Federated Learning at the Network Edge: When Not All Nodes are Created Equal

Federated Learning at the Network Edge: When Not All Nodes are Created Equal

Francesco Malandrino, Carla Fabiana Chiasserini

***Abstract*—Under the federated learning paradigm, a set of nodes can cooperatively train a machine learning model with the help of a centralized server. Such a server is also tasked with assigning a weight to the information received from each node, and often also to drop too-slow nodes from the learning process. Both decisions have major impact on the resulting learning performance, and can interfere with each other in counter-intuitive ways. In this paper, we focus on edge networking scenarios and investigate existing and novel approaches to such *model-weighting* and *node-dropping* decisions. Leveraging a set of real-world experiments, we find that popular, straightforward decision-making approaches may yield poor performance, and that considering the quality of data in addition to its quantity can substantially improve learning.**

I. Introduction

Federated learning (FL) is a distributed machine learning paradigm whereby a set of *learning nodes* cooperate in training a model (e.g., a neural network) with the assistance of a centralized *model server* and without the need to share their local data. FL has been introduced [1] in 2015 by Google, with the goal of leveraging the computational power of end-user devices – most notably, smartphones – without the privacy and security concerns arising from sharing the potentially sensitive information they own. As discussed in Sec. II, it has since been widely adopted in edge computing scenarios, owing to its ability to blend device- and server-based computation, and to enable cooperation between devices regardless of their location.

FL includes the following high-level steps, summarized in Fig. 1:

1) learning nodes train a local model based on local, on-device data;
2) learning nodes send the model parameters – and nothing else – to the server;

This work has been partly funded by the European Commission through the H2020 project Hexa-X (Grant Agreement no. 101015956).

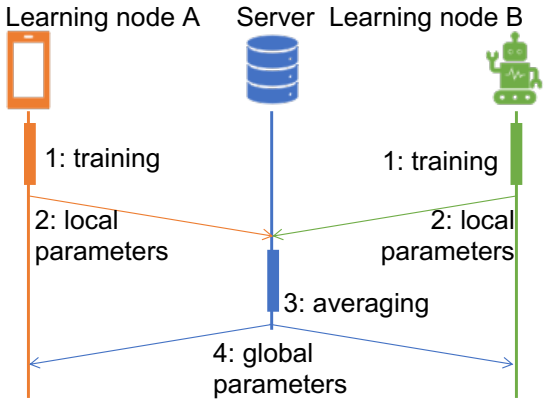

Fig. 1. Main steps of each iteration of the federated learning paradigm: learning nodes train their local model (1) and send the local parameters to the server (2); the server performs a weighted averaging of the model (3) and sends the global parameters back to the learning nodes (4).

3) the server combines the parameters coming from different learning nodes;
4) the server sends the combined, *global* parameters back to the learning nodes;
5) the learning nodes replace their local parameters with the global ones, and move to a new training iteration (step 1), till the desired accuracy is achieved.

Step 3 usually takes the form of a *weighted averaging* of the local parameters [1], [2]. Weights are assigned to individual learners and reflect the magnitude of their (expected) contribution to the learning process; indeed, as discussed next, properly assigned model weights is one of the main decisions learning servers can make.

Originally envisioned for homogeneous nodes dealing with homogeneous datasets (e.g., in a classification problem, datasets adequately representing all classes), FL has also the potential to deal with *heterogeneity*, both in the capabilities of learning nodes [2], [3] and in their local data [2], [4]. In both cases, the key is to endow the model server with additional responsibilities: the *weights* given to

different local models in the averaging phase (step 3 in Fig. 1) can account for the quantity of data they are trained upon; at the same time, if some nodes are consistently slower than others, they can be *dropped* from the learning process [3], [5].

Our key observation is that these two decisions, *model-weighting* and *node-dropping*, are deceptively simple, and their impact on the overall learning process is often misunderstood and underestimated. Many state-of-the-art works take straightforward approaches to these decisions, which may result in poor performance under richer, more complex scenarios. Our contribution is therefore to shed a light on the model-weighting and node-dropping decisions, studying how they should account for the quantity *and quality, e.g., variety,* of data available to local nodes, as well as for the data processing time. In so doing, we focus on an edge network scenario and leverage a set of experiments using the popular `tensorflow` library and the recent Fashion-MNIST dataset.

In the remainder of the paper, we describe federated learning in edge scenarios and the associated challenges in Sec. II, before narrowing the focus on model-weighting and node-dropping strategies in Sec. III. We then describe our experiments and results in Sec. IV, and the main lessons learned in Sec. V, along with pointers to further promising research directions. Finally, Sec. VI concludes the paper.

II. Federated Learning in Edge Scenarios

Edge computing is a distributed paradigm predicated upon performing the computation as close as possible to the user nodes requesting it, i.e., at servers located at the *edge* of the network infrastructure. It also includes scenarios where user nodes themselves have computational and/or storage capabilities, and require edge support for coordination, or to offload the heaviest computation tasks. FL has long been identified as an excellent match for edge computing scenarios, and many research works aim at making it in such scenarios as efficient as possible. At the same time, *communication* is a major issue for FL in edge scenarios. Nodes can be connected with the edge-based server in many ways and through different technologies; thus, their connectivity has a major impact on the latency incurred when sending model updates – and, indeed, on whether or not such updates are received in the first place.

Specifically, as detailed in [6], edge computing is more effective than fully-distributed, device-to-device networks at tackling the main factors hindering the performance of FL, namely, the different node capabilities, available data, and unpredictable communication delays and shortages. Narrowing its focus to node capabilities, [5] aims at choosing the set of learning nodes that results in the shortest learning time, solving a double-edged conundrum. On the one hand, more nodes mean that convergence can be reached in fewer iterations; on the other hand, the duration of each iteration is determined by the slowest node [7]. In a similar spirit, [2] addresses the problem of jointly selecting the learning nodes to use for the learning process and assigning them the wireless resources they need to communicate effectively. Setting in a fully-decentralized *fog* scenario where no learning server may be present, [8] tackles many issues relevant to edge computing, including device mobility and the possibility of offloading computation from a node to another.

Shifting the focus towards the major issue of communication between FL nodes and server, the authors of [9] seek to reduce the communication overhead of FL by proposing a compression algorithm suited for federated learning settings. Their algorithm outperforms existing schemes when local datasets are heterogeneous, thus making the high-frequency communication required by FL viable in low-bandwidth scenarios. Heterogeneous datasets are also identified as a major problem in [4], which envisions extracting a homogeneous subset from each local dataset in order to avoid bias and training errors.

Following an orthogonal, more theoretical approach, several works [7], [10] aim at characterizing the learning performance, deriving closed-form expressions for their (expected) training time. Such a characterization is then exploited to make optimal or near-optimal decisions on the cooperation among nodes [7] and the equilibrium between local learning and global updates [10]. In order to obtain manageable closed-form expressions, some of these works make simplifying assumptions (e.g., that local datasets be homogeneous), or target specific parameter optimization algorithms (e.g., stochastic gradient descent).

One scenario where nontrivial model-weighting

decisions are routinely used is *asynchronous* FL [11], where nodes may join the learning process at different times and model-weighting serves the purpose of quickly including newly-arrived nodes in the learning process. In our case, the purpose is different, namely, to adapt model weights to the contribution each node can give to the overall learning process, and weed out those nodes that may have a negative impact on the learning performance.

III. Model-weighting and Node-dropping Strategies in Federated Learning

Model-weighting and node-dropping are the most fundamental decisions the learning server can make, and arguably among the simplest to enact. At the same time, as discussed in the following, these decisions can be leveraged to address all the main issues of FL, either in combination with the strategies reviewed in Sec. II, or as an alternative to them.

Insufficient quantity of data: In many FL scenarios, some learning nodes may not have enough local data, thus being unable to properly train their local models; in this case, a popular solution is *augmenting* local datasets. As an example, the authors of [12] propose to combine actual data samples from other learning nodes in a privacy-preserving way, and adding them to the local dataset.

In both cases, data augmentation is able to increase the quantity of data available to learning nodes, without jeopardizing FL's privacy properties. On the negative side, it may increase the complexity of the system and its overhead; furthermore, the augmented samples come from processing of already-existing ones, hence, do not increase the total quantity of information.

Model-weighting decisions represent an additional, simpler way to deal with learning nodes with small local datasets. The basic idea [1], [2] is that model weights shall account for the quantity of data local models are based upon; however, as we will demonstrate next, also accounting for the variety of such data yields even better results. In both cases, relying on model-weighting to tackle insufficient dataset sizes has the benefit of reducing complexity and avoid tampering with the nodes' own data.

Non-homogeneous data: When it comes to training machine-learning algorithms, the quality of data is as important as its quantity. There is no universally-acknowledged definition of data quality, as it is scenario- and application-dependent. For classification applications, a high-quality dataset is expected to adequately represent all existing classes, so that the classifier can be properly trained. In this sense, quality can be expressed as the number of classes existing in a given dataset or, more formally, through entropy [13].

On the other hand, non-i.i.d. data, where classes are under- or over-represented, is universally characterized as low-quality, and has immediately been identified as one of the primary threats to successful FL. In addition to the augmentation approaches described earlier [12], several works propose training the local model on a subset of the local data [4], chosen in such a way to be i.i.d. An alternative to ignoring data is allowing all learning nodes to use all their data, and then weight their local models accounting for the quality of such data. Examples of this approach include entropy [13], but simpler approaches, e.g., counting the labels observed, can yield similarly good performance.

Nodes with different capabilities: As with other distributed learning schemes, in FL it is possible to proceed from an iteration to the next one only when *all* learning nodes have sent their local models, i.e., have performed step 2 in Fig. 1. It follows that the pace of the learning process as a whole is determined by the *slowest* learning node, which becomes an issue when different learning nodes take very different times to perform their iterations [7]. Owing to the limited amount of control that can be exerted on FL nodes, the most viable solution is often to exclude overly-slow nodes from the learning process [2], [5], [7].

However, making such node-dropping decisions solely on the basis of their response times may actually hurt the learning process; indeed, longer response times can be associated with larger, higher-quality local datasets, hence, with the nodes that may contribute the most to the learning. A way to decrease the likelihood of this unwanted outcome is to consider additional aspects in making node-dropping decisions, e.g., the quantity and quality or variety of local data. By so doing, it is possible to differentiate between nodes that are slow due to limited capabilities (or poor connectivity [2], [5]) and those that have simply more data to process.

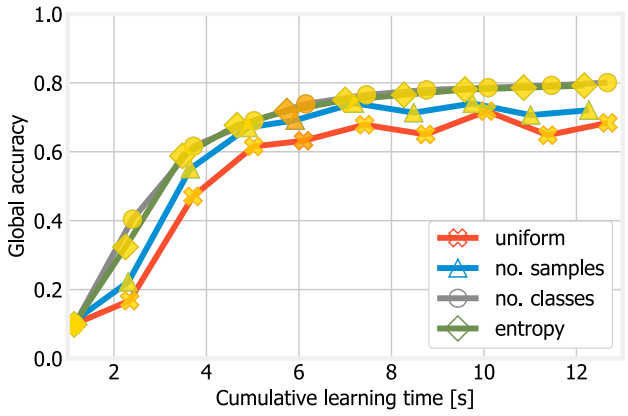

Fig. 2. Relationship between elapsed time and global accuracy for different model-weighting strategies, when no nodes are dropped from the learning process. The color of markers corresponds to the category of the slowest node in that particular iteration (gold: yellow, bronze: orange, no other category appears).

IV. Experiment Design and Results

In this section, we demonstrate how model-weighting and node-dropping decisions can deal with heterogeneity in the quality and quantity of the local datasets at learning nodes. Using the set of real-world experiments described in Sec. IV-A, we obtain the results described in Sec. IV-B.

A. Experiment setup

Dataset and neural network structure. Fashion-MNIST is a dataset released by Zalando research and aimed at providing a more challenging, drop-in replacement for the classic MNIST handwritten digits dataset. Owing to the relative simplicity of the dataset, we use a relatively small neural network for classification. Specifically, we create a dense network with four layers, with sizes $[28^2, 200, 100, 200]$ neurons (notice that the size of the first layer must match the size of the input, i.e., 28×28 pixels). Neurons use the `softmax` activation function, and parameters are optimized using stochastic gradient descent (SGD), with a learning rate of 10^{-2}. The network is implemented using the popular `tensorflow` library, originally developed by Google.

Network scenario and local datasets. Our experiments feature a typical medium-scale edge scenario [6], with 20 learning nodes connected with, and coordinated by, an edge-based server. Five out of 20 nodes belong to each of the following four categories so that the total amount of data remains constant:

- *gold*, having 500 samples each, representing all 10 classes (i.e., articles of clothing) present in the Fashion-MNIST dataset;
- *silver*, with 200 samples each, still belonging to all classes;
- *bronze*, with 500 samples each, belonging to only two classes per node;
- *garbage*, with 200 samples each, belonging to two classes.

With the exception of "gold" ones, nodes suffer from either low quantity or low diversity, hence, low quality, of local data. Our experiments establish a correlation between the category of each node and its contribution to the learning process, thus allowing us to identify the best strategies to decide whether and how to integrate each node in the learning process.

Model-weighting and node-dropping strategies. As discussed in Sec. III, the weights assigned to local models during the averaging phase (step 3 of Fig. 1) can account for the quality and/or quantity of their local data. Specifically, we consider the following options:

- *uniform*: all local models are given equal weight;
- *no. samples*: weights are proportional to the number of samples in each local dataset;
- *no. classes*: weights are proportional to the number of classes in each local dataset;
- *entropy*: weights are proportional to the *entropy* of local data, an information-theoretic metric expressing, intuitively, how difficult it is to predict the class of a randomly-chosen local sample.

The "samples" strategy accounts for the quantity of local data, the "classes" one for its quality, and the "entropy" one for both. "Uniform", where all weights are equal, is added as a benchmark.

For node-dropping, we assume that five learning nodes are dropped after iteration 1, and compare the state-of-the-art strategy of dropping the slowest nodes [2], [5], [7] against the alternative one of dropping the nodes with the lowest weight. The rationale behind the latter strategy is that weights are linked to how significant the contribution that nodes can give to the learning process is, thus, dropping the lowest-weight nodes can reduce learning times without impairing the learning quality.

B. Experiment results

The first aspect of interest is the progress of the overall learning, i.e., the accuracy of the *global*,

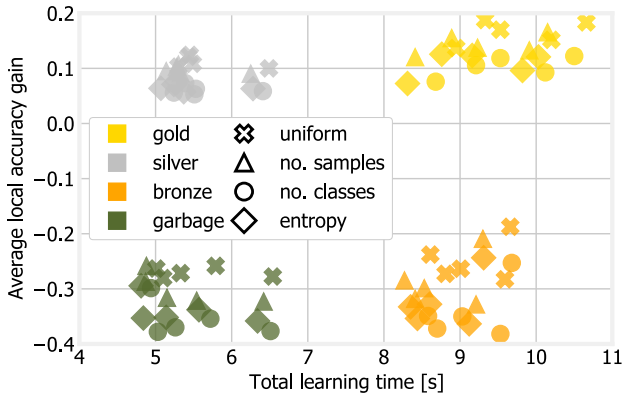

Fig. 3. Relationship between the average per-node learning time and the local accuracy gain for different categories of nodes (identified by the color of markers), under different model-weighting strategies (identified by the shape of markers).

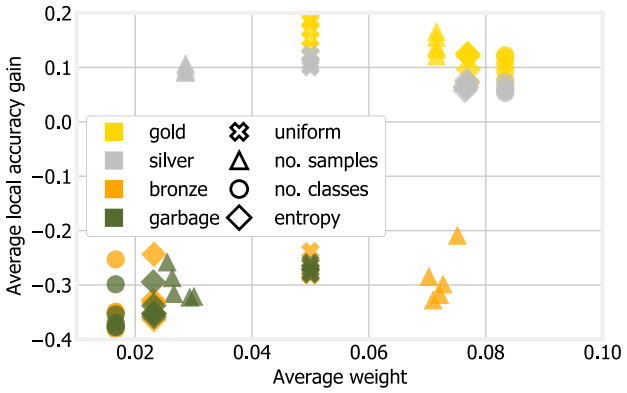

Fig. 4. Relationship between the weight assigned to local models and the local accuracy gain for different categories of nodes (identified by the color of markers), under different model-weighting strategies (identified by the shape of markers). Correlation coefficients for the "uniform", "no. samples", "no. classes" and "entropy" strategies are, respectively, 0, 0.07, 0.98 and 0.99.

averaged model (step 3 of Fig. 1), portrayed in Fig. 2. Each marker therein represents the state of the learning process after an iteration: its position along the x- and y-axis represent (respectively) the elapsed time and the global classification accuracy, while its color corresponds to the category of the slowest node in that particular iteration. Different lines correspond to different model-weighting strategies.

We can immediately observe that iteration times do not differ substantially across model-weighting strategies, and that they are usually determined by "gold" or "bronze" nodes – which makes intuitive sense, as those nodes have the largest local datasets. Even more interestingly, we can observe a clear difference in the classification accuracy obtained by different weighting strategies: giving the same weight to all nodes, or only accounting for the size of the local datasets, results in a lower accuracy than accounting for data quality. Furthermore, there is little difference between the "no. classes" and "entropy" strategies, suggesting that simply counting the observed classes can be as effective as adopting more complex metrics.

Next, Fig. 3 displays the relationship between the time taken by local learning iterations and the local accuracy gain, i.e., the improvement in classification accuracy obtained during local training (step 1 in Fig. 1). The latter metric can be seen as a measure of how much individual nodes contribute to the global learning process. In the plot, the *color* of each marker represents the category of the corresponding node, while the *shape* of each marker represents the model-weighting strategy. It is clear that, for all model-weighting strategies, local learning times are strongly correlated with the quantity of local data, while local accuracy gains are more strongly linked with the number of classes, i.e., the data quality. These results also suggest that only relying on local learning times for node-dropping decisions may result in removing nodes with large, hence, potentially valuable, datasets.

Fig. 4 shows the relationship between the weight assigned to each local model and the corresponding local accuracy gain. Similar to Fig. 3, the color and shape of markers represent, respectively, the node category and model-weighting strategy. We quantify the relationship between weights and accuracy gains, by *correlation coefficients*, expressing to which extent changes in one quantity are reflected by changes in the other: values close to 1 indicate strong correlation, values close to 0 little to no correlation.

In our case, it is clear that weights only considering the quantity of data (i.e., the "no. samples" strategy) may not be able to identify the nodes that can contribute the most to the learning process, e.g., it gives similar weights to the "silver" and "garbage" nodes. On the other hand, "no. classes" and "entropy" weights are very well correlated with accuracy gains, which again suggests how the quality of data has a high impact on the learning effectiveness. From both Fig. 3 and Fig. 4, it is also possible to see how better local accuracy gains

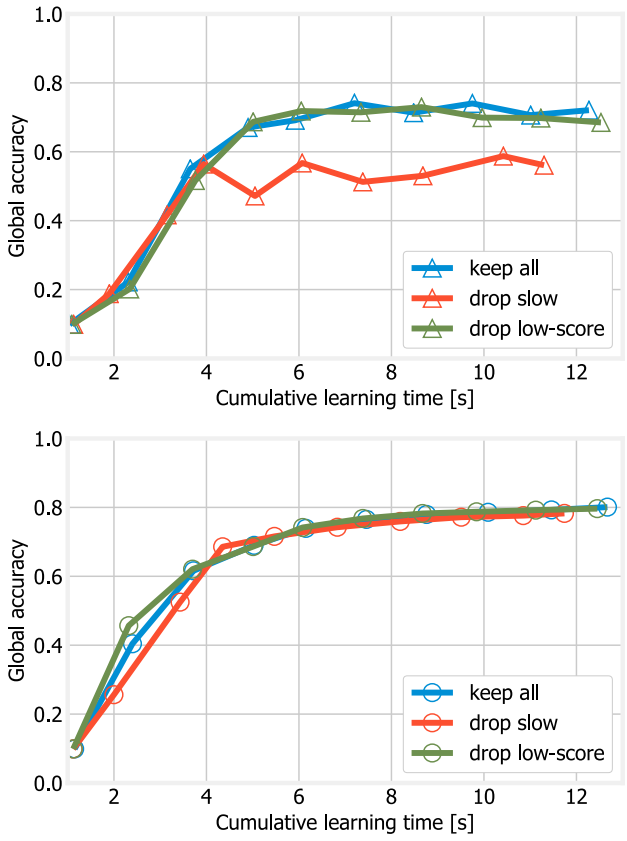

Fig. 5. Relationship between elapsed time and global accuracy for different node-dropping schemes, under the "no. samples" (top) and "no. classes" (bottom) model-weighting strategies.

do not necessarily coincide with better global classification accuracy. Indeed, good model-weighting decisions are necessary to consolidate local learning from different nodes into a consistent, high-quality global model.

Last, Fig. 5 shows the effect of different node-dropping strategies on the learning process, for the "no. samples' and "no. classes" model-weighting strategies. Specifically, we wait for the first five iterations, and then drop the five nodes with the lowest score, computed according to each node-dropping strategy. We can observe that, when weights only reflect the quantity of local data, dropping the slowest nodes significantly hurts the learning accuracy. On the other hand, more sophisticate model-weighting *or* node-dropping strategies yield virtually the same accuracy as keeping all nodes. This also highlights how model-weighting and node-dropping decisions interact with one another, and can represent different, complementary ways to achieve the same goals.

Summary. In conclusion, our results show that the *quantity* of data drives the computation time of local nodes; however, it is the *quality* of data that determines its usefulness to the global learning process. It is thus of paramount importance that model-weighting decisions do not solely account for data quantity or computation time, as that may adversely impact performance.

V. Take-away Messages and Challenges

Based on both the existing works discussed in Sec. II and Sec. III and the experiments reported in Sec. IV, we can highlight the following high-level lessons learned, which also point at interesting directions for future research.

Model weights matter: Assigning the right weights to local nodes during the averaging phase can have a very significant impact on the learning process, as highlighted in Fig. 2 and Fig. 5. Although few works in the literature have explored this option, Fig. 4 shows how weights accounting for the quality of data as well as its quantity are much more likely to identify the nodes that can contribute the most to the learning process. Importantly, the information needed to compute such weights is either already available or easy to collect for the learning server, and does not jeopardize the privacy properties of FL.

Data quality matters: Our experiments strongly underline the importance of dataset quality. An example is provided in Fig. 3 and Fig. 4, showing how nodes with more diverse data ("gold" and "silver") are able to offer much greater contributions to the overall learning process. Quantifying data quality is not a trivial task, however, our experiments show that even simple definitions based on counting the classes present in a given dataset yield very good results. Further research can explore additional aspects of data quality, e.g., its freshness [14].

Check why nodes straggle before dropping them: The global learning time of FL depends on the slowest node in each iteration; therefore, it is often tempting to try and speed learning up by dropping the slowest nodes [7]. Such a strategy is appropriate when the slowest nodes are indeed stragglers, with limited computational capabilities or poor connectivity [2], [5]; however, this may result in unduly excluding nodes with valuable, rich datasets. Privacy concerns often prevent the server from obtaining additional information on individual learning nodes; however, already-available data like

the size of local datasets can provide significant help to tell genuine stragglers apart from nodes that simply have a lot of data. If warranted, the latter can be directed to sample their own datasets, in a similar spirit to [4], so as to provide good contributions to the learning process with a smaller latency. This also points at the exciting research direction of extending the FL paradigm by allowing additional interaction between the learning server and learning nodes, striking the right balance between simplicity, privacy, and effectiveness.

FL is robust: This is not very surprising, since FL has been introduced for the very purpose of exploiting local, potentially heterogeneous, data from devices that cannot be centrally controlled. It is however interesting to highlight how the robustness of FL extends beyond tackling low-quality data, to tackling suboptimal configurations. An example is provided in Fig. 5, where it is sufficient to make high-quality model-weighting *or* node-dropping decisions to obtain very good learning performance. This suggests that FL is indeed a viable choice in those environments and scenarios where there is a significant likelihood that configuration decisions be suboptimal. Robustness to incorrect configuration is a relevant research area in distributed computing scenarios, and – so far – a neglected one.

VI. Conclusion

In the context of federated learning, we have considered the problems of model-weighting, i.e., assigning weights to local models during the averaging phase, and node-dropping, i.e., selecting the nodes to exclude from the learning process. After observing how those two decisions can tackle most of the issues and hurdles of FL, we have reviewed existing approaches thereto and found them to seldom depart from straightforward solutions based on the quantity of local data learning nodes have and their response time. Leveraging a set of real-world experiments, we have observed how more comprehensive approaches, accounting for the quality of local data and for the reasons behind longer node latency, can yield substantially better learning performance.

References

[1] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[2] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2020.

[3] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[4] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM*, 2020.

[5] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," in *IEEE ICC 2019*, 2019.

[6] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[7] G. Neglia, G. Calbi, D. Towsley, and G. Vardoyan, "The role of network topology for distributed machine learning," in *IEEE INFOCOM*, 2019.

[8] Y. Tu, Y. Ruan, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-Aware Optimization of Distributed Learning for Fog Computing," in *IEEE INFOCOM*, 2020.

[9] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020.

[10] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[11] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, 2019.

[12] M. Shin, C. Hwang, J. Kim, J. Park, M. Bennis, and S.-L. Kim, "XOR Mixup: Privacy-Preserving Data Augmentation for One-Shot Federated Learning," *arXiv preprint arXiv:2006.05148*, 2020.

[13] H. Huang, J. Huang, Y. Feng, J. Zhang, Z. Liu, Q. Wang, and L. Chen, "On the improvement of reinforcement active learning with the involvement of cross entropy to address one-shot learning problem," *PloS one*, vol. 14, no. 6, p. e0217408, 2019.

[14] A. A. Abdellatif, C. F. Chiasserini, and F. Malandrino, "Active learning-based classification in automated connected vehicles," in *IEEE INFOCOM PERSIST-IoT Workshop*, 2020.

Francesco Malandrino (M'09, SM'19) earned his Ph.D. degree from Politecnico di Torino in 2012 and is now a researcher at the National Research Council of Italy (CNR-IEIIT). His research interests include the architecture and management of wireless, cellular, and vehicular networks.

Carla Fabiana Chiasserini (M'98, SM'09, F'18) received her Ph.D. from Politecnico di Torino in 2000. She is currently a Full Professor with the Department of Electronic Engineering and Telecommunications at Politecnico di Torino, as well as the Vice Rector for Alumni and Career Orientation. Her research interests include architectures, protocols, and performance analysis of wireless networks.