Federated vs. Centralized Machine Learning under Privacy-elastic Users: A Comparative Analysis

Georgios Drainakis*, Konstantinos V. Katsaros†, Panagiotis Pantazopoulos‡, Vasilis Sourlas§ and Angelos Amditis¶
Institute of Communication and Computer Systems (ICCS)
Iroon Polytechniou Str. 9, GR-15773, Athens, Greece
Email: *giorgos.drainakis@iccs.gr, †k.katsaros@iccs.gr, ‡ppantaz@iccs.gr, §v.sourlas@iccs.gr, ¶a.amditis@iccs.gr

***Abstract*—The proliferation of machine learning (ML) applications has lately witnessed a considerable shift to more distributed settings, even reaching hand-held mobile devices; there, contrary to typical Centralized learning (CL) whereby the involved (large amounts of) training data are centrally gathered to train models, the load of training tasks is distributed across a set of capable mobile learners at the expense of their own energy. The idea of Federated learning (FL) has emerged as a privacy-preserving mechanism suggesting that the ML model parameters rather than data, are sent over the network to a central point of aggregation. However, when relaxing the privacy concerns, the debate strongly relates to the available network resources. Interestingly, the so-far theoretical or even experimental comparison of the two approaches overlooks network conditions and remains of low realism.**

In this work we rely on past measurement studies to introduce a realistic system model that accounts for *all* involved mobile network conditions such as bandwidth and data availability (affecting training accuracy and model aggregation) as well as user mobility patterns (affecting data loss). A dedicated simulation framework we have developed replays rich mobile-traces allowing for a comprehensive comparison of the two ML approaches over a large set of training data shedding light on network-resources utilization, energy efficiency and training convergence. Intuitively, our results suggest that the ratio between the employed raw data and the corresponding ML model shapes the conditions under which FL acts as a network-efficient alternative to CL. Interestingly enough, asymmetry in data availability across users as well as their varying number are shown to hardly affect the FL approach in traffic and energy needs, pointing both to its promising potential and the need for further research.

***Index Terms*—Federated Learning, Network Optimization, Simulation.**

I. Introduction

Machine learning techniques have demonstrated great potential in solving various classification and regression problems, where analytical or approximation methods are far less effective. By exploiting available recorded data, ML frameworks are able to identify patterns, analyse behaviors and even predict future values [1]. As such, these data-driven methods are typically employed to deal with challenges that emerge in mobile (network) environments and require information exchange between network devices and a central entity realising the learning process. Relevant applications span across different domains (*e.g.*, mobile applications, autonomous vehicles, smart sensors, Internet of Things) and serve various objectives (*e.g.*, image recognition, natural language processing, e-health *etc.*).

In the traditional approach of centralized learning (CL), clients acquire raw data from the environment (*e.g.*, measurements, audio, images, video *etc.*) and after the initial pre-processing, transmit it to a central server, which in turn performs the respective computationally-heavy model training task. Such a scheme however, places significant traffic overheads to the underlying (wireless) network, since training of complex tasks generally requires the exchange of large chunks of data, and also aggregates huge processing loads at a single location.

Distributed learning (DL) techniques have risen, as an alternative, seeking scalability by allocating the (learning) computational load to the distributed devices rather than the central server. Federated learning (FL), a Google driven DL scheme [2], initially emerged as a privacy-preserving mechanism, whereby the learning task is performed in a distributed manner by the potentially mobile clients and only the parameters of the learning model are communicated to the central server. By delegating the computational load to the clients, their (own) data is protected from third-parties access. The server acts as a controller thereafter, in charge of client selection, model parameter aggregation and scheduling. As a result, in FL the ML model is frequently transferred over the network, as opposed to the data in the case of CL. FL has already been employed in applications such as Google's predictive keyboards [3], healthcare with distributed data [4] and object detection using images from a vehicle [5].

In this paper, we argue that once the focus is shifted from privacy-concerns, to the ability of FL to distribute the load across clients in a (mobile) network, a comparison between CL and FL becomes increasingly interesting in understanding the impact of the differentiation between the traffic patterns in these two cases i.e., the training data exchange in CL *vs.* the exchange of model parameters in FL. A differentiation that in the presence of mobility has been so-far under-explored. In this context, our goal is to explore and compare the performance of the two approaches, in terms of *i*) network resource utilization, *ii*) energy efficiency and *iii*) ML (training) convergence, examining a realistically complex set of operating conditions including: *1*) Bandwidth availability: training a complex ML task in general requires exchange of large amount of data, which can lead to congestion in the network, affecting also the data/model transfer duration and the corresponding model convergence, *2*) Mobility: in a dynamic environment, client availability depends on user's mobility; if the user is offline, this can lead to communication failure, *3*) Data availability and distribution: even when a client is available, existence of adequate data is not always guaranteed, which in turn can cause delays in the training process, *4*) Client selection: refers to the configuration of the training process, in what concerns the selection of the total client set that will contribute to the learning process.

Existing works so far have attempted to address these questions in a restrictive manner. While there is extensive research both theoretical [6] and experimental [7] on the comparison between CL and FL in terms of ML convergence, other parameters that can affect the system's performance like user mobility and network resources are largely neglected. On the other hand, while there is considerable effort in optimizing FL techniques *e.g.*, in terms of communication efficiency [8] or bandwidth efficiency [9], the focus is on privacy-sensitive only environments, and hence an extensive evaluation and comparison with the CL alternative is missing.

Our work seeks to fill these gaps by developing a concrete and pragmatic system model and a correspondingly realistic evaluation environment. To this end, we use a dedicated Artificial Intelligence (AI)/ML software environment to accurately reproduce the ML process, we replay real-world mobility traces to capture mobility patterns and we employ accurate, measurement-based modeling assumptions e.g., regarding bandwidth availability, to study the above debate.

The results point to a series of interesting outcomes; the data/model size ratio determines the achievable accuracy of the trained models with the centrally-derived one exhibiting slightly better values, in line with previous observations. Furthermore, the above ratio largely shapes both the point of equal accommodated traffic and the point of equal energy consumption needed for the learning process. Interestingly, when more users share the available raw-data and contribute to the two approaches, the federated instance exhibits low sensitivity in both accommodated traffic and energy consumption terms. Finally, a welcome FL property is revealed as it is shown to remain robust when the raw-data is unevenly distributed across the participating learners. On a more forward-looking note, we highlight the need for in-depth exploration of both ML processes in the course of time to gain further insights, especially on the FL applicability.

The remainder of the paper is structured as follows. In Section II we detail the relevant literature. The system model is introduced in Section III. In Section IV, our extensive simulation results are provided, followed by concluding remarks and pointers for future investigation in Section V.

II. Related Work

A. Research on CL-FL comparison

Prior work performed on the comparison between CL and FL is limited to certain system parameters. While such an approach enabled the development of smart optimization algorithms to address a specific problem, *e.g.*, enhance classification accuracy, the absence of a thorough parameter investigation undermined a systematic and therefore more realistic view on the CL-FL comparison.

Optimizing ML accuracy in relation to the communication costs has lately attracted much attention. In [10], a clustering FL algorithm is employed for energy demand prediction by charging station providers, using data collected from charging stations in electric vehicle networks. FL is reported to achieve higher accuracy and reduction of communication costs by 83%, compared to CL algorithms, however it refers to a static setup. A FL protocol which adapts to sudden state changes (*i.e.*, drifts) is presented in [11]. The protocol, which is incorporated in the ML training phase, is compared against CL, in terms of training loss and communication costs. In [12], a collaborative learning framework is proposed to predict battery failure on electric vehicles, focusing on the comparison of CL-FL in regards to throughput, convergence time and accuracy and suggest fast convergence with similar-to-CL performance.

Reliability is another aspect that has been examined. In specific, FL is employed for vehicle-to-vehicle (V2V) communication in [13], enabling vehicular users (VUE) to accurately estimate the probability of an extreme event and thus facilitating ultra-reliable low-latency communication (URLLC) in a vehicular network. Based on this, a joint power control and resource allocation strategy is designed and compared against CL at the Road Side Units (RSU). The results suggest that the FL solution achieves higher levels of reliability, when the number of VUEs exceeds a certain value.

Finally, an interesting approach is taken in [14], where FL and CL are compared in a mobile network environment so that an optimal resource block allocation policy can be established, under imperfect channel state information (CSI). The study focuses mainly on the transmission parameters of the physical layer and on secondary factors, such as client fairness.

B. Research on FL optimization

Numerous studies have also been directed towards the optimization of FL methods in wireless and mobile environments. While these works study the effect of multiple parameters, they lack comparison to the CL alternative. However, the development of our system model was based on this research of various system parameters.

Most of these studies focus on client selection (to carry-out local training tasks) and the respective algorithms to improve network-related issues. In [15], the functionality of FL in a cellular environment is analysed. A closed-form optimization problem is formulated that aims to increase training accuracy compared to previous FL methods, using a joint client selection and resource allocation scheme. A similar process is followed in [16] whereby a scheduling policy is employed to promote clients with recently acquired data, in order to accelerate the convergence time of the training. Peer-to-peer bandwidth is leveraged in [17] to offload the main transmission channel. A gossip mechanism is designed to choose clients taking into account the available bandwidth and inter-client communication criteria. Finally, a holistic approach to client-device selection is presented in [18]. A selection algorithm is devised to maximize the number of available clients under physical layer and communication restrictions, aiming to decrease convergence time, without affecting accuracy.

Another group of studies, focuses on reducing the effects of mobility in a wireless environment for the training process. An asynchronous communication client-server mechanism that provides resilience to dropouts is shown in [19]. Similarly, in [20] a surveillance mechanism is introduced as a means of inspecting the system as a whole during training. Additionally, this study accounts for the incorporation of different heterogeneous devices in the process. A distributed optimization algorithm is presented in [21], as a solution for training in environments with intermittent client availability. Lastly, the distribution of available client data has also drawn attention and specifically the effect on the training process, under the assumption that data is distributed in an non independent and identically distributed manner (*i.e.*, non-i.i.d). In specific, [22] provides the theoretical framework for accuracy reduction under non-i.i.d. data. The same problem is analysed in [23], but is countered using compression techniques, while the theoretical conditions for convergence for non-i.i.d. data training are provided in [24].

In our work, we aim to address the gaps appearing along the two above-mentioned threads of research, exploring in-detail the involved parameters that determine the corresponding system's performance. Contrary to these works, we establish a system model that encapsulates all aforementioned parameters bounded to the constraints of a realistic system, as captured by measurement-based models and real mobility traces, enabling the accurate experimental comparison of CL and FL performance in dynamic mobile environments.

III. System Model

In our model, we consider a network environment comprising of several mobile clients and a central entity-server. The system's objective is to utilize available client data in order for the server, which operates as a training control manager for the whole process, to perform a machine-learning task. In the CL setup, the clients are asked to upload their local datasets to the server. When every transfer is completed, the server trains the model using the data acquired in that round. This process is repeated for several rounds, each time with a selection of a different group of clients, until a specified time limit is reached. For the FL example, the setup runs in a similar manner, with the exception of communication being limited to exchange of model parameters, instead of the complete datasets (Fig. 1). After the server shares the model to the clients, each selected client trains the model locally using its own data and computing resources. Then, it communicates the updated model parameters to the server. The server acts as a model aggregator, whose only subsequent task is to (re)-distribute the updated model to the clients that will participate in the next round.

A. Network model

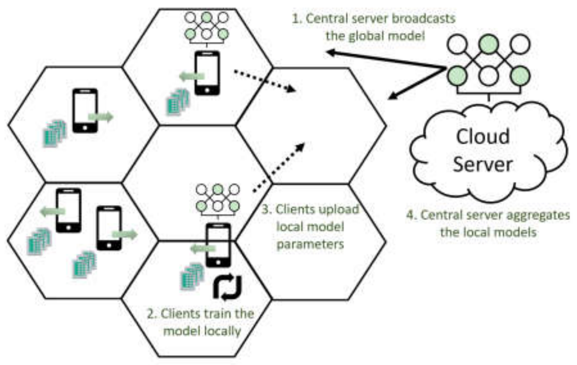
1. Central server broadcasts the global model
Cloud Server
3. Clients upload local model parameters
4. Central server aggregates the local models
2. Clients train the model locally

Fig. 1. In CL, clients moving (green arrows) between cells (hexagons) upload their data to the cloud server for training, while in FL they perform the training themselves (recursive arrows) and only upload the model's parameters.

We consider a mobile Long-Term Evolution (LTE) network, where the central entity refers to a cloud server or a data centre. We assume a total of K_T mobile clients in a time-period of T_{TOTAL}, within the overall network area under consideration. In each communication round the server identifies which clients are currently online (K_O) and selects a portion of them (K_S) to participate in the round's training.

The client throughput (R) in both the uplink (UL) and the downlink (DL) is described by the user equipment's (UE) upload/download speed (in MBytes/sec), which is modelled as a Gaussian random variable (Ñ). Its mean value is assumed equal to the average cell throughput (C)[1], divided by the number of online clients, while an artificial standard deviation parameter (sigma) is introduced, equal to 20% of the mean value, to account for throughput variations *e.g.*, due to path loss and interference. This choice allows to factor-in the way the locally-present number of users shapes the throughput provision in the considered area. Given that a client is online, we also assume that there exists a minimum throughput threshold (C_{min}) to enable server-client communication, both in DL and in UL. For LTE, C_{min} is assumed equal to the 5% cell edge rate[2]. Thus, client throughput is given by:

$$R = max\{\tilde{N}(\frac{C}{\|K_O\|}, sigma), C_{min}\} \tag{1}$$

where $\|.\|$ denotes a vector's norm.

B. Mobility model

Real-world traces promise realistic performance evaluation and credible results compared to the use of synthetic ones. However, concerns are typically expressed on their representativeness and generality of the evaluation results. Along this line, performing multiple iterations on a public dataset,

[1]Average cell throughput values (speed) for LTE are 5.9 (UL) / 6.7 (DL) MBytes/sec [25]

[2]That is, 0.24 (UL)/ 0.22 (DL) MBytes/sec [25]

increases the generality of our results. In specific, we have chosen the Shanghai Telecom Dataset [26] for LTE traces.

The Shanghai Telecom Dataset contains records of UEs accessing the Internet through base stations in a total period of 15 days. Each node denotes a base station in Shanghai, China. The database includes timestamps (taken every minutes, which is the dataset's time granularity) for connection initialization and termination, thus online presence can be calculated. The clients are monitored and are considered online, as long as they remain in the network, while communication is performed between the various clients and the cloud server. When a client moves to another cell, a handover (HO) is assumed, in order to capture service continuity.

We ignore the communication disruption during the HO process. However, a change of cell (and therefore base-station) will affect client throughput, depending on the corresponding congestion (III-A). The dataset provides a large set of clients, thus an initial pool for K_T is randomly generated.

Handover across network cells is also taken into account. A mobile client is thus considered able to communicate with the central server according to the Boolean rule:

$$\begin{aligned} HO(t) =& \textit{not } online(t) \ AND \\ & online(t - T_H) \ AND \\ & online(t + T_H), \end{aligned} \tag{2}$$

where t denotes the current time and T_H a threshold period for the HO. In our experiments, we assume a threshold equal to 1.5 times the time granularity of the LTE dataset i.e., the minimum that can be obtained in the given dataset, resulting in $T_H = 90$ secs. This might deviate from the achievable HO times in practice, but allows us to be consistent with the dataset semantics. We also note that communication disruption during the HO process is ignored.

C. Dataset acquisition & distribution

In general, there are numerous issues that arise in terms of raw data acquisition, related to each client's acquisition rate, the UE's data storage capacity and the data staleness level [16]. In our configuration, adequate in-volume data is assumed to become available in the clients' devices. Thus, the investigation is mainly focused on the distribution of the existing data. Two distinctive dataset distributions are studied, namely the evenly distributed setting (e.d), whereby data samples can be found in different users with the same probability and the non-e.d. setting, whereby data is distributed asymmetrically across clients. This distinction refers to the dataset size distribution, though the class distribution of the training is always considered independent and identically distributed (i.i.d.).

1) On e.d. setting: When the system runs under the e.d. assumption the total training dataset (data and labels) is equally divided to the ($\mathrm{K_T}$) clients. To ensure true data randomness, the original dataset is firstly randomly shuffled and afterwards the client that will receive each dataset record is determined by the following rule:

$$client\_id = mod(dataset\_sample\_id, K_{\mathrm{T}}) \tag{3}$$

where mod denotes the modulo operation, ensuring that all clients will receive the same amount of data in a sequential, yet independent manner.

2) On non-e.d. setting: For the non-e.d. setting, we assume a Gaussian distribution of data across the clients with a mean value equal to 50% and a standard deviation parameter (sd). Thus, Eq. (3) becomes:

$$client\_id = \tilde{N}(K_{\mathrm{T}} * 50\%, K_{\mathrm{T}} * sd) \tag{4}$$

D. Client selection model

As discussed in Section II-B, various communication models have been so-far studied in order to enhance the performance of machine learning in wireless/mobile environments. In the present analysis however, no optimization technique has been introduced to the model, so that the strengths and weaknesses of the two methods can be revealed. As a result, the client selection process is performed in a randomized manner. The decision-making algorithm operates as follows:

In each communication round, the server identifies online clients (K_O) as a subset of total clients (K_T). The server keeps track of all clients that participated in the training process, so it excludes the ones that their data has already been used, resulting in the subset of available clients (K_A). Then, it aims to randomly select a (predetermined) fixed number of clients (K_N) that are required for the training round. If there are not enough K_A, all currently available clients are selected. Selection process can then be described by the following rule:

$$K_{\mathrm{S}} = \begin{cases} rand(K_A, K_{\mathrm{N}}), & \text{if } K_A > K_N \\ K_A, & \text{otherwise} \end{cases} \tag{5}$$

where $rand(S, x)$ is a random selection of x from a set of S and $\mathrm{K_S}$ are the selected clients. If no clients are found or all clients have already participated in the process, the round is repeated after a waiting period of 60 secs, to account for the mobility dataset semantics (III-B).

In the CL case, the selected players upload their local datasets to the central server in a parallel manner. The time required for each upload is simply modelled as $T = Dataset\_Size/Client\_Throughput$ [18], where $Client\_Throughput$ follows the Section III-A description.

When all players upload their datasets, the central server performs the ML task and marks the end of the round (we assume infinite resources for the central server *i.e.*, zero computation time for the ML task [18]). The time to upload all datasets equals to that of the "slowest" client. This procedure is repeated until a global time limit, set by the server is reached. In case a player goes offline during upload (irrespective of the upload completion percentage), we define a communication failure. If such a failure occurs, the client's contribution is neglected by the central server. However, to account for the time and resources spent for the (partial) communication, we consider the delay time equal to the estimated upload time, assuming the worst-case scenario that connection is lost, when almost all the data was uploaded. The estimated upload time can be calculated, given our constant throughput model, as described in Eq. (1) and the a-priori known per-client dataset size.

The same settings overall apply to the FL case. The main difference is that the server shares the training model with the clients, then the clients train the model with their local individual datasets and finally return the updated model to the server (Fig. 1). Again, this is performed in a parallel manner. In the FL case, communication failure can also occur when the server shares the model with the clients and vice versa.

E. Computational and Energy Consumption model

The computational throughput (V_C) of a mobile device to perform a ML task, measured in (processed) training samples/sec depends on the dataset content, the UE's capabilities and the training model's complexity. A good approximation for some popular large-scale classification tasks can however be deducted from [27], since different models have been tested in various configurations. To enhance precision, we calculate the computational time needed for a UE's training round by measuring the actual time spent by our setup to perform the task, since our system's processing capacity (CPU) is similar to that of a modern smartphone equipped with a graphics (GPU) or neural (NPU) processing unit [28].

Regarding energy consumption, the UE's functions related to the study of the present system are considered in isolation. Any consumption related *e.g.*, to the device's operating system functionalities or displaying, is neglected and we focus on energy expenditure due to transmission (TX)/ receiption (RX) of data and ML training tasks. Thus, the energy consumption (battery discharge) of a device ($\mathrm{E_i}$) is then the sum of all these functions, E_i=E_i^{TX}+E_i^{RX}+E_i^{ML}.

Energy consumption in a period (t) per UE can be calculated as $E_\mathrm{i} = P_\mathrm{i} * t$, where P_i stands for the respective power consumption. For the transmission expenditure we are based on [29], where average power consumption valuesare reported P_i^{TX}=2.2 Watts, P_i^{RX}=1.5 Watts for LTE. For the training expenditure, based on [27], we assume P_i^{ML}=2 Watts for our training task of SVHN (see Section III-F). The sum of all E_i is used to compared the total (system) energy consumption between the FL and the CL setup.

F. Machine Learning

Fig. 2. Representation of SVHN dataset

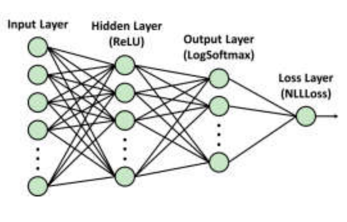

Fig. 3. Neural network model architecture

A representative machine-learning task, namely the Street View House Numbers (SVHN) dataset is selected as an application model. SVHN is a dataset corresponding to an image classification problem and is widely used in bibliography *e.g.*, [30]. It is based on a real-world image dataset (Fig. 2), with digits from natural scene images (house numbers in Google Street View). It contains 531,131 32x32 colour training images (1.3 GB) in 10 classes (for digits 0-9) and 26,032 test images.

To solve the above-mentioned task, a neural network was built (Fig. 3), to be used in both CL and FL case. The network is comprised of an input layer of 3072 neurons, which correspond to the total pixels of the input images of SVHN (32x32x3), an output layer of 10 neurons, equal to the total output classes of SVHN and an intermediate (hidden) layer of 512 neurons. Rectified Linear Unit (ReLU) activation is applied on the hidden linear layer (ReLU functions as a filter, allowing only positive values to pass through), while on the output layer LogSoftmax activation [31] is selected, being more effective for N-element classification tasks. To calculate training loss (loss layer), the negative log-likelihood loss function (NLLLoss) is preferred [31], since it couples together with LogSoftmax for classification tasks. In regards to hyper-parameter settings, a batch size of 64 samples was chosen, along with a learning rate of 0.1. The total model size reaches 6.1 MB.

IV. Experimental Evaluation

Based on the above system model, here we focus on evaluating the performance of the two considered ML setups (*i.e.*, FL and CL) in an effort to gain a better understanding of the impact of the considerably differentiated traffic and computing patters (distributed vs. centralized). Through extensive simulation study we mainly seek to address the following questions: *1*) Network resource consumption is obviously expected to be affected by the differentiated traffic patterns. As the FL and CL traffic loads are mainly dictated by the ML model size and the training dataset size (*i.e.*, the set of data delivered by a client at each round of the learning process), respectively exchanged over the network, we expect the dataset/model size ratio to heavily shape the resulting comparison in what concerns network resource consumption *i.e.*, a large ratio points to a larger impact of a CL setup on network resource utilization, while a low value is expected to associate FL with higher network loads. This obviously affects a series of aspects related not only to network resource consumption, but also to energy consumption, the impact of network disconnections *etc*. An emerging question then is: what is the impact of the dataset/model size ratio on the CL *vs.* FL comparison, in presence of the effects of mobility and a varying throughput? *2*) The raw data distribution (e.d. *vs.* non-e.d.) is expected to mainly shape the resulted ML accuracy. Does the non-e.d. setting favor CL over FL? And if so, to what extend are network resources and energy consumption (indirectly) affected? *3*) The option to (dynamically) scale the number of participating users per round is essential from a system management point of view to mitigate the effects of mobility or to speed up the process, when quick convergence is desired. What is the effect of such scaling in the resulted accuracy? Is there any dependency between the per round participants and the total resource consumption?

A. Simulation setup

The simulations were performed on a single desktop machine with the following characteristics: Intel Core i7-10700 CPU @ 2.9 GHz, 64-bit, RAM 16 GB, OS Windows 10. To emulate the distributed learning environment Pysyft library [32] was used. For each measurement the following process is assumed for CL and FL case. An initial pool of K_T clients is selected. The total training dataset is distributed among them, as described in Section III-C. Also, the number of clients that need to participate per round (K_N) is set. Then, using the mobility dataset's timeframe as global base, the training process (realised as a Pysyft simulation) runs for a total duration of $T_{TOTAL} = 2$ hr. All relevant measurements are taken on the expiration of the global time limit.

B. Evaluation metrics

The following metrics are used to evaluate the performance of CL *vs.* FL.

1) Training Loss: Refers to neural network's loss function, which is the function we aim to minimize during the learning process optimization. This is a parameter related to the training data. Ideal value is equal to 0 (dimensionless).

2) Test Accuracy: The effectiveness of the trained model is evaluated on the test data. The percentage of successful to total classifications provides the test accuracy metric. Ideal value is equal to 100%.

3) Traffic Volume: The network traffic volume, due to the transmission of the training data or the ML model parameters. In particular, it is calculated by aggregating all the data exchanged between the central server and the clients during the training process (measured in MB), as described in III-A-III-D. In case of CL, the exchanged data refers to the actual data uploaded by the clients during the training rounds, while in FL it refers to the model parameters uploaded by the clients plus the ones shared by the central server (Fig. 1). It is noted that the above calculation does not account for multi-hop transmissions; we assume one direct transfer to the central server. For the sake of clear representation, traffic volume is normalized to the total dataset size (1.3 GB).

4) Energy Consumption: Captures the users' energy expenditure during the learning tasks. It is calculated by combining the total energy spent for ML processing and server-client communication (Section III-E).

C. Effect of the client dataset/model size ratio

To study the importance of client dataset size to model size ratio, the following setup is assumed. The training dataset is uniformly divided into a number of initial clients, *i.e.*, K_T, which varies from a minimum of 20 to a maximum of 800 (16 values totally). Since the model size if fixed, this essentially is equal to a varying value of the per client dataset size. We also assume a fixed $K_N = 5$, in each round. Each experiment is repeated for 12 different time periods, randomly chosen from the LTE mobility dataset, with a constant duration of $T_{TOTAL} = 2$ hr each. Effectively, this gives a total of 192 pairs of CL-FL experiments. The results *i.e.*, mean values out of the 12 samples together with the corresponding standard deviations, are depicted in Figs. 4-7.

In the presence of enough data per user, at least 2 times the size of the model, FL is shown to achieve an equivalent performance to that of CL, in terms of both ML theoretical convergence (as analysed in IV-B) *i.e.*, reduction of training loss (Fig. 4) and practical implementation *i.e.*, accuracy of classification of test data (Fig. 5). On the other hand, FL fails to converge when data size is close to the model size, since the individuals learners do not have enough local data to train the model [33], thus propagate erroneous parameters. As a result, in the area of the ratio equals unity or less, FL, in our setup, consumes ineffectively large quantities of bandwidth and energy (due to transmission). Given that in FL, models are exchanged twice (upload/download) per round, traffic volume exceeds the total dataset size.

Interestingly, there is an area of a relative equilibrium, for small values of data/model ratio. When the considered ratio equals 2, CL has similar network performance with FL (Fig. 6), though outperforms FL in energy consumption by 21% (Fig. 7). For this ratio value, the network traffic induced is by definition equivalent, while, as expected, the FL suffers from the additional energy costs of the training process. For ratio=3, FL has similar energy performance with CL, though outperforms it, in terms of network resource consumption by 27%. We deduct that there is no straightforward choice for this area and the selection depends on the viewpoint of the problem. For example, from the user's perspective, CL is preferable for ratio=2, in order to minimize battery depletion effects. On the other hand, for ratio=3, a user would be indifferent, in what concerns battery depletion, but from the providers point of view, FL is preferable, to minimize network costs. For a higher ratio values, namely in the area of 10, FL on average outperforms CL, both in regards to network (by 82%) and energy consumption (by 42%). In sharp contrast to the case of small ratio values, the relatively small model size clearly benefits FL applications allowing for cost-effective transfers.

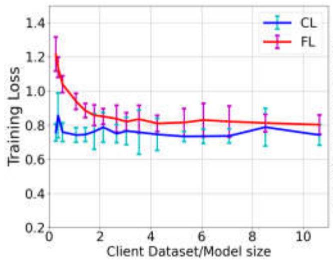

Fig. 4. Training loss *vs.* client data-model ratio

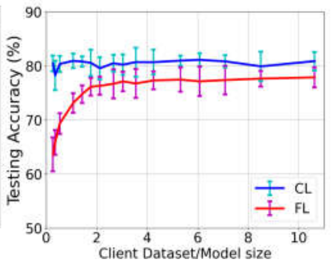

Fig. 5. Testing accuracy *vs.* client data-model ratio

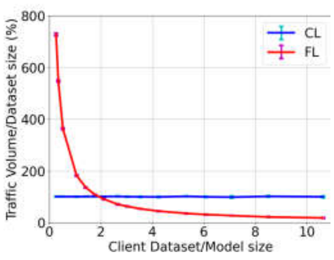

Fig. 6. Traffic volume as percentage of the total dataset *vs.* client data-model ratio

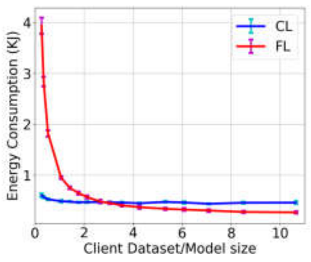

Fig. 7. Energy consumption *vs.* client data-model ratio

D. Effect of data distribution

To understand how different training data (volume) distribution settings affects the system performance, we assume non-e.d. settings and choose varying values for sd in Eq. (4), namely [20%, 40%, 80%, 160%, 320%], resulting in a mixture of balanced and unbalanced distributions (of raw data availability across the clients), as shown in Fig. 8. We fix the values of K_T=100 and K_N=5 and repeat each experiment for 5 different time periods. The accuracy averages are depicted in Fig. 9.

Unlike FL's vulnerability in unbalanced class distribution [22], in the case of unbalanced client dataset distribution, FL has proven to exhibit considerable robustness, in what concerns the achieved accuracy. Under asymmetry, a limited number of users hold considerable amounts of data and contribute accurately-trained models; that is shown to be enough for achieving a stable yet closely approximating (less than 8%) the centralised case.

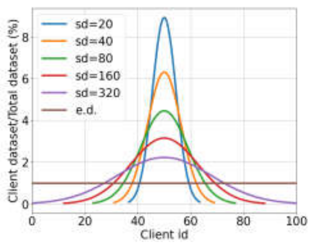

Fig. 8. Settings for data distribution among clients

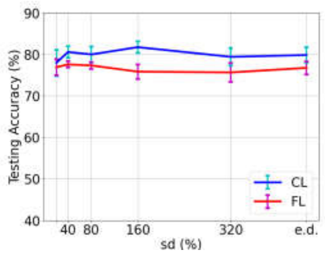

Fig. 9. Effect of data distribution on accuracy

E. Effect of scaling the number of users per round

For this set of simulations, K_T is fixed to 100 clients (suggesting a client dataset to model ratio equal to 2). Given the above-mentioned results, we have chosen this value, in the area where CL's performance is similar to that of FL. To examine the effect of scaling the number of users participating at each ML training round, we vary K_N=[5, 10, 15, 20, 25]. Each experiment is repeated for 5 different time periods. The results are depicted in Fig. 10-11. Increasing the participating users in each round does not have an effect on the final accuracy or network resource expenditure for CL or FL. In other words, given enough total data, with a few or many contributions per round, whether data chunks or model parameters, the ML task is effectively presenting a fairly stable behavior at the end of the overall training process. Thus, in terms of network management, the central server is flexible to decide whether to accept a burst of clients in order to accelerate the ML process, or proceed gradually with the process in case of network resource restrictions, without affecting the outcome of the ML task.

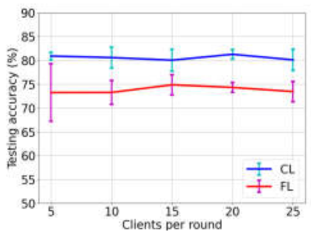

Fig. 10. Effect of K_N on Accuracy

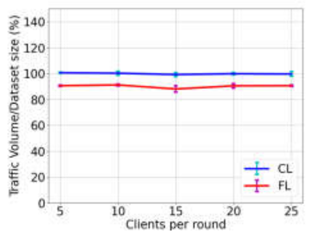

Fig. 11. Effect of K_N on Traffic Volume

F. The data loss dimension

The selection of $T_H = 90$ sec, as described in Eq. (2), is dataset-specific *i.e.*, determined essentially by the dataset specifications even if not reflecting highly realistic HO time values. To estimate the effect of this assumption on the performance metrics, we -in all cases- measure the total data lost. That is due to the users' mobility and the corresponding offline periods. We report the mean and max values of data lost, as a percentage of the total dataset size in TABLE I. We deduct that losses are minor in relation to the size of the dataset, thus our initial assumption has not considerably affected the derived results. However, in other congested network environments (potentially captured by other mobility traces) the question of data loss remains open and seems of particular interest for future investigation.

TABLE I
DATA LOST PER SIMULATION

Data Lost/Dataset Size (%)	Sec. IV-C	Sec. IV-D	Sec. IV-E
Mean (CL)	0.40	0.51	0.49
Max (CL)	1.06	0.76	0.82
Mean (FL)	0.26	0.15	0.18
Max (FL)	1.07	0.47	0.28

V. CONCLUSIONS AND FUTURE WORK

We have presented a novel systematic model, based on previous measurement studies and realistic modelling assumptions aiming to establish a solid framework for the comparison of centralized (CL) and federated (FL) learning instances in mobile environments. Our carefully designed model accounts for all involved network conditions (*e.g.*, bandwith availability) as well as user characteristics (*e.g.*, mobility profiles) and assess the way that the two ML instances compare (in terms of accuracy) and at the same time shape network resources and energy consumption. Simulation results suggest that FL exhibits robustness against non-uniform data volume distributions, while the number of clients sharing the available raw data per training round is not a key factor for the achievable accuracy and network performance; importantly, the critical role for the render one approach outperform the other in the mobile environment is played by the data to model size ratio.

Our current findings call for some interesting extensions; one would involve the training of more complex models and exploration in greater depth the effect of data to model ratio for the selection of CL *vs.* FL. This could also be extended to the training of regression ML tasks. The exploration of other wireless environment and the relevant mobility patterns *e.g.*, in wireless local area network (WLAN) hot-spots could add to completeness of this study. Finally, grasping the way that the training evolves across time remains an open issue. Along this line, the questions of the achievable accuracy in each CL or FL training round as well as the implications of large datasets, complex models and number of users in the training completion time constitute the subjects of our upcoming work.

References

[1] M. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *arXiv preprint arXiv:1908.00080*, 2019.

[2] B. McMahan and D. Ramage, "Google ai blog: Federated learning: Collaborative machine learning without centralized training data," https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, April 2017, (Accessed on 10/27/2020).

[3] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[4] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Differential privacy-enabled federated learning for sensitive health data," *arXiv preprint arXiv:1910.02578*, 2019.

[5] A. M. Elbir and S. Coleri, "Federated learning for vehicular networks," *arXiv preprint arXiv:2006.01412*, 2020.

[6] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 63–71.

[7] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020.

[8] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[9] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[10] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[11] M. Kamp, L. Adilova, J. Sicking, F. Hüger, P. Schlicht, T. Wirtz, and S. Wrobel, "Efficient decentralized deep learning by dynamic model averaging," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 393–409.

[12] S. Lu, Y. Yao, and W. Shi, "Collaborative learning on the edges: A case study on connected vehicles," in *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[13] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency v2v communications," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[14] M. M. Wadu, S. Samarakoon, and M. Bennis, "Federated learning under channel uncertainty: Joint client scheduling and resource allocation," *arXiv preprint arXiv:2002.00802*, 2020.

[15] 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," *arXiv preprint arXiv:1909.07972*, 2019.

[16] H. H. Yang, A. Arafa, T. Q. Quek, and H. V. Poor, "Age-based scheduling policy for federated learning in mobile edge networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8743–8747.

[17] J. Jiang, L. Hu, C. Hu, J. Liu, and Z. Wang, "Bacombo—bandwidth-aware decentralized federated learning," *Electronics*, vol. 9, no. 3, p. 440, 2020.

[18] 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)*. IEEE, 2019, pp. 1–7.

[19] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices," *arXiv preprint arXiv:1911.02134*, 2019.

[20] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[21] Y. Yan, C. Niu, Y. Ding, Z. Zheng, F. Wu, G. Chen, S. Tang, and Z. Wu, "Distributed non-convex optimization with sublinear speedup under intermittent client availability," *arXiv preprint arXiv:2002.07399*, 2020.

[22] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[23] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, 2019.

[24] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[25] M. R. Akdeniz, Y. Liu, M. K. Samimi, S. Sun, S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter wave channel modeling and cellular capacity evaluation," *IEEE journal on selected areas in communications*, vol. 32, no. 6, pp. 1164–1179, 2014.

[26] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160–168, 2019.

[27] J. Liu, J. Liu, W. Du, and D. Li, "Performance analysis and characterization of training deep learning models on mobile device," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2019, pp. 506–515.

[28] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool, "Ai benchmark: All about deep learning on smartphones in 2019," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019, pp. 3617–3635.

[29] A. Nika, Y. Zhu, N. Ding, A. Jindal, Y. C. Hu, X. Zhou, B. Y. Zhao, and H. Zheng, "Energy and performance of smartphone radio bundling in outdoor environments," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 809–819.

[30] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in neural information processing systems*, 2017, pp. 1195–1204.

[31] Pytorch neural network api. [Online]. Available: https://pytorch.org/docs/stable/nn.html

[32] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017*, 2018.

[33] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.