability, we use a real user trace of events collected, over a period of one week, from over 136k devices from various countries [11].

Experimental Results: We use the same setup as in Oort [13] and enable the real-world behavioural trace (i.e., **VarAvail**). We compare *A2FL*, Random, and Oort selection in terms of the average test loss versus the training time.

Accuracy Results: Fig. 3a and Fig. 3b show that, whether using Oort's data or non-IID data mapping, *A2FL* achieves noticeable lower test loss compared to the other methods. *A2FL* has slightly higher time compared to Oort which biases the selection to lower round time impacting the quality negatively esp. in the non-IID case. This shows that *A2FL* is a superior selection method to enhance model quality irrespective of clients' data distribution.

Behaviour Analysis: Fig. 4a shows that both *A2FL* and Random have a high rate of unique clients contributing to

model training compared to Oort. *A2FL* selects the least available participants first which results in a higher total unique number of clients which explains the better model over other methods. By selecting, the least available clients first, *A2FL* is able to harness their data early in the training to boost the clients' representation in the model especially when data is non-IID. Fig. 4b shows that the total aggregated updates are lower for *A2FL*, yet it achieves the best model quality. This shows that having more aggregated updates is not the only factor that contributes to the model convergence, rather the uniqueness of the data points on the model is trained. Note that, the lower number of updates for *A2FL* is because some clients become unavailable after selection during the training. We think the model quality can be improved if these updates were incorporated. This improvement in reducing the number of failed updates is left for future exploration.

VII. Conclusion

We study the impact of behavioural heterogeneity on federated learning. We dissected the main cause for the low-quality models trained by state-of-the-art selection methods in realistic non-IID scenarios. Hence, we revisited the selection scheme which plays a key role in model quality. To this end, we propose *A2FL* to maximize the diversity of the clients' pool on which the model is trained. Experiments with a real FL benchmark show *A2FL* improves model quality with minimal training cost compared to existing methods.

References

[1] K. Bonawitz *et al.*, "Towards Federated Learning at Scale: System Design," in *MLSys*, 2019.

[2] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," *arXiv 1912.04977*, 2019.

[3] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied Federated Learning: Improving Google Keyboard Query Suggestions," *arXiv 1812.02903*, 2018.

[4] FedAI, "Federated AI Technology Enabler," 2021.

[5] FaceBook, "High-speed library for applying differential privacy for pytorch," 2021.

[6] A. D. P. Team, "Learning with privacy at scale," *Apple Machine Learning Journal*, 2017.

[7] T. Hsu, H. Qi, and M. Brown, "Federated Visual Classification with Real-World Data Distribution," in *ECCV*, 2020.

[8] W. Li *et al.*, "Privacy-Preserving Federated Brain Tumour Segmentation," in *Machine Learning in Medical Imaging*, 2019.

[9] J. Jiang, Y. Zhou, G. Ananthanarayanan, Y. Shu, and A. A. Chien, "Networked Cameras Are the New Big Data Clusters," in *Proceedings of the ACM Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo)*, 2019.

[10] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *MLSys*, 2020.

[11] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *The Web Conference*, 2021.

[12] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A Semi-Asynchronous Protocol for Fast Federated Learning With Low Overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, 2021.

[13] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Efficient Federated Learning via Guided Participant Selection," in *USENIX OSDI*, 2021.

[14] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation.," *IEEE TNNLS*, 2019.

[15] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *ICML*, 2019.

[16] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *NeurIPS*, 2020.

[17] 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, 2020.

[18] L. Yang, C. Beliard, and D. Rossi, "Heterogeneous Data-Aware Federated Learning," in *IJCAI - Federated learning workshop*, 2020.

[19] R. Liu, F. Wu, C. Wu, Y. Wang, L. Lyu, H. Chen, and X. Xie, "No one left behind: Inclusive federated learning over heterogeneous devices," in *ACM SIGKDD*, 2022.

[20] R. Balakrishnan, T. Li, T. Zhou, N. Himayat, V. Smith, and J. Bilmes, "Diverse client selection for federated learning via submodular maximization," in *ICLR*, 2022.

[21] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," *ArXiv 1804.08333*, 2018.

[22] A. M. Abdelmoniem, C.-Y. Ho, P. Papageorgiou, and M. Canini, "Empirical analysis of federated learning in heterogeneous environments," in *2nd Workshop on Machine Learning and Systems (EuroMLSys)*, 2022.

[23] K. Bonawitz *et al.*, "Practical Secure Aggregation for Privacy-Preserving Machine Learning," in *CCS*, 2017.

[24] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[25] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv 1812.01097*, 2018.

[26] F. Lai, Y. Dai, X. Zhu, and M. Chowdhury, "FedScale: Benchmarking Model and System Performance of Federated Learning," *arXiv 2105.11367*, 2021.

[27] Y. Ruan, X. Zhang, S.-C. Liang, and C. Joe-Wong, "Towards Flexible Device Participation in Federated Learning," in *AISTATS*, 2021.

[28] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv 2010.01243*, 2020.

[29] B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning Differentially Private Recurrent Language Models," in *ICLR*, 2018.

[30] A. M. Abdelmoniem and M. Canini, "Towards mitigating device heterogeneity in federated learning via adaptive model quantization," in *1st Workshop on Machine Learning and Systems (EuroMLSys)*, 2021.

[31] A. Arouj and A. M. Abdelmoniem, "Towards energy-aware federated learning on battery-powered clients," in *ACM FedEdge Workshop*, 2022.

[32] L. Li, H. Xiong, Z. Guo, J. Wang, and C. Xu, "Smartpc: Hierarchical pace control in real-time federated learning system," in *RTSS*, 2019.

[33] K. Wang, Y. Ma, M. B. Mashhadi, C. H. Foh, R. Tafazolli, and Z. Ding, "Age of information in federated learning over wireless networks," *arXiv 2209.06623*, 2022.

[34] Y. Jiang, J. Konečný, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv 1909.12488*, 2019.

[35] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated Learning on Non-IID Data Silos: An Experimental Study," *arXiv 2102.02079*, 2021.

[36] J. Huang, C. Hong, L. Y. Chen, and S. Roos, "Is Shapley Value fair? Improving Client Selection for Mavericks in Federated Learning," *arXiv 2106.10734*, 2021.

[37] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice, 2nd edition*. Melbourne, Australia: OTexts, 2018.

[38] S. J. Taylor and B. Letham, "Forecasting at scale," *PeerJ Preprints 5:e3190v2*, 2017.

[39] Z. Szabó, K. Téglás, A. Berta, M. Jelasity, and V. Bilicki, "Stunner: A Smart Phone Trace for Developing Decentralized Edge Systems," in *19th International Conference on Distributed Applications and Interoperable Systems (DAIS)*, 2019.

[40] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *ArXiv 1804.03209*, 2018.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016.

[42] A. Benchmark, "Performance ranking," 2021.

[43] M-Lab, "MobiPerf: an open source application for measuring network performance on mobile platforms," 2021.

[44] Pytorch.org, "PyTorch," 2022. https://pytorch.org/.

[45] S. Ramaswamy, O. Thakkar, R. Mathews, G. Andrew, H. B. McMahan, and F. Beaufays, "Training Production Language Models without Memorizing User Data," *arXiv 2009.10031*, 2020.