An Analysis of Federated Learning Performance in Mobile Networks

Kaylani Bochie[1], Matteo Sammarco[2], and Miguel Elias M. Campista[1]

[1]*Universidade Federal do Rio de Janeiro – UFRJ*, Brazil
[2]*Stellantis*, Poissy, France
{kaylani,miguel}@gta.ufrj.br, matteo.sammarco@stellantis.com

***Abstract*—The increase in personal data collection for service customization threatens users' privacy. In federated learning, privacy can be preserved by distributing the learning process, where only neural networks' weights are shared between clients and servers. This work evaluates the impact of different networks parameters on the performance of federated learning models in a mobile-oriented scenario. Particularly, we consider the performance of convolutional neural networks for image classification. The experiments use the `Flower` framework and the CIFAR-10 dataset for image classification. Typical parameters in mobile networks, such as latency, connectivity, data volume, and client availability, are evaluated. Our results show that, in addition to the increase in total training time, a higher number of disconnected users in a training round can negatively impact the model performance. These results indicate the need for client-server orchestration to perform dynamic adaptation of network conditions.**

***Index Terms*—Federated learning, wireless mobile networks.**

I. Introduction

The popularization of deep-learning-based solutions for computer networks is undeniable [1]. From action recognition [2] to botnet detection in the Internet of Things (IoT) [3], deep learning has overtaken traditional algorithms for developing new solutions. However, training deep learning models requires a large amount of computational power, usually available at the network core in central servers. Therefore, the common approach is to aggregate the data from producing devices in a central server to train machine learning models. This approach raises some privacy concerns, specifically for privacy-sensitive applications, such as in insurance [4] or the healthcare domain [5].

Recently, federated learning has emerged as a new machine learning paradigm to tackle privacy concerns in deep-learning-based applications [6]. Federated learning eliminates raw data transmission from data-producing devices to the central server. Instead, each data-producing device trains a machine-learning model locally on its own data. Each participating client sends the resulting machine-learning model to the central server, which aggregates the received parameters. The computed model is sent to the participating clients, repeating the procedure until the algorithm converges or a final condition is reached. Since privacy concerns contribute to the consistent growth of federated learning popularity, research must be conducted to understand better issues such as performance, explainability, security, and robustness.

An appealing scenario for extensively investigating federated learning performance is mobile networking. The lessons learned with wired networks may not suit these networks well, given the data transmission dynamics and the heterogeneous nature of the participating devices. Federated learning on mobile networks has to deal with highly variable device density and computational power availability, which may negatively affect the model convergence and performance [7]. Therefore, experiments on implementations of federated learning algorithms in typical mobile network scenarios are required as a first step toward proposing improvements to both performance and convergence time.

This work evaluates the impact of mobile network parameters on the performance of federated learning algorithms[1]. We analyze parameters such data reception probability, participating hardware constraints, device availability, and transmission delay. We start with a revision step to identify datasets, learning algorithms, and frameworks commonly used in the corresponding literature. After this initial step, we run centralized learning models to establish a performance baseline for future comparison. The ideal federated learning scenario is then simulated, *i.e.*, we ignore parameters such as transmission delay and device availability. Next, we analyze the impact of different mobile network parameters on federated learning and finally propose and evaluate improvements for its operation in mobile networks.

The remainder of this work is organized as follows. Section II reviews papers related to this work. Section III explains federated learning and the relationship with mobile networking parameters. Section IV presents the experimental setup, while Section V presents the obtained results. Finally, Section VI concludes this work and suggests future directions.

II. Related Work

This section considers papers that evaluate the federated learning performance in wireless and mobile settings. We also

[1]A preliminary version of this paper was published in SBRC 2021, a Brazilian conference in Portuguese. The paper can be accessed in http://www.gta.ufrj.br/ftp/gta/TechReports/BSD21.pdf

review contributions proposing client clustering and aggregation methods, which is a future step for our work.

Ghosh et al. tackle client clustering on federated learning settings [8]. They proposed a framework named Iterative Federated Clustering Algorithm (IFCA), which dynamically allocates federated learning clients into different clusters. The decision to group clients in real-time occurs because client data is not transferred through the network, hindering clusters from being defined beforehand. The novelty of their paper consists of sharing representation layers, *i.e.*, shallower layers, between all users to train the final layers of each cluster. Their proposal borrows this concept from multi-task learning. The authors perform their experiments using image recognition datasets and observe performance improvements up to 7 and 57% compared with traditional federated learning and centralized machine learning, respectively. In the last approach, each client trains machine learning models on their local datasets.

Nishio et al. consider the impact of mobile edge computer scenarios on federated learning performance [9]. The authors explore the heterogeneity of mobile clients, namely, clients in a cellular network, to dynamically select which clients participate in a training round. The variation in computational power, data resources, and channel conditions directly impact their proposal. The authors then propose FedCS (Federated Learning with Client Selection), which takes a "resource oriented" approach to client selection. The FedCS approach relies on a resource request step that provides the information of each client. Information such as channel state and GPU (Graphical Processing Unit) availability is used to build client groups, which perform individual federated training rounds. The authors evaluate their proposal on standard image classification datasets and show that their proposal can improve classification performance compared with standard federated learning algorithms. Their proposal, however, leverages client metadata, such as hardware capabilities, which may not be available for some applications.

Yang et al. evaluate three scheduling policies for federated training on wireless networks and their impact on accuracy and communication rounds required for convergence [10], [11]. The authors also evaluate the impact of some physical layer parameters, *e.g.*, Signal-to-Noise Ratio (SNR), on model convergence in federated settings. Despite implementing classical scheduling policies, their experiments consider complete client availability, where only the communication channel limits performance. This assumption, however, may not represent some wireless scenarios, such as mobile networks.

Nguyen et al. develop a system capable of detecting botnet in IoT network [12] by leveraging IoT gateways as federated learning clients [13]. The authors perform a centralized performance evaluation to configure the federated setting. The research performed by Nguyen et al. explains that a real application cannot use centralized learning as a first step to adjust the configuration of federated learning parameters. Nevertheless, our work follows the same trend since our goal is to understand the impact of mobile network parameters on the overall model performance, instead of building and deploying a federated learning application.

Tran et al. analyze how local processing latency, *e.g.*, the time a client takes to finish processing its data during a federated training round, impacts federated training performance [14]. Their proposal uses synchronized updates to improve federated performance, which may not be viable if a single base station does not directly control the clients, as is the case for large-scale mobile applications.

Even though our work focuses on mobile networks' applications, some federated learning settings are deployed on stable and highly available networks to preserve privacy without optimizing communication [6]. Kairouz et al. highlight the availability of reproducible results as an open issue in federated learning research [7]. Multiple studies evaluate federated learning performance on devices optimized for DNNs (Deep Neural Networks) [15]. These works, however, do not represent the mobile setting on which clients cannot leverage models with billions of parameters. This work tackles the study of mobile network's parameters on federated learning, where the number of available clients can vary, the data available for training is unknown *a priori*, and faulty connections are more present than on Ethernet-based networks.

III. Federated Learning

Federated learning is an emerging decentralized and distributed machine learning paradigm [6]. It enables devices to leverage their computing power to collaboratively train a global model on private and locally available data, without transmitting it to another location, usually high-performance servers [16]. The main motivation of federated learning is to enable the development of privacy algorithms coupled with high performance DNNs.

Classical machine learning approaches require that the clients send their data to a centralized server, which may be an issue for privacy sensitive applications, such as healthcare applications [17]. Federated learning, alternatively, distributes a learning model to clients and requires them to send back only the learning model's parameters. Note that each client trains the model on its locally available data. All these steps form a federated learning round and the server may choose to perform more training rounds according to some criteria, such as client availability, inference performance, elapsed real time, and model convergence. Once the training is finished the obtained model can be distributed to the clients and then be used to perform inferences.

Although federated learning can aid in privacy preservation, this new paradigm suffers from new and interesting challenges. These challenges can be divided into four. Firstly, in traditionally studied distributed machine learning, where a server distributes data to a number of clients for training, the clients receive data samples from the same data distribution, which are consequently IID (Independent and Identically Distributed). In contrast to the aforementioned distributed machine learning scenario and centralized machine learning, each client participating in a federated learning application has its own collected data, which introduces the notion of "non-IIDness". Secondly,

data volume also varies from client to client. Thirdly, the number of participating clients in a federated training round can vary, and this fact is more pronounced in mobile scenarios. A common strategy is to query clients for availability before starting the training process. Mobile clients will only participate in a training round if some requirements are met, *i.e.*, the device is connected to a charger and it has not been used for several hours. Finally, hardware heterogeneity can affect performance. Again, this characteristic is more prominent in mobile scenarios. All of these challenges motivate an in-depth and thorough study of federated learning techniques and their applications [6], [7]. This work strives to explore how the previously presented challenges affect federated learning performance and how to improve it.

It is possible to identify multiple stages where mobile networks can introduce errors that impact the overall final model performance. Clients must have enough computational power to both hold the machine learning models in memory and train the models during the allocated time for a federated training round. Furthermore, client availability can increase the number of training rounds needed to achieve convergence. Finally, the communication latency between server and clients, besides impacting model performance when the communication interval exceeds the time allocated by the server for a training round, increases the time it takes for the global model to finish training rounds.

IV. Experimental Setup

The neural network was firstly trained and evaluated in a centralized setting, aiming to establish a performance baseline for the federated learning results. The baseline is obtained using an early stop mechanism to achieve improved performance on the centralized setting. Classification metrics are then used to qualitatively estimate if the machine learning model achieved convergence.

After concluding the centralized analysis, the neural network was deployed on a federated learning setting with ideal network conditions, *i.e.*, communication and processing delays are ignored, and the clients had the failure probability set to zero percent. The results obtained in this step are used to evaluate the number of rounds needed for the model convergence. At this point, it is important to highlight the term "round" as the process of: (i) transmitting the model from the server to the selected clients, (ii) training the model on each client with their particular dataset for some number of local epochs, and (iii) transmitting the trained models from the clients back to the server for aggregation. During the ideal federated training, a family of curves is obtained by varying the following simulation parameters: the number of participating clients (N_c), the number of training epochs for each client (E_t), and the local batch sizes for each client (B_c).

Additionally, each client has another parameter named "steps per epoch", which limits the maximum number of samples used by the client on each round. This parameter is used to prevent full access to the training dataset along all rounds, to better reflect federated learning scenarios. This decision was made since in real-world applications, participating clients possess datasets with different statistical properties.

Finally, mobile networks' parameters are emulated by considering a scenario where clients may disconnect during training and may not be able to process the data during training. This was achieved by instantiating each client in the simulation with a given failure probability.

CIFAR-10 is an image classification dataset widely used for computer vision tasks [18]. The low-resolution images result in a compact dataset with numerous samples, which enables fast training. This characteristic make the dataset highly popular for benchmarking in computer vision tasks. The dataset comprises 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck) and is a subset of the CIFAR-100 dataset [18].

Model selection is a sensitive issue for mobile applications, since the model's performance must be balanced with memory consumption and processing times. Even though commonly available mobile devices are capable of processing models which consume hundreds of MBs of memory [19], it is expected that multiple applications share devices' resources. As such, typical deep-learning-based mobile applications leverage models consisting of a few MBs. This work follows the trend of minimizing the use of fully-connected layers and, instead, uses deep convolutional neural networks, which are more appropriate for computer vision tasks [1]. Naturally, real-world applications may employ additional techniques to further reduce the neural network size, such as weight quantization [20]. This work does not use these techniques since neural network compression is considered to be orthogonal to this work.

The dataset chosen is commonly used as a benchmark for computer vision applications. As a result, there is a multitude of papers published which evaluate the performance of centralized neural networks on such dataset [21]. The architecture used on the CIFAR-10 dataset was selected using the design principles of VGG models [22]. The neural network has 550,570 parameters and the corresponding file in Hierarchical Data Format (HDF) has 4.3 MB. The Flower framework, which is being developed to enable federated applications, was used in the simulations to deploy the FedAVG algorithm [6]. We run our experiments on an Intel Core $\mathrm{i5-10400}$ $2{,}90$ GHz computer with 6 processing cores and 32 GB RAM.

V. Results

A. Baseline Models on Centralized Training

Centralized machine learning is already well-explored in the literature. This work applies previously researched workflows to obtain the centralized results [1]. It is important to mention that the overall model size, *i.e.*, the volume of data transmitted to each client, must be limited due to the mobile network limitations discussed in Section III. Even though devices may choose to participate in federated training rounds during periods of low activity, it is important to guarantee that memory, CPU, and GPU usage does not hinder other applications. Although it is possible to achieve better results with deeper and more complex models, communication and

processing costs can make their use unfeasible. We argue that the values obtained during centralized training only shift the performance practical upper bound used for evaluation and do not contradict the results discussed in Section V.

The centralized experiments use SGD (Stochastic Gradient Descent) as the optimizer, the learning rate is 0.0001, and a momentum is 0.9. Training is performed for 25 epochs with a batch size of 128 samples. The centralized learning algorithm achieved 67.82% accuracy on the test set. This value was used as a practical upper bound to evaluate distributed convergence.

B. Federated Learning on Ideal Wireless Conditions

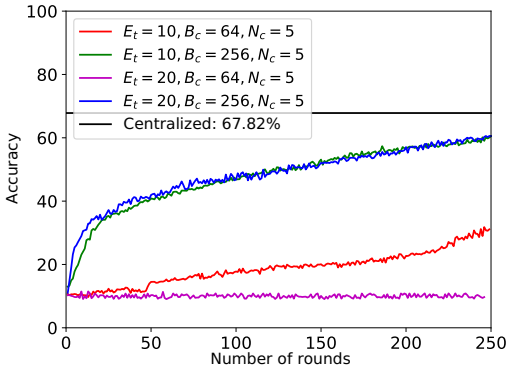
DNNs are distributed to the clients and, after each round, their performances are evaluated on two clients chosen at random. Figure 1 shows that the number of local training epochs has a positive impact on model convergence. This is expected, since for a given number of federated training rounds, a high number of local training epochs allows each client to look at more data during training, accelerating the global model convergence. Furthermore, local batch sizes, which assume a regularizing role during centralized machine learning [23], can act as a catalyst for model convergence. This occurs because clients are limited to a given number of steps per epoch. Thus, increasing the batch size also enables the clients to look at more data during each federated training round.

Another important federated learning parameter can be seen when comparing Figure 1(a) and Figure 1(b). The number of clients has direct impact on model convergence and model performance. However, in mobile settings, the number of available clients may vary greatly. Therefore, techniques to evaluate if the server must execute a training round for a given set of clients must be developed. The blue curve of Figure 1(b) shows that the model performance may decrease if the number of available clients is insufficient, so the server must be able to decide if another training round must be performed. As Figure 1(a) shows, a model training on just a few clients can achieve satisfying performance. however, the training process becomes more sensitive to the federated parameters, which is shown by the poor performance when using 10 training epochs and 64 data samples per batch. Figure 1(c) confirms the previous results in a scenario with 50 clients.

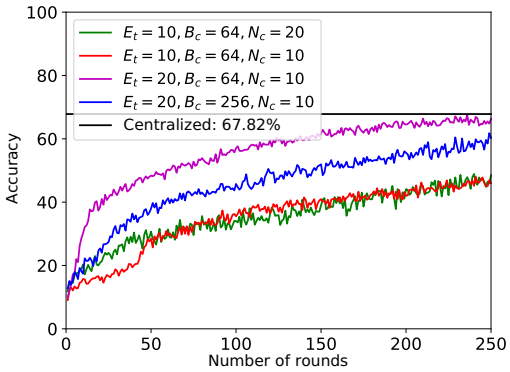
C. Federated Learning on More Realistic Wireless Conditions

In this section, wireless mobile network parameters are introduced in the form of failure probability, which corresponds to the chance of a client being disconnected or not being able to finish its training during a training round. Since a client may stop responding to the server altogether, the centralized server can only infer that the client failed during training, because of either high processing time or transmission delay between the client and the server. The results are compared to those obtained with ideal network conditions.

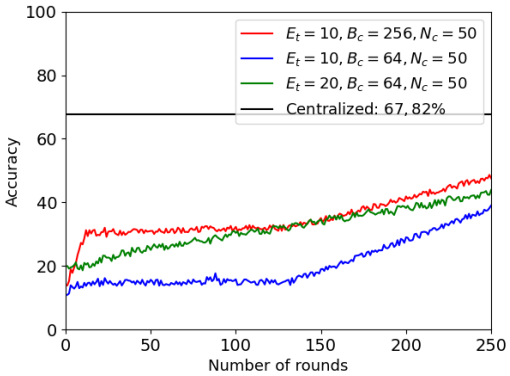
Failure Probability: When a client's latency surpasses the time allocated by the server for a federated training round, the client is "dropped" during that particular round, *i.e.*, the server infers that the client got disconnected during training. For simulation purposes, if a client presents faulty behavior, the server ignores the faulty client during the current round. Hence, model convergence is impaired by these faulty connections. The failure probability values were select to cover both critical and non-critical settings. It is also worth noting that the model's performance presents similar behavior for failure probability in the range between 5% and 25%, thus we chose to omit these results.



(a) Curves obtained by simulating 5 clients.



(b) Curves obtained by simulating 10 and 20 clients.



(c) Curves obtained by simulating 50 clients.

Fig. 1. Model convergence on the CIFAR-10 dataset with ideal network conditions.

Figure 2 presents one of the models seen in Figure 1(b), but with different failure probabilities (P_f) applied to all clients. We can observe that a large number of disconnections degrades model performance and slows down model convergence. The red curve (P_f = 25%) in Figure 2 shows that the algorithm is able to converge when disconnections occur, but the model achieves lower performance. However, at least in the simulated scenarios, depending on the number of participating clients, the effect of disconnections on model convergence can be amplified. This can be seen in the blue curve (P_f = 75%), which shows superior performance in relation to the green curve (P_f = 50%), even though the number of faulty connections is greater. This result can be attributed to a regularization effect introduced by removing some clients from each training round, similarly to the way a dropout layer can reduce neuron dependency on neural networks and increase its performance. It is worth noting that the red, green, and blue curves do not seem to converge at 250 rounds.

Latency: The server is responsible for allocating a maximum time interval for each training round. If the sum of a client's processing and transmission times is greater than this maximum interval, the client is dropped during this training round. This may result in performance degradation. However, an increase in client's latency always increases the elapsed real time required for the model to converge. The overall latency to train a model can be directly calculated by Equation 1 below:

$$\text{Total latency} = \sum_{i=1}^{R} \max_{\forall c \in C_i} (\min(L_c, T)), \tag{1}$$

where C_i denotes the set of clients involved in a training round, L_c denotes the latency of a single client, and T denotes a timeout parameter configured by the server, and R denotes the number of training rounds. In our model, a client's latency (L_c) is determined by the sum of its processing time and the transmission time to the server. It is important to note that failure or disconnection probabilities have direct impact on the total latency, since a single client failing during training is enough to set the latency of a training round to T.

Figure 3 presents a Monte Carlo simulation of Equation 1 for different failure probabilities. The timeout (T) parameter was selected as twice the worst case scenario for the devices. This parameter is application dependent and thus must be adjusted depending on the applications. Transmission latency was estimated using the software `iPerf3`[2] to measure the RTT (Round-Trip Time) between three servers in Brazil, Canada, and Russia. The position of each client is chosen at random during the simulation. Processing times of each device was selected based on our previous simulations.

A possible strategy to reduce the impact of faulty clients is to prematurely end a training round when some defined subset of clients report their results to the server. Thus, the server may start a new training round with those clients or select a different subset of clients to reduce latency. This strategy is presented in Figure 3, where the total training time is greatly reduced by a large interval of failure probabilities (P_f). It is worth mentioning the tradeoff between the total training time and the computational cost on each client, since the fraction of training performed in slower clients is not used.

Convergence Analysis: Figure 4 summarizes the number of rounds required for the algorithm to achieve convergence for some configurations. The triangles pointing to the right represent models that still showed signs of improvement at round number 250, while the $\times$ symbol represents a model that showed signs of deterioration during training. The convergence was estimated by observing if the model's performance is kept stable during multiple training rounds. The figure shows that a high number of disconnections, in addition to delaying convergence, also can harm the final model's performance. This result coupled with the knowledge extracted from Figure 1(a), it is possible to highlight that if the same group of clients contributes to training across multiple rounds, the model's performance can even decrease. In that case, the server must identify the drop in performance and stop the training process until the appropriate number of clients is available to join the training. We can also highlight the better performance of the

[2]Accessed on https://iperf.fr/.

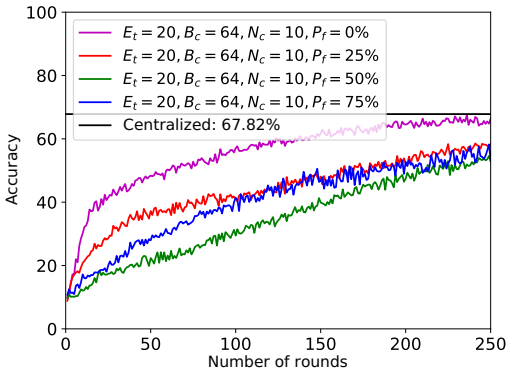


Fig. 2. Impact of disconnection probability during federated training.

Fig. 3. Latency comparison using Equation 1 with 5 clients (left) and 10 clients, but with early round stopping when 5 clients report their results.

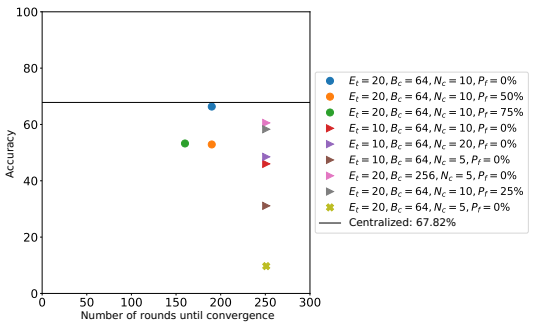


Fig. 4. Number of rounds for each scenario to achieve convergence.

model with $P_f = 75\%$ in relation to the model with $P_f = 50\%$. Even though more clients presented problems during training, the model converges faster and achieves better performance.

VI. Conclusion and Future Scope

This work studied mobile networks parameters and their effect on federated learning performance. The data was distributed among the clients to simulate the random data collection process on real-world applications. The results show how federated learning can help maintaining users' privacy while achieving comparable performance to centralized machine learning approaches. Some mobile networks parameters were found to have counter-intuitive impact on federated learning performance, such as a higher fault probability in clients improving a federated learning model's performance. Additionally, traditional hyperparameters of deep neural networks, such as batch size, also have slightly different behaviors in a federated setting. Obtaining representative datasets of real world federated learning applications is still a challenge. To enable appropriate evaluations the datasets must contain information regarding the origins of the data, *i.e.*, the authors of each sample.

As future work, we intend to explore more complex wireless scenarios, such as when disconnection probabilities vary over time. Additionally, we intend to explore more complex neural network architectures to cover scenarios where devices have more computational power and are constrained by extremely low latency applications, such as those in vehicular and industrial networks. Finally, we plan to research techniques to improve federated performance through client clustering and selection without using client metadata.

Acknowledgment

This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. It was also supported by CNPq, FAPERJ Grants E-26/211.144/2019 and E-26/202.689/2018, and FAPESP Grant 15/24494-8.

References

[1] Bochie et al., "A survey on deep learning for challenged networks: Applications and trends," *Journal of Network and Computer Applications*, vol. 194, p. 103213, Nov 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521002149

[2] Li et al., "Skeleton-based action recognition with convolutional neural networks," *CoRR*, vol. abs/1704.07595, 2017.

[3] Meidan et al., "N-BaIoT—network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[4] Y. Wang and W. Xu, "Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud," *Decision Support Systems*, vol. 105, pp. 87–95, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923617302130

[5] Zhou et al., "Deep-learning-enhanced human activity recognition for internet of healthcare things," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6429–6438, 2020.

[6] McMahan et al., "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[7] Kairouz et al., "Advances and open problems in federated learning," in *Foundations and Trends in Machine Learning*, 2019. [Online]. Available: https://arxiv.org/abs/1912.04977

[8] Ghosh et al., "An efficient framework for clustered federated learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 19 586–19 597.

[9] 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.

[10] Yang et al., "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, Jan 2020.

[11] A. Ghasempour and M. Martínez-Ramón, "Short-term electric load prediction in smart grid using multi-output gaussian processes regression," in *2023 IEEE Kansas Power and Energy Conference (KPEC)*, 2023, pp. 1–6.

[12] A. Ghasempour, "Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges," *Inventions*, vol. 4, no. 1, p. 22, Mar 2019. [Online]. Available: http://dx.doi.org/10.3390/inventions4010022

[13] Nguyen et al., "DÏoT: A federated self-learning anomaly detection system for IoT," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, July 2019, pp. 756–767.

[14] Tran et al., "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1387–1395.

[15] Li et al., "Privacy-preserving federated brain tumour segmentation," in *Machine Learning in Medical Imaging*, H.-I. Suk, M. Liu, P. Yan, and C. Lian, Eds. Cham: Springer International Publishing, 2019, pp. 133–141.

[16] Abdulrahman et al., "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, 2021.

[17] Roth et al., "Federated learning for breast density classification: A real-world implementation," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, S. Albarqouni, S. Bakas, K. Kamnitsas, M. J. Cardoso, B. Landman, W. Li, F. Milletari, N. Rieke, H. Roth, D. Xu, and Z. Xu, Eds. Cham: Springer International Publishing, 2020, pp. 181–191.

[18] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[19] Benedetto et al., "Deep neural networks on mobile healthcare applications: Practical recommendations," *Proceedings*, vol. 2, no. 19, 2018. [Online]. Available: https://www.mdpi.com/2504-3900/2/19/550

[20] Gong et al., "VecQ: Minimal loss DNN model compression with vectorized weight quantization," *IEEE Transactions on Computers*, vol. 70, no. 05, pp. 696–710, may 2021.

[21] Baldominos et al., "A survey of handwritten character recognition with MNIST and EMNIST," *Applied Sciences*, vol. 9, no. 15, 2019.

[22] J. Brownlee, *Better Deep Learning*, v1.8 ed. Jason Brownlee, 2018.

[23] L. et al., "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.