Universidade Federal
do Rio de Janeiro
Escola Politécnica

AN ANALYSIS OF FEDERATED LEARNING ON MOBILE NETWORKS

Kaylani Bochie

Final Undergraduate Project presented to the Electronics and Computer Engineering Course, Polytechnic School, Federal University of Rio de Janeiro, as part of the requirements necessary to obtain the degree of Engineer.

Supervisor: Miguel Elias Mitre Campista, D.Sc.

Rio de Janeiro – RJ, Brazil

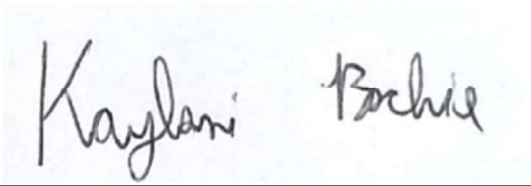
October, 2021

AN ANALYSIS OF FEDERATED LEARNING ON MOBILE NETWORKS

Kaylani Bochie

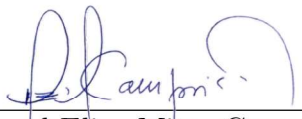
PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:



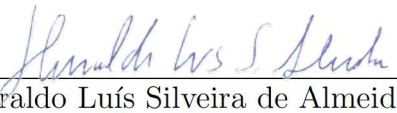
Kaylani Bochie

Orientador:



Prof. Miguel Elias Mitre Campista, D.Sc.

Examinador:



Prof. Heraldo Luís Silveira de Almeida, D.Sc.

Examinador:

Prof. Rodrigo de Souza Couto, D.Sc.

Examinador:

Matteo Sammarco, Dr.

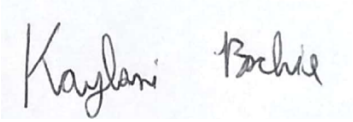
Rio de Janeiro – RJ, Brazil

October, 2021

Declaração de Autoria e de Direitos

Eu, *Kaylani Bochie*, CPF *149.326.847-38*, autor da monografia *an analysis of federated learning on mobile networks*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.
2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e ideias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.
4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.
5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.
6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/indenização nesse sentido.
7. Por ser verdade, firmo a presente declaração.



Kaylani Bochie

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

ACKNOWLEDGMENTS

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, Brazil, FAPERJ, Brazil Grants E26/211.144/2019 and E-26/202.689/2018, and FAPESP, Brazil Grant 15/24494-8.

ABSTRACT

The increase in data production has enabled a newly found interest in deep-learning-based solutions. From service customization to healthcare applications, deep neural networks have successfully been deployed to enable a multitude of uses. However, traditional deep learning applications must collect large volumes of data to guarantee good model performance. This pattern of data collection paves the way for recent privacy concerns, such as misuse of personal data and personal information leaks. In this context, federated learning arises as an alternative to traditional centralized deep learning applications. Federated learning enables clients to collectively train a global model without sharing their private data by using a central management server. Nevertheless, parameters such as network conditions, client availability, and hardware heterogeneity can impact federated learning performance, specially in wireless mobile network scenarios. As such, this work evaluates how mobile network parameters impact federated learning performance. This is achieved by simulating ideal federated learning conditions, where the network and the clients do fail, and introducing real wireless network conditions for comparison. Our results show how faulty clients impact inference performance, overall latency, and model convergence. These results reinforce the need for client-server orchestration to dynamically adapt the algorithm depending on network conditions.

Keywords: machine learning, deep learning, federated learning, neural networks, privacy, mobile networks.

Abbreviations

CIFAR - Canadian Institute for Advanced Research

CMFL - Communication-Mitigating Federated Learning

CNN - Convolutional Neural Network

DNN - Deep Neural Network

FedCS - Federated Learning with Client Selection

FedMMD - Federated Maximum and Mean Discrepancy

GPU - Graphical Processing Unit

HDF - Hierarchical Data Format

IFCA - Iterative Federated Clustering Algorithm

IoT - Internet of Things

NLP - Natural Language Processing

RTT - Round-Trip Time

SGD - Stochastic Gradient Descent

SNR - Signal-to-Noise Ratio

UFRJ - Universidade Federal do Rio de Janeiro

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The popularization of deep-learning-based solutions for computer networks is undeniable [2, 3]. From action recognition [4] to botnet detection in the Internet of Things (IoT) [5], deep learning has overtaken traditional algorithms for developing new solutions. However, training deep learning models requires a large amount of computational power, which is usually available at the network core in the form of central servers. Therefore, the common approach is to aggregate the data from the data producing devices in a central server to train the learning models. This approach raises some privacy concerns, specifically for privacy sensitive applications, such as in insurance [6] or the healthcare domain [7].

By trying to diminish privacy concerns on deep learning applications, federated learning then emerges as a new paradigm to solve these privacy issues [8]. This is achieved by eliminating the data transmission from the data producing devices to the central server. Instead, each data producing device locally trains a learning model on its own data. Afterwards, the client sends the resulting machine learning model to a central server, which then aggregates the parameters received from different clients and coordinates the model distribution to all devices. Since privacy concerns are contributing to a consistent growth in popularity of the federated learning paradigm, similarly to other technologies, studies must be performed to better understand its performance, explainability, security, robustness, *etc.*

Due to its relatively novel *status*, since the "official" creation of the federated learning paradigm, credited to McMahan et al. in 2017 [8], this new paradigm still requires extensive exploration to justify its use in multiple applications. More-

over, given the dynamic and heterogeneous nature of data in mobile networks, it is possible that concepts learned from experiments on high availability networks may not properly be applied to mobile networks. This is due to the effects that low client availability and highly variable hardware configurations may have on federated learning performance [9]. Therefore it is still required to investigate how federated learning algorithms behave in typical scenarios of mobile networks, with the goal of proposing improvements to both overall performance and latency.

This work evaluates the impact of mobile network parameters on the performance of federated learning algorithms. Parameters such as failure probability, hardware limitation, low availability, and transmission delay are evaluated. To enable reproducibility only open source datasets and software are used. The code used on this work can be found in an open repository[1].

This work starts with a revision step, with the goal of identifying datasets, learning algorithm, and frameworks commonly used in the literature. After this initial framing step, the evaluation of centralized learning models is performed, with the goal of establishing a performance baseline for future comparison. Afterwards the ideal federated learning scenario is simulated, *i.e.*, factors such as communication delay and reduced availability are ignored. Subsequently the mobile networks parameters are introduced and their impact is evaluated. Finally, possible improvements for the federated learning setting under mobile networks constraints are proposed and evaluated.

The remainder of this work is organized as follows. Chapter 2 explains the concepts behind federated learning and the relationship with mobile networking parameters. Chapter 3 presents the experimental setup, *i.e.*, the methodology, the dataset, the tools used, and the experiments' configuration. Chapter 4 presents the obtained results, along with the conclusions taken from the experiments. Chapter 5 reviews papers related to this work. Finally, Chapter 6 concludes this work and presents future research directions. Appendix A presents the publications made during the development of this work.

[1] `https://github.com/kaylani2/sbrc2021`.

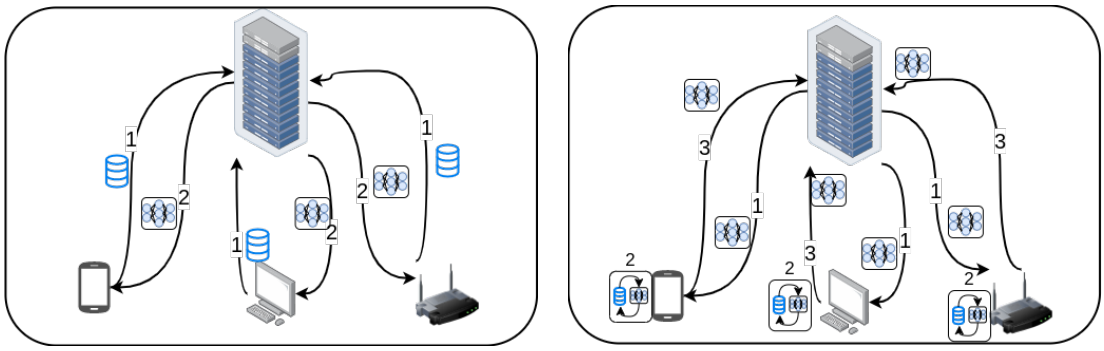
Chapter 2

Federated Learning

2.1 Introduction

Federated learning is an emerging decentralized and distributed machine learning paradigm [8]. It enables devices to leverage their own computing power to collaboratively train a global model on private and locally available data, without transmitting the data to another location, usually high-performance servers [10, 11]. The main motivation of federated learning is to enable the development of privacy algorithms coupled with high performance Deep Neural Networks (DNNs).

It is worth noting how different categories of possible federated learning settings and how their individual characteristics can impact performance during training and deployment. Considering that, Figure 2.1 illustrates the main differences from traditional, *i.e.*, centralized learning settings and federated learning settings. Figure 2.1(a) shows that the traditional approach requires that the clients send their data to a centralized server, which may be an issue for privacy sensitive applications, such as healthcare applications [12, 13, 14, 15]. As Figure 2.1(b) shows, federated learning, alternatively, distributes a learning model to clients and requires that clients send back only the learning model's parameters, after 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 trained model can be distributed to the clients and then be used to performance inferences.



(a) Traditional (centralized) learning setting.

(b) Federated learning setting.

Figure 2.1: Traditional learning and federated learning settings, reproduced from [1]. (a) During step 1 the data is sent from the clients to a centralized server to be used during training; and step 2 consists of the server sending back trained machine learning models to the clients. (b) Step 1 consists of the server initializing a learning model and distributing it to the clients; during step 2 the clients train the models on their respective local datasets; finally, on step 3 the model's parameters are sent back to the server to be aggregated.

Algorithm 1 presents a federated learning training algorithm, namely, FederatedAveraging.

Although federated learning can aid in privacy preservation, this new paradigm suffers from new and interesting challenges. These challenges can be divided into four main categories. Firstly, on 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 data collected by the device, 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 variation is more pronounced in the mobile scenario. 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 the device has not been used for several hours. Finally, hardware heterogeneity can affect performance.

Algorithm 1: `FederatedAveraging` [8]. Here, the scenario uses K clients indexed by k, uses B as the local minibatch size, uses E as the number of local epochs, uses ρ_k as the local dataset of the kth client, and η is the local training rate. Additionally, the parameter C controls the number of clients used in a training round.

Input: w_0

foreach *round in* $t = 1, 2, ...$ **do**
  $m \leftarrow max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  foreach *each client* $k \in S_t$ ***in parallel*** **do**
    $w_{t+1}^k \leftarrow$ ClientUpdate (k, w_t)
  end
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{K} w_{t+1}^k$
end
ClientUpdate (k, w):
$\beta \leftarrow$ (split ρ_k into batches of size B)
foreach *each local epoch* i *from* 1 *to* E **do**
  foreach *batch* $b \in \beta$ **do**
    $w \leftarrow w - \eta \nabla l(w; b)$
  end
end
return w to server

Again, this characteristic is more prominent in mobile scenarios. All of these challenges motivate a in-depth and thorough study of federated learning techniques and their applications [8, 9]. This work strives to explore how the previously presented challenges affect federated learning performance and how to improve it.

2.2 Mobile Network Parameters

By examining Figure 2.1(b) we can 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, obviously increases the time it takes for the global model to finish training rounds.

2.3 Cross-Silo vs. Cross-Device

After understanding how federated learning contributes to users' privacy, it is important to separate its two main scenarios: **cross-silo** and **cross-device**.

In cross-silo federated learning, clients are managed by a single entity, *i.e.* a centralized server. Therefore, parameters such availability and computational power can be defined *a priori*. It must be noted that this scenario also does not concentrate the data in a central location, and the training is distributed among the clients. However, since the clients are managed by the server, it is possible to orchestrate and define strategies to maximize performance with complete client coordination. Additionally, some of these applications are deployed with a specific set of tasks in mind, so hardware selection can be performed before deploying the application, or even as a service, such as the NVIDIA Clara framework [16]. Alternatively, on cross-device scenarios, clients usually have varied hardware configurations and are available at quasi-random times. Even though the server sends the model to each client, clients may not participate during training, and this decision can be "made" by the device, not by the server. Naturally, federated learning on mobile networks falls under the category of cross-device federated learning, where mobile network parameters greatly impact the overall model performance. If devices have limited hardware, they may not execute multiple local training epochs; memory availability limits the size of the global model; and finally, available bandwidth hinders the transmission of large volumes of data.

Chapter 3

Experimental Setup

3.1 Methodology

A simplified workflow representing the methodology employed in this work can be seen in Figure 3.1. The neural network was firstly trained and evaluated in a centralized setting, aiming to establish a performance baseline to serve as comparison for the federated results. The baseline was obtained using an early stop mechanism to achieve improved performance on the centralized setting [17]. Classification metrics were 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 were ignored, and the clients had the failure probability set to zero percent. The results obtained on this step were used to evaluate the number of rounds necessary for the model to achieve 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 clients, (ii) training the model on each client with their particular datasets 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:

- N_c: number of participating clients.

- E_t: number of training epochs for each client.
- B_c: local batch sizes for each client.

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 on real world applications of federated learning, 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.

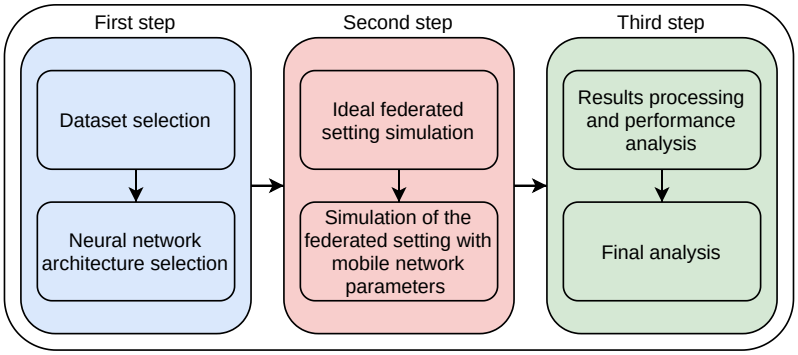


Figure 3.1: Employed methodology.

3.2 The CIFAR-10 Image Recognition Dataset

CIFAR-10 is an image classification dataset widely used for computer vision tasks [18]. Figure 3.2 shows samples from the CIFAR-10 dataset. The low resolution results in a small dataset with numerous samples, which enables fast training. These 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].

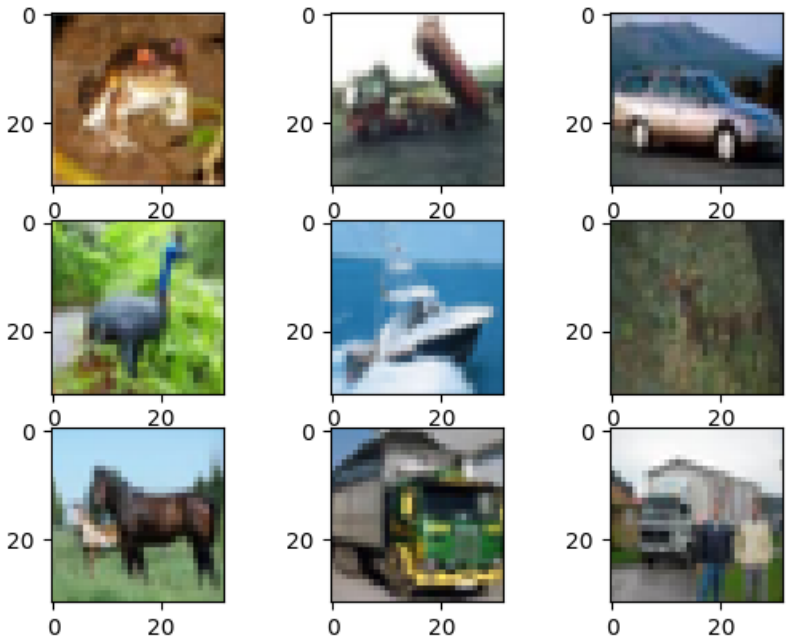


Figure 3.2: Random samples from the CIFAR-10 dataset.

3.3 Neural Network Selection

Model selection is a sensitive issue for mobile applications, since the model's performance must be balanced with memory consumption and processing times [19]. Even though commonly available mobile devices are capable of processing models which consume hundreds of MBs of memory [20], it is expected that multiple applications share devices' resources. As such, typical deep-learning-based mobile applications leverage models consisting of a few MBs [21]. 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 [2]. Naturally, real-world applications may additionally employ other techniques to further reduce the neural network size, such as weight pruning [22, 23] and weight quantization [24]. This work does not use these techniques since neural network compression is considered to be orthogonal to this work.

The dataset used in this project is commonly used as a benchmark for computer vision applications. As a consequence, there are a multitude of papers published which evaluate the performance of centralized neural networks on such

dataset [25].

The architecture used on the CIFAR-10 dataset was selected using the design principles of VGG models [26, 27]. The neural network has 550,570 parameters and the corresponding file in Hierarchical Data Format (HDF) has 4.3 MB. The neural network architecture used for the CIFAR-10 dataset can be seen in Code 3.1.

Code 3.1: Convolutional neural network architecture for the CIFAR-10 dataset.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
dropout (Dropout)            (None, 16, 16, 32)        0
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
max_pooling2d_1 (MaxPooling2D) (None, 8, 8, 64)        0
dropout_1 (Dropout)          (None, 8, 8, 64)          0
conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856
conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584
max_pooling2d_2 (MaxPooling2D) (None, 4, 4, 128)       0
dropout_2 (Dropout)          (None, 4, 4, 128)         0
flatten (Flatten)            (None, 2048)              0
dense (Dense)                (None, 128)               262272
dropout_3 (Dropout)          (None, 128)               0
dense_1 (Dense)              (None, 10)                1290
Total params: 550,570
```

3.4 Mobile Network Parameters

Table 3.1 presents the values used to simulate the federated learning setting using the CIFAR-10 dataset.

3.5 Software

The results presented in Chapter 4 were obtained using Python, the standard data exploration suite, *e.g.*, Pandas, Matplotlib, and Numpy, along with the

Table 3.1: Parameters that affect federated learning performance, symbols, meanings, and studied values.

Symbol	**Meaning**	**Values explored on Chapter 4**
N_c	Number of clients	[5, 10, 20]
E_l	Local training epochs	[10, 20]
B_c	Local batch sizes	[64, 256]
P_f	Client failure probability	[0%, 25%, 50%, 75%]
L_c	Client latency	Simulated values
R	Number of federated training rounds	[1..250]

following machine learning frameworks:

- Scikit-learn: an open source machine learning library commonly used for data pre-processing [28].
- TensorFlow: an open-source platform for machine learning and, moreover, for deep learning [29].
- Keras: an interface to provide easier access to some TensorFlow tools [30].
- Flower: a framework designed to simplify the deployment of federated learning scenarios [31].

3.6 Hardware

An Intel Core i5 − 10400 2,90 GHz computer with 6 processing cores and 32 GB RAM was used to perform the experiments.

Chapter 4

Results

4.1 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 [2].

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 Chapter 2. 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 usages do not hinder other applications. Although it is possible to achieve better results with deeper and more complex models, communication and processing costs could 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 negate the results discussed in Chapter 4.

The complete neural network architecture can be seen in Code 3.1. The centralized experiments used SGD (Stochastic Gradient Descent) as the optimizer, the learning rate was set to 0.0001 and a momentum of 0.9. Training was performed for 25 epochs with a batch size of 128 samples. Table 4.1 presents these hyperparameters.

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.

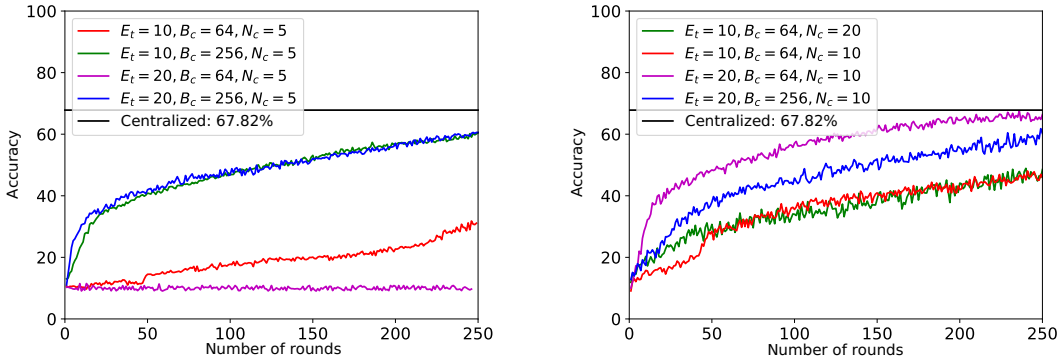
Table 4.1: Hyperparameters used in centralized training.

Hyperparameter	**Value**
Optimizer	SGD
Learning rate	0.0001
Momentum	0.9
Training epochs	25
Batch size	128 samples

4.2 Ideal Wireless Network Conditions on a Federated Learning Setting

The DNNs are distributed to the clients and, after each round, their performances are evaluated on 2 clients chosen at random. Figure 4.1 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, which accelerates global model convergence. Furthermore, local batch sizes, which assume a regularizing role during centralized machine learning [32], can act as a catalyst for model convergence. This is due to the fact that clients are limited to a given number of steps per epoch, so 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 4.1(a) and Figure 4.1(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 4.1(b) shows that the model performance may decrease if the number of available clients is insufficient, so the server must be able to dynamically decide if another training round must be performed. As Figure 4.1(a) shows, a model training on just a few clients can achieve satisfying performance, however, the training process becomes much more sensitive to the federated parameters, which is



(a) Curves obtained by simulating 5 clients.

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

Figure 4.1: Model convergence on the CIFAR-10 dataset with ideal network conditions.

shown by the poor performance when using 10 training epochs and 64 data samples per batch.

4.3 Wireless Mobile Network Parameters on a Federated Learning Setting

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.

4.3.1 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. Accordingly, 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%, therefore we chose to omit these results.

Figure 4.2 presents one of the models seen in Figure 4.1(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 4.2 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. This result must be further explored to better compose a possible conclusion.

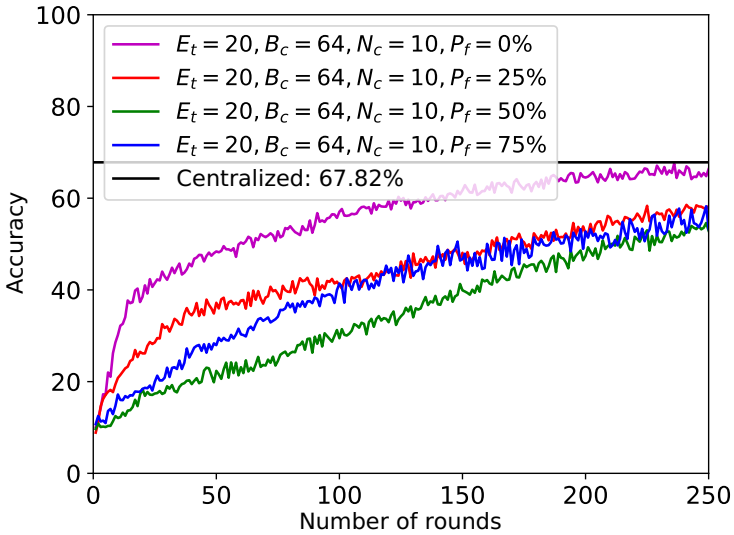


Figure 4.2: Impact of disconnection probability during federated training.

4.3.2 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 4.1 below:

$$\text{Total latency} = \sum_{i=1}^{R} \max_{\forall c \in C_i} (\min(L_c, T)), \tag{4.1}$$

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 4.3 presents a Monte Carlo simulation of Equation 4.1 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`[1] 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 4.3, 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

[1] Accessed on `https://iperf.fr/`.

training time and the computational cost on each client, since the fraction of training performed in slower clients is not used.

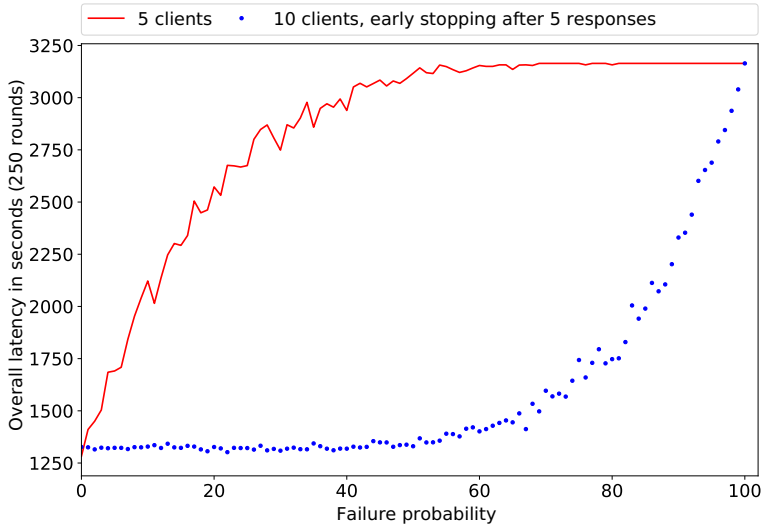


Figure 4.3: Latency comparison using Equation 4.1 with 5 clients (left) and 10 clients, but with early round stopping when 5 clients report their results.

4.3.3 Convergence Analysis

Figure 4.4 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 4.1(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 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.

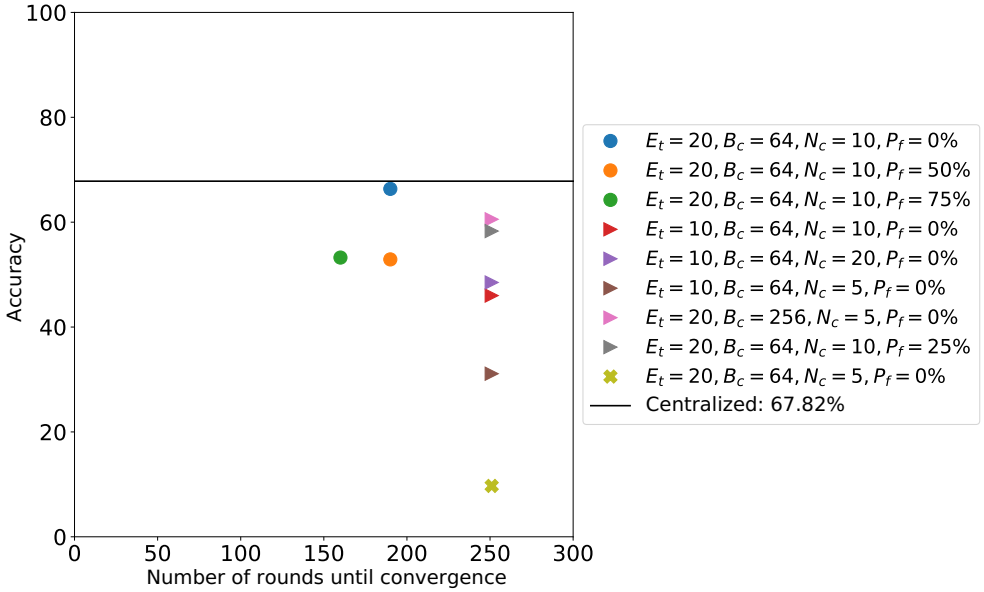


Figure 4.4: Number of rounds for each scenario to achieve convergence.

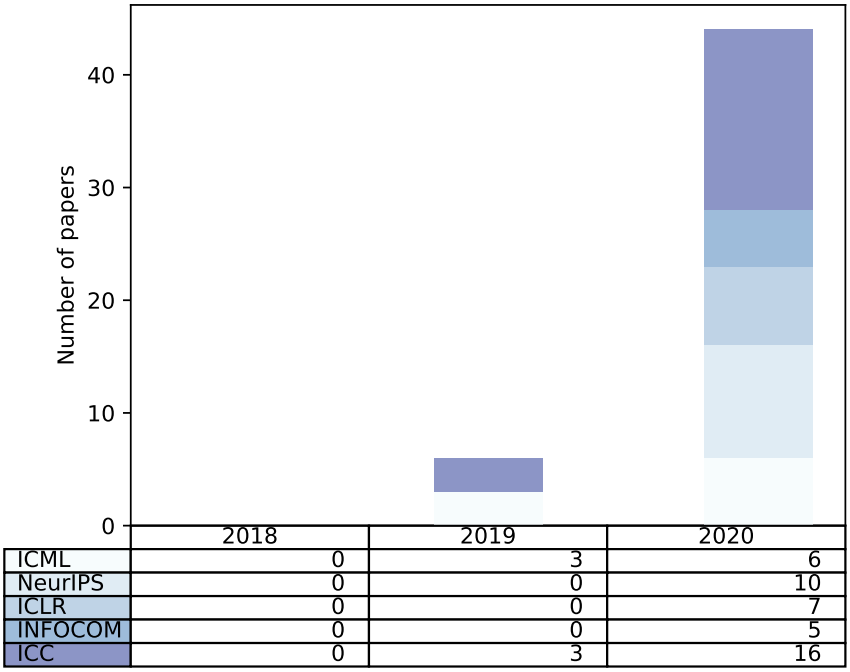
Chapter 5

Related Work

This section discusses works related to this one. We considered papers that evaluate their contributions for federated learning in wireless and mobile settings. We also review research that propose methods for client clustering and aggregation, which was selected as future steps for this work. Tables 5.1, 5.2, 5.3, 5.4, and 5.5 presents the works found by directly querying "federated" and "learning" on flagship conferences. These papers, with the addition of related references, were considered when elaborating this research.

Figure 5.1 presents a plot that illustrates the novel status and sudden interest of federated learning as a research topic. The data was obtained by querying the corresponding conferences' databases for the terms "federated" and "learning".

Ghosh et al. tackle client clustering on federated learning settings [33]. Their proposed framework is named Iterative Federated Clustering Algorithm (IFCA) and dynamically allocates federated learning clients into different clusters. The decision to group clients in real time is due to the fact that, since client data is not transferred through the network, cluster identities cannot be defined beforehand. The novelty of their paper consists of sharing representation layers, *i.e.*, shallower layers, between all users, and finally training the final layers of each individual cluster. Their proposal borrow this concept from multi-task learning. The authors perform their experiments using image recognition datasets and observe increases in performance up to 7% when compared to traditional federated learning, and up to 57% when compared to localized machine learning, where clients single-handedly train a machine learning model on their local datasets. Additional and similar efforts are



	2018	2019	2020
ICML	0	3	6
NeurIPS	0	0	10
ICLR	0	0	7
INFOCOM	0	0	5
ICC	0	3	16

Figure 5.1: Federated learning growing interest in flagship conferences.

being done in the field of federated meta-learning [34, 35, 36, 37], which consists of training an initial global model that must be tuned by individual users.

Nishio et al. consider how mobile edge computer scenarios impact federated learning performance [38]. The authors explore the heterogeneity of mobile clients, namely, clients in a cellular network, to dynamically select which clients are fit to 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 protocol bases itself on a resource request step that provides the information of each client. Information such as channel state and GPU (Graphical Processing Unit) availability are then used to form client groups, which then perform individual federated training rounds. The authors evaluate their proposal on standard image classification datasets and show that their proposal is capable of improving classification performance when compared to standard federated learning algorithms. However, their proposal leverages client metadata, such as hardware capabilities, which may not be available for some applications.

Table 5.1: Federated learning papers published on ICML. An "-" in one of the fields means that the authors did not specify or it was unclear.

Year	Paper	Datasets	Clients
2020	FedBoost: Communication-Efficient Algorithms for Federated Learning	Synthetic and Shakespeare	-
	SCAFFOLD: Stochastic Controlled Averaging for Federated Learning	Simulated and EMNIST	100
	Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization	a5a, mushrooms, a9a, and w6a (LIBSVM)	20
	From Local SGD to Local Fixed-Point Methods for Federated Learning	a94 and a4a (LIBSVM)	-
	FetchSGD: Communication-Efficient Federated Learning with Sketching	CIFAR-10/100, FEMNIST, and PersonaChat	[3, 10k, 17568, 50k]
	Federated Learning with Only Positive Labels	CIFAR-10/100, AmazonCat, WikiLSHTC, Amazon670K	4k
2019	Analyzing Federated Learning through an Adversarial Lens	Fashion MNIST and UCI Adult Census	[10, 100]
	Agnostic Federated Learning	Adult, Fashion MNIST, Cornell Movie dataset, and Penn Tree-Bank (PTB) dataset	-
	Bayesian Nonparametric Federated Learning of Neural Networks	MNIST and CIFAR-10	-
2018	-	-	

Yang et al. evaluate three scheduling policies for federated training on wireless networks and their impact on both accuracy and communication rounds required for convergence [39]. The authors also evaluate how some physical layer parameters, *e.g.*, Signal-to-Noise Ratio (SNR), negatively impact model convergence on federated settings. Despite implementing classical scheduling policies, the experiments consider complete client availability, where only the communication channel limits its performance, which may not represent some wireless scenarios, such as mobile networks.

Nguyen et al. develop a system capable of performing botnet detection on IoT (Internet of Things) networks by leveraging IoT gateways as federated learning clients [40]. The authors perform a centralized performance evaluation to configure the federated setting. The research performed by Nguyen et al. explain how a real application would not be able to use centralized learning as a first step to adjust the configuration of federated parameters. Nevertheless, our work uses this technique, since the goal of this work is to better understand how introducing the

Table 5.2: Federated learning papers published on NeurIPS. An "-" in one of the fields means that the authors did not specify or it was unclear.

Year	Paper	Datasets	Clients
2020	An Efficient Framework for Clustered Federated Learning	MNIST/CIFAR-10 (augmented), and FEMNIST	[50, 100, 150, 200, 400, 600, 800]
	Attack of the Tails: Yes, You Really Can Backdoor Federated Learning	CIFAR-10, EMNIST, ImageNet, Sentiment140, and Reddit	[10, 30, 100]
	Ensemble Distillation for Robust Model Fusion in Federated Learning	CIFAR-10/100, ImageNet, AG News, and SST2	[10, 20, 21, 150]
	Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge	CIFAR-10/100 and CINIC-10	16
	Inverting Gradients - How easy is it to break privacy in federated learning?	CIFAR-10 and ImageNet	-
	Lower Bounds and Optimal Algorithms for Personalized Federated Learning	LIBSVM datasets	-
	Personalized Federated Learning with Moreau Envelopes	MNIST and synthetic	20
	Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach	MNIST and CIFAR-10	50
	Robust Federated Learning: The Case of Affine Distribution Shifts	MNIST and CIFAR-10	10
	Throughput-Optimal Topology Design for Cross-Silo Federated Learning	Shakespeare, FEMNIST, Sentiment140, and iNaturalist	-
2019	-	-	-
2018	-	-	-

characteristics of mobile networks impact 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 [41]. Their proposal uses synchronized updates as a mechanism to improve federated performance, which may not be a viable approach if the clients are not directly controlled by a single base station, as is the case for large scale mobile applications.

Li et al. use the NVDIA Clara healthcare framework to segment brain tumor images [12]. The authors use a centralized baseline to compare the results for federated learning settings. Similarly, Roth et al. use the NVDIA Clara framework to implement a system capable of performing breast density classification, trying to detect and provide early treatment for breast cancer [13]. Both papers present

Table 5.3: Federated learning papers published on ICLR. An "-" in one of the fields means that the authors did not specify or it was unclear.

Year	Paper	Datasets	Clients
2020	DBA: Distributed Backdoor Attacks Against Federated Learning	MNIST, CIFAR-10, Tiny-imagenet and Lending Club Loan Data	-
	Differentially Private Meta-Learning	Shakespeare and proprietary collected from Wikipedia	-
	Fair Resource Allocation in Federating Learning	Synthetic, Vehicle Sensor, Sent140, Shakespeare and Omniglot	[100, 23, 1101, 31]
	Federated Adversarial Domain Adaptation	Digit-Five, Office-Caltech 10, DomainNet and Amazon-Review	-
	Federated Learning with Matched Averaging	MNIST, CIFAR-10 (augmented), and Shakespeare	[16, 66]
	Generative Models for Effective ML on Private, Decentralized Datasets	EMNIST	3400
	On the Convergence of FedAVG on Non-IID Data	Synthetic and MNIST	-
	Towards Blockchain-Based Reputation-Aware Federated Learning	-	-
2019	-	-	-
2018	-	-	-

Table 5.4: Federated learning papers published on INFOCOM. An "-" in one of the fields means that the authors did not specify or it was unclear.

Year	Paper	Datasets	Clients
2020	A Unified Federated Learning Framework for Wireless Communications: towards Privacy, Efficiency, and Security	MNIST	20
	Blockchain Technology and Neural Networks for the Internet of Medical Things	Tuberculosis Chest X-ray Image datasets	20
	Federated Routing Scheme for Large-scale Cross Domain Network	-	-
	Signal Recognition Based On Federated Learning	RML2016.10a	[5,10,20]
	Towards Blockchain-Based Reputation-Aware Federated Learning	-	-
2019	-	-	-
2018	-	-	-

their results when compared directly to a centralized learning approach, while using clients with high computational power, similar characteristics, and high availability. Alternatively, Asad et al. evaluate different federated learning variations, such as Communication-Mitigating Federated Learning (CMFL) and Federated Maximum and Mean Discrepancy (FedMMD), to understand their impacts on communication efficiency, although the number of clients participating of the federated training

Table 5.5: Federated learning papers published on ICC. An "-" in one of the fields means that the authors did not specify or it was unclear.

Year	Paper	Datasets	Clients
2020	Dynamic Sample Selection for Federated Learning with Heterogeneous Data in Fog Computing	MNIST	100
	Federated Learning for Energy-Efficient Task Computing in Wireless Networks	MNIST	[6, 7, 8, 9, 10, 11, 12]
	A Privacy-Preserving and Verifiable Federated Learning Scheme	MNIST	20
	Content Popularity Prediction in Fog Radio Access Networks: A Federated Learning Based Approach	MovieLens	943
	Federated Learning in the Sky: Joint Power Allocation and Scheduling with UAV Swarms	Proprietary	5
	GAN Enhanced Membership Inference: A Passive Local Attack in Federated Learning	MNIST	100
	Convergence Time Minimization of Federated Learning over Wireless Networks	MNIST	[3,6,9,12,15]
	Energy-Aware Analog Aggregation for Federated Learning with Redundant Data	MNIST	50
	Client-Edge-Cloud Hierarchical Federated Learning	MNIST and CIFAR-10	50
	Electrical Load Forecasting Using Edge Computing and Federated Learning	Pecan Street Inc's Dataport	200
	Privacy-aware and Resource-saving Collaborative Learning for Healthcare in Cloud Computing	CIFAR-10	-
	GGS: General Gradient Sparsification for Federated Learning in Edge Computing	MNIST, CIFAR-10, and ImageNet	[2,4,8]
	Device Scheduling with Fast Convergence for Wireless Federated Learning	MNIST	[4,8,12]
	Privacy-Preserving Personalized Federated Learning	HAR	-
	Hybrid-FL for Wireless Networks: Cooperative Learning Mechanism Using Non-IID Data	CIFAR-10 and Fashion MNIST	1000
	Achieving Privacy-preserving Federated Learning with Irrelevant Updates over E-Health Applications	MNIST	[100, 150, 200, 250, 300, 350, 400, 450]
2019	Towards Efficient and Privacy-preserving Federated Deep Learning	MNIST	[100, 200, 300, 400, 500]
	Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge	CIFAR-10 and Fashion MNIST	1000
	Federated Learning Based on Over-the-Air Computation	CIFAR-10	20
2018	-	-	-

rounds remains constant [42].

Even though our work focuses on mobile networks' applications, some federated learning settings are deployed on stable and highly available networks with the goal of preserving privacy without optimizing communication [13]. Kairouz et

al. highlight the availability of reproducible results as an open issue on federated learning research [9]. There are multiple studies that evaluate federated learning performance on devices optimized for DNNs [12, 13]. These works, however, do not represent the mobile setting, on which clients are not able to 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.

Chapter 6

Conclusion and Future Scope

6.1 Final Consideration

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 federated learning 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 increasing fault probability in clients can improve 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 datasets representative 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. Some frameworks provide datasets tailored to federated learning evaluation, such as LEAF [43], but the datasets are tightly coupled with the functionalities provided by the framework, hampering their usage with different tools.

6.2 Future Directions

As future work we intend to expand our simulation environment to cover other typical applications on mobile networks, such as traffic classification and Natural Language Processing (NLP) tasks. We also intend to explore datasets that could be better suited to federated learning applications, such as the FEMNIST and Shakespeare datasets.

We will shift the development tools to another framework, namely, TensorFlow Federated. This is due to current limitations of the Flower framework, such as transparent data selection for each client, which limits the type of experiments available to perform. 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 intend to research techniques to improve federated performance through client clustering and selection without using client metadata.

Bibliography

[1] BOCHIE, K., SAMMARCO, M., DETYNIECKI, M., *et al.*, "Análise do Aprendizado Federado em Redes Móveis", *XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, , Aug 2021.

[2] Bochie et al., "Aprendizado Profundo em Redes Desafiadoras: Conceitos e Aplicações". In: *Minicursos do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 12 2020.

[3] BOCHIE, K., GILBERT, M. S., GANTERT, L., *et al.*, "A survey on deep learning for challenged networks: Applications and trends", *Journal of Network and Computer Applications*, v. 194, pp. 103213, Nov 2021.

[4] LI, C., ZHONG, Q., XIE, D., *et al.*, "Skeleton-based Action Recognition with Convolutional Neural Networks", *CoRR*, v. abs/1704.07595, 2017.

[5] MEIDAN, Y., BOHADANA, M., MATHOV, Y., *et al.*, "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders", *IEEE Pervasive Computing*, v. 17, n. 3, pp. 12–22, 2018.

[6] WANG, Y., XU, W., "Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud", *Decision Support Systems*, v. 105, pp. 87–95, 2018.

[7] ZHOU, X., LIANG, W., WANG, K. I.-K., *et al.*, "Deep-Learning-Enhanced Human Activity Recognition for Internet of Healthcare Things", *IEEE Internet of Things Journal*, v. 7, n. 7, pp. 6429–6438, 2020.

[8] MCMAHAN, B., MOORE, E., RAMAGE, D., *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: Singh, A., Zhu,

J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, v. 54, *Proceedings of Machine Learning Research*, pp. 1273–1282, Fort Lauderdale, FL, USA, 20–22 Apr 2017.

[9] Kairouz et al., "Advances and Open Problems in Federated Learning". In: *Foundations and Trends in Machine Learning*, 2019.

[10] ZHANG, C., XIE, Y., BAI, H., *et al.*, "A survey on federated learning", *Knowledge-Based Systems*, v. 216, pp. 106775, 2021.

[11] ABDULRAHMAN, S., TOUT, H., OULD-SLIMANE, H., *et al.*, "A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond", *IEEE Internet of Things Journal*, v. 8, n. 7, pp. 5476–5497, 2021.

[12] Li et al., "Privacy-Preserving Federated Brain Tumour Segmentation". In: Suk, H.-I., Liu, M., Yan, P., *et al.* (eds.), *Machine Learning in Medical Imaging*, pp. 133–141, Cham, 2019.

[13] Roth et al., "Federated Learning for Breast Density Classification: A Real-World Implementation". In: Albarqouni, S., Bakas, S., Kamnitsas, K., *et al.* (eds.), *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, pp. 181–191, Cham, 2020.

[14] ZHU, W., ZHAO, C., LI, W., *et al.*, "LAMP: Large Deep Nets with Automated Model Parallelism for Image Segmentation". In: Martel, A. L., Abolmaesumi, P., Stoyanov, D., *et al.* (eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, pp. 374–384, Cham, 2020.

[15] FERRARINI, M. G., LAL, A., REBOLLO, R., *et al.*, "Genome-wide bioinformatic analyses predict key host and viral factors in SARS-CoV-2 pathogenesis", *Communications Biology*, v. 4, n. 1, pp. 590, May 2021.

[16] CORPORATION, N., "NVIDIA Clara: Intelligent Computing for Healthcare", `https://www.nvidia.com/en-us/clara/`, 2021.

[17] LI, M., SOLTANOLKOTABI, M., OYMAK, S., "Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks". In: Chiappa, S., Calandra, R. (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, v. 108, *Proceedings of Machine Learning Research*, pp. 4313–4324, Aug 2020.

[18] KRIZHEVSKY, A., "Learning Multiple Layers of Features from Tiny Images", *University of Toronto*, , 05 2012.

[19] ISUYAMA, V., ALBERTINI, B., "Comparison of Convolutional Neural Network Models for Mobile Devices". In: *Anais do XX Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, pp. 73–83, Porto Alegre, RS, Brasil, 2021.

[20] BENEDETTO, J. I., SANABRIA, P., NEYEM, A., *et al.*, "Deep Neural Networks on Mobile Healthcare Applications: Practical Recommendations", *Proceedings*, v. 2, n. 19, 2018.

[21] LI, D., WANG, X., KONG, D., "DeepRebirth: Accelerating Deep Neural Network Execution on Mobile Devices". In: McIlraith, S. A., Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 2322–2330, 2018.

[22] JIANG, C., LI, G., QIAN, C., *et al.*, "Efficient DNN Neuron Pruning by Minimizing Layer-wise Nonlinear Reconstruction Error". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 2298–2304, 7 2018.

[23] ZHANG, T., YE, S., ZHANG, K., *et al.*, "A systematic DNN weight pruning framework using alternating direction method of multipliers". In: Ferrari, V., Sminchisescu, C., Weiss, Y., *et al.* (eds.), *Computer Vision – ECCV 2018 - 15th European Conference, 2018, Proceedings*, Lecture Notes in Computer Science

(including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 191–207, 2018.

[24] GONG, C., CHEN, Y., LU, Y., *et al.*, "VecQ: Minimal Loss DNN Model Compression With Vectorized Weight Quantization", *IEEE Transactions on Computers*, v. 70, n. 05, pp. 696–710, may 2021.

[25] BALDOMINOS, A., SAEZ, Y., ISASI, P., "A Survey of Handwritten Character Recognition with MNIST and EMNIST", *Applied Sciences*, v. 9, n. 15, 2019.

[26] SIMONYAN, K., ZISSERMAN, A., "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: Bengio, Y., LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

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

[28] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., *et al.*, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, v. 12, pp. 2825–2830, 2011.

[29] ABADI, M., AGARWAL, A., BARHAM, P., *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems", 2015, Software available from tensorflow.org.

[30] CHOLLET, F., OTHERS, "Keras", `https://keras.io`, 2015.

[31] Beutel et al., "Flower: A Friendly Federated Learning Research Framework", *ArXiv*, v. abs/2007.14390, 2020.

[32] LECUN, Y., BENGIO, Y., HINTON, G., "Deep learning", *nature*, v. 521, n. 7553, pp. 436–444, 2015.

[33] GHOSH, A., CHUNG, J., YIN, D., *et al.*, "An Efficient Framework for Clustered Federated Learning". In: Larochelle, H., Ranzato, M., Hadsell, R., *et al.* (eds.), *Advances in Neural Information Processing Systems*, v. 33, pp. 19586–19597, 2020.

[34] CHEN, F., LUO, M., DONG, Z., *et al.*, "Federated Meta-Learning with Fast Convergence and Efficient Communication", *arXiv: Learning*, , 2018.

[35] YUE, S., REN, J., XIN, J., *et al.*, "Inexact-ADMM Based Federated Meta-Learning for Fast and Continual Edge Learning". In: *Proceedings of the Twenty-Second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, MobiHoc '21, p. 91–100, New York, NY, USA, 2021.

[36] FALLAH, A., MOKHTARI, A., OZDAGLAR, A. E., "Personalized Federated Learning: A Meta-Learning Approach", *ArXiv*, v. abs/2002.07948, 2020.

[37] JIANG, Y., KONEČNÝ, J., RUSH, K., *et al.*, "Improving Federated Learning Personalization via Model Agnostic Meta Learning", *CoRR*, v. abs/1909.12488, 2019.

[38] NISHIO, T., YONETANI, R., "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2019.

[39] Yang et al., "Scheduling Policies for Federated Learning in Wireless Networks", *IEEE Transactions on Communications*, v. 68, n. 1, pp. 317–333, Jan 2020.

[40] Nguyen et al., "DÏoT: A Federated Self-learning Anomaly Detection System for IoT". In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, July 2019.

[41] Tran et al., "Federated Learning over Wireless Networks: Optimization Model Design and Analysis". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1387–1395, 2019.

[42] Asad et al., "Evaluating the Communication Efficiency in Federated Learning Algorithms". In: *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 552–557, 2021.

[43] CALDAS, S., WU, P., LI, T., *et al.*, "LEAF: A Benchmark for Federated Settings", *Workshop on Federated Learning for Data Privacy and Confidentiality (NeurIPS 2019)*, , 2019.

Appendix A

Publications

This work was developed during a period of Scientific Initiation in Grupo de Teleinformática e Automação (GTA) on UFRJ. During the development of this work one paper was published on an international journal and four papers were published on Brazilian conferences, one of which was awarded best paper of the main track. They are listed below:

- Bochie, K., Gilbert, M. S., Gantert, L., Barbosa, M. S. M., Medeiros, D. S. V. e Campista, M. E. M. - "A Survey on Deep Learning for Challenged Networks: Applications and Trends", in Journal of Network and Computer Applications, vol. 194, pp. 103213, Elsevier, ISSN: 1084-8045, DOI 10.1016/j.jnca.2021.103213, 2021.

- Bochie, K., Sammarco, M., Detyniecki, M. e Campista, M. E. M. - "Análise do Aprendizado Federado em Redes Móveis", in XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2021), Uberlândia, MG, Brazil, August 2021. Best paper award on the main track.

- Bochie, K., Gilbert, M. S., Gantert, L., Barbosa, M. S. M., Medeiros, D. S. V. e Campista, M. E. M. - "Aprendizado Profundo em Redes Desafiadoras: Conceitos e Aplicações", in Minicursos do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2020), Rio de Janeiro, RJ,

Brazil, December 2020.

- Bochie, K., Gonzalez, E. R., Giserman, L. F., Campista, M. E. M. e Costa, L. H. M. K., "Detecção de Ataques a Redes IoT Usando Técnicas de Aprendizado de Máquina e Aprendizado Profundo", in XX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2020), Petrópolis, RJ, Brazil, October 2020.

- Bochie, K., Campista, M. E. M. - "LabSensing: Um Sistema de Sensoriamento para Laboratórios Científicos com Computação Inteligente nas Bordas", in II Workshop de Trabalhos de Iniciação Científica e de Graduação - WTG 2019, Gramado, RS, Brazil, May 2019.