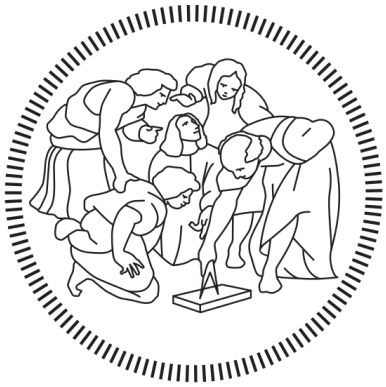
Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria
Master of Science in Telecommunication Engineering



Federated-Learning-Assisted Failure Management in Microwave Networks

Supervisor:
Prof. Massimo Tornatore
Co-Supevisors:
Prof. Francesco Musumeci
Dr. Omran Ayoub

Candidate:
Tara Tandel
Matricola: 927088

Academic Year 2020 - 2021

Acknowledgements

I would like to express my sincere gratitude to Prof. Massimo Tornatore to give me this incredible opportunity to expand my knowledge and trusting me to implement them. I would like to extend my sincere thanks to Prof. Francesco Musumeci for his assistance, invaluable advice, continuous support, and patience during every stage of the research project, and to Dr. Omran Ayoub that was a big source of positiveness and motivation. Without their tremendous understanding and encouragement in the past year, it would be impossible for me to complete my study. My appreciation also goes out to my family, my boyfriend Giovanni and my good friends for their encouragement and support all through my studies.

In the end, I would like to thank the Polimi University and all my professors and staff that made this possible.

Milan, August 25 T.T.

Abstract

Faults are unavoidable and cause network downtime and degradation of large and complex communication networks. The need for fault management capabilities for improving network reliability is critical to rectifying these faults. Current communication networks are moving towards the distributed environment enabling these networks to transport heterogeneous multimedia information across end-to-end connections. An advanced fault management system is thus required for such communication networks. Fault Management provides information on the status of the network by locating, detecting, and identifying, network problems thereby increasing network reliability. A large portion of next-generation (6G) services and applications, such as cloud computing, video streaming, and smart working platforms, have strict availability requirements and must be available at all times from any location on the network. Because of the increased use of these apps, internet service providers (ISPs) are being forced to develop new solutions to deliver Ultra-Reliable Low-Latency Communication (URLLC).

As 5G communication networks become more widely implemented worldwide, both business and academics have begun to go beyond 5G and towards 6G communications. It is widely assumed that 6G will be founded on pervasive Artificial Intelligence (AI) to provide data-driven Machine Learning (ML) solutions in heterogeneous and massive-scale networks. Traditional ML approaches, on the other hand, need centralized data gathering and processing by a single server, which is becoming a bottleneck to large-scale applications in daily life as privacy concerns grow.

In this context we present a failure management system based on Federated Learning that protects data privacy by performing local detection model training and inference. Not only is privacy protected in this method, but devices may also profit from their peers' expertise by transmitting just their changes to a distant server, which aggregates the latter and provides an enhanced detection model with participating devices. We do extensive tests on the SIAE dataset, an Italian company that provided us with genuine data to assess the efficacy of the suggested strategy. Experiment findings show that

the suggested Federated Learning detection model outperforms the isolated model in terms of accuracy.

Abstract (Italiano)

I guasti sono inevitabili e causano tempi di inattività della rete e il degrado di reti di comunicazione grandi e complesse. Diventa fondamentale quindi la capacità di gestione degli errori per migliorare l'affidabilità della rete e correggere questi errori. Le attuali reti di comunicazione si stanno spostando verso l'ambiente distribuito che consente a queste reti di trasportare informazioni multimediali eterogenee attraverso connessioni end-to-end. Per tali reti di comunicazione è quindi necessario un sistema avanzato di gestione dei guasti. La gestione dei guasti fornisce informazioni sullo stato della rete individuando, rilevando e identificando i problemi di rete, aumentando così l'affidabilità della rete. Una gran parte dei servizi e delle applicazioni di nuova generazione (6G), come il cloud computing, lo streaming video e le piattaforme di smart working, hanno requisiti di disponibilità rigorosi e devono essere sempre disponibili da qualsiasi punto della rete. A causa del maggiore utilizzo di queste applicazioni, i provider di servizi Internet (ISP) sono costretti a sviluppare nuove soluzioni per fornire una comunicazione a bassa latenza ultra affidabile (Ultra-Reliable Low-Latency Communication - URLLC).

Man mano che le reti di comunicazione 5G si sviluppano sempre di più in tutto il mondo, sia le imprese che gli istituti accademici hanno iniziato ad andare oltre il 5G e verso le comunicazioni 6G. È ampiamente ipotizzato che il 6G sarà fondato sull'intelligenza artificiale pervasiva (AI) per fornire soluzioni di Machine Learning (ML) basate sui dati in reti eterogenee e su larga scala. Gli approcci ML tradizionali, d'altro canto, richiedono la raccolta e l'elaborazione centralizzata dei dati da parte di un singolo server, che sta diventando un collo di bottiglia per le applicazioni su larga scala nella vita quotidiana man mano che crescono i problemi di privacy.

In questo contesto presentiamo un sistema di gestione dei guasti basato su Federated Learning che protegge la privacy dei dati eseguendo l'addestramento e l'inferenza del modello di rilevamento locale. Non solo la privacy è protetta con questo metodo, ma i dispositivi possono anche trarre vantaggio dall'esperienza di altri dispositivi trasmettendo solo le loro modifiche ad un

server remoto che aggrega il dispositivo e fornisce un modello di rilevamento avanzato con i dispositivi partecipanti. Effettuiamo test approfonditi sul dataset SIAE, azienda italiana che ci ha fornito dati veritieri per valutare l'efficacia della strategia suggerita. I risultati dell'esperimento mostrano che il modello di rilevamento del Federated Learning suggerito supera il modello distribuito in termini di accuratezza.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Microwave networks, which are made up of radio links, use a communication technology that travels in free space. Because the communication in a microwave network is not insulated, as it is with wired technology, the power measurement of the link might fluctuate over time owing to physical limitations and changes in the environment. Because of the numerous different causes of failure, their nature might be either temporary or permanent. Deep fading events can cause both temporary and permanent failures; in these cases, the received power measure suffers severe attenuation due to some impairments (i.e., atmospheric events, physical obstacles); the duration of these deep fading events can provide better intuition on its root-cause and aid in the identification of the proper countermeasures. Failures in the network can arise as a result of flaws in the design phase, which have a lasting influence on the link performances and power measurements; they do not allow the connection to operate correctly, resulting in a large number of failure events and an unavailability time. These are the examples of the failure root causes that can happen in a network.

In the new era, we are moving forward to be a virtual society. We can see that the classes, meetings, and significant jobs can be done remotely using the internet; besides, services and applications, such as cloud computing, video streaming, and smart working platform, have strong availability constraints and must always be reachable. The increase of usage of these applications forces internet service providers (ISPs) to research new solutions in order to provide Ultra-Reliable Low-Latency Communication (URLLC). However, the implementation of these solutions must be inexpensive, in order to provide the required high-quality service connection at low costs. The failure

identification is one solution to this problem

Nowadays, the failure identification procedure is carried out by human experts who analyze graphs of radio power measures related to the failure event, correlating them with the alarms, and then, based on experience, possible root causes of the failure can be identified, and proper countermeasures can be implemented to restore the service.

Because sometimes the root cause identification is required, and this method may take a long time, simply detecting the failure is insufficient to start the procedure to avoid the loss of communication. As a result, a solid mechanism for failure detection (recognizing anomalies caused by failure occurrences), localization (identifying where the failure occurred in the network), and identification (understanding the actual cause of the failure) is critical, as it may be used by operators to perform traffic re-routing and rapid failure recovery.

The number of network links to be examined can be enormous, increasing the time required to restore service. URLLC imposes a strict time constraint for restoring service after a failure; this constraint can be met by using data analysis techniques capable of working with a massive amount of data in a very short period of time. Machine learning techniques are viewed as a strong candidate to address this issue, as machine learning enables quick decision-making by effectively leveraging the plethora of data that can be retrieved via network monitors.

Edge devices, such as network equipment, gather and store data. Because of their specializations, organizations such as ISPs frequently have limited perspectives on network data. However, privacy and security concerns make it more difficult to integrate data from multiple companies in a straightforward manner. In this setting, federated machine learning (or federated learning (FL)) appears as a functional solution that may assist in the development of high-performance models shared across many parties while still adhering to user privacy and data confidentiality standards.

Aside from privacy and security concerns, another compelling reason for FL is to maximize computational capacity at a cloud system's edge devices, where devices only need to train a limited amount of data; this purpose is appropriate in cases where our computational power is insufficient for large-scale machine learning. Moreover, in centralized machine learning, where we constantly want the data from the devices like network edges, the communication is most effective when only calculated results, rather than raw data,

are transferred between devices and servers. Routers, for example, may handle the majority of computation locally and communicate the needed findings with the cloud at regular intervals. Access points can complete the majority of the calculation for the information they need to acquire and communicate with central servers using only a few communication channels.

An analogy may be used to describe FL in our work. In other words, an ML model is a final model that wants to recognize the failure in the network, and the data is stored in each network edge device. Traditionally, the ML are raised by purchasing data and transporting it to where the a central machine is situated. However, due to privacy concerns and laws, we are unable to physically move the data.

In our instance, the data can no longer migrate outside of its immediate surroundings. FL, on the other hand, utilizes a dual technique. We may let the data be trained over several network devices, similar to how our ML model is created in a distributed way, with no data going outside of its local area. In the end, the ML model grows from everyone's data.

Horizontal and vertical FL are the two forms of FL. The Google GBoard system employs horizontal FL and demonstrates B2C (business-to-consumer) applications. It may also be used to enable edge computing, in which devices at the periphery of a cloud system do many of the computational activities, reducing the need to interact with the central servers through raw data. Horizontal FL, as suggested and expanded by WeBank, is a B2B (business-to-business) approach in which several businesses form an alliance to construct and use a shared machine learning model.The model is created with the goal of guaranteeing that no local data leaves any locations and that the model's performance meets business requirements. In our work we cover only the B2B model.

We examine the failure management problem in microwave networks in our study using FL, and we primarily focus on the solution of failure identification via FL. Our study is part of a research initiative funded and shared by SIAE Microelettronica, an Italian telecommunications company specialized in microwave networks,equipment manufacturing.
Microwave networks, which are made up of radio connections, employ a communication technique that travels through free space. Because the communication in a microwave network is not isolated, as it is with wired technology, the power measurement of the link might fluctuate over time owing to physical limitations and changes in the environment. Because of the numerous

different causes of failure, their nature might be either temporary or permanent. Deep fading events can cause both temporary and permanent failures; in these cases, the received power measure suffers severe attenuation due to some impairments (i.e., atmospheric events, physical obstacles); the duration of these deep fading events can provide better intuition on its root-cause and aid in the identification of the proper countermeasures.

Failures in the network can arise as a result of design flaws, with long-term consequences for link performance and power measurements. They do not allow the link to function correctly, resulting in a large number of failure occurrences and unavailability period. This sort of failure might be attributed to incorrect radio connection configuration or interference from other lines in the same region. Other failures can be caused by device aging or hardware breakdown; in these situations, the connection may perform poorly or cease to function, resulting in a permanent failure that must be resolved by direct intervention on the radio link devices.

Techniques from the machine learning discipline are regarded as a strong candidate to address failure identification issue, as machine learning enables fast decision-making by effectively leveraging the plethora of data retrieved via network monitors. However, the final models are not as robust as needed and can't provide enough accuracy with limited amount of data, so there is still space for more sophisticated approaches. As the data are scattered through different operators, and new regulations for privacy limits the movements of the actual data, to extend the machine learning methods to solve the failure identification problem, we also consider FL to tackle the problem of data distribution across multiple operators, resulting in data on isolated islands.

As a result, the first primary goal of this thesis is to use FL a reliable failure classifier that takes as input a new network measure associated with a failure event, and to provide as output the root cause that provides the specific event. One use of the suggested method is when the operators are unwilling to disclose their data. For example, we may have one tiny operator that does not have all failure causes in its dataset can work with other operators to make more reliable predictions. Alternatively, large operators can work together to construct a more solid classifier without sharing data and simply communicating the specification of their locally trained model. The network operator may utilize this solution to reduce analysis time by giving

the failure reason and a probable restoration process to execute rapid failure repair and build a more accurate failure-cause identification model .
In this thesis, we propose a method for training a machine learning model collaboratively without sharing the data, with the goal of benefiting from the data of other operators in order to improve classification performance compared to the case where only one operator trains its data alone without collaborating with others.

1.1 Thesis Contribution

On this thesis we concentrate on failure-cause identification in microwave networks. Specifically, the main contributions of this work can be listed as follows:

- We model the failure identification problem in radio link as a machine learning classification problem.
- We use an FL algorithm to train a model without sharing data between different parties. Also, we propose and define a condition for the convergence of this algorithm.
- We investigate the possible divisions of the clients concerning the repetition of each failure root cause.
- We compare the FL algorithm with centralized learning (the case when all the data is gathered together) and isolated learning (the case when an operator train only with its own data)

1.2 Thesis Outline

The remainder of the thesis is organized as follows:

In Chapter 2, an overview of previous work is presented. First we present state of the art work on failure management in different types of communication networks. Then related work is presented, in which FL techniques are implemented in networking use cases.

In Chapter 3, the background knowledge of the thesis is presented. The first part is a methodological background on machine learning. Then, the algorithm used in the thesis development are described. The second part

discusses a background knowledge on FL. The third part presents a technological background related to the microwave networks. An overview of the hardware and software components is provided; also the channel model and the different failure classes are discussed.

The problem addressed in the thesis is formally described in Chapter 4, where we also provide a description of the available data set.

In Chapter 5, the approaches to solve the problem are presented. Initially we describe data preprocessing. Then, the FL approach used in the context of failure-cause identification is presented.

In Chapter 6, we evaluate the performance of the different approaches proposed in chapter 5; we compare the different techniques and we discuss the results.

In chapter 7 we conclude the thesis and identify possible extension of this work.

Chapter 2

Related work

This chapter summarizes the recent developments and the ongoing research related to failure management in communication networks. We emphasize the most relevant existing works and discuss other applications where FL is implemented in the networking domain.

2.1 Failure Management

Network domains have become more and more advanced in terms of their size, complexity, and level of heterogeneity. Comprehensive fault management is one of the most significant challenge in network management. A variety of techniques for failure management were developed in different networking contexts.

Services involved in an service-oriented architecture(SOA) often do not operate under a single processing environment and must communicate using different protocols over a network. Under such conditions, designing a fault management system that is both efficient and extensible is challenging. In [1] the author performed an analysis and also proposed a self-healing algorithm over such networks. In [2], the authors provide a survey of data mining and machine learning-based techniques for network fault management, including a description of their characteristics, similarities, differences, and shortcomings. In [3] the problem of the failure detection was discussed. The authors propose a technology based on sequence numbers for network elements connected via two or more routes to solve this problem. The authors of [4] present an advanced method to solve the multiple failure localization prob-

lems, using the network topology information and services information; the presented approach is based on a Hopfield neural network (HNN), which receives as input a bipartite graph with the candidate failure link and the alarm set. The HNN is used for optimizing the uncertainty between failures and alarms to evaluate the exact position of the failures. In [5], the authors use the alarms in the microwave network to solve the failure identification problem. The troubleshooting of the failure is solved with a neural network that inputs the alarms produced by each base station and provides the cause of the alarm burst.

In [6], the authors provide a framework divided into two parts to solving failure detection and identification problems in an optical link. Their framework takes the Bit Error Rate (BER) measures from the network as input, and the authors compare various machine learning algorithms to identify the best solution for both problems. In [7], failure-cause identification is addressed considering different machine learning algorithms, namely Support Vector Machine(SVM), Random Forest(RF), and Artificial Neural Network(ANN). Also, the authors address the automation of failure identification in microwave networks, and they provide supervised and semi-supervised techniques. The authors identify six categories of failure causes in microwave networks and compare various ML algorithms. They also investigate an automated labeling procedure, based on autoencoders-like Artificial Neural Networks, to combine the knowledge of the few manually-labeled data with extensive unlabeled data.

Failure management is also applied in Wi-Fi networks to simplify the troubleshooting to not experts users. For example in [8] the authors evaluate different machine learning algorithms to identify the most common diseases that can affect the network, providing as input some statistical measures on the accessibility of the wireless channel.

The authors of the survey [9] demonstrated, there are many areas in the optical networks in which the use of machine learning is investigated. One of these areas is the quality of transmission (QOT) estimation, in which the goal is to compute the transmission quality metrics starting from the physical measurements evaluated at the receiver. The QOT estimation can be applied in two scenarios: the first is related to the prediction for unestablished lightpath metrics, and the second is related to monitoring the already present light paths.

2.2 Federated Learning in Networks

AI techniques are widely used in the networking field. However, FL has been barely used in the context of network failure management. In this chapter, we present some examples of FL applications in networking areas different from the failure management.

In [10] A specially designed federated learning method is proposed for machinery fault diagnosis problems. Also, a self-supervised learning algorithm is proposed for better explorations of time-series machinery data. Moreover, a dynamic validation scheme is proposed to adaptively implement model averaging operation. The challenging scenarios with non-independent and identically distributed user data are addressed. The proposed data privacy-preserving learning scheme is validated through experiments on two rotating machinery datasets. As presented by the authors of the article [11], FL can be used for IoT systems; they mentioned that smart cities, smart homes, and smart medical systems had challenged the functionality and connectivity of the large-scale Internet of Things (IoT) devices. Thus, with the idea of offloading intensive computing tasks to edge nodes (ENs), edge computing emerged to supplement these limited devices. They aimed to make DRL-based decisions feasible and further reducing the transmission costs between the IoT devices, and edge nodes, FL is used to train DRL agents in a distributed fashion. The experimental results demonstrate the effectiveness of the decision scheme and FL in the dynamic IoT system.

In [12] they demonstrate the development of a QoT classifier over an autonomous machine-learning pipeline, the trading of the classifier over a federated marketplace, and eventually its deployment in the customer's network as a cloud-native micro-service. In [13] they introduced a framework, FedProx, to tackle heterogeneity in federated networks. FedProx can be viewed as a generalization and re-parametrization of FedAvg, the current state-of-the-art method for FL. While this re-parameterization makes only minor modifications to the method itself, these modifications have important ramifications in theory and practice. Practically, they demonstrate that FedProx allows for more robust convergence than FedAvg across a suite of realistic federated datasets. In particular, FedProx demonstrates significantly more stable and accurate convergence behavior in highly heterogeneous settings than FedAvg–improving absolute test accuracy by 22% on average. It is proposed in [14] an autoencoder-based anomaly-detection for optical failure

detection using FL with unbalanced data (<3%), this means that in each dataset the number of points differ by 3 percent, which identifies implicit failure and achieves detection accuracy of 96.8% and an F1 value of 0.9224.

Based on a DCN with OCS, [15] proposes a pattern-aware scheduling and fas convergence strategy for the distributed machine learning jobs. Experimental results show significant accelerations for completion time and convergence of the jobs. In [16] two-step aggregation is introduced to facilitate scalable FL(SFL) over passive optical networks(PONs). Results reveal that the SFL keeps the required PON upstream bandwidth constant regardless of the number of involved clients while bringing 10% learning accuracy improvement.

Chapter 3

Background

3.1 Machine Learning Methodologies

3.1.1 Machine Learning Definition

Machine learning is part of the computer science field, defined 60 years ago, in which the algorithms can learn how to perform a task, receiving as input the data and the desired output, and not specific instruction. There are different fields in machine learning. We can divide the machine learning algorithms into the following categories:

- *Supervised learning :* These algorithms are based on labeled data, i.e., data where each set of inputs (the features) is associated with a specific output value, either numerical or categorical; the goal of these algorithms is to find a function able to map the new observations of input data into an output value, learning from the examples observed previously, during training phase of the algorithm.

- *Unsupervised learning :* These algorithms work with unlabelled data, after the execution of the algorithms, the system will be able to describe some hidden structures of the unlabelled data exploring them and drawing inferences from the data-set.

- *Semi-Supervised learning :* These algorithms are the connecting link between supervised and unsupervised learning, where there is a vast amount of data, but only a tiny part of them is labeled; the algorithms can improve their prediction accuracy, w.r.t. a pure supervised

approach, exploiting the presence of the unlabelled data to train the models.

- *Reinforcement learning :* This method can learn how to behave in a specific context where the system must maximize its performance. In this case, the learning procedure is based on a trial and error search, where the system interacts with the environment by producing action and discovering errors or rewards. The data to train the machine need to specify the set of actions, the set of states reachable in the environment, and the reward function that from each couple (state, action) provide a reward.

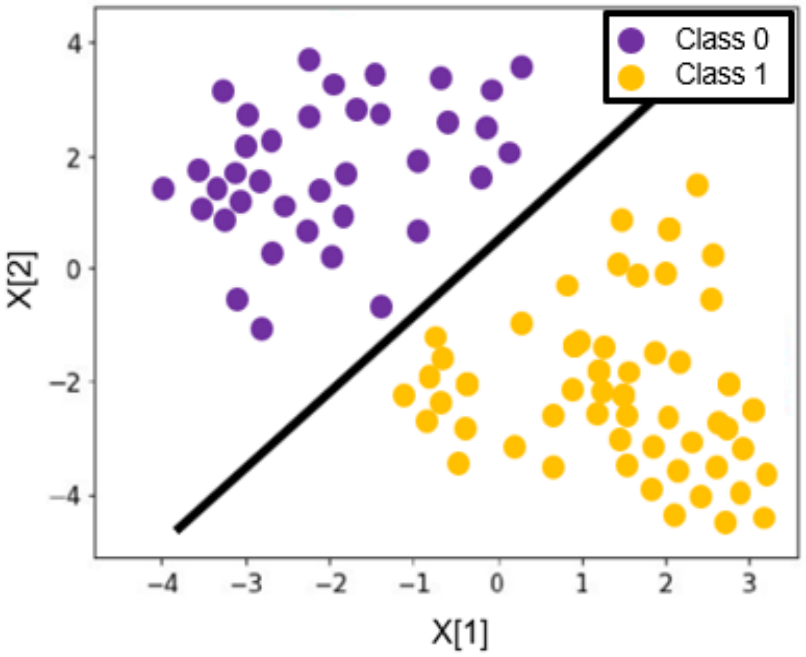


Figure 3.1: Example of binary classification where data points are constituted by two features X[1] and X[2]

3.1.2 Classification

The main focus of this work is to identify the root cause of failures in microwave links using local data.

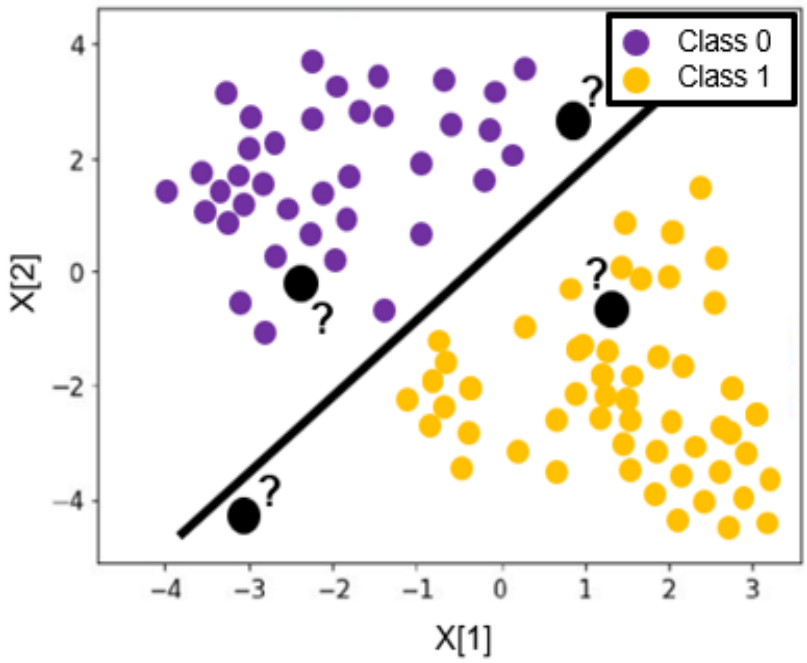


Figure 3.2: How we can use the model to perform classification

We model this task as a machine learning classification problem, where our target labels are elements of a discrete non-ordered set representing different root causes of failures. The goal is to map a set of input variables, in one of the output variables, i.e., the classes.

Figure 3.1 is an example of a classification problem, in which we have two-dimensional points, described by the features X[1] and X[2], divided into two classes, class 0 and class 1. The machine learning algorithm output a model that discriminates the two classes, described by the solid line. According to the model, a new unseen point will be classified in Fig.3.2 the points below the line will be classified as class 1 and above class 0. Usually, the data are represented in a tabular format; in these cases, the row of our tables are called data points or data instances. Instead, the columns are called attributes or features, and the target variable that must be predicted is labeled or class. In the end, the classification goal is to predict a label for each data point based on its features.

Some high-level steps are usually taken to solve a machine learning classification problem, as in the following [17]:

1. *Gathering Data:* As we need data to train our model, the first step is

to collect the data. As the data quality directly influences the classifier that will be developed, the gathered data must describe all the plethora of data behavior; they have labeled accurately and do not contain missing values. Some attributes of the gathered data may be missing using accurate data, and this problem is faced in step 2.

2. *Data Preparation:* After the collection of the data that are not suitable for machine learning algorithms, we have to pre-process it according to the problem we want to solve. In this step, several methods for data manipulation are available. For example, visualizing the data can be an excellent approach to finding a relationship between the features, which we can use to create new features that simplify the problem. This step is the one in which the missing data must be processed, and the data set will be split into train and test set so that our final model can be evaluated on the data contained in the test set and never used for training. Usually, the train-test split is about 80/20 or 70/30, but this also depends on the quantity of data available.

3. *Choosing a model:* A vast amount of algorithms can be used for classification, so we need to choose the most suitable one for our problem, according to the target variable to predict and the type of data we have as input. Each algorithm has a set of tunable parameters, called hyperparameters. There are methods used to select the hyperparameters to implement the most suitable model to solve our problem. The methods to perform the hyperparameters selection are presented in the following. In the following, some classification algorithms will be explained in detail.

4. *Training:* In this step, our model is trained to learn from the available data. The models are generally constituted by some variables called weights(W) and biases(b). These are the components that we modify during the training steps in order to improve the model accuracy. The training phase starts with the initialization of W and b. Then the model is used to predict the output of our training data, the prediction is compared with the actual output, and this information is used to improve W and b; the procedure is repeated using as start condition for W and b the output of the previous iteration.

5. *Evaluation:* After training the model, to check its performance, the

test data-set is used. This phase allows us to test our model against labeled data that has never been used for training. The results should be representative of how the trained classifier might perform in the real world.

Evaluation Metrics

The performance of a classifier can be evaluated using different metrics calculated over the test data. To describe the most commonly used performance metrics used to evaluate a classification algorithm, we refer to a binary classification problem[1], where we assume labels 0 and 1 to be negative and positive class respectively.

Table 3.1: Confusion matrix for a binary classification problem [18]

	Actual Class: 1	Actual Class: 0
Predicted Class: 1	tp	fp
Predicted Class: 0	fn	tn

To calculate most of the metrics, we need to build the confusion matrix as in Table 3.1; to create it, we need to predict the classes for the points in the test set, and to fill the cells of the matrix, we need to take in account both the classes provided in output by the classifier and the ground truth classes to the test points. the elements in the cells are integer numbers and are calculated as follows in Table 3.1: *tp* stands for True positive, it is the number of points for which the classifier output the class 1 and the actual class is one too; in the same way *tn*, that stands for True Negative, is the number of points classified as 0 which are belonging to the class 0, so this two number refer to the correct classification, as *fp* and *fn*, that stands for False Positive and False Negative respectively, refer to the wrong classifications, indeed *fp* is the number of points belonging to the class 0 but classified as 1, and at the same time *fn* is the number of points belonging to the class 1 and classified as class 0.

When the confusion matrix is built, we can calculate the different performance metrics, as they are presented and compared in [19]; the most commonly used performance measure is the *Accuracy (Acc)*, that is defined

[1]The concept can be extended also to multiclass classification problems.

as the ratio between the correctly classified samples to the total number of samples, i.e.:

$$Acc = \frac{tp + tn}{tp + tn + fp + fn}$$

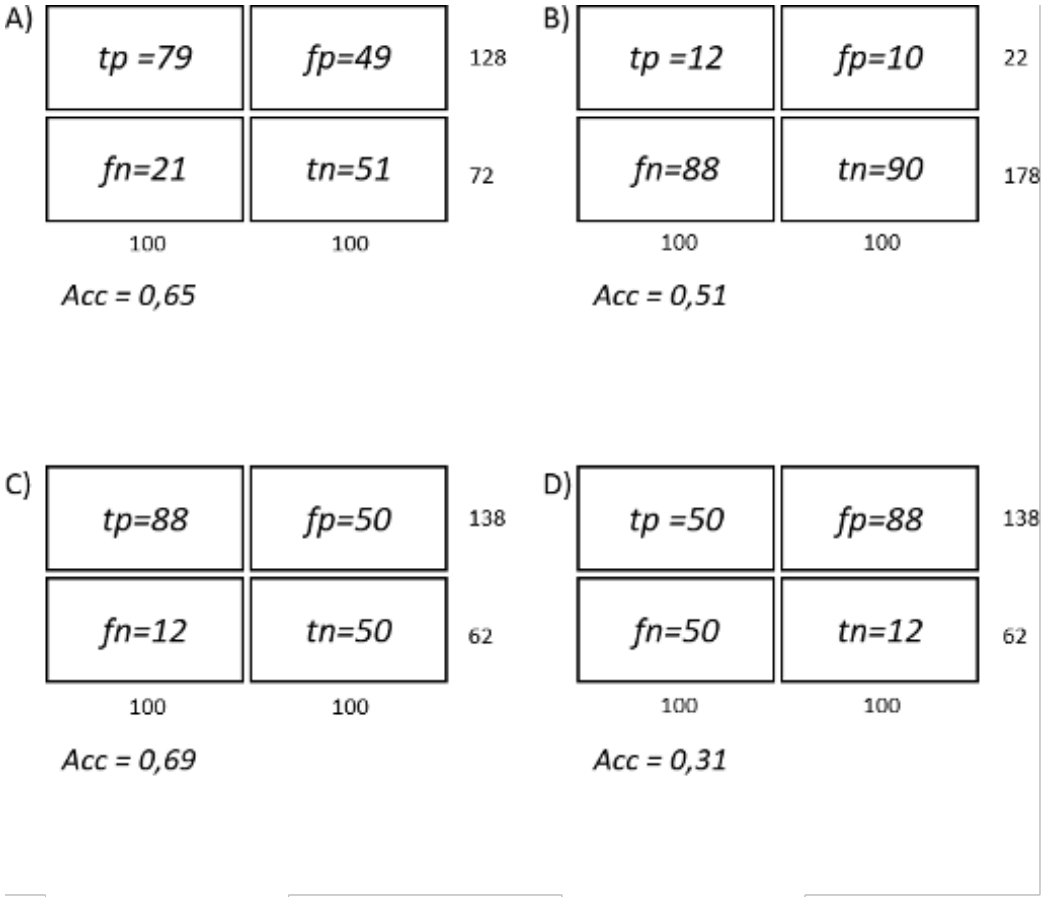


Figure 3.3: Examples of confusion matrices for different models.

Another metric is the *Precision (Prec)*, which represents the portion of positive samples that were correctly classified out of the total of the positive predicted samples, *Recall (Rec)* represents the positive correctly classified samples to the total number of positive samples *Rec* can be considered as an accuracy only related to the actual positive points; another one instead is the F_1*-score* that represents the harmonic mean of precision and recall.
Accuracy suffers from imbalanced classes since predicting the most representative class will provide a high value even if the classification is wrong. To deal with that, Precision and Recall are calculated jointly. Analyzing them

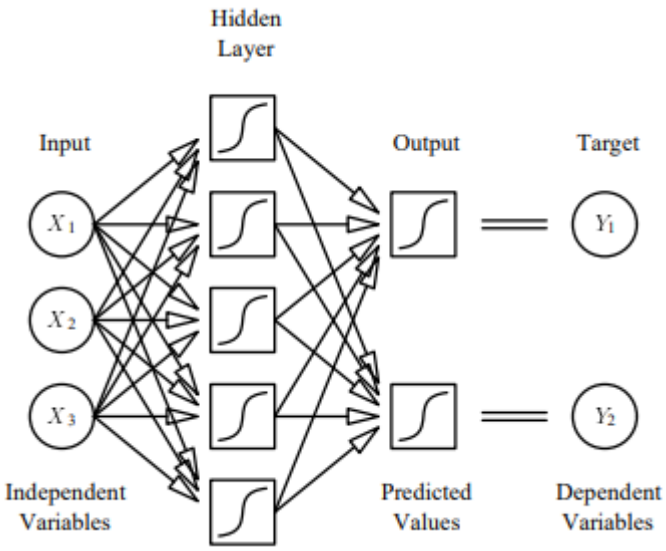


Figure 3.4: Multilayer perceptron structure [20].

alone can provide misleading results because classifying correctly only one instance yields 100% precision but a very low recall, and respectively classifying all instances as positive yields 100% recall, but very low precision. In order to avoid these situations, the F_1-the score was defined. In Fig. 3.3 examples of confusion matrix are presented with the accuracy calculated for different models.

3.1.3 Classification algorithms

This section will describe the supervised learning algorithms used in our analysis from a mathematical point of view. In case the output of the prediction is a continuous value, we are facing a regression problem. Conversely, when a discrete value is predicted, we are solving a classification problem. The challenges to get good performance from these algorithms are the number of samples, which in some cases must be huge, and the number of samples per class, which generally must be more or less balanced. In our case, we are facing a classification problem because we are interested in predicting which problem affects our radio link.

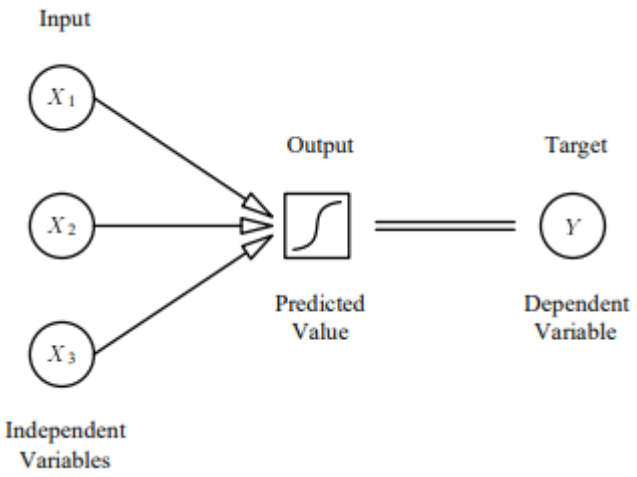


Figure 3.5: Perceptron illustration [20].

Artificial Neural Network

Artificial Neural Network (ANN) is an algorithm inspired by the biological structure of the human brain, where different neurons work in parallel to solve the same task. ANN is the basic structure for deep learning; indeed, in the last few years, with the increase of computational capacity, many variants of the simple multilayer perceptron were created. Most of these variants are used to analyze images or sequences of data, and they are also solving different machine learning problems; in this section, we focus on the feedforward neural networks used for the classification task.

So an ANN has different neurons interconnected with each other and organized in layers, in Fig.3.4 circles and boxes represent the neurons, and the connections between neurons are represented by the arrows that are shown how the information flow inside the network.

To better understand the functioning of the ANN, we start from the fundamental element that is the neuron. The neurons are also called perceptrons in machine learning literature [21]; as explained in [20] and shown in Fig.3.5, the perceptron computes a linear combination of the inputs, also adding a biasing term, this combination is called *net input*, then a possibly nonlinear activation function is applied to the net input and produce an output; typically the activation functions are bounded between -1 and 1, or between 0 and 1, in these cases, they are called *squashing* functions.

We can define the output(O) of the perceptron through the formula:

$$O = f(net) = f(\overline{W} \cdot \overline{X}) = f\left(\sum_{j=1}^{n+1} W_j X j\right) = f\left(\sum_{j=1}^{n} W_j X j + \theta\right)$$

where X is the set of input, W is set of weight associated to the arrows in Fig.3.5 and f is the activation function.

The perceptron with the proper activation function can classify the points into two different classes. In our case, a more complex structure is needed to identify more classes, so we use a network of perceptron as in Fig.3.4 this means that we will have more hidden layers and also more neurons for each layer to calculate the final output we need to use the recursive formulation of the network that is:

$$\begin{aligned} &a^0 = X = \textit{input data} \\ &z^i = (W^{i-1})^T a^{i-1} + b^{i-1} \qquad i \in [1, .., L] \\ &a^i = f(z^i) \\ &a^L = Y = \textit{target variables} \end{aligned}$$

where a^i is a vector that identify the value of the neurons in the specific layer i, the maximum number of layer is L, b^i is a vector too and identify the bias value of the layer i, W^i represents a matrix where the elements are the weights of the connection between the neurons of the two different layers i and $i+1$, and f is the activation function used in our neurons.

The neural network, like all the others models, need to be trained to predict in the most accurate way the target that we want, the training process at a high level is similar to the one used by much other data science model, define a cost function and use gradient descent optimization to minimize it.

The gradient is calculated over the cost function that defines the quantity of the error provided by our model. This error can be calculated only at the end of the network, but to modify all the weights and the bias term of the network, we need to propagate back the predicted error. This technique is called *Backpropagation*, so the training of a neural network can be divided into two phases:

- *Forward propagation:* Using the recursive definition of the neural network presented above, we give data points in the input, and the neural

network provides us a prediction.

- *Backpropagation:* In this phase, we initially calculate the error at the end of the network, that error is reported at each layer, in this way, we know which is the impact of that weights in the prediction error and we can correct them with the right quantity. As shown in [22] we can calculate the error at each layer and consequently the gradient to improve our weights in this way:

Output error:
$$\delta^L = \nabla_a C \odot f'\left(z^L\right)$$
Backpropagate the error:
$$\delta^l = \left(\left(W^{l+1}\right)^T \delta^{l+1}\right) \odot f'\left(z^l\right)$$
Gradient calculation:
$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \text{ and } \frac{\partial C}{\partial b^l_j} = \delta^l_j$$

To get the best performances from the neural network, we can tune some hyperparameters as for all the other algorithms. In our case, we decide to tune our network's structure, which means to decide the number of hidden layers and the number of neurons per layer. Another tunable hyperparameter is the activation function used in the neurons. According to the problems, this can be a crucial choice to improve the performances.

3.2 Federated Learning Methodologies

3.2.1 Federated Learning Definition

FL seeks to create a combined machine learning model based on data from several locations. In FL, there are two processes: model training and model inference. Information, but not data, can be transferred between parties throughout the model training process. At each site, the exchange does not reveal any protected private sections of the data. The trained model might be kept by one party or shared among several. The model is applied to a fresh data instance at inference time. In a B2B environment, for example, a federated medical-imaging system may receive a new patient with diagnoses

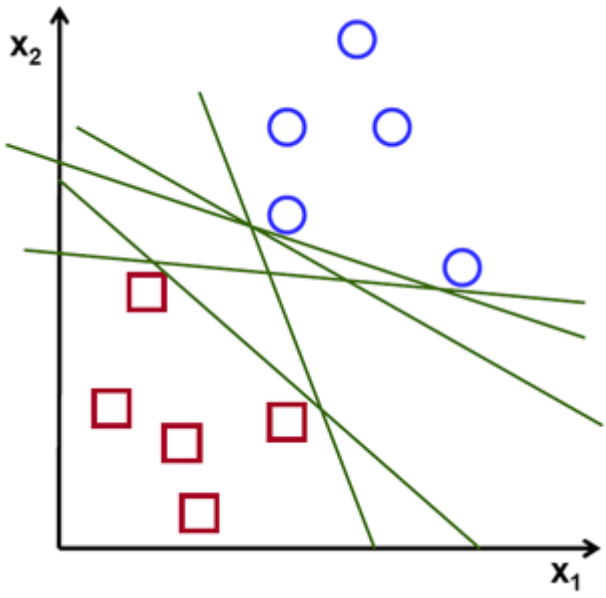


Figure 3.6: Possible separating linear hyperplane [23]

from multiple hospitals. In this case, the parties work together to make a prediction. Finally, a fair value-distribution mechanism should be in place to share the profit generated by the collaborative model. Mechanism design should be done in such a way that the federation can be sustained.

In broad terms, FL is an algorithmic framework for building ML models that can be characterized by the following features, where a model is a function mapping a data instance at some party to an outcome [24]. Applications of FL are suitable in contexts where the following conditions hold:

- There are two or more parties interested in jointly building an ML model. Each party holds some data that it wishes to contribute to training the model.
- In the model-training process, the data held by each party does not leave that party.
- The model can be transferred in part from one party to another under an encryption scheme, such that other parties cannot re-engineer the data at any given party.
- The performance of the resulting model is a good approximation of the ideal model built with all data transferred to a single party.

Consider N data owners $\{F_i\})_{i=1}^N$ who want to train a machine learning model with their respective data sets $\{D_i\}_{i=1}^N$. A common method is to gather all data $\{D_i\}_{i=1}^N$ on a single data server and train an ML model M_{SUM} on the server using the consolidated data. In the traditional method, any data owner $\{F_i$, as well as other data owners, will disclose their data $\{D_i$ to the service.

FL is an ML process in which the data owners collaboratively train a model M_{FED} without collecting all data $\{D_i\}_{i=1}^N$. Denote ν_{SUM} and ν_{FED} as the performance measure (e.g., accuracy) of the centralized model M_{SUM} and the federated model M_{FED}, respectively.

We can capture what we mean by performance guarantee more precisely. Let δ be a non-negative real number. We say that the FL model M_{FED} has δ-performance loss if

$$\|\nu_{SUM} - \nu_{FED}\| < \delta$$

The previous equation expresses the following intuition: if we use secure FL to build an ML model on distributed data sources, this model's performance on future data is approximately the same as the model built on joining all data sources together.

We allow the FL system to perform a little less than a joint model because, in FL, data owners do not expose their data to a central server or any other data owners. This additional security and privacy guarantee can be worth a lot more than the loss inaccuracy, which is the δ value.

An FL system may or may not involve a central coordinating computer depending on the application. An example involving a coordinator in a FL architecture is shown in Figure 3.7 In this setting, the coordinator is a central aggregation server (a.k.a. the parameter server), which sends an initial model to the local data owners A–C (a.k.a. clients or participants). The local data owners A–C train a model using their respective dataset and send the model weight updates to the aggregation server. The aggregation server then combines the model updates received from the data owners (e.g., using federated averaging [25]) and sends the combined model updates back to the local data owners. This procedure is repeated until the model converges or until the maximum number of iterations is reached. Under this architecture, the raw data of the local data owners never leave the local data owners. This approach ensures user privacy and data security and saves the communication overhead needed to send raw data. The communication between the central

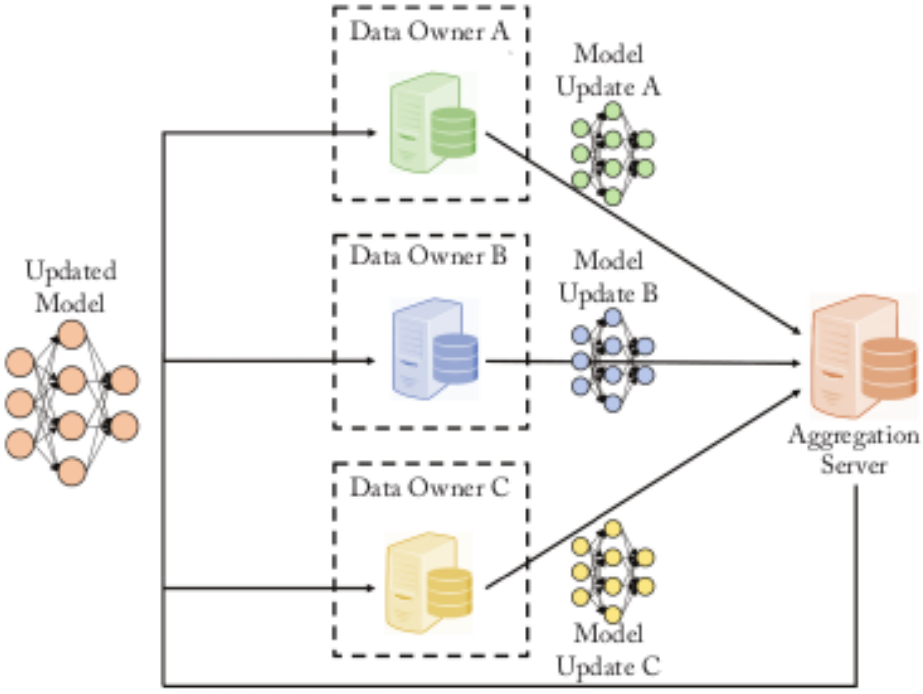


Figure 3.7: An example of FL architecture: client-serve model [24]

aggregation server and the local data owners can be encrypted (e.g., using homomorphic encryption [24, 26]) to guard against information leakage.

The FL architecture can also be designed in a peer to peer manner, which does not require a coordinator. This ensures further security guarantee in which the parties communicate directly without the help of a third party, as illustrated in Figure 3.8. The advantage of this architecture is increased security, but a drawback is potentially more computation to encrypt and decrypt messages since each peer has its own security technique.

FL brings several benefits. It preserves user privacy and data security by design since no data transfer is required. FL also enables several parties to collaboratively train an ML model so that each of the parties can enjoy a better model than what it can achieve alone. For example, FL can be used by private commercial banks to detect multi-party borrowing, which has always been a headache in the banking industry, especially in the Internet finance industry [27]. There is no need to establish a central database with FL, and any financial institution participating in FL can initiate new user queries to other agencies within the federation. Other agencies only need to answer

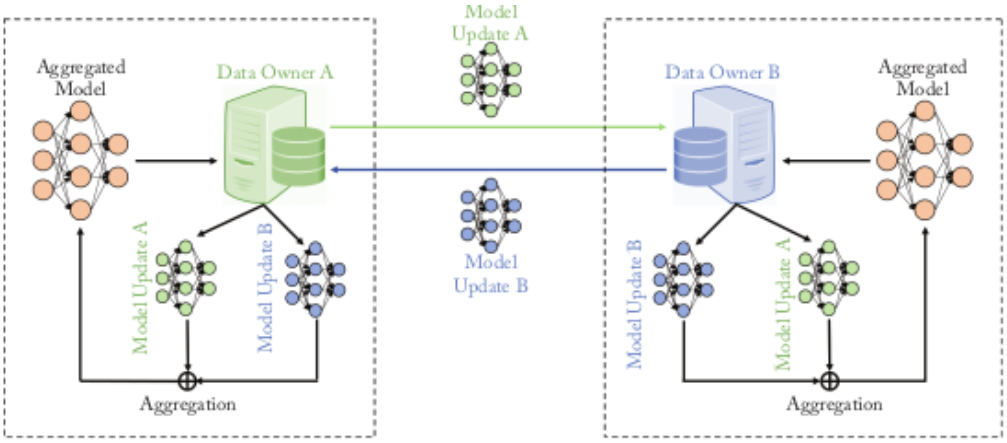


Figure 3.8: An example of FL architecture: peer-to-peer model. [24]

questions about local lending without knowing the specific information of the user. This approach protects user privacy and data integrity and achieves a critical business objective of identifying multi-party lending.

While FL has great potential, it also faces several challenges. The communication link between the local data owner and the aggregation server may be slow and unstable [28]. There may be a vast number of local data owners (e.g., mobile users). In theory, every mobile user can participate in FL, making the system unstable and unpredictable. Data from different participants in FL may follow non-identical distributions [29–31], and different participants may have unbalanced numbers of data samples, which may result in a biased model or even failure of training a model. As the participants are distributed and difficult to authenticate, FL model poisoning attacks [32], in which one or more malicious participants send ruinous model updates to make the federated model useless, can take place and confound the whole operation.

3.2.1.1 Categories of Federated Learning

Matrix D_i denote the data held by the ith data owner. Suppose that each row of the matrix D_i represents a data sample, and each column represents a specific feature. At the same time, some data sets may also contain label data. We denote the feature space as X, the label space as Y, and we use I to denote the sample ID space. For example, in the financial field, labels may be users' credit. In the marketing field, labels may be the user's purchasing

desire. In the education field, Y may be the students' scores. The feature X, label Y, and sample IDs I constitute the complete training data-set $(I; X; Y)$. The feature and sample spaces of the data sets of the participants may not be identical. We classify FL into horizontal FL (HFL), vertical FL (VFL), and federated transfer learning (FTL), according to how data is partitioned among various parties in the feature and sample spaces. Figures 3.9,3.10,3.11 show the three FL categories for a two-party scenario [24].

HFL refers to the case where the participants in FL share overlapping data features, i.e., the data features are aligned across the participants, but they differ in data samples(also the samples can have overlaps in some sense). It resembles the situation that the data is horizontally partitioned inside a tabular view. Hence, we also call HFL sample-partitioned FL or example-partitioned FL [33]. In our work we used the overlapped features of our data for different operatos so we used HFL to identify failure root causes. Also, in our separation assume that we don't have any overlapping sample.

Different from HFL, VFL applies to the scenario where the participants in FL share overlapping data samples, i.e., the data samples are aligned amongst the participants, but they differ in data features. It resembles the situation that data is vertically partitioned inside a tabular view. Thus, we also name VFL as feature-partitioned FL. FTL is applicable for the case when neither is overlapping in data samples nor features.

For example, when the two parties are two banks that serve two different regional markets, they may share only a handful of users, but their data may have very similar feature spaces due to similar business models. That is, with limited overlap in users but significant overlap in data features, the two banks can collaborate in building ML models through horizontal FL [24, 26].

When two parties are providing different services but sharing many users (e.g., a bank and an e-commerce company), they can collaborate on the different feature spaces that they own, leading to a better ML model for both. That is, with considerable overlap in users but little overlap in data features, the two companies can collaborate in building ML models through vertical FL [24, 26]. Split learning, recently proposed by [34], and [35], is regarded here as a particular case of vertical FL, which enables vertically federated training of deep neural networks (DNNs). That is, split learning facilitates training DNNs in FL settings over vertically partitioned data [35].

In scenarios where participating parties have highly heterogeneous data (e.g., distribution mismatch, domain shift, limited overlapping samples, and

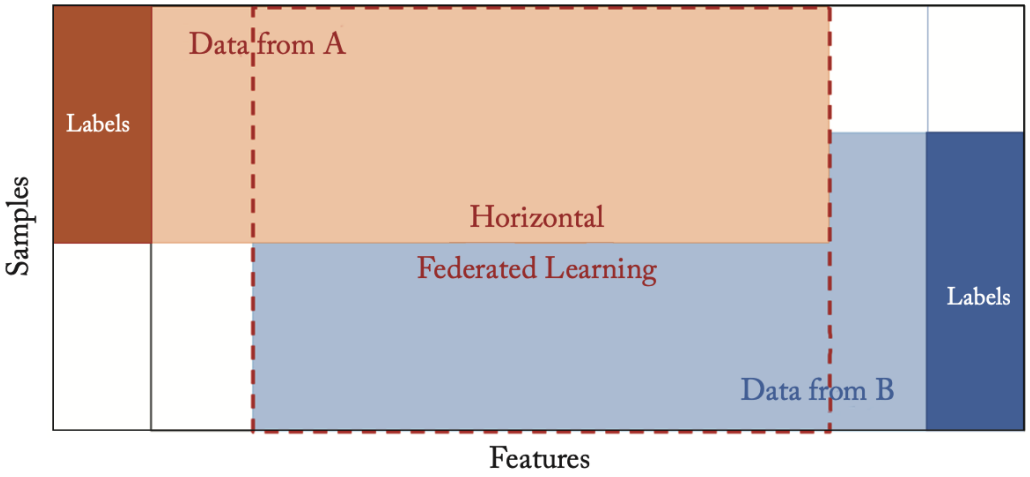


Figure 3.9: Illustration of HFL, a.k.a. sample-partitioned FL where the over- lapping features from data samples held by different participants are taken to jointly train a model [24].

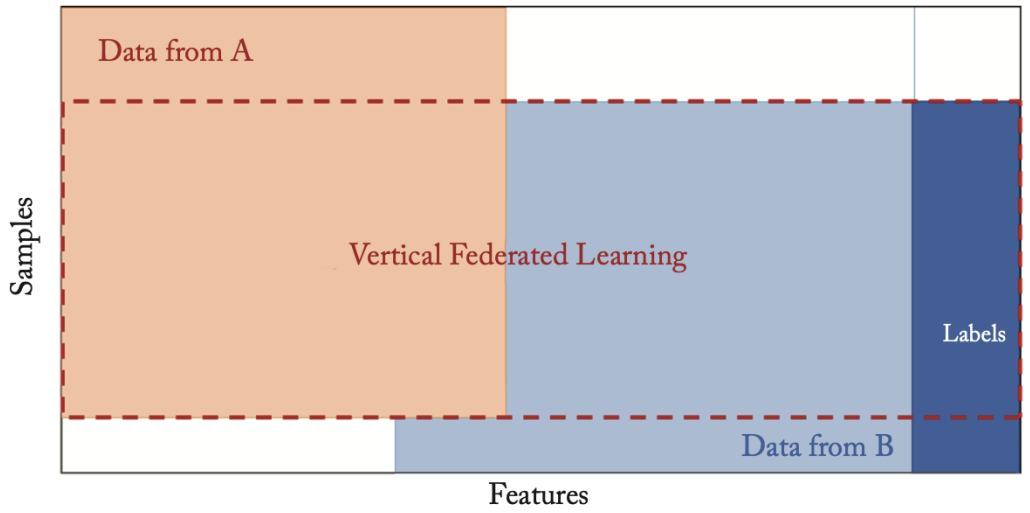


Figure 3.10: Illustration of VFL, a.k.a feature-partitioned FL where the overlapping data samples that have non-overlapping or partially overlapping features held by multiple participants are taken to jointly train a model [24]

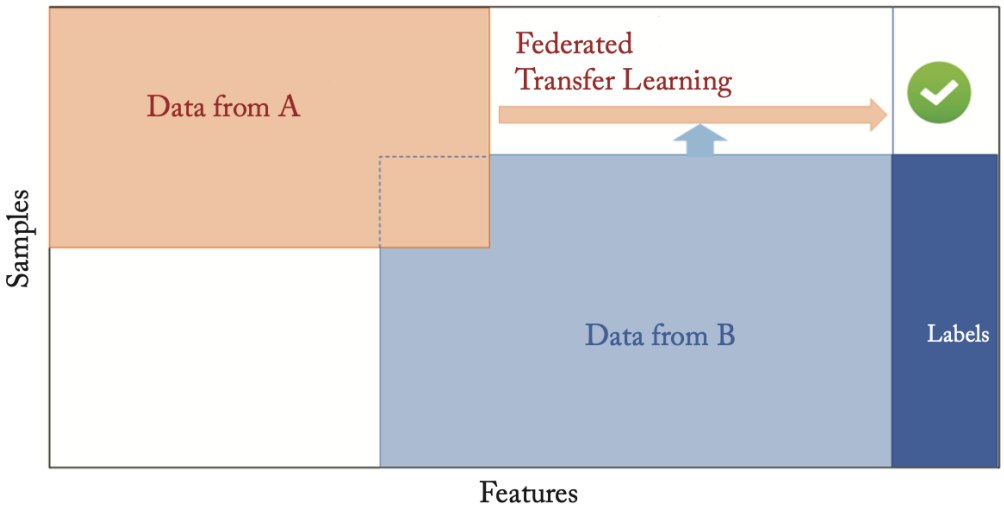


Figure 3.11: Federated transfer learning (FTL). A predictive model learned from feature representations of aligned samples belonging to party A and party B is utilized to predict labels for unlabeled samples of party A. [24].

scarce labels), HFL and VFL may not build effective ML models. In those scenarios, we can leverage transfer learning techniques to bridge the heterogeneous data owned by different parties. We refer to FL leveraging transfer learning techniques as FTL.

Transfer learning aims to build effective ML models in a resource-scarce target domain by exploiting or transferring knowledge learned from a resource-rich source domain, which naturally fits the FL setting where parties are typically from different domains. [36] divides transfer learning into mainly three categories: (i) instance-based transfer, (ii) feature-based transfer, and (iii) model-based transfer. Here, we provide brief descriptions of how these three categories of transfer learning techniques can be applied to federated settings.

- **Instance-based FTL.** Participating parties selectively pick or re-weight their training data samples such that the distance among domain distributions can be minimized, thereby minimizing the objective loss function.

- **Feature-based FTL.** Participating parties collaboratively learn a common feature representation space, in which the distribution and seman-

tic difference among feature representations transformed from raw data can be relieved and such that knowledge can be transferable across different domains to build more robust and accurate shared ML models.

Figure 3.11 illustrates an FTL scenario where a predictive model learned from feature representations of aligned samples belonging to party A and party B is utilized to predict labels for unlabeled samples of party A.

- **Model-basedFTL.** Participating parties collaboratively learn shared models that can benefit transfer learning. Alternatively, participating parties can utilize pre-trained models as the whole or part of the initial models for a FL task.

In our work, the data separation is in the domain of HFL. We will elaborate more on that in chapter 4.

Besides these categories, we can classify FL by the distribution of the clients.

3.2.2 Federated Learning Local Data Owners Distribution

FL is a machine learning setting where many clients (e.g., mobile devices or whole organizations) collaboratively train a model under the orchestration of a central server (e.g., service provider) while keeping the training data decentralized. It embodies the principles of focused collection and data minimization and can mitigate many of the systemic privacy risks and costs resulting from traditional, centralized machine learning [33].

The term Federated Leaning was introduced in 2016 by McMahan et al. [25]: "We term our approach FL since the learning task is solved by a loose federation of participating devices (which we refer to as clients) which are coordinated by a central server." An unbalanced and non-IID (identically and independently distributed) data partitioning across many unreliable devices with limited communication bandwidth was introduced as the defining set of challenges.

The categories that were introduced above can have one of the classifications as follow:

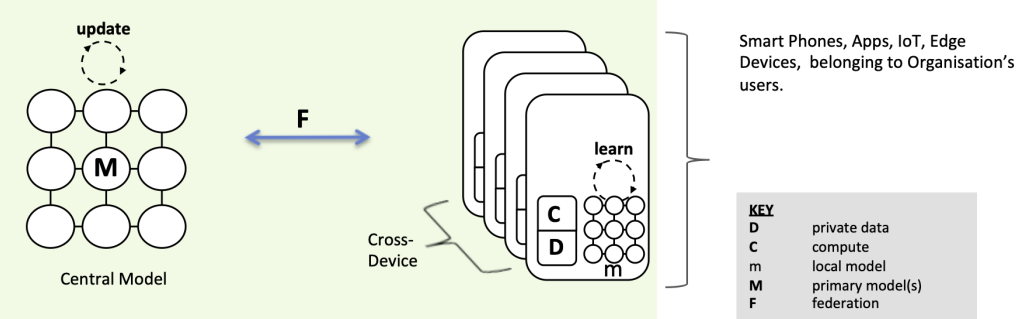


Figure 3.12: Single Organisation, Cross-Device FL [37]

Cross-Device Federated Learning

To restate some of the challenges, we are working at scale, potentially needing millions of devices for the federation to work. Devices may be offline, and we need to be careful about when we consume compute and impact user experience. Again, with their control of the OS, Google is in a unique position versus app developers, for example, and it may be more difficult to implement a solution if we are a smaller scale, standard business with an app. The data in this scenario partitioned horizontally [37].

Cross-Silo Federated Learning

Cross-silo FL is usually used in B2B communications; most of our work also falls into this classification. We are now looking to unlock the value of data that is more widely distributed. In this case, we have a few local data owners with lots of data, for example, between hospitals and banks, perhaps distributed aggregate data from consumer wearable in different fitness app businesses we have cross-silo FL.

While the aim for Cross-Silo is generally said to be the same — to update and enhance a central, and in this instance, shared model — there are potentially more substantial problems on the security front. At the same time, each company can employ more consistent, resilient, and scalable computing.

3.2.3 Federated Optimization

To distinguish it from distributed optimization, the optimization issue emerging from FL is referred to as federated optimization [25, 38]. In real-

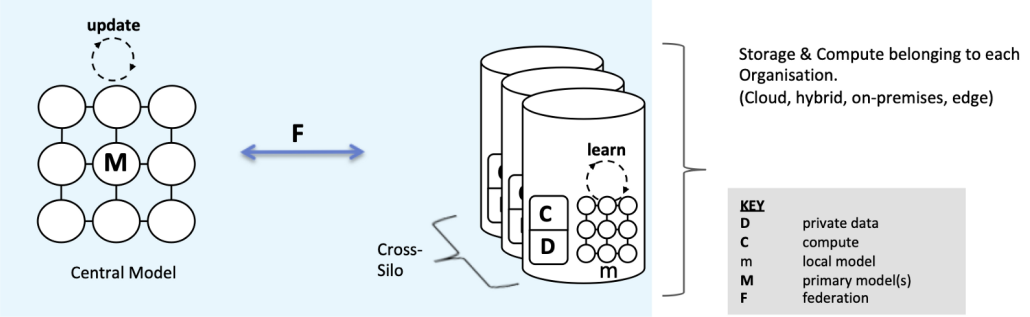


Figure 3.13: Multiple Organisations, Cross-Silo FL [37]

ity, federated optimization differs from traditional distributed optimization problems in numerous ways [25, 39, 40].

- **Datasets with non-independent identical distributions (Non-IID).** It is possible to guarantee that various compute nodes have IID datasets for distributed optimization inside a data center, such that all local updates appear pretty similar. IID datasets cannot be ensured in federated optimization. We cannot use IID assumptions for decentralized datasets in FL because the data held by various participants may have entirely different distributions [41]. While similar participants may have similar local training data, two randomly selected individuals may yield substantially different model weight updates or gradient updates.

- **Unbalanced number of data points.** It is feasible to split the data evenly across the computing nodes in a data center for distributed optimization. However, in practical settings, different participants typically have vastly varying quantities of training datasets [38, 42, 43]. Some individuals, for example, may have a few data points, while others may have a significant amount of data.

- **Huge number of participants.** The number of parallel computing nodes inside a data center may be readily adjusted for distributed optimization. However, because ML or DL typically requires a large amount of data, FL applications may need to engage many users, particularly with mobile devices [44]. Every one of these participants has the potential to engage in FL, making it considerably more dispersed than that found in a data center.

- **Slow and unreliable communication links.** In a data center, it is anticipated that nodes interact swiftly with one another and that packets are nearly never lost. However, with FL, communication between clients and the server is based on existing Internet connections. Uploads (from client to server, for example) are significantly slower than downloads, especially if the connection is made from a mobile terminal. Some clients may also experience brief Internet connectivity outages [28].

[25] first used the FedAvg algorithm. He focuses on non-convex objective functions that are often encountered when training DNNs. FedAvg may be used to calculate any finite-sum objective function of the following form:

$$\min_{w \in \mathbb{R}^d} f(w), \quad f(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w),$$

Where n is the number of data points and $w \in R^d$ is the dimension d of model parameters (e.g., model weights of an ANN).

For an ML or DL issue, we usually use $f_i(w) = L(x_i, y_i; w)$, which is the prediction loss on sample (x_i, y_i). For the given model parameters w, where x_i and y_i indicate the ith data point and the accompanying label, respectively. Assume there are K participants (also known as data owners or clients) in an HFL system, with D_k representing the dataset held by the kth participant and P_k denoting the collection of data point indexes on client k. As the cardinality of P_k, define $n_k = |P_k|$. That is, the ith participant is expected to have n_k training data points. As a consequence, given K participants, Equation 3.2.3 may be rewritten as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(W), \; where \; F_k(w) := \frac{1}{n_k} \sum_{i \in P_k} f_i(w)$$

When the data points held by the K participants are distributed independently and identically (IID), we get $\mathbb{E}_{D_k}[F_k(w)] = f(w)$, where the expectation $\mathbb{E}_{D_k}[.]$ is taken over the set of data points owned by the kth participant. In the DML paradigm, distributed optimization algorithms generally make this IID assumption. If the IID assumption is violated, as stated above in the non-IID setting, the loss function $F_k(.)$ preserved at the kth participant may be an arbitrarily poor approximation of the function $f(.)$ [29, 45].

3.2.4 Federated Learning Algorithms

The algorithms that we consider optimize the finite-sum objective

$$\min_{w \in \mathbb{R}^d} f(w), \quad f(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w),$$

where w is a vector that contains d model parameters. In supervised learning, we treat the function $f_i(w)$ as loss function $f_i(w) = l\left(x_i, y_i; w\right)$, where an input-output pair (x_i, y_i) is one of n given labeled examples, often referred to as *training examples*. the objective function $f_i(w)$ is defined by the model parameters w conditioned on n labeled examples. The problem can this be interpreted as finding the w which minimized the average loss over all n training examples.

In a significant data context, where the number of training examples is too large to be stored on one computer, we must distribute the computation to multiple computers. If the number of training examples held by client k is denoted by $n_k = |P_k|$, then we can rewrite the objective function as a weighted sum over all $F_k(w)$:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(W), \; where \; F_k(w) := \frac{1}{n_k} \sum_{i \in P_k} f_i(w)$$

Distributing the data and computational burden leads us to re-formulate the objective function $f(w)$ from Eq. 3.2.4 to 3.2.4. Assume that there are K clients across which data and computation are distributed. Each client then holds a part P_k of all training examples and computes $F_K(w)$, the average loss on client k.

Our evaluation includes the synchronous FedAvg and FSVRG algorithms as well as the asynchronous CO-OP algorithm. In our work we picked FedAvg algorithm since it is most suited for HFL. This section describes these algorithms and gives their pseudocode [46].

3.2.4.1 Federated Averaging (FedAvg)

FedAvg orchestrates training via a central server that hosts the shared global model w_t, where t is the communication round. However, the actual optimization is done locally on clients using, for instance, Stochastic Gradient

Algorithm 1: FederatedAveraging

1 initialize w_0
2 **for** *each round* $t = 0, 1, \ldots$ **do**
3 $\quad m \leftarrow \max(\lfloor C \cdot K \rfloor, 1)$
4 $\quad S_t$ = random set of m clients
5 $\quad$ **for** *each client* $k \in S_t$ ***in parallel*** **do**
6 $\quad\quad w_{t+1}^k = \texttt{ClientUpdate}(k, w_t)$
7 $\quad w_{t+1} = \sum_{k \in S_t} \frac{n_k}{n_\sigma} w_{t+1}^k, \quad n_\sigma = \sum_{k \in S_t} n_k$

Decent (SGD). FedAvg has five hyperparameters: the fraction of clients C to select for training, the local mini-batch size B, the number of local epochs E, a learning rate η, and possibly a learning rate decay λ. The parameters B, E, η, and λ are commonly used when training with SGD. However, here E stands for the total number of iterations through the local data before the global model is updated.

The algorithm starts by randomly initializing the global model w_0. One communication round of FedAvg then consists of the following: The server selects a subset of clients S_t, $|S_t| = CK \geq 1$, and distributes the current global model w_t to all clients in S_t. After updating their local models w_t^k to the shared model, $w_t^k \leftarrow w_t$, each client partitions its local data into batches of size B and performs E epochs of SGD. Finally, clients upload their trained local models w_{t+1}^k to the server, which then generates the new global model w_{t+1} by computing a weighted sum of all received local models. The weighting scheme is dependent on the number of local training examples, as described in Algorithm 1 on line 7 [46].

3.2.4.2 Federated Stochastic Variance Reduced Gradient (FSVRG)

The idea behind FSVRG is to perform one expensive full gradient computation centrally, followed by many distributed stochastic updates on each client. With each iteration through a random permutation of the local data, a stochastic update is performed by performing one update per data point. Standard FSVRG only has one hyperparameter: the step size h. However, this step size is not used directly. Instead, client k has a local step size h_k

that is inversely proportional to n_k , i.e. $h_k = \frac{h}{n_k}$. The motivation behind h_k is that clients should make roughly the same amount of progress when n_k varies greatly from client to client [47].

Algorithm 2 gives a complete description of FSVRG, where one iteration is performed as follows: First, to compute a full gradient, all clients download the current model w_t and compute loss gradients to their local data. Clients then upload their gradients, which the server aggregates to form the full gradient $\nabla f(w_t)$. Next, all clients download $\nabla f(w_t)$ and initialize their local model w_t^k and local step-size h_k . After creating a random permutation of their local data, clients will iteratively perform n_k SVRG updates using both the local and the full gradient as well as a client-specific step size h_k. Finally, when all clients have computed and uploaded their final w_{t+1}^k, the server combines all w_{t+1}^k to form a new global model w_{t+1}, similar to FedAvg.

SVG in its original form [[47], Alg. 4] is primarily concerned with sparse data in the sense that some features are seldom represented in the dataset or are only present on few clients. This sparsity structure is exploited by multiplying gradients and model parameters with diagonal matrices that contain how frequently features are represented. However, this scaling is only possible because the dimension of the model is the same as the dimension of the input in the Support Vector Machine (SVM) model they consider. However, in a neural network model, the number of parameters is generally much larger than the input dimension.

3.2.4.3 CO-OP

Whereas FedAvg and FSVRG rely on synchronized model updates, CO-OP [48] proposes an asynchronous approach. This approach will immediately merge any received client model with the global model. Each client k has an age a_k associated with its model, and the global model has age a. The model age difference, $a - a_k$, is used to compute a weight when merging models. This idea is motivated by the fact that some clients will train on outdated models in an asynchronous framework while others will train on more up-to-date ones.

A local model will only be merged if $b_l \leq a - a_k \leq b_u$,for some choice of integers $b_l < b_u$. The intuition behind this acceptance rule is that we neither want to merge outdated models ($a - a_k > b_u$) nor models from overactive clients ($a - a_k < b_u$). The lower and upper bounds, b_l and b_u , can therefore

Algorithm 2: Federated SVRG

initialize w_0
$h \leftarrow$ stepsize
$\{\mathcal{P}_k\}_{k=1}^{K}$ = data partition
for *each round* $t = 0, 1, \ldots$ **do**
    Compute $\nabla f(w_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_t)$
    for *all* K *clients* ***in parallel*** **do**
        initialize: $w_{t+1}^k \leftarrow w_t$, and $h_k = \frac{h}{n_k}$
        let $\{i_s\}_{s=1}^{n_k}$ be a permutation of $\mathcal{P}_k$
        for $s = 1, \ldots, n_k$ **do**
            $\Theta \leftarrow \nabla f_{i_s}(w_{t+1}^k) - \nabla f_{i_s}(w_t) + \nabla f(w_t)$
            $w_{t+1}^k \leftarrow w_{t+1}^k - h_k \cdot \Theta$
    $w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$ `// update global model`

be thought of as an age filter. CO-OP also inherits all hyperparameters from its underlying optimization algorithm, for instance SGD.

The training procedure is as follows: Each client has its training data and performs E rounds of an optimization algorithm before requesting the current global model age a from the server. The client now decides whether or not its age difference meets the restrictions. Should the local model be outdated, the client reconciles with the global model and starts over. Should the client instead be overactive, it just continues training. Otherwise, the local model is uploaded to the server for merging. The pseudocode of CO-OP is presented in Algorithm 3.

3.3 Microwave Networks Technologies

3.3.1 Hardware Components

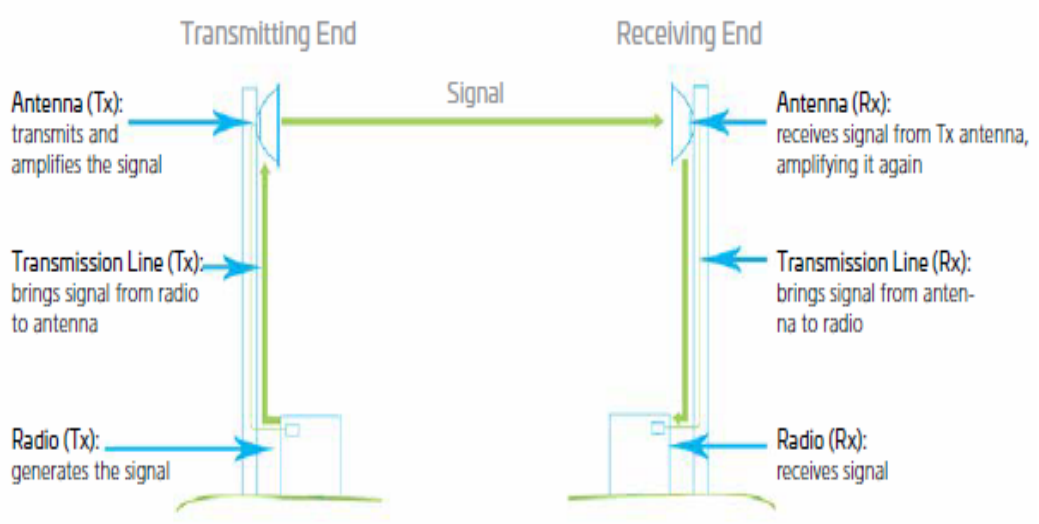


Figure 3.14: Basic components that allow LOS microwave communications [49].

The basic structure of a microwave communication system with its building blocks is shown in Fig.3.14. The structure includes a microwave radio at the transmitter site, connected to a directional antenna via a transmission line. The outbound signal from the directional antenna is aligned to a receiving antenna connected to a radio receiver. This section discusses the

Algorithm 3: CO-OP

$w = w_1 = \ldots = w_K \leftarrow w_0$
$a \leftarrow b_l$
$a_1 = \ldots = a_K \leftarrow 0$
// Each client k independently runs:
while *true* **do**
  $w_k \leftarrow$ ClientUpdate(w_k)
  Request and receive the model age a from the server
  if $a - a_k > b_u$ **then**
    // Client is outdated
    Fetch w, a from the server
    $w_k \leftarrow w, a_k \leftarrow a$
  else if $a - a_k < b_l$ **then**
    continue // Client is overactive
  else
    // Normal update
    $w_k, a_k \leftarrow$ UpdateServer$(w_k, a_k) = \{$
      $w \leftarrow (1-\alpha) \cdot w + \alpha \cdot w_k\,, \quad \alpha \leftarrow (a - a_k + 1)^{-\frac{1}{2}}$
      $a \leftarrow a + 1$
      return w, a
    $\}$

characteristics of three main elements and their impact on the link's functionality.

Radio

The radio element is the only active element in the system. The transmitter site (TX) generates the signal to transmit and code it, then aggregate and compress it in a relatively small radio channel. This procedure is called Modulation. After the Modulation, the radio up-converts the radio channel to the right microwave frequency to be transmitted.

The radio on the receiver site (RX) operates the opposite procedure, i.e., it down-converts the radio channel from the microwave frequency and then demodulates the signal to be sent again on the cable communication.

The radio has three basic configurations used in microwave communications systems [49]:

- *Full Indoor:* All active components are located inside a building or shelter, allowing easy maintenance and upgrades, without requiring tower climbs, for instance. Being farther from the antenna may introduce higher transmission line losses than other configurations, however.
- *Full Outdoor:* All the electronic devices are mounted outside, eliminating the need and cost for indoor space. However, they can be difficult to access for maintenance or upgrades, requiring tower climbs as they are located on the tower. In some cases, rooftop access mitigates this challenge.
- *Split-Mount:* Electronic devices are distributed into an outdoor unit (ODU) and the indoor unit (IDU), eliminating transmission line losses with the easy maintenance of the IDU. However, it also combines the disadvantages of the other two configurations by requiring indoor storage and tower climbs for the ODU.

Transmission Line

The transmission line is the physical media connecting the radio and directional antenna. Because of the amount of signal loss they can introduce, the choice of transmission line type is determined mainly by the frequencies in use. There are two possible implementations for the transmission lines [49]:

- *Coaxial Cable:* It is suitable in applications using frequencies up to, or just above, 2 GHz. Above this frequency range, the physical medium introduces unacceptable signal loss.

- *Waveguide:* It is suitable for higher frequencies. An elliptical waveguide featuring an elliptical cross-section can support frequencies up to around 40 GHz. However, it is rarely used in applications above 13 GHz. Microwave waveguides are maintained under dry air or dry nitrogen pressure to avoid moisture condensation that degrades their performance.

Antenna

Antennas are devices that radiate or receive electromagnetic waves of specific frequencies. The antenna is a transition structure between the transmission line and the open air that makes the generation of radiated electromagnetic power as efficiently as possible. An antenna designed to radiate and receive microwave frequencies, therefore, is called a microwave antenna.

A directional antenna in a microwave system is typically parabolic in shape, as this permits the greatest focus of energy possible in a single beam. The antennas are usually polarized, vertical or horizontal, based on the location of their feed connection.

There are different types of antennas, selected according to the link characteristics and evaluating some specific antenna parameters, i.e.:

- *Directivity function:* It expresses the antenna capability of radiating and receiving electromagnetic waves in a generic direction, usually identified by the azimuth and elevation angles. The Directivity Function shows how, even if the parabolic microwave antennas are highly directional, some signal energy is lost due to transmission towards the undesired direction.

- *Gain:* It represents the gain provided by the antenna in its direction of maximum directivity for the isotropic antenna; in other words, the gain is the measure of how the transmitter antenna concentrates power density in the direction of maximum irradiation, as defined by the directivity function.

- *Effective Area*: The size of the antenna dish is a vital part of its design, function, and role within the network. Bigger antenna dishes yield extraordinary power, but they are more challenging to be installed and introduce limitations regarding tower space, tower loading, leasing costs, and local zoning regulations. The effective area returns the amount of power flux density captured by the antenna in the reception phase.

3.3.2 Channel Characterization

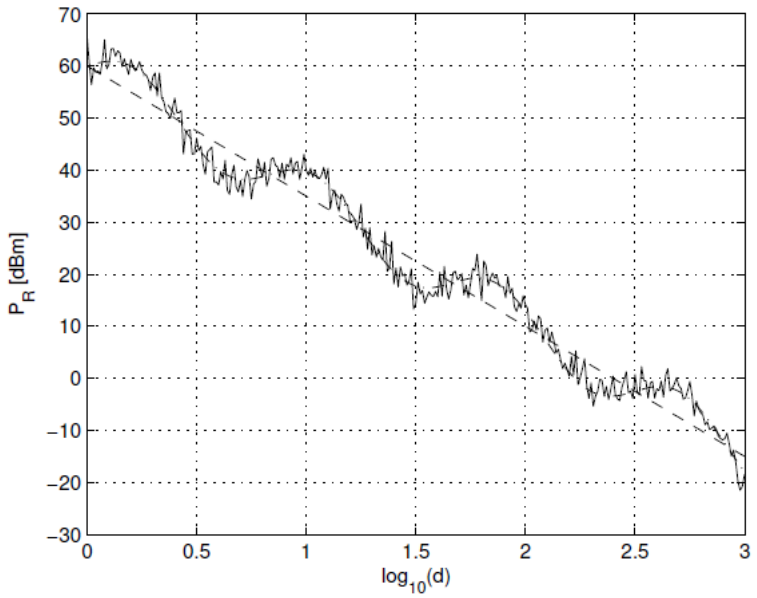


Figure 3.15: Representation of the large and small scale channel components [50].

In wired networks, the electromagnetic waves used for the communication are guided, so the attenuation at which they are subjected is provided mainly by internal effects. Instead, in microwave communications, another element that affects the transmission is the channel behavior; the electromagnetic waves are radiated in the free space and suffer the channel effects. The channeling effect can be divided into three components [50], in Fig.3.15 we can see the attenuation effect of the channel provided by the different components. The elements that compose the channel effects will be addressed more deeply in the following sections, referring to Fig.3.15 in order to explain them.

Path Loss

Path loss is a phenomenon whose variations are associated with considerable distance modifications regarding wavelengths (large-scale components). The path loss expresses the deterministic relation between the link distance *d*, the frequency, and the mean channel attenuation. Path loss can be expressed in a linear unit or dB.

This component allows using of a straightforward model derived from the empirical observation, as the channel path loss in dB decreases linearly with *log(d)*. This relation is shown in Fig.3.15 as a straight dotted line.

Shadowing

The shadowing is a random attenuation that provides fluctuations around the path loss value; this model component is due to obstacles that shadow entirely or partially the propagation path.

The fluctuations of the attenuation are well described in linear units by a log-normal distribution, which means that the logarithmic representation of the attenuation assumes a Gaussian density.

The model of these components is justified by thinking that the global attenuation, in dB, is given by the sum of numerous components corresponding to the attenuation of single fractions of the overall path (Central Limit theorem). The shadowing effect is represented in Fig.3.15 as the sinusoidal dotted line around the straight line is associated with the path loss.

Multipath Fading

In any terrestrial radio communications system, the signal will reach the receiver via the direct path and as a result of reflections from objects such as buildings, hills, ground, water, etc., adjacent to the main path.

Multipath fading is the small scale component, represents a random attenuation fluctuation around the large scale attenuation (path loss + shadowing). This component derives from the combination of more signal components at the receiver, as reflections or echos, with different phases and amplitude. In Fig.3.15 its effect is represented with the sharp line around the sinusoidal line representing the shadowing.

Multipath fading may also distort the radio signal. As the various paths taken by the signals vary in length, the signal transmitted at a particular

instance will arrive at the receiver multiple times along the different paths. This phenomenon can cause problems as phase distortion and inter-symbol interference. As a result, it may be necessary to incorporate features within the radio communications system that reduce the effect of these problems.

Multipath fading can affect radio communications channels in two different ways [51]:

- *Flat fading*: This form of multipath fading affects all the frequencies across a given channel, either equally or almost equally. When flat multipath fading is experienced, the signal changes in amplitude, rising and falling over a while.

- *Selective fading*: Selective fading occurs when the multipath fading affects different frequencies across the channel in different ways. This effect happens when phases and amplitudes of the signal vary across the channel. Sometimes relatively deep attenuation may be experienced, and this can give rise to some reception problems. Simply maintaining the overall amplitude of the received signal will not overcome the effects of selective fading, and some form of equalization may be needed.

3.3.3 ACM

ACM stands for adaptive code and Modulation. It is a software element used to fight the effects of fading, and this means that the ACM tries to match Modulation, coding, and other signal and protocol parameters to the conditions on the link (i.e., to the quality of the radio channel). The ACM prevent unavailability dynamically adapting the signal to become more robust even to bad channel condition (e.g., rain or snow).

The ACM goal is to improve the efficiency of the radio link by increasing network capacity over the existing infrastructure while reducing sensitivity to environmental interferences. Adaptive Modulation means dynamically varying the modulation in an errorless manner to maximize the throughput under momentary propagation conditions. In other words, a system can operate at its maximum throughput under clear sky conditions and decrease it gradually under rain fade (e.g., links can change from 1024QAM down to QPSK to keep “link alive” without losing connection). Before the invention of the ACM, the designer must set up the links relying on the "worst-case" conditions to avoid link outage.

Adapting coding and Modulation provide benefits to the distances achievable with the radio links, to the availability that it is considered more important than the reduction of capacity during the heavy fade phenomena on the channel. This phrase is acceptable when the connection is provided using IP, that can work with a variable-capacity, and in the end, allows the operator to use smaller antennas that will minimize costs and avoid aesthetic and plan constraints in dense urban areas and regions of natural beauty, where planners or building owners may prohibit large antennas.

3.3.4 Performance Metrics

The performance of radio links concerning its availability constraints is evaluated through specific metrics defined by ITU-T. Recommendations G.826 and G.828 [52] describe how to measure the performances in terms of errors and availability. The measurements refer to the concept of the block that is a set of consecutive bits associated with the path, where each bit belongs to one and only one block.

The recommendations G.826 and G.828, as presented in [52], contain the following error event counters that are defined as follows:

- *Errored Block (EB):* A block in which one or more bits are in error.
- *Errored Second (ES):* one second with one or more errored blocks or at least one defect; where the defects are different errors from an EB defined according to the technology.
- *Severely Errored Second (SES):* A one-second period which contains 30% errored blocks or at least one defect.
- *Background Block Error (BBE):* An errored block not occurring as part of an SES.
- *Severely Errored Period (SEP):* A time during which at least three but not more than nine consecutive severely errored seconds (SES) occur.
- *Errored Second Ratio (ESR):* The ratio of ES out of total time in seconds in available time during a fixed measurement interval.
- *Severely Errored Second Ratio (SESR):* The ratio of SES out of total time in seconds in available time during a fixed measurement interval.

- *Background Block Error Ratio (BBER):* The ratio of Background Block Errors (BBE) out of a total number of blocks in available time during a fixed measurement interval.

- *Severely Errored Period Intensity (SEPI):* The number of SEP events in the available time, divided by the total available time in seconds.

The previous metrics are related to communication errors, but none explained how to measure the unavailability. The system is defined as unavailable when being measured ten consecutive severely errored seconds (SES). It becomes available again after ten consecutive seconds that are not severely errored. To measure the unavailability, the "Unavailable seconds (UAS)" measure is defined, it contains the number of seconds when the system is unavailable in a measurement interval; the UAS of a measurement interval is computed as the sum of all the time intervals containing at least ten consecutive severely errored seconds in at least one direction of transmission.

3.3.5 Categories Of Failures In Microwave Links

Microwave communication is susceptible to failures caused by various factors such as rain, vegetation, interference from other radio connections. These failure reasons can be transient or chronic and can result in channel unavailability, lowering the link's dependability and causing massive data loss. In our challenge, we wish to create a technique for doing automatic failure troubleshooting that outputs the source of the failure. The failure factors that we explore in this paper are provided below.

Six distinct failure root causes are identified. Although these six root causes can coexist and contribute to a failure in actual microwave connections, we believe that the channel can only be affected by one sort of failure root cause at a time in our study.

- *Deep Fading:* The fading is a variation of the channel attenuation. Fading is provided by different variables like seasonality, geographical position, or radiofrequency. It is a random event and represents the shadowing and the multipath effects, presented in subsection 3.3.2. The deep fading occurs when high attenuation is provided by the channel, which causes the unavailability of the link. This phenomenon can be associated with some obstacles in the LOS of the link that was not

present in the design phase (e.g., growth of vegetation), or since the links work at high frequency, it can be provided by some meteorological phenomena (e.g., rain, snow and even fog at very high frequencies). It is a temporary source of unavailability, and normally, no action is required (e.g., as it happens when some meteorological phenomena occur). When the shadowing effect occurs, due to some fixed obstacle, and the LOS is not provided anymore, the problem becomes complex, and some experts must solve the problem by working actively on it (e.g., eliminating the obstacle in LOS or changing the radio link position).

- *Extra Attenuation:* When the link is designed, the highest Modulation available on the link is defined, and in turn, the nominal received power is defined, which provides a BER threshold able to satisfy the availability constraints of the link. In normal conditions, the maximum received power must match or be near the nominal received power, but when the gap between the measured data and the project value increases above a certain threshold (i.e., 6 dB), the link is in an extra attenuation case. This failure may not cause unavailability, but it is synonymous with some hardware (e.g., crushing of transmission lines) or configuration problems. The different attenuation problem does not change over time, and to fix it, human action is needed; this may involve intervention in the field or remotely.

- *Interference:* In this case, the radio link is subject to a consistent number of errors; this means that the actual BER of the link overcomes the threshold to respect the required link availability. The errors in the link are provided by the fact that the received signal is not as pristine as the receiver would expect. This event happens because the signal arrives in the radio station with the addition of the channel noise, which is expected, plus another interferent signal; this makes the received signal unintelligible and causes errors in the communication. This cause of failure results from a bad design of the link, particularly a lousy interference analysis. The interference problem does not change over time. Human action is needed to turn off the interfering link or change its carrier frequency to solve the interference problem.

- *Low Margin:* This problem is the representation of one of the other known problems that cause the presence of UAS. However, these prob-

lems could have been avoided; a lousy design phase generates the problem. Usually, the problem is because the ACM is disabled or because the lowest reachable Modulation defined to the ACM is not the lowest configurable one; if the ACM had been correctly configured, UAS would not have been detected for the same channel phenomenon.

- *Self-Interference:* The radio links work in a Full-Duplex way, this means that each site needs to transmit and receive the signal; in each site are present two radio components, one for transmitting and one for receiving. The transmission line that connects the antenna to the two radio components is split in one point; usually, the transmitted signal cannot reach the received signal path because they are on two different bands, but because of the presence of some nonlinear components(e.g., amplifiers) of the microwave radio, they can create some spurious signal in the received band that is difficult to filter; this unwanted part of the transmitted signal reaches the receiver radio with a higher power concerning the legitimate signal, creating interference and errors and consecutively unavailability. The problem can be caused by a wrong design procedure or by the degradation of the hardware needed to delete the spurious components of the signals (e.g., filters). The self-interference problem is a constant failure, and to be solved, human action is needed.

- *Hardware Failure:* This category of problems encloses all the failure causes that are not easy to identify analyzing the power measurements. For the Hardware Failure events, there is not a specific way to identify the hardware component targeted by the failure due to the presence of a significant lack of measurements given by the inactivity of the component and by the incompleteness of the information to identify the right hardware components targeted by the failure, as the alarms. The hardware failure is not the only ones present in this class; also, some failure root causes explained before can be included in this category due to the not apparent behavior of the radio measures provided in the failure event. This class of problems contains both permanent and temporary failure sources; the need for human action can be identified by analyzing the performance measure, explained in section 3.3.4, and checking the duration of the unavailability period.

Chapter summary

This chapter's primary focus is the presentation of the FL concept. For a better understanding of this concept, we first needed to introduce machine learning. Knowing these definitions and concepts is helpful to understand the approach used to solve our problem. In general, there is a high-level presentation of the concept of machine learning, a description of the classification problem with its performance measures, and all the steps to provide a classifier able to solve the problem. After that, we gave a brief of FL and its categories; also, we provided the most used algorithms in this field. How the presented algorithm was used in our framework will be explained in the following chapters.

The main hardware and software components that comprise a radio connection were discussed in this chapter. The hardware components and their purpose on the connection are presented first, followed by an examination of the channel components that impact the sent signal. The paper discusses some hardware and software remedies for limiting channel effects. We describe the G.828 guidelines' performance indicators for radio links, as well as the failure causes we evaluate in our research.

Chapter 4

Problem Statement

In this chapter, we will first discuss the underlying reasons for failure. Then we will describe the data that we are trying to categorize and how recognize them. Furthermore, we will go through the many scenarios that might occur in real life and our suggested solution based on FL.

4.1 Failure Identification

4.1.1 Input Data

Our problem's raw data comprises of measurements taken from an Italian microwave network with 10841 radio links.

The dataset consists of a period of time from 26/11/2017 to 10/07/2019. Part of the dataset has been labelled by a human experts with labels indicating failure causes described in section 3.3.5. The labels correspond to a 45-minutes slot and have been applied to windows where at least 1 UAS is present in the last 15 minuets of the slot. Measurements are repeated for each link every 15 minutes. A single sample analysis gives a static perspective of the channel. This analysis can result in circumstances where distinct failure root-causes behave similarly in terms of power measurements collected from the connection. To reduce this difficulty, we choose 45-minute intervals, and the items analyzed indicate the trend of the measurements over time. Analyzing the power measures in the 15-minutes window affected by the failure, in conjunction with information on their evolution in the previous 30-minutes window, can help identify different failure root-causes that behave in the same way during the failure event, but affect the evolution of

the power measures in different ways [53].

4.1.1.1 Data Description

There are 44 fields in the input data. We regard the last field as our output (failure causes) and the rest as our features, as seen below.

General link information ($X_1 \div X_4$): This part, includes information that uniquely identifies link i.e., they include an identification number (ID-Link), the data and time when the measures have been collected for three 15-minutes windows. We will discard this part of the data in our analysis because they do not add any information to our data.

Design information ($X_5 \div X_13$): This information is related to the parameters that are fixed during design phase of the link that does not change over time. The data includes the type of equipment used in the link (X_5). Three types of equipment are considered in our dataset, namely, AlcPlus2e and two devices of SMOS family, all provided by SIAE Microelettronica. The lowest modulation format configured in the ACM software (X_6), the carrier frequency and the bandwidth associated with the link ($X_7 - X_8$), and a flag indicating if the ACM is enabled on the link (X_9). In our analysis we will discard this part of the data because they don't add any information. From X_10 until X_13 provide design features (RxNominal, lowthr, Ptx, LowThr), which describe the design characteristics of the link, we will use these information to train our model.

Propagation Measures [$X_{13} \div X_{43}$]: The performance and power measure sampled during the last 15-minutes windows in which there is at least a UAS (i.e., EsN, txMaxAN, RxminBN,$\cdots$), Also, the link performance and power measure sampled 15 minutes before (i.e., EsN-1, txMaxAN-1, RxminBN-1,$\cdots$), and the same radio link measures sampled 30-minutes before (i.e., EsN-2, txMaxAN-2, RxminBN-2,$\cdots$). Note that in all the 15-minutes slots composing the 45-minutes window, the link measures features are collected at both side of the link, i.e., site A and site B, we will use these information to train our model.

The labels associated to part of the data are identified with the variable name y. The labels identify the root-causes of the failure, i.e., the events producing at least one UAS, they represent one of the known failure causes as identified by the human expert as described in section 3.3.5. The considered labels are the following:

- y_0: Deep Fading
- y_1: Extra attenuation
- y_2: Interference
- y_3: Low margin
- y_4: Self-interference
- y_5: Hardware failures

4.2 Failure Identification With Traditional Machine Learning

The FL framework considered in this thesis is compared against a traditional machine-learning-based approach where we assume all available data points are collected in one central machine where training. This approach has been developed in [53] and is described in the following.

4.2.1 Supervised Failure identification

We can evaluate the failure identification problem as a classification problem in the machine learning domain. The solution of the machine learning problem is no more the root cause of the failure event, but it is a classifier that, receiving as input some information X related to the failure event, is able to identify the failure cause y of the evaluated event with the highest classification accuracy.

In the supervised method, we address the machine learning problem with particular algorithms designed for classification, which require data including information about the failure occurrences connected with the radio link's power measurements and the underlying cause.

Formally, a labeled dataset [X,y] is supplied in input, with each piece of X connected to network measures impacted by at least one unavailability second and designated by the characteristics $[X_1 \div X_{43}]$. [1], with a label y attached.We give as output a classifier that can accept as input network measurements X related with failure occurrences and output a projected failure

[1]Our machine learning algorithm's input characteristics are an elaboration of $[X_1 \div X_{43}]$

cause y while optimizing its classification accuracy.

The goal of the supervised algorithm is to discriminate between the six classes related to the failure cause, explained in chapter 3.3.5.

The authors of [53] evaluated the performances ANN machine learning algorithms.
To assess the algorithms fairly, They developed a performance evaluation process that combines the Holdout and k-fold cross-validation techniques. As to perform a more accurate analysis on the algorithm, they defined a fixed training/test partition (80% Training - 20% Test), as in the Holdout procedure, in this way the classifier will be tested on the same set of unseen points. The hyperparameters must be chosen in order to maximize the classifier performance. To fine tune hyperparameters, They used a K-fold cross-validation method where K is equal to 10, the cross validation procedure is executed only on the training set. note that, in their procedure the splitting of the dataset, always maintain the proportion between the classes, i.e., in their test set all the classes will be represented by the 20% of their points, and in a single iteration of the cross-validation, the classes inside the validation fold are represented by the 10% of their points inside the training dataset.
At the end of the model assessment procedure, all the training set is used to train the selected model, then the performance is evaluated on the test set containing unseen points.The performance evaluated in this case are the accuracy of the classifiers.
Note that, the performance calculated using this procedure are biased by the initial fixed split between training and test set, so, the performance can be generalized varying also the test set. Doing so, they adopted two nested K-fold cross-validation procedures. The "outer" cross-validation selects the test set and calculate the accuracy of the specific classifier, as in Fig.4.1. Instead the "inner" one perform model selection using the training set identified by the "outer" cross-validation, as shown in Fig.4.2. In the end the performance result is obtained as the average of the accuracy performed on the test sets, as shown in Fig.4.1. The drawback of this approach is related to the fact that each train/test split (i.e., for each iteration of the "outer" crossvalidation step) the model with the highest accuracy changes in order to provide the highest accuracy on average on the train set, this means that they cannot provide a unique classifier.

4.2.1.1 Results of Traditional Machine Learning Approach

The model selection for ANN, provide as a result the hyperparameters presented in tables 4.1.

In the ANN models They obtained 5 hidden layers, all composed by 100 neurons; each neuron uses as activation function the Rectified Linear Unit function (RELU), that is mathematically defined as $f(x) = max(0, x)$.

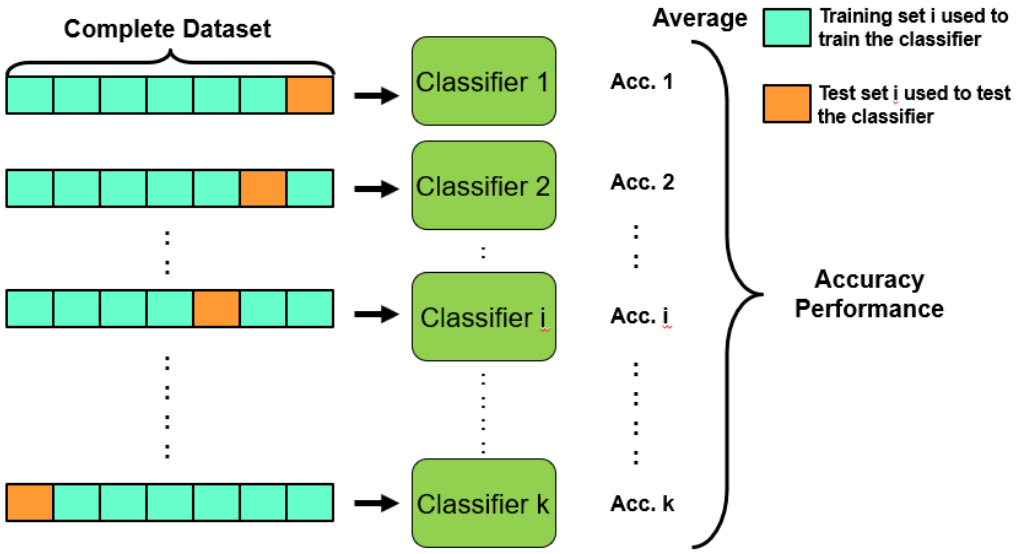


Figure 4.1: Outer K-fold crossvalidation for performance assessment.

Table 4.1: Hyperparameters of the model based on ANN

Parameter	Value
Number of Hidden Layers	5
Number of Nodes per Layer	100
Activation Function	Relu

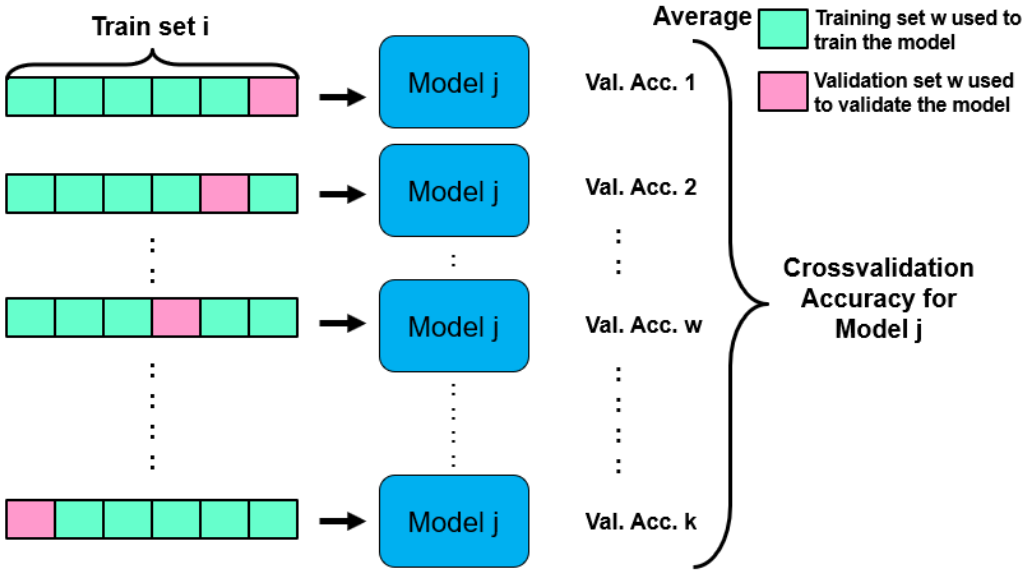


Figure 4.2: Inner K-fold crossvalidation for model selection.

At the end of the work [53] they obtained the accuracy for ANN 90.2%. This accuracy is obtained with hyperparameters in table 4.1 which we will use for our future work.

4.3 Strategies for Training Machine Learning Model in Distributed Datasets

Consider the Failure Identification issue, where several operators wish to train a model cooperatively. We have three options for resolving this issue. Classical machine learning, in particular, includes centralized data training, in which data is collected, and the entire training process is carried out on a single server. Distributed machine learning relies on basic assumptions for data training, whereas FL relies on more resilient assumptions. 4.3

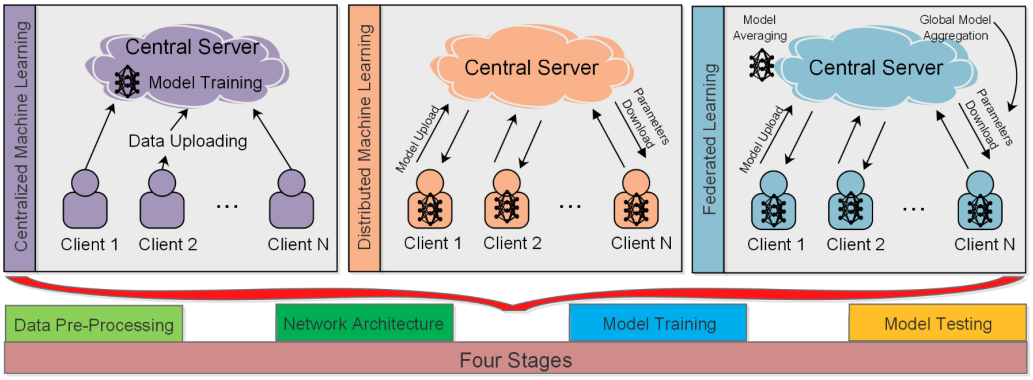


Figure 4.3: Model Classifier of Centralized, Distributed and FL [54]

4.3.1 Centralized Machine learning

Traditional data analysis strategies include sending compressed sensor data from all operators to a central server and conducting the data analysis operations [55]. However, the modern equipments of the network can generate hundreds of gigabytes of data in a single day, which means the data transfer and the storage of this data are virtually impossible [56]. Also, participants' data contain private information, which poses questions about the transmission and storage of these data on a central server.

On the one hand, centralized training is computationally efficient for participants since they are relieved of computing duties, demanding more resources. Instead, participants' private data is very vulnerable since the server might be malevolent or infiltrated by enemies. Meanwhile, uploading a large amount of data might cause communication overhead between participants and the server [54]. This approach was thoroughly discussed at section 4.2.1

4.3.2 Distributed Machine learning

While there are many different strategies to increase the processing power of a single machine for large-scale machine learning, there are reasons to prefer a scale-out design or combine the two approaches, as often seen in HPC (high performance computing). The first reason is the generally lower equipment cost, both in terms of initial investment and maintenance. The second reason is the resilience against failures because, when a single processor fails within an HPC application, the system can still continue operating by ini-

tiating a partial recovery (e.g., based on communication-driven checkpointing [57] or partial re-computation [58]). The third reason is the increase in aggregate I/O bandwidth compared to a single machine [59]. Training ML models is a highly data-intensive task and the ingestion of data can become a serious performance bottleneck. Since every node has a dedicated I/O subsystem, scaling out is an effective technique for reducing the impact of I/O on the workload performance by effectively parallelizing the reads and writes over multiple machines. A major challenge of scaling-out is that not all ML algorithms lend themselves to a distributed computing model which can thus only be used for algorithms that can achieve a high degree of parallelism [60]. These existing approaches make fundamental assumptions for the data training, which are much more robust in FL [33]. Below are the common assumptions made during the execution of traditional machine learning algorithms. [54]

1. Data on the participants are sampled as independent and identically distributed *(i.i.d)*. Whereas, FL assumes *non-i.i.d* as different users contain different types of data.

2. Data is evenly distributed among all the participants. This assumption is technically impossible, as the expected number and the actual number of participants are different in real-time scenarios. Therefore, FL divides the number of shards among the participated participants so that each participant can receive an equal amount of data.

3. Total number of participants is smaller than the available local training examples per participant. This assumption cannot be made in FL as FL is designed for large scale scenarios where the participants can be higher in number.

4.3.3 Federated Learning Solution

As described in the sections 4.3.2 and 4.3.1, traditional ways to solve our problem have certain drawbacks, such as privacy issues, computational challenges, and distribution issues. FL may be a suitable answer to these problems. FL is a newly introduced approach where only learning parameters of Artificial Neural Network (ANN) are required to be communicated between the central server and the participants [54]. At the same time, the

whole training process is executed collaboratively on the individual participants This technique reduces the amount of data transfer and minimizes the privacy concerns of individual's private information. Hence, the load on powerful central servers in traditional machine learning has been distributed among individual low-power participants such as mobile devices or vehicles or in the case of cross-silo operators or hospitals. Therefore, we can say that FL has the potential to challenge the dominant paradigm of distributed computation FL varies in many ways from traditional machine learning problems (e.g., distribution of data centres). While both approaches strive to optimize their learning goal, FL algorithms have to consider the reality that contact with edge devices occurs over unstable networks with limited upload bandwidth [47]. As the communication overhead in FL is more often than computation overhead, therefore minimizing the communication overhead is crucial. This communication overhead can be measured either by uploading the gradients or through the communication rounds between the central server and participants [24]. The performance in FL can define with the achieved classification accuracy after specific numbers of communication rounds [54].

Chapter 5

Federated-Learning-Assisted Failure Management in Microwave Networks

In this chapter, we detail the methodological approaches used in the use case described in Chapter 4. In the first section, we describe the data preprocessing process. In the second section, our evaluation approach for the *supervised failure identification using FL* case is proposed. In the third section, we describe the possible cases of having three network operators which want to train a machine learning model collaboratively without sharing their data and possibly with one or two of them lack some information, and our approach towards them.

5.1 Data Preprocessing

The data used in our work are obtained on an anonymized real microwave network and consisting of SIAE Microelettronica equipment, see Fig.5.1. As the nature of the collected data is not suitable to be handled directly by machine learning algorithms, we used the refined data that is provided in the work of [53]. Moreover, we performed some further preprocessing to make the data suitable for our use case..

	idlink	area	provider	ramo	data	ip a	ip b	protection	uas	es	ses	RxNominu	txMaxA	txminA	rxmaxA	rxminA	txMaxB	txminB	rxmaxB	rxminB	acmMax	isG828Ok	isRxPwrOk	isTxPwrOk	sn	rxRef	txRef	snA	snB	acmSnA	acmSnB	rxSnA	rxSnB	eqtype	acmLowerA	acmUpperMo	acmEnqin	freqband	bandwidtl
1	198	1	1	1	2018-02-21 00:00:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
2	198	1	1	1	2018-02-21 00:15:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
3	198	1	1	1	2018-02-21 00:30:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
4	198	1	1	1	2018-02-21 00:45:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
5	198	1	1	1	2018-02-21 01:00:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
6	198	1	1	1	2018-02-21 01:15:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
7	198	1	1	1	2018-02-21 01:30:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
8	198	1	1	1	2018-02-21 01:45:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
9	198	1	1	1	2018-02-21 02:00:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
10	198	1	1	1	2018-02-21 02:15:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
11	198	1	1	1	2018-02-21 02:30:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
12	198	1	1	1	2018-02-21 02:45:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-40	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
13	198	1	1	1	2018-02-21 03:00:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
14	198	1	1	1	2018-02-21 03:15:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-41	19	19	-38	-40	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
15	198	1	1	1	2018-02-21 03:30:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
16	198	1	1	1	2018-02-21 03:45:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
17	198	1	1	1	2018-02-21 04:00:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-41	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
18	198	1	1	1	2018-02-21 04:15:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-41	19	19	-38	-40	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
19	198	1	1	1	2018-02-21 04:30:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
20	198	1	1	1	2018-02-21 04:45:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
21	198	1	1	1	2018-02-21 05:00:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
22	198	1	1	1	2018-02-21 05:15:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
23	198	1	1	1	2018-02-21 05:30:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
24	198	1	1	1	2018-02-21 05:45:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
25	198	1	1	1	2018-02-21 06:00:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
26	198	1	1	1	2018-02-21 06:15:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-39	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
27	198	1	1	1	2018-02-21 06:30:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
28	198	1	1	1	2018-02-21 06:45:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
29	198	1	1	1	2018-02-21 07:00:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-41	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
30	198	1	1	1	2018-02-21 07:15:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-41	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
31	198	1	1	1	2018-02-21 07:30:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-38	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0
32	198	1	1	1	2018-02-21 07:45:00	10.200.152.236	10.200.152.238	1=0	0	0	0	-39	18	18	-40	-40	19	19	-38	-39	7	1	1	1	0	-39	17	null	null	null	null	null	null	19	acm-4QAM	acm-256QAM	1	18.0	28.0

Figure 5.1: Snapshot of the real raw data provided by the network management system.

5.1.1 Handling Incomplete Information

In most real-world settings, data is partial, i.e., some measurements might be missing owing to, for example, a disconnection between the equipment and the management system. As a result, appropriately handling missing data is critical to avoid severely affecting machine learning algorithms. Three distinct ways to solving the missing data problem are described in [61]:

- *Ignoring and Discarding Data:* In this method, we either discard the entire row containing the missing value, or we remove simply the attributes that have a large number of missing values and do not give meaningful information.

- *Parameter Estimation:* The parameters of a Maximum likelihood model are calculated using accurate data, and the model will be used to predict the probable missing values. An example of this can be: When we are asking people their salaries statistics showed that the people who have higher salaries are more probable to refuse to answer. So, our estimation for empty data about salaries would be in higher ranges.

- *Imputation:* This method fills the empty data with values calculated using statistical procedures (i.e., mean or median) or a fixed value determined by some previously established connection.

We use a mix of the preceding techniques to tackle the missing value problem in our database. Specifically, we opt to exclude feature acmMax (i.e., an attribute reflecting the lowest modulation attained in the 15 minutes measurements) due to its scarcity in our dataset.

The missing values phenomena can influence received and transmitted power measurements; however, we already knew that missing values imply a software or hardware fault unrelated to signal propagation, which causes unavailability occurrences. We examine the values that received and transmitted power features may reach in order to determine the best method to handle the missing information. We categorize transmitted and received power levels into three groups:

- *Normal*: the values provided by the measure are in the operational ranges, that are for transmitted power [-20, 28] and for received power [-99, -22].

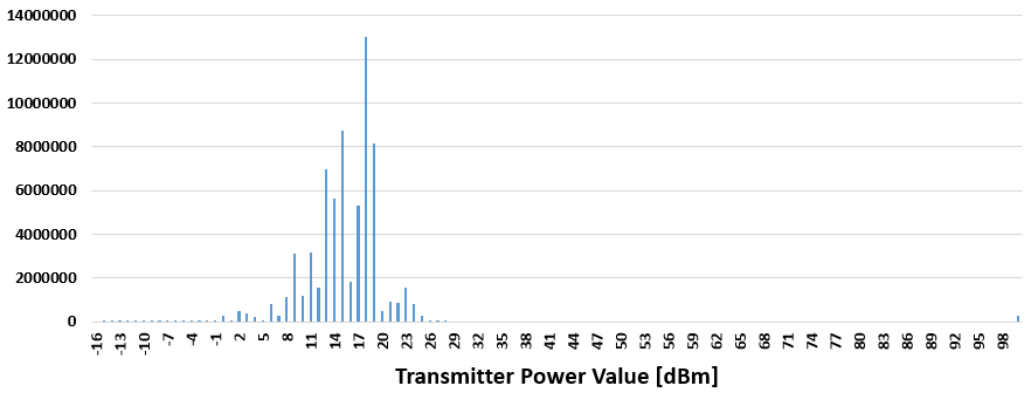


Figure 5.2: Distribution of the transmitted power values [53].

- *Anomalous*: the values of the features take one value that is not feasible and that indicates the presence of some problem. The anomalous values for transmitted power are {Null, -99}, and for the received power are {Null, -100, -1, 0≈[-2, -8]}. The Null value indicates a missing information, the one that we want to handle.

- *Out of range*: The values of the features are not in the operational range, but they are not anomalous values too.

The entries containing a *Null* value which are not suitable for ML algorithms were replaced with a value that corresponds to the presence of a failure. The exact value of the feature in this case was determined by investigating the distribution of the received and transmitted power values. As an example the normal range which is appear in the distribution for transmitted power is [-20,28] and for received power [-99, -22].

Figures 6.2 and 6.1, shown the distribution of the values associated to the transmitted and received power respectively. The two distribution, as explained in the *Normal values* definition, cover two different ranges of values. In machine learning, it is a good rule standardize the features, this means transform all features to have mean equal to zero and standard deviation equal to one. In order to do not eliminate the differences of the points, provided by their feature distribution, we set different values for the transmitted and received powers. These values will be out of the distribution depicted by the Figs. 6.2 and 6.1, in order to clearly identify the presence of a lack of information, but at the same moment the standard deviation of the

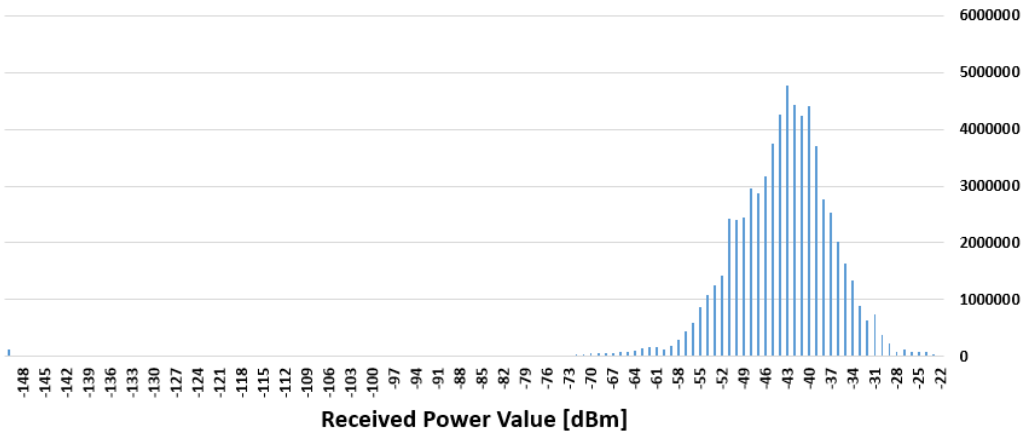


Figure 5.3: Distribution of the received power values [53].

features does not have to be too big, otherwise the feature standardization procedure will decrease too much the differences between the possible values.

5.1.2 Features Normalization

After the data preparation, the data samples are not yet suitable to be used directly in machine learning algorithms, as they must be normalized (or standardized). The data normalization is the process to transform features so that they all range in a 0-mean interval and the scale of the various features are comparable. The normalization procedure is useful when the attributes of the data have different scales or range of values for the following reasons:

- After feature normalization, all features contribute proportionally to the output of the machine learning algorithm, therefore in training phase the algorithm will be less sensitive to scale of data.
- The speed of optimization in machine learning algorithms (specially the ones that use gradient descent) is related to the scale of data.

In our case we achieve the standardization goal transforming the data to have mean value zero and variance one in all their features. This means that for the feature X_i we calculate its mean value $\bar{X}_i$, and its standard deviation σ_i,

to compute the standardize feature we use the formula

$$\frac{X_i - \bar{X}_i}{\sigma_i}$$

the formula can be described also in a vector space, where X is the point described by the features $[X_4 \div X_{43}]$, $\bar{X}$ is a vector in which each element is the mean value of the specific feature (i.e., $\bar{X}[i] = \bar{X}_i$), and σ is the standard deviation vector, in which each element i is the standard deviation of the i^{th} feature (i.e., $\sigma[i] = \sigma_i$). Then the standardized point can be described through the formula

$$\frac{X - \bar{X}}{\sigma}$$

5.2 Failure Identification Using Federated Learning

Now we focus on using FL algorithm for failure-root-cause identification in microwave networks. We use our labeled data to solve a multi-class classification problem, so this analysis compares centralized machine learning with FL and shows the benefit of FL over isolated data islands.
The goal of this chapter is to create a classifier that can detect failure root causes from an unobserved data point. This classifier does not use pooled data, and it is built only by sharing each client's local model. So, in order to attain this aim, we followed a number of actions. In the first section, we explain our model's training method using separated datasets; in the second section, we look for the optimal hyperparameters to utilize.

5.2.1 Training

To be able to identify the failure root-cause with distributed datasets, we need to be able to build a classifier that is trained collaboratively between several clients and is able to predict the failure root-cause when as input we give a data point with the mentioned features in the section 4.1.1.1. To acquire the final classifier, we went through several processes. Our approach is based on the FL algorithm called Federated Averaging, which was described in Algorithm 1. As the name implies, this method operates by averaging the weights provided by clients.

Assume we have many operators, each with a subset of the data described in section 4.1.1, these operators are referred to as clients. Also, we have a central server in charge of receiving and averaging the weights received from clients; this server is known as the coordinator, see fig.?? .

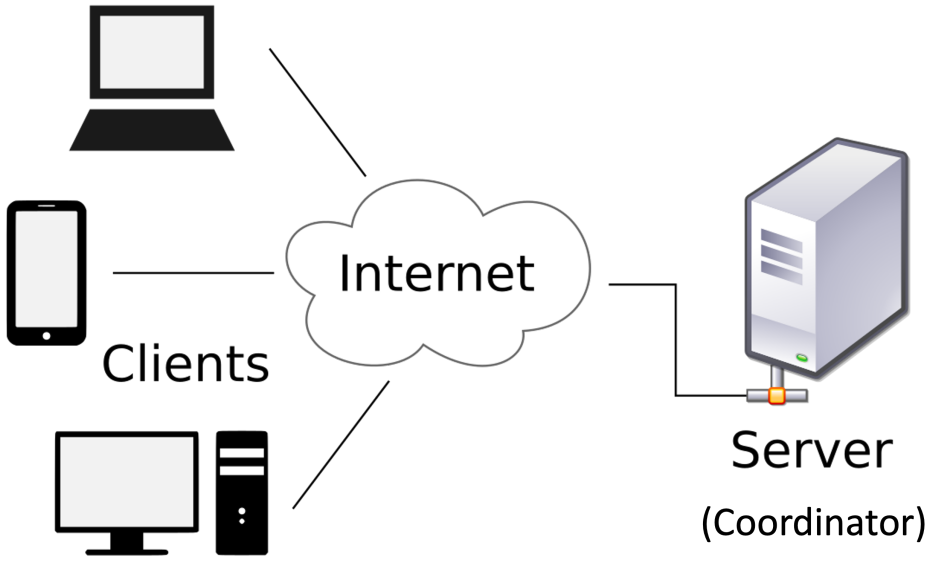


Figure 5.4: Client-Server(coordinator) architecture of FL

For federated model training in HFL systems(see section 3.2.1.1, the federated averaging (FedAvg) method was used in [25]. Also, in [62] a comparison was conducted, and results are shown in the table 5.2. as it is shown, the FedAvg algorithm has the best performance between all the other algorithms that have been shown in section 3.2.4. Also, in [54] they made a comparison on the MNIST dataset between distributed machine learning and centralized machine learning, which FL outperformed the other two, the result of their work in shown in the Fig. **??**. The figure compares the three algorithms of FedAVG, CO-OP, and FSRVG, compared to centralized machine learning. We can see that FedAvg is better than CO-OP and FSRVG. However, it can perform the same as a centralized case.

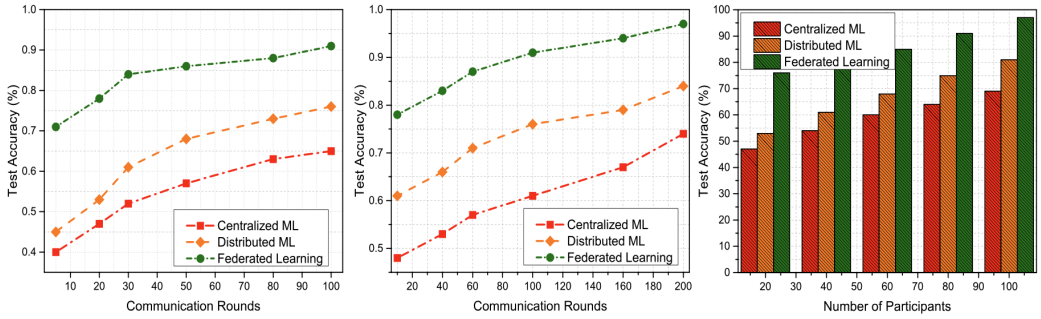
5.2.1.1 Federated Optimization

The optimization issue emerging from FL is referred to as federated optimization [25, 38]. In reality, federated optimization differs from traditional

Table 5.1: Convergence comparison on MNIST dataset in three different scenarios: a) 100 communication rounds, b) 200 communication rounds, c) various number of participants [54]

	FedAvg	CO-OP	FSRVG	Centr
FedAvg	×	+	+	=
CO-OP	−	×	=	−
FSVRG	−	=	×	−

Table 5.2: Summary of algorithm comparisons, showing if the algorithm in a row is better (+) , worse (−) , or practically equivalent (=) compared to the algorithm in a column [62]



(a) Scenario one (b) Scenario two (c) Scenario three

distributed optimization problems in numerous ways [25, 39, 40].

- **Datasets with Non-Independent Identical Distributions (Non-IID).** It is possible to guarantee that various compute nodes have IID datasets for distributed optimization inside a data center, such that all local updates appear pretty similar. IID datasets cannot be ensured in federated optimization. We cannot use IID assumptions for decentralized datasets in FL because the data held by various participants may have entirely different distributions [41]. While similar participants may have similar local training data, two randomly selected individuals may yield substantially different model weight updates or gradient updates.

- **Unbalanced number of data points.** It is feasible to split the data evenly across the computing nodes in a data center for distributed opti-

mization. However, in practical settings, different participants typically have vastly varying quantities of training datasets [38, 42, 43]. Some individuals, for example, may have a few data points, while others may have a significant amount of data.

- **Huge number of participants.** The number of parallel computing nodes inside a data center may be readily adjusted for distributed optimization. However, because ML or DL typically requires a large amount of data, FL applications may need to engage many users, particularly with mobile devices [44]. In our work we investigated the difference in the sense of accuracy between ML and FL. Every one of these participants has the potential to engage in FL, making it considerably more dispersed than that found in a data center.

- **Slow and unreliable communication links.** In a data center, it is anticipated that nodes interact swiftly with one another and that packets are nearly never lost. However, with FL, communication between clients and the server is based on existing Internet connections. Uploads (from client to server, for example) are significantly slower than downloads, especially if the connection is made from a mobile terminal. Some clients may also experience brief Internet connectivity outages [28]. We considered in our investigation that the connections might not be available. In this instance, the client on the not-available list will be replaced by a new client, and the server will compute the average of just received data.

SGD and its variations (for example, mini-batch gradient descent) are the most often used DL optimization methods [10]. Many advancements in DL may be seen as modifying the structure of the model (and therefore the loss function) to make it more accessible to optimization using basic gradient-based approaches [10, 45]]. Given the ubiquitous use of DL, it is only logical to create new federated optimization methods based on SGD [[25].

SGD may be used naively in federated optimization by performing a single mini-batch gradient computation (e.g., on a randomly selected subset of participants) during each round of federated training. The processes of transmitting updates from the participants to the server and the server back to the participants, i.e., Steps 1–4 of Figure **??**, are referred to as "one round" in this context. This approach is computationally efficient, but it necessitates

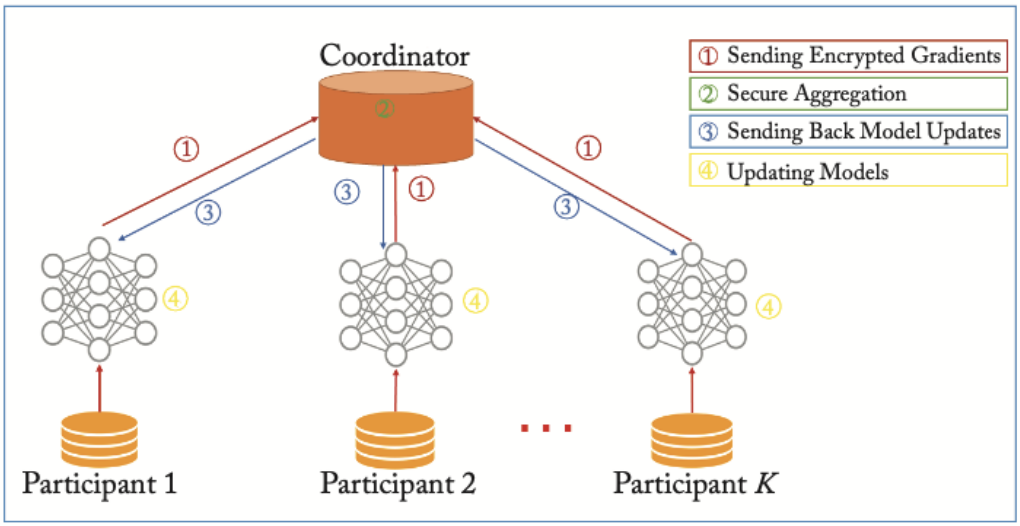


Figure 5.5: Exemplary client-server architecture for an HFL system [24]

many communication rounds of training to produce satisfactory models; for example, even using an advanced approach like batch normalization (BN) [63] training on the MNIST dataset necessitates 50,000 rounds mini-batches of size 60 [25].

Communication costs are generally low for DML with parallel training within data centers or computer clusters, and computational costs dominate. Recent techniques emphasize the use of graphics processing units (GPUs) to reduce these expenses. In contrast, communication costs dominate in FL since communication occurs via the Internet or wide area networks (WANs), including wireless and mobile networks.

A single on-site dataset in FL is often tiny in comparison to the entire dataset size, and current terminals (such as network equipment) have reasonably powerful processors. As a result, for many model types in FL, compute costs are minimal when compared to communication expenses. As a result, we may employ more computing to reduce the number of communication rounds required to train a model. The following are the two major methods for include computation [25].

- **Increased parallelism.** We can engage more participants working independently in be- tween client-server communication rounds.
- **Increased computation on each participant.** Rather than performing a simple computation like a gradient calculation, each client

performs a more complex calculation in between communication rounds, such as performing multiple model weight update over a training epoch.

The optimization process is fully explained in the section 3.2.3

5.2.1.2 The FedAvg Algorithm

The FedAvg algorithm family, as described by [25], allows us to add computation using both techniques discussed above. Three key parameters govern the amount of computation: (1) rho, the fraction of clients that perform computation during each round; (2) S, the number of training steps each client performs over its local dataset during each round (i.e., the number of local epochs); and (3) M, the mini-batch size used for client updates. $M = \infty$ denotes that the whole local dataset is handled as a single mini-batch.

We can use $M = \infty$ and $S = \infty$ to generate SGD with different mini-batch sizes. During each round, this method picks a ρ-fraction of participants and computes the gradient and loss function across all of the data possessed by these players. As a result, ρ controls the *global* batch size in this method, with $\rho = \infty$ equivalent to full-batch gradient descent using all data stored by all participants. We refer to this basic baseline technique as FederatedSGD since we still pick batches by using all of the data on the chosen participants. While the batch selection method differs from picking a batch by randomly selecting individual samples, the batch gradients g calculated by the FederatedSGD algorithm still fulfill $\mathbb{E}[g] = \nabla f(w)$, given that the datasets owned by various participants are IID [24].

It is usually believed that the coordinator or server knows the initial ML model and that the participants are aware of the optimizer parameters. In a typical implementation of distributed gradient descent with a fixed learning rate, the kth participant computes $g_k = \nabla F_k(w_t)$, the average gradient on its local data points at the current model weight w_t, and the coordinator aggregates these gradients and applies the model weight update described by [25].

Alternatively, the coordinator can transmit the averaged gradients $\overline{g}_t \sum_{k=1}^{K} \frac{n_k}{n} g_k$ back to the participants, who will then compute the updated model weights w_{t+1} using Equation line 20 of Algorithm 4. Gradient averaging is the name given to this approach [42].

That is, each client performs one (or more) steps of gradient descent on the current model weights $\bar{w}_t$ using its local data, and then transmits the

Algorithm 4 The FedAvg Algorithm (adapted from McMahan et al. [2016b])

The coordinator executes:
Initializes model weight w_0, and broadcasts the initial model weight w_0 to all participants.

for each global model update round $t = 1, 2, \ldots$ **do**
The coordinator determines $\mathcal{C}_t$, which is the set of randomly selected $\max(K\rho, 1)$ participants.
for each participant $k \in \mathcal{C}_t$ **in parallel do**
Updates model weight locally: $w_{t+1}^k \leftarrow$ **Participant Update** $(k, \overline{w}_t)$.
Sends the updated model weight w_{t+1}^k to the coordinator.
end for
The coordinator aggregates the received model weights, i.e., taking the weighted average of the received model weights: $\overline{w}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
The coordinator checks whether the model weights converges. If yes, then the coordinator signals the participants to stop.
The coordinator broadcasts the aggregated model weights $\overline{w}_{t+1}$ to all the participants.
end for

Participant Update $(k, \overline{w}_t)$:
(This is executed by participant k, $\forall k = 1, 2, ..., K$.)
Obtain the latest model weight from the server, i.e., set $w_{1,1}^k = \overline{w}_t$
for each local epoch i from 1 to number of epochs S **do**
batches $\leftarrow$ randomly divides dataset $\mathcal{D}_k$ into batches of size M.
Obtain the local model weight from last epoch, i.e., set $w_{1,i}^k = w_{B,i-1}^k$
for batch index b from 1 to number of batches $B = \frac{n_k}{M}$ **do**
Computes the batch gradient g_k^b.
Updates model weights locally: $w_{b+1,i}^k \leftarrow w_{b,i}^k - \eta g_k^b$.
end for
end for
Obtains the local model weight update $w_{t+1}^k = w_{B,S}^k$, and sends it to the coordinator.

Parameter	Value
Number of Hidden Layers	5
Number of Nodes per Layer	100
Activation Function	Relu

Figure 5.6: Hyperparameters of the model based on ANN

locally updated model weights w_{t+1}^k to the server. The server then computes a weighted average of the resulting models using Algorithm 4 line 9, and returns the aggregated model weights $\bar{w}_{t+1}$ to the client. *Model averaging* is the term used to describe this approach [25].

Algorithm 4 summarizes the model averaging variation of the FedAvg algorithm. After writing the algorithm in this manner, it is reasonable to wonder what happens when the participant iterates the local update many times before proceeding to the averaging phase. The number of local updates each round for a participant with n_k local data points is provided by $u_k = \frac{n_k}{M} S$. Algorithm 4 contains the entire pseudo-code for the FedAvg algorithm with model averaging.

5.2.2 Model and Hyperparameters Search

As stated before, it is assumed that all the clients know the ML algorithm that has been chosen. We decided to choose Artificial Neural Network(ANN) as our algorithm. The parameters are taken from [53] so we created an ANN with the parameter that are shown in the table 5.6.

This model is supplied to all clients with all of the hyperparameters preselected. We must also initialize our weights. These weights will be transmitted to all selected clients in each round after that. Because of privacy concerns and the possibility of malevolent coordinators and other privacy attacks, the server should typically have no knowledge of the data in each client. It is difficult to select an appropriate initial weight without prior knowledge. As a result, the optimum choice is to assume that all of the weights of the specified ANN architecture are zero. So, the server will broadcast 0 as the initial weights to the selected clients.

5.2.2.1 Convergence Condition

An iterative algorithm is said to converge when as the iterations proceed the output gets closer and closer to a specific value. In some circumstances, an algorithm will diverge; its output will undergo larger and larger oscillations, never approaching a useful result [64].

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point, because this is the direction of steepest descent. Conversely, stepping in the direction of the gradient will lead to a local maximum of that function. The procedure is then known as gradient descent.

In fact, getting to this maximum value takes a long time, hence a convergence condition is required. The fluctuation of the gradient results is minimized when we get close to the minimal point. So we may claim the function has converged if the gradient does not change or only changes for a limited amount after multiple iterations.

We experimented with several convergence conditions, which are specified by the percentage of change of the loss function and the number of iterations that it should stay within the change percentage defined, after which the convergence time was measured. The change percentage 0.1 with 20 rounds provided a fair compromise between the accuracy achieved and the time spent, as indicated in the 5.3 [1]. It's worth noting that after 1000 cycles of learning, all of these prerequisites remain valid.

Table 5.3: Convergence Time in minuets with different conditions.

%	NO. rounds	duration(min)
0.1%	50	∞
1%	50	∞
0.1%	30	42
1%	30	35
0.1%	20	28
1%	20	20

[1]These measurements has been taken with apple MacBook with **Memory** 8 GB 2133 MHz LPDDR3 and **Processor** 1.4 GHz Quad-Core Intel Core i5

Hyperparameter Search

This section shows the hyperparameter search that we conducted. The hayperparameters that we had to tune were (1) rho, the fraction of clients that perform computation during each round; (2) S, the number of training steps each client performs over its local dataset during each round (i.e., the number of local epochs); and (3) M, the mini-batch size used for client updates, Also, we could tune the learning rate.

To begin, we partitioned the data into three sub-datasets at random. There was a hard constraint for all of them: no link which is indicated in the data set with the *IDLink* should be repeated in any two sub-datasets. For speedier results, we relaxed the number of rounds requirement above. After 500 cycles of learning, the same conditions apply.

We intended to adjust the number of training steps each client takes (local epochs), S, as well as the mini-batch size used for client updates, as indicated by the letter M. Weighted accuracy is the point of comparison, with the number of points in each customer. Also, this technique was repeated ten times for **all** the combinations of S and M according to below numbers, and the results were averaged.

$$S = [10, 100, 200, 500] and M = [32, 64, 128]$$

The best ones were $S = 500$ and $M = 64$. To speed up the training process, we need to fine-tune the learning rate *eta*, which was set between $[0.1, 0.001, 0.00001, 0.0000001]$. The trick is to train a network by starting with a low learning rate and gradually increasing it for each batch. To better understand the outcome, we needed to plot the learning curve for each of them. The test and train errors make up the learning curve. The examination of train-test mistakes allows for the detection of potential issues. [65]:

- **high bias:** Training error is close to test error but they are both higher than expected
- **high variance:** Training error is smaller than expected and it slowly approaches the test error

After elimination of the ones with the problem the best learning rate turned out to be 0.00001. For the relatively small number of clients, ρ is one, so we can select all the clients. At the end in the hyperparameters of table

5.4 with the initialize weights of our model will be bradcasted at each round to our clients.

Table 5.4: Hyperparameters for the FedAvg Algorithm

hyperparameter	value
ρ	1
S	500
M	64
η	0.00001

Chapter 6

Numerical Results

Now we focus on evaluating our proposed frameworks. The evaluation is executed using the performance metrics presented in Chapter 3.
Initially, we go deep into the dataset analysis after performing the data preparation procedures presented in Chapter 5. Then, we provide details of how we created real-life use cases and how we evaluated the performance of our FL algorithm. In the end, we will we discuss numerical results.

6.1 Data analysis and presentation

As previously stated, our raw data is collected every 15 minutes. We construct 45-minute frames in order to capture the fluctuation of radio power measurements across time. These windows are defined by 43 features, including six features evaluating the *Es* and *Ses* performances (Chap.3.3.4), twelve features presenting the maximum and minimum transmitted power on both sites, and twelve features relating to the received power. Four features define different power thresholds related to the link's design, and one binary feature indicates whether the AC is operational. Also, eight of them that represent the design information are discarded.

Human specialists manually labeled the windows, assigning a label to each window based on the behavior of the radio power measurements over 45 minutes. The labels are applied to the 45-minute time intervals and specify the reason for the failure event within that period. The indicated windows are two, and their class occurrences are displayed in Table 6.1, indicating

Table 6.1: Class occurrences in the labeled data.

Failure Cause	Occurrences
Deep Fading	284
Extra Attenuation	581
Interference	49
Low Margin	190
Self-Interference	187
Hardware Failure	1222

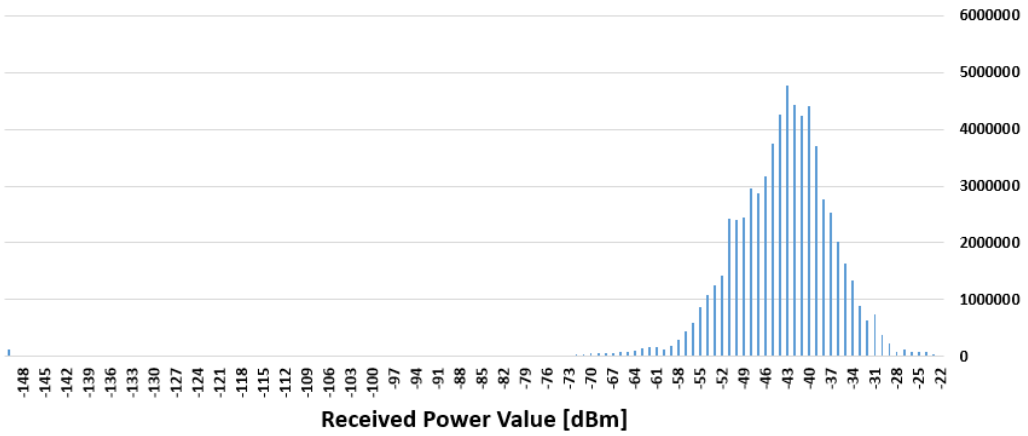


Figure 6.1: Distribution of the values for the received power.

a significant imbalance between several classes. This is due to the rarity of some events (e.g., Interference events), as well as the fact that a large number of occurrences, classified as "Hardware Failure", have no apparent behavior that can be safely ascribed to one of the "well-known" root-causes.

The missing values problem in the power measurement is solved by imputation. The missing values in received power attributes are replaced with the value -150 dBm because the distribution of the values is entirely in the negative part of the axis, as shown in Fig. 6.1. For the transmitted power attributes, missing values are replaced with the value 100 dBm; the choice is driven by the fact that the distribution of the points is mainly in the positive part of the axis, as shown in Fig. 6.2.

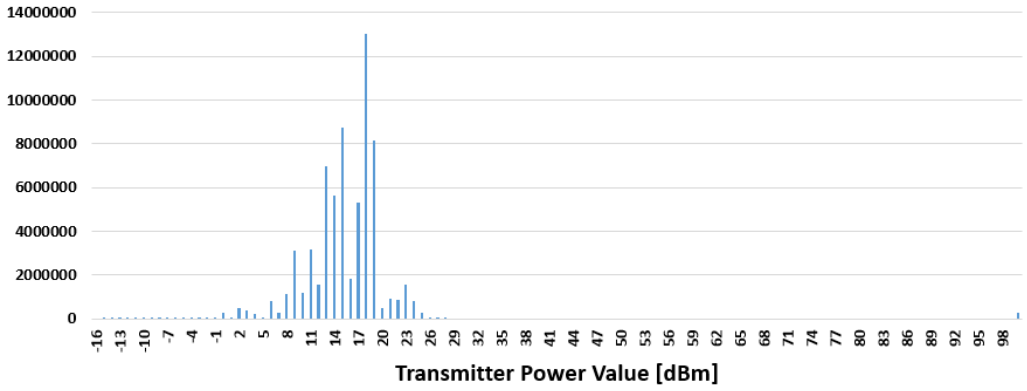


Figure 6.2: Distribution of the values for the transmitted power.

6.2 Data Separation

As previously stated, the goal of our FL use case is to solve the classification problem where the classifier is able to distinguish a newly unobserved given point with aforementioned features and classify it in one of the 6 failure root-causes, of a certain number of participants (clients) without sharing any data and just sharing the model characteristics. We decided to create our clients in a way to replicate this behavior. As discussed in sections 3.2.2, there are two forms of device distribution for FL: cross-silo and cross-device. In cross-device, we require hundreds of devices, which necessitates more data points than we currently have. To restate some of the challenges, we are working at scale, potentially needing many millions of devices for federation to work. Devices may be offline, we need to be careful about when we consume compute and impact user experience. In the case of cross-silo, we may have as many devices as we like as long as the data in them is large enough. The goal is the same as cross-device, to update and improve a central, and in this case shared model, there are arguably greater challenges on the security side. At the same time, there's scope to use more consistent, powerful and scalable compute within each organisation.

We began with three participants for this purpose. We cannot do less than three since there will be no cross-silo case for FL. We established a strict limitation for our separation in this case: no two datasets can have the same link. This restriction exists because three distinct operators who wish to participate in the FL classification to categorize freshly incoming data

points cannot share a link in their equipment. So, in order to simulate this real-world behavior, we imposed this tight limitation on our separation. We chose three sub-datasets totally at random for separation, but we nevertheless adhered to our strict constraint.

We used a systematic approach to handle training datasets and clients for these separations. We want to examine the impact of missing labels on each client and how FL may enhance or degrade accuracy whether the clients were trained alone or if we put the time and space to collect the data in one location and violated the privacy of the operators (we call the former ***Isolated knowledge*** and the latter ***Complete knowledge***).

To be fair to all clients, we attempted to have around the same number of data points in each of them. in this case we can eliminate a portion of space state on the available cases. Since we have the rigid constraint of link ID, we could not have precisely the same number of points. To provide a fair comparison, we also ensured each customer had at least 5 data points for each failure cause. We removed 20% of these datasets to have 20% of data points from each class since we want to analyze the effect of the missing label in FL for testing and maintain the remaining 80% for training.

To that aim, we should remove the labels one by one to observe what happens. If we remove all of the labels from three clients one by one and do all of the conceivable combinations, we will obtain 262144. There are six labels in such a manner that each client can have a label or not have one. Thus $26 = 64$ modes exist for a client, we have three clients, and we can have any combination of three clients. As is clear, the state space is too vast to attempt everything. As a result, we need to condense our states into a smaller space.

Table 6.1 shows the number of data points for each label. Our analysis focuses on the domain of repetition of each class, as indicated in the table above 6.1, we may categorize our labels into three sub-categories: *least repeated, medium repeated, most repeated*, see Table 6.2.

We want to select three labels, one label from each category that mentioned in Table 6.2, from all 6 available labels. Later, we will remove in a systematic manner all the points that are associated to the selected label. To include in our experiment, we require a label from the moderately repeated ones, as a result, of the four medium repeated labels (Deep Fading, Extra Attenuation, Low Margin, Self-Interference, summing-up *Deep Fading* will be adequate for our purposes. Furthermore, we have one label with a few

Table 6.2: Categories of label distribution of dataset.

label N. in dataset	label	Cat. of repetition
0	Deep Fading	Medium
1	Extra Attenuation	Medium
2	Interference	Least
3	Low Margin	Medium
4	Self-Interference	Medium
5	Hardware Failure	Most

repetitions (*Interference*) and one with a high number of repetitions (*Hardware Failure*); both are unique, and no other labels have nearly the same amount of repetition.

Table 6.3: State space of removing labels for one client.

case	removed label
1	0
2	2
3	5
4	0,2
5	0,5
6	2,5
7	0,2,5
8	None

In the end, we chose three labels from a total of five. These labels will be removed in a systematic way from the clients. The process is as follows: we have three clients in each one, we can remove three labels, 0, 2, 5. So each client can be constructed in 2^3 ways. All the ways are shown in the table 6.3. Each client can have eight modes, and we can have **any** combination of these three. So we can have 8^3 modes which equal to 512.
In real-world settings, there is always one operator who has seen all the failure causes due to it's dimension. This implies that we have an operator working for a long time and have seen all potential failure causes, so there is no need to remove label from this client and the labels inside should always be fixed. We make one of our clients fixed operator, so it has to have all the labels at all times. With that in mind, the overall number of instances would be reduced. We now only have two clients that can miss labels, and they are

our benefit participants. They can each have eight states, as shown in the table 6.3. We can also have **any** combination of these two. As a result, we shall limit our state-space to 64 modes.

Now, we want to extract only the one-of-a-kind instances. Because we distributed our clients randomly and equitably, there is minimal variation in the number of points and failure causes amongst our three clients. Consider the scenario when client 2 lacks the label "0" in its training set, while client 1, as previously stated, has all the labels, and we suppose that client 3 can likewise have all the labels. Also, another case is similar to the last one, except that client 3 is the missing label "0" while client 2 has all of the labels in its training set. Cases like these are referred to be recurrent cases. We approach the topic as if it were a statistical problem.

Now, assume we have two baskets and eight different colors. We also offer 16 balls in eight different colors. As a result, every two balls are the same color. We want to choose two balls such that we may select one ball from each basket, and there should be no duplicate occurrences. We answer this issue in statistics by first computing all of the potential combinations, which are 64, as previously mentioned; secondly, we must eliminate the duplicates; to remove the duplicates, we must first determine how many there are. To do so, we declare we wish to pick two colors from a set of eight. (This implies we picked four of them with the same combination two by two.)

If we extend this simple problem of statistics to our case we can say that balls indicated the combination in the table 6.3, and the baskets are clients. So in conclusion we can have:

$$64 - \binom{8}{2} = 64 - 28 = 36$$

All these cases are shown in the table 6.4.

6.3 Failure Identification

The FedAvg algorithm's objective is to train a model collaboratively to distinguish between the six classes linked to the failure cause, as stated in chapter 3.3.5, without exchanging data amongst participants.

To further understand this, we trained our model with training data of each client using three distinct methods. The first is to train a client inde-

Table 6.4: Space state for 3 clients. the showed numbers are missing labels.

N.O. case	client 1	client 2	client 3
1	None	0	0
2	None	0	2
3	None	0	5
4	None	0	2 and 5
5	None	0	0 and 2
6	None	0	0 and 5
7	None	2	2
8	None	2	5
9	None	2	2 and 5
10	None	2	0 and 2
11	None	2	0 and 5
12	None	5	5
13	None	5	2 and 5
14	None	5	0 and 2
15	None	5	0 and 5
16	None	0 and 2	0 and 2
17	None	0 and 2	0 and 5
18	None	0 and 2	2 and 5
19	None	0 and 5	0 and 5
20	None	0 and 5	2 and 5
21	None	2 and 5	2 and 5
22	None	None	None
23	None	None	0
24	None	None	2
25	None	None	5
26	None	None	0 and 2
27	None	None	0 and 5
28	None	None	2 and 5
29	None	0 and 2 and 5	None
30	None	0 and 2 and 5	0
31	None	0 and 2 and 5	2
32	None	0 and 2 and 5	5
33	None	0 and 2 and 5	0 and 2
34	None	0 and 2 and 5	0 and 5
35	None	0 and 2 and 5	2 and 5
36	None	0 and 2 and 5	0 and 2 and 5

pendently with its training data set and then test it with the separated test set described in 6.2 we call this isolated model. It is important to note that we might have a label in our test set because the test set was split before removing the labels but not in our train set. This method will demonstrate the impact of FL more effectively. The second is to train a model collaboratively using FedAvg (Algorithm 4) and test it independently on each client's train set. We refer to this model FedAvg. The next step is to collect all of the data from all clients and train them in a centralized machine learning model, which we call the centralized model.

We evaluate the performances of three different approaches to solve the problem as mentioned earlier, to identify which one is the most suitable for classifying a new point (i.e., a 45-minutes window within UAS in the last 15-minutes slot) in different use cases.

The three models are as follow:

- *Isolated model:* This model trains one client with only its local data. The separated test data will be tested on the model that is obtain for each client using only its local data.

- *FedAvg model:* We will train this model collaboratively between all three clients. The separated test data of each client will be tested separately on the model that has been obtained using FedAvg between three clients, without sharing any data and just by sharing the model characteristics.

- *Centralized model:* We will train this model on the pooled dataset of three clients. The separated test data of each client will be tested separately on the model that has been obtained using the pooled data.

For all three techniques, we used the hyperparameters as in tables 5.6 and 5.4 for our ANN algorithm, which will be broadcasted to all clients before starting the training procedure. In addition, we maintain the train and test sets the same for all methods. Furthermore, these techniques employ an Artificial Neural Network as the foundation algorithm and SGD for optimization. We decided to fine-tune parameters to maximize the performance of the algorithms, so we took case 22 in the table 6.4 and ran the FedAvg ten times with a combination of hyperparameters to ensure we had the best values; this procedure was thoroughly explained in the section 5.2.2.1.

Our instances may be divided into three sub-categories. The grouping can occur in a wide range of variables, but because we are evaluating the influence of label repetition on the outcome of FL, the category based on the different forms of missing labels appeared appropriate, and we can arrange our instances as shown in the figure 6.3

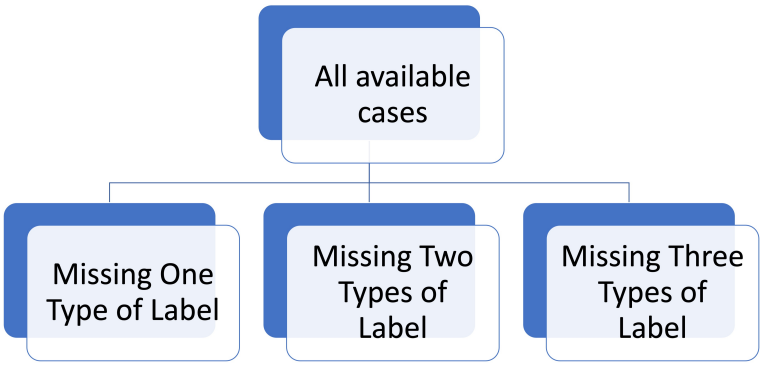


Figure 6.3: Resulting categories of FL. left: *scenario 1*, middle: *scenario 2* and right: *scenario 3*

So we can break the table of 6.4 into three tables. Moreover, analyze them one by one. we start with scenario 1 of figure 6.3.

6.3.1 Scenario 1: where only one type of label is missing

Scenario 1: Investigate the case where one or two clients miss only one type of label.

Objective: To understand in which cases a collaboration is profitable.

we try to investigate the case where one or two clients miss only one type of label. This helps us understand if it is preferable for two clients to perform FL or just train them separately or agree on some type of data aggregation of centralized training if two clients are lacking in their data set the same failure cause.

Table 6.5 shows all the cases where only one type of label is missing. This label can be missing in one or two clients, and we have a strong client with all the labels included. We compared the accuracy of isolated machine learning, FL, and centralized machine learning techniques. We report the results in the Fig. 6.4.

As shown in Fig. 6.4 in case number 24(the fifth set of columns from the bottom), we see that for client 3, the FL degraded the accuracy by 2%. The

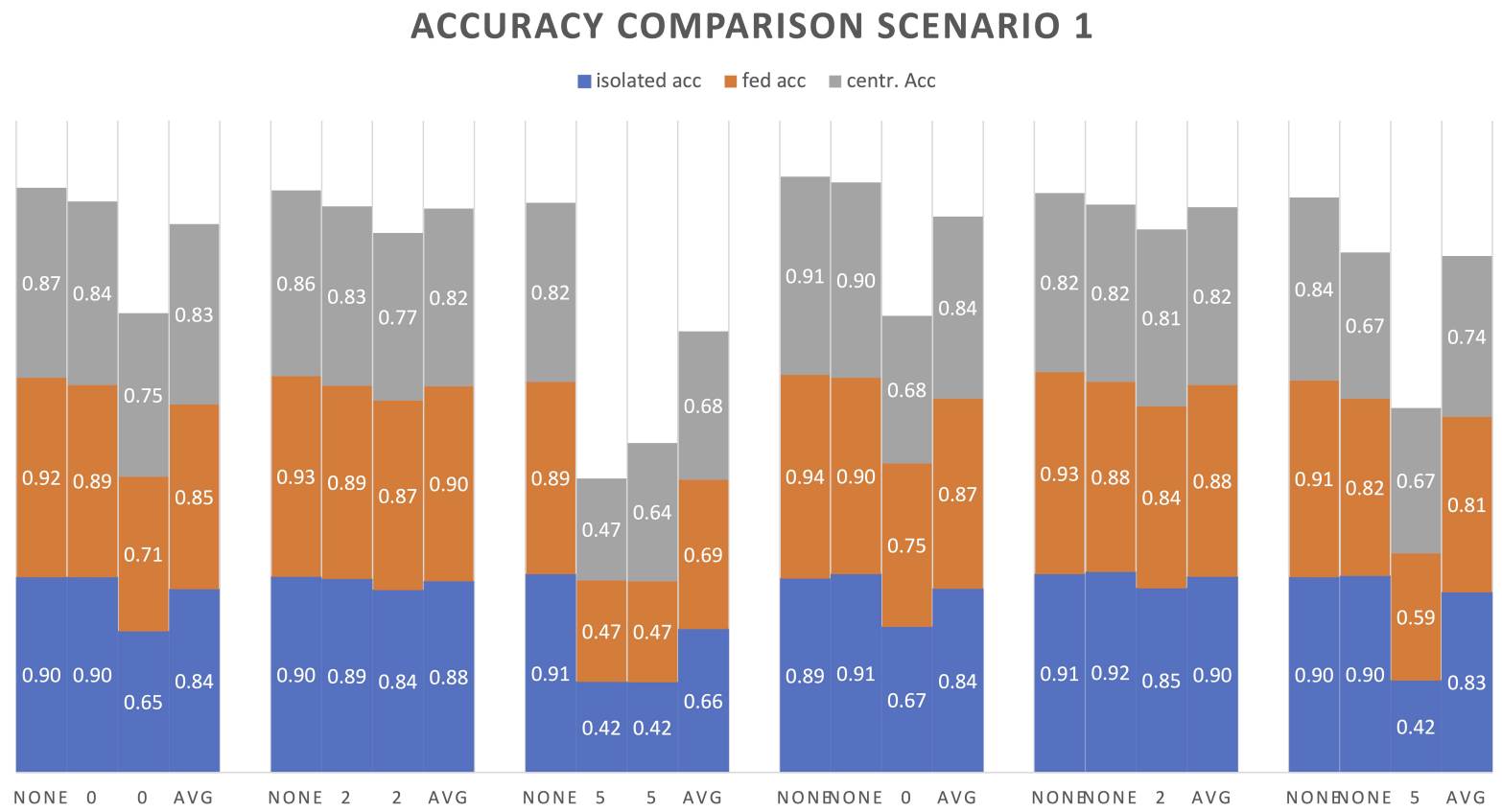


Figure 6.4: Comparison of accuracy for scenario 1. each set of three columns displaying one case corresponding to the table 6.5 (groups from right to left in the figure match with the cases in the table from up to down). Furthermore, each column represents a client, and the numbers under each column indicate the client's missing label. Each column displays isolated accuracy, FedAvg accuracy, and centralized accuracy for one client.

Table 6.5: Scenario 1 where only type of label is missing in each case. The numbers correspond to the numbers of table of 6.4.

N.O. case	client 1	client 2	client 3
1	None	0	0
7	None	2	2
12	None	5	5
23	None	None	0
24	None	None	2
25	None	None	5

reason for this could be the extra information that is coming from another client. The model of client 3 is focused on understanding better other labels rather than label 2. When we mix new knowledge with the model, this focus is also now in label 2, and since we only have one data point in the test case of client 3 with label 2, this effect will show itself by reducing the accuracy of other labels. In case number 23 (the fourth set of columns from the bottom), we can see that for client 3, the accuracy improved by 8% compared to the isolated model. Moreover, we see 3% improvement in comparison with the centralized model. In case 25 (the first set of columns from the top), we can see a much better improvement in the accuracy of the client 3 models. FedAvg is better than isolated but 17%. This significant difference is because of many data points with label 5. Since label 5 is the most repeated, we expect to have more data points with label 5 also in the test set.

In case number 7 (the second set of columns from the bottom), we can see no apparent difference between the isolated case and the FedAvg model client 2 has the same 89% accuracy for both, and client 3 has the accuracy 84% for the former and 87% for the latter. However, on average, we see a 2% improvement for the FedAvg according to the centralized model. In conclusion, we have the best improvement compared to the isolated case when the most repeated label is missing (label 5) in one client but not in two. This difference can be interpreted as follows: we do not have enough data to support two clients even though the repetition is high in our main client. Also, when the least repeated label is missing in both situations, we do not observe much of an improvement. However, this slight improvement in the case where the least repeated one is missing can show that now all the clients can recognize and classify the missing failure cause without seeing it in their database.

Now we will look at Scenario 2. We may further divide scenario two into three sub-scenarios, which are depicted in Fig. 6.5.

6.3.2 Scenario 2: where two types are labeled is missing

Scenario 2: Investigate the case where one or two clients miss two types of label.

Objective: To understand in which cases a collaboration is profitable. We can claim that any combination of missing two kinds of labels in two clients can impact the final judgments of the firms to develop a more robust model since we simulated the instances where two types of labels between the categories of (high, medium, and low repetition are absent) in this section.

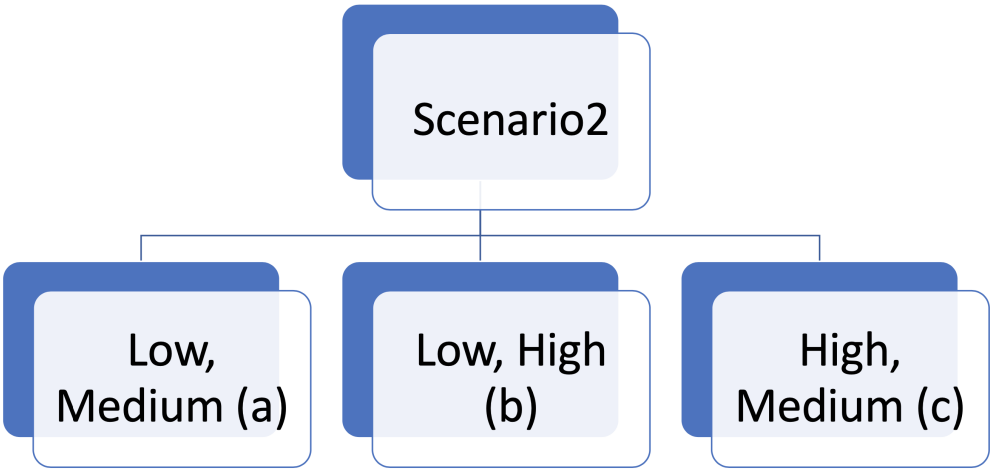


Figure 6.5: sub-division of scenario 2. Each sub-division shows a combination of two types of label that can be missing from one or two clients at the same time. The letters indicated near, are the reference inside the text

First, we discuss case "a" shown in the table 6.6, where only label types medium (0) and low (2) are missing from one or two clients. We start analyzing the results in Fig. 6.6, case 2 (the first set of columns from bottom). It shows an improvement of 7% in FedAvg compared to the centralized model for client 2 and 4% in client 3 of FedAvg, compared to the centralized model. Also, we can see the overall average is improved for FedAvg according to the centralized and isolated models. This behavior depends on the fact that we have two clients that have focused on their isolated training on two different sets of five labels. When we aggregate these focused models, we can see an improvement in the FedAvg of 7% percent in average compared to centralized. Now we move on the case 5 (the second set of columns from the

Table 6.6: sub-division of scenario 2, case "a", where only label types medium (0) and low (2) are missing from one or two clients. The numbers correspond to the numbers of table of 6.4.

N.O. case	client 1	client 2	client 3
2	None	0	2
5	None	0	0 and 2
10	None	2	0 and 2
16	None	0 and 2	0 and 2
26	None	None	0 and 2

bottom); the picture shows the for client 2 we have FedAvg as 90% whereas the centralized is 85%, also for the third client, we can see FedAvgt working better with 3% different compared to centralized the same reasoning as the previous case can be applied here.

For the third set of columns (case 10), the FedAvg algorithm decreased the accuracy by 3% according to isolated accuracy as was expected and explained in case 24 of the first scenario. The third client shows the exact behavior of the previous case. In case 16 (second set of columns from the top), we can see that FedAvg has an accuracy of 74%, whereas centralized has 75% and isolated has 73% which are relatively close. For the second client, we can see that FedAvg improved the accuracy by 10%. This phenomenon is the number of zeros in the test case of client 3, which are relatively higher than in client 2. Moreover, for the last case (first set of columns from the top), we can see that FedAvg has an accuracy of 78%, whereas the isolated model has 69% and centralized has 79% of accuracy. The difference between the FedAvg and isolated is 9%, which can be considered as a good improvement. Fig. 6.6 depicts the outcome of this treatment. In comparison to the isolated example, this case shows no improvement in the FL model. Compared to the centralized model, on the other hand, FL is improving. This gain may be because, in isolated training, our models are more focused on detecting the most often repeated label, label 5. We can observe this impact in FL since it is the average of these models, but in the centralized model, we remove this emphasis, making it more likely to misclassify label 5, which has a high recurrence in the test set.

After that, we move on to case "b" of scenario 2, where only label types low (2) and high (5) are missing from one or two clients. We demonstrated these cases in the table 6.7.

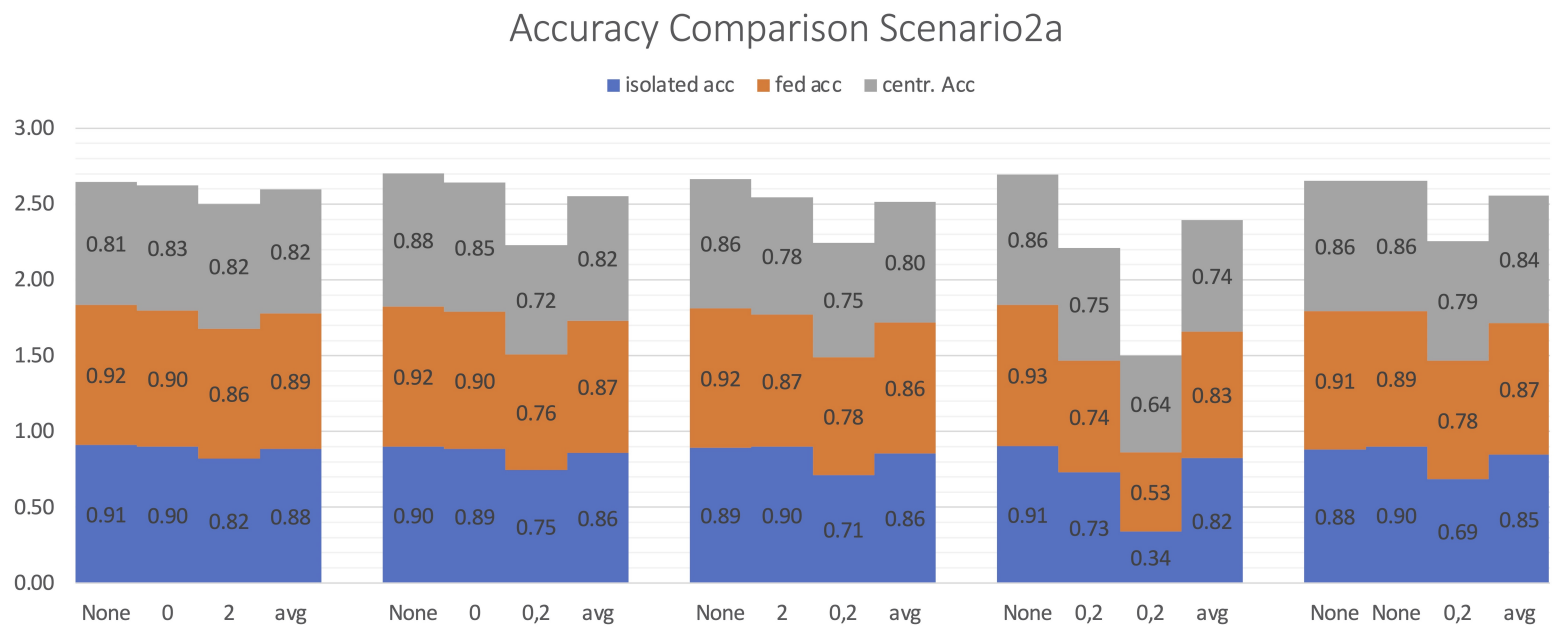


Figure 6.6: Comparison of accuracy for scenario 2, the case "a." each set of three columns displaying one case corresponding to the table 6.6 (groups from right to left in the figure is matching with the cases in the table from up to down). Furthermore, each column represents a client, and the numbers under each column indicate the client's missing label. Each column displays isolated accuracy, FedAvg accuracy, and centralized accuracy for one client.

Table 6.7: sub-division of scenario 2, case "b", where only label types high(2) and low(5) are missing from one or two clients. The numbers correspond to the numbers of table of 6.4.

N.O. case	client 1	client 2	client 3
8	None	2	5
9	None	2	2 and 5
13	None	5	2 and 5
21	None	2 and 5	2 and 5
28	None	None	2 and 5

Fig. 6.7 shows the result of the case "b" in scenario 2, where only label types low (2) and high (5) are missing from one or two clients. We can see in cases 8 and 9 (the first and the second set of columns from the bottom), we have a degradation of 5% in the accuracy for client 2 comparing FedAvg and isolated. Nevertheless, we can see a good improvement in client 3 with 11% and 9% of improvement. In client 3, centralized learning has an accuracy of 70% in both cases, making it a promising approach for client 3 to use the pooled data if it has the chance. The subsequent two cases (13, the third set of columns from the bottom, and 21, the second set of columns from the top) has the same behavior as client 2. FedAvg improved accuracy 7% and 6% compared to isolated model respectively. These relatively close numbers show that when we have more than one label missing, the lowest repeated label (2) does not affect FL. The last case (fist set of columns from the top) shows that in client 3, we had an increase of 11% in FedAvg with respect to the isolated case. This good improvement is related to the fact that label 5 is repeated the most in our test set, and when client 3 can recognize it without ever seeing it in its data set, the accuracy improves a lot. We interpret the figure as follows: we can see that when label 2 is missing with label 5, and both do not exist in one client, the effect of the low repeated label (2) is negligible, and the pattern of scenario 1 for label 5 is appearing. However, when both are missing in one client, we see a rapid degradation of FL compared to the centralized case.

Following that, we go to case "c" of scenario 2, where only the label types medium (0) and high (5) are absent from one or two customers. These examples are included in the table 6.8.

The situation with two missing labels, in which the medium and high labels are absent from one or two customers, differs from the preceding one.

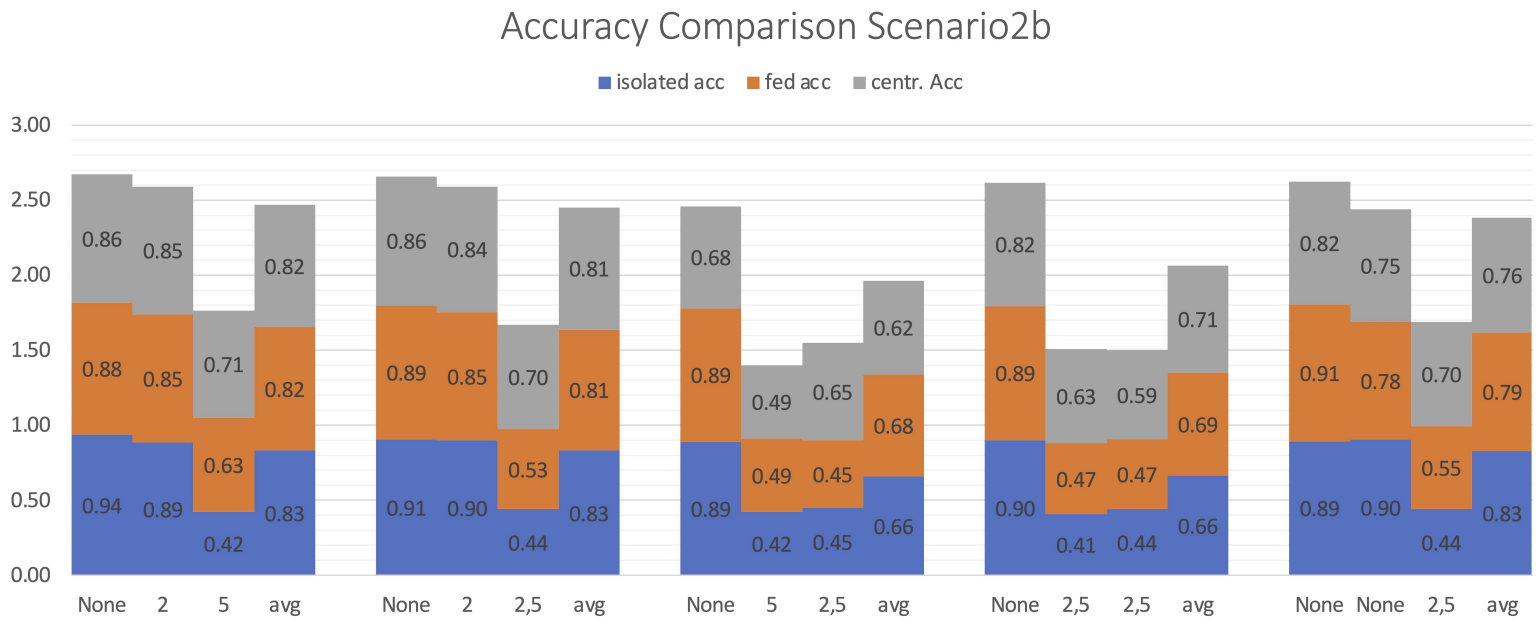


Figure 6.7: Comparison of accuracy for scenario 2, case "b." each set of three columns displaying one case corresponding to the table 6.7 (groups from right to left in the figure is matching with the cases in the table from up to down). Furthermore, each column represents a client, and the numbers under each column indicate the client's missing label. Each column displays isolated accuracy, FedAvg accuracy, and centralized accuracy for one client.

Table 6.8: sub-division of scenario 2, case "c", where only label types medium (0) and high (5) are missing from one or two clients. The numbers correspond to the numbers of table of 6.4.

N.O. case	client 1	client 2	client 3
3	None	0	5
6	None	0	0 and 5
15	None	5	0 and 5
19	None	0 and 5	0 and 5
27	None	None	0 and 5

Fig. 6.8 shows the result of the cases which are shown in Table 6.8. In case 3 (the first column from the bottom), we do not see much improvement in client 2 where the medium repeated label (0) is missing, as the pattern is the same in previous cases. Nevertheless, we see a good improvement in client 3 with a 19% increase in FedAvg compared to the isolated case. The reason, as told, is because the number of test points with label 5 is more than others, so when a client finds knowledge to recognize it, we immediately see the difference. However, we see that the centralized model has 70% accuracy in client 3 whereas the FedAvg accuracy is 62%, again the best option for client 3, in this case, can be a centralized model.

Case 6 (the second set of columns from the bottom) shows the same trend for client 2 as the previous case, but now that we have even more labels missing in client 3, inaccuracy of FedAvg improved 19% as before. We can say that when label 5 is missing if any other label is missing, we can see the same improvement as if only label 5 is missing from the client. Now we move on to the next case, 15 (the middle set of columns). We can see interesting results here; we do not see much improvement comparing FedAvg and isolated model in client 2 and client 3, only 6% improvement for both. The reason for this number is that now too much label is missing from the client which can affect badly the aggregated results we cannot have enough knowledge about label 5 which is the most repeated one so just the knowledge of client 1 is added to them which is not enough the same reasoning goes for case 19 (the second set of columns from the top). Case 27 (the first one from the top) shows the most improvement between other cases comparing FedAvg with the isolated model. FedAvg improved the isolated accuracy by 22% which by far is the best increase that we have. The reason for that is because the knowledge of client 3 is few, but client 2 and client 1 has complete

knowledge about all the 6 failure causes; this can, of course, improve a lot the client 3 accuracies because now client 3 can recognize 2 types of the label without every observing them in the training phase. Now by comparing average values, we may conclude that FL works better than isolated learning but is near to centralized learning.

Now we can move on to the third scenario; Here, we cannot divide clearly by category, so our examination would be general and on all cases.

6.3.3 Scenario 3: where three types of the label is missing

Scenario 1: Investigate the case where one or two clients miss three types of label.

Objective: To understand in which cases a collaboration is profitable.

This scenario considers cases in which we have a very powerful operator will all the labels inside the dataset but one or two of the other operators are pretty weak and miss some labels in their own data. This knowledge of the missing label is crucial for the more vital operators to understand which situations they should avoid or require more and which cases they may safely work on.

Table 6.9: Space state for scenario 3. The showed numbers are missing labels.

N.O. case	client 1	client 2	client 3
4	None	0	2 and 5
11	None	2	0 and 5
14	None	5	0 and 2
17	None	0 and 2	0 and 5
18	None	0 and 2	2 and 5
20	None	0 and 5	2 and 5
22	None	None	None
30	None	0 and 2 and 5	0
31	None	0 and 2 and 5	2
32	None	0 and 2 and 5	5
33	None	0 and 2 and 5	0 and 2
34	None	0 and 2 and 5	0 and 5
35	None	0 and 2 and 5	2 and 5
36	None	0 and 2 and 5	0 and 2 and 5

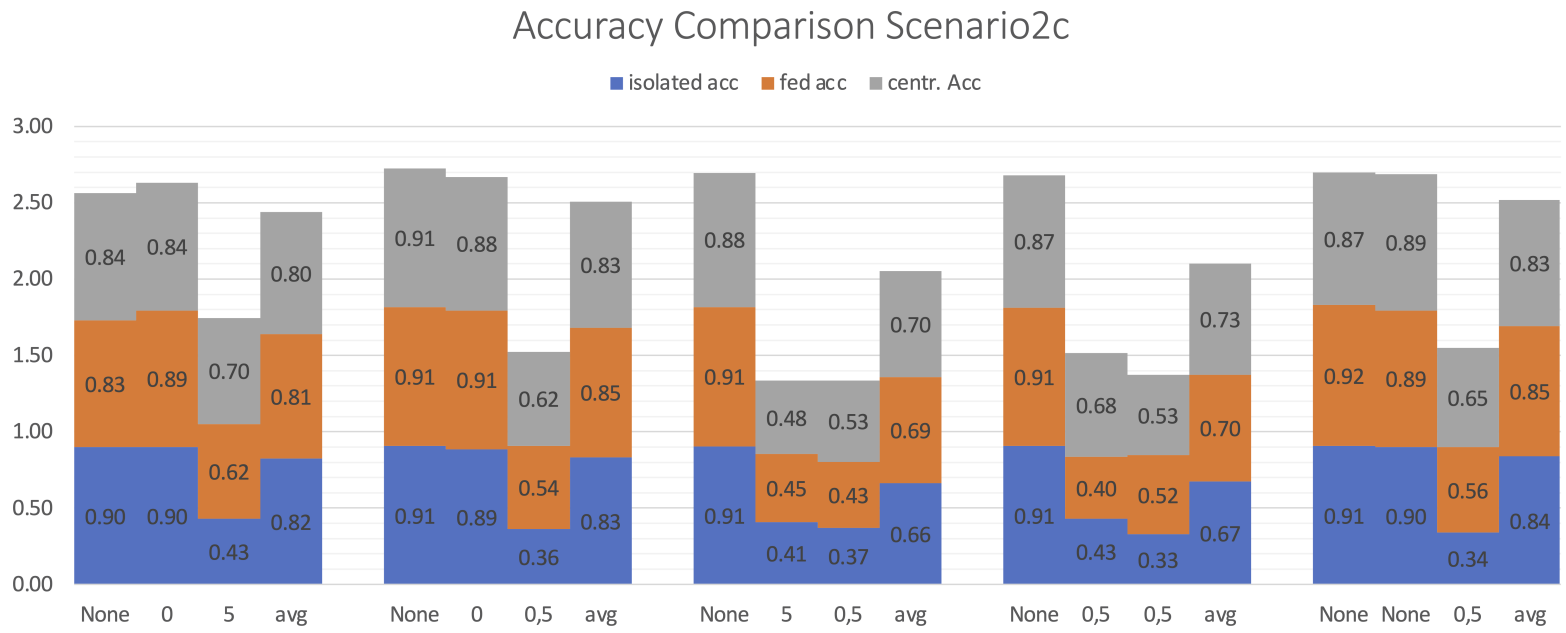


Figure 6.8: Comparison of accuracy for scenario 2, case "c." each set of three columns displaying one case corresponding to the table 6.7 (groups from right to left in the figure is matching with the cases in the table from up to down). Furthermore, each column represents a client, and the numbers under each column indicate the client's missing label. Each column displays isolated accuracy, FedAvg accuracy, and centralized accuracy for one client.

The results of this experiment are depicted in the figure 6.9. Cases 4, 11, 14, 17, 18, and 20 (the first six sets of columns from the button) are following the same trend as has been explained in scenario 1, see 6.3.1, and scenario 2, see 6.3.2. So we move on to case 22 (the seventh set of columns from the bottom). In this case, we can see that we have all the labels available in all three clients; the question is, will FL improve the accuracy if they collaborate? The answer is yes, and we can see in client 1 and client 3, 3% and 2% improvement in FedAvg compared to isolated model respectively. Also, we can see a better improvement of FedAvg concerning the centralized approach in average 12% of improvement. The reason is that when we have three big clients with all the failure causes available in their data sets. Each can train a model more focused on labels to recognize some labels better than another based on the number of specific points. The focus can then be averaged in FedAvg and improve accuracy since this focus will not exist while training on a whole dataset.

In cases 30, 31, 32 (the seventh, sixth and fifth set of columns from the top), we see the same pattern as discussed before on client 3. However, for client 2 we degradation of 1% inaccuracy of FedAvg in comparison with isolated. The reason for this is based on the fact that client 2, due to the lower number of variations in the label, now can recognize much better the rest of the labels, so the accuracy of the isolated model is high (89%) considering the fact the client 2 does not know labels 0, 2, 5. So, giving client 2 the knowledge of these three labels does not change the accuracy much since now the ability of the model to recognize the rest of the labels except for 0, 2, 5 is lower. For the rest of the cases, the same reasoning will apply except that the degradation in FedAvg compared to the isolated model is based on the combination of the labels missing from the third client. For example, in case 35(the second set of columns from the top), the degradation is bout 10%. Here we can see that when many labels are missing in two clients, see the last case (number 36 in the table 6.9), the FedAvg accuracy of the client 1 which has all the labels on its dataset will descrease by 5% in comparison with isolated case. This case is helpful for the second and third operators but not convenient for the first party. Also, merging all the data will not help much because it violates privacy and requires a large database unit, and the accuracy would be the same as FL. The other examples follow the patterns illustrated and discussed in the sections 6.3.1 and 6.3.2 since the label with the lowest repetition will not have a significant impact on the final accuracy.

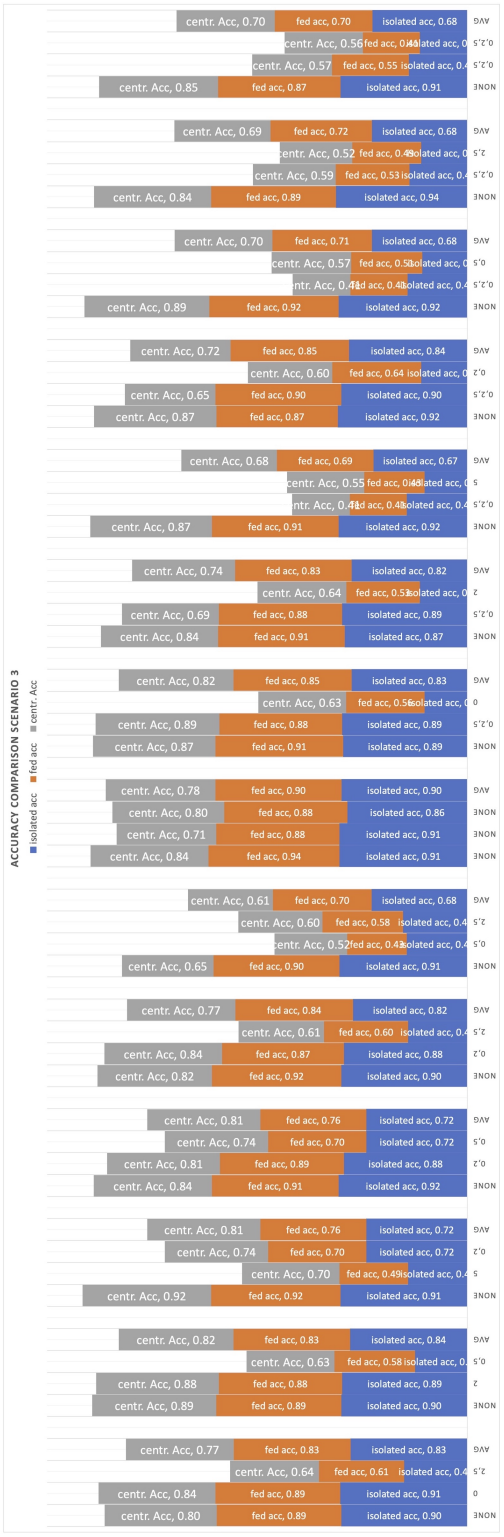


Figure 6.9: Comparison of accuracy for scenario 3. each set of three columns displaying one case corresponding to the table 6.5 (groups from right to left in the figure match with the cases in the table from up to down). Furthermore, each column represents a client, and the numbers in each column indicate the client's missing label. Each column displays isolated accuracy, FedAvg accuracy, and centralized accuracy.

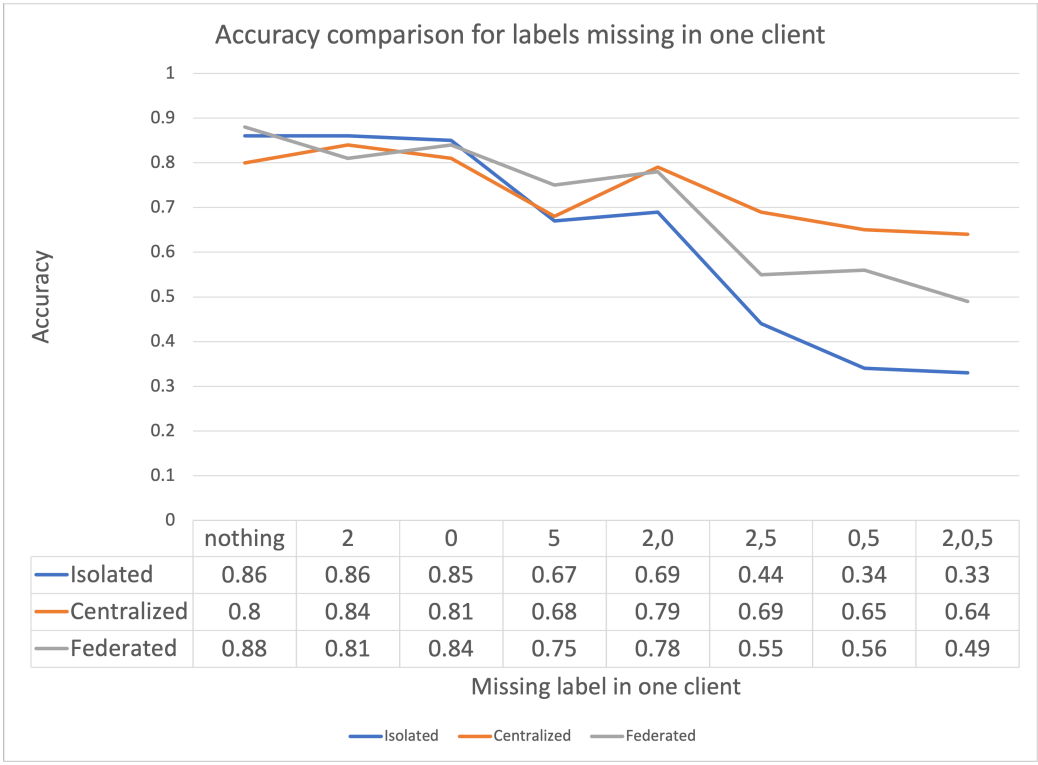
6.4 Effect of missing labels on one client

We investigate the impact of missing labels on only one client. This study will help you understand how FL can assist increase accuracy as the number of missing labels grows.

We assumed in this case that we have two clients with all of the accessible labels. Furthermore, one client can profit from the expertise of its two peers. As seen in Fig 6.10, When there is no label missing in the last client, we can see that FL is improving the accuracy by 2% in comparison with isolated model and 8% compared to centralized. We can see how FL aids in the development of a more accurate failure-case detection model where we can classify a newly seen data point with the features mentioned earlier in the six failure root-cause categories. Following this, we will go to the low repetition label (label 2); this label is missing from the third client. We can see that FedAvg decreases the accuracy concerning isolated and centralized models by 5% and 3%, respectively. This decrease happens because of the added knowledge of label 2 where we only have one of it in our test set, and it can bring the accuracy of our classifier to recognize other labels with more repetition correctly.

As we move on to the fourth column of the table with label 5, which has the highest repetition is missing, we can see the effect of FL on accuracy here FedAvg has the accuracy of 75% whereas isolated has 67% and centralized 68%. We can see here that an operator is missing a high repeated label. The gain of the accuracy improvement with FedAvg is about 8%. Now we explain the following case where labels 0 and 2 are missing, and we can that even though when each one of them is missing alone in a client, we would not see a good improvement in accuracy of FedAvg compared to isolated when both of them are missing from a client, we can see that the improvement is about 9%. This improvement can be significant for this client since from now on, and it can classify *two* more failure causes without ever seeing them. In the cases where label 5 is missing with labels 0 and 2, we can see that the improvement of FedAvg is noticeable compared to the isolated model. To be precise, when labels 2 and 5 are missing from one client, the improvement of FedAvg is 9%; however, when labels 2 and 0 are missing, the improvement is 22%. Here we can see that the more labels with more repetition are missing in one client, the improvement of FedAvg in comparison to an isolated case in increasing; for example, this improvement in when label 2 is missing is -5%

but when labels 0,5 are missing which have 22% and difference between them 27%. So, the best option for this client is FedAvg according to the isolated model. We have not seen any improvement after the number of low-response sites was reduced. We do not notice any progress since the low repeated label points are too little, which does not impact the model and reduces FL accuracy. This effect can be explained by the fact that the accuracy of predicting other labels might be lowered compared to the isolated instance. The critical aspect here is that the more labels, and thus the more points missing from one client, the more the FL may assist in having an accuracy approaching centralized model.



	nothing	2	0	5	2,0	2,5	0,5	2,0,5
Isolated	0.86	0.86	0.85	0.67	0.69	0.44	0.34	0.33
Centralized	0.8	0.84	0.81	0.68	0.79	0.69	0.65	0.64
Federated	0.88	0.81	0.84	0.75	0.78	0.55	0.56	0.49

Figure 6.10: Accuracy comparison for missing labels in one client. The numbers on x axis define the missing labels in the client

Chapter 7

Conclusions and Future Work

This thesis presented the failure identification problem in microwave networks as a federated learning classification problem. We took a data sequence from a genuine Italian microwave network and showed our data preparation technique to make them acceptable for our algorithm.

We examine the performance of three alternative classification approaches (Isolated, Federated, and Centralized) and find the trade-offs in classification accuracy, privacy, and needed space. We also assess the algorithms' performance using varying amounts of labeled data. Based on our findings, we conclude that federated learning may enhance accuracy significantly more than isolated learning and can be as good as or better than centralized learning, assuming we have the proper state.
To decrease the cost of sharing the model, we should carefully examine the missing label. In our work, the focus was on the repeating of the failure; we examined three types of labels with low (2), medium (0), and high(5) repetition.
We can say that for the most repeated label(5), we can see an increase of 20% in accuracy compared to isolated. Also, when all the participants share their data through federated learning, the final result can be better. Nevertheless, we can see a slight improvement in the least repeated label which means the client who lacks it can now recognize and classify it. However, if we have a big operator with all the data available, we may see a degradation inaccuracy in all the cases.

So, in conclusion, federated learning can be beneficial in terms of the users' privacy and communications of the data. However, operators should

first consider the missing label and then examine if federated learning can benefit them or not.

As future work, the followed research directions can be identified:

- Evaluate other aspects like the number of labels that are missing in the client.
- Analyse other classification algorithms able to classify and can be integrated with federated learning.
- Do other data separation based on other features like geographical separation.
- Try with aggregation biased towards the specific client

Bibliography

[1] A. Alhosban, Z. Malik, K. Hashmi, B. Medjahed, and H. Al-Ababneh, "A two phases self-healing framework for service-oriented systems," *ACM Trans. Web*, vol. 15, Apr. 2021.

[2] M. Nouioua, P. Fournier-Viger, G. He, F. Nouioua, and Z. Min, "A survey of machine learning for network fault management," *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics*, p. 1–28, 2021.

[3] B. C. Wyld, "Failure detection method in a communication channel with several routes," Apr. 7 1998. US Patent 5,737,311.

[4] B. Wang, H. Yang, Q. Yao, A. Yu, T. Hong, J. Zhang, M. Kadoch, and M. Cheriet, "Hopfield neural network-based fault location in wireless and optical networks for smart city iot," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 1696–1701, IEEE, 2019.

[5] H. Wietgrefe, K.-D. Tuchs, K. Jobmann, G. Carls, P. Fröhlich, W. Nejdl, and S. Steinfeld, "Using neural networks for alarm correlation in cellular phone networks," in *International Workshop on Applications of Neural Networks to Telecommunications (IWANNT)*, pp. 248–255, Citeseer, 1997.

[6] S. Shahkarami, F. Musumeci, F. Cugini, and M. Tornatore, "Machine-learning-based soft-failure detection and identification in optical networks," in *Optical Fiber Communication Conference*, p. M3A.5, Optical Society of America, 2018.

[7] F. Musumeci, L. Magni, O. Ayoub, R. Rubino, M. Capacchione, G. Rigamonti, M. Milano, C. Passera, and M. Tornatore, "Supervised and semi-supervised learning for failure identification in microwave networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1934–1945, 2021.

[8] I. Syrigos, N. Sakellariou, S. Keranidis, and T. Korakis, "On the employment of machine learning techniques for troubleshooting wifi networks," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, IEEE, 2019.

[9] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "An overview on application of machine learning techniques in optical networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1383–1408, 2018.

[10] W. Zhang, X. Li, H. Ma, Z. Luo, and X. Li, "Federated learning for machinery fault diagnosis with dynamic validation and self-supervision," *Knowledge-Based Systems*, vol. 213, p. 106679, 2021.

[11] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported internet of things," *IEEE Access*, vol. 7, pp. 69194–69201, 2019.

[12] B. Shariati, P. Safari, G. Bergk, F. I. Oertel, and J. K. Fischer, "Inter-operator machine learning model trading over acumos ai federated marketplace," in *Optical Fiber Communication Conference (OFC) 2021*, p. M2B.7, Optical Society of America, 2021.

[13] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018.

[14] S. Liu, D. Wang, C. Zhang, L. Wang, and M. Zhang, "Semi-supervised anomaly detection with imbalanced data for failure detection in optical networks," in *Optical Fiber Communication Conference (OFC) 2021*, p. Th1A.24, Optical Society of America, 2021.

[15] C. Wang, N. Yoshikane, F. Balasis, and T. Tsuritani, "Acceleration and efficiency warranty for distributed machine learning jobs over data center network with optical circuit switching," in *2021 Optical Fiber Communications Conference and Exhibition (OFC)*, pp. 1–3, 2021.

[16] J. Li, L. Chen, and J. Chen, "Scalable federated learning over passive optical networks," *CoRR*, vol. abs/2010.15454, 2020.

[17] G. Yufeng, "The 7 steps of machine learning." `https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e`. Accessed : 2019-09-11.

[18] F. Trovò, "Exercises notes in machine learning," 2019.

[19] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, 2018.

[20] W. S. Sarle, "Neural networks and statistical models," 1994.

[21] S. S. Haykin *et al.*, *Neural networks and learning machines/Simon Haykin.* New York: Prentice Hall,, 2009.

[22] M. Nielsen, "Neural networks and deep learning." `http://neuralnetworksanddeeplearning.com/chap2.html`. Accessed : 2019-09-13.

[23] "Introduction to support vector machines." `https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html`. Accessed : 2019-09-16.

[24] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2019.

[25] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[26] Y. Lie, W. Yang, T. Chen, and Z. Wei, "Federated learning and transfer learning for privacy, security and confidentiality." AAAI 2019 Tutorial, `https://aisp-1251170195.file.myqcloud.com/fedweb/1552916850679.pdf`.

[27] W. A. Department, "Federated learning white paper v1.0 webank, shenzhen, china." `https://aisp-1251170195.cos.ap-hongkong.myqcloud.com/fedweb/1552917186945.pdf2,5`, September 2018.

[28] F. hartmann, "Federated learning." `https://florian.github.io/federated-learning/1,56,66`, August 2018.

[29] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, vol. abs/1806.00582, 2018.

[30] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *CoRR*, vol. abs/1903.02891, 2019.

[31] V. Lier, "Robustness of federated averaging for non-iid data," Master's thesis, Computer Science, 2018.

[32] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. B. Calo, "Analyzing federated learning through an adversarial lens," *CoRR*, vol. abs/1811.12470, 2018.

[33] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019.

[34] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *CoRR*, vol. abs/1810.06060, 2018.

[35] P. Vepakomma and O. Gupta, "Mit media lab's split learning: Distributed and collaborative learning."

[36] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[37] A. Gooday, "Understanding federated learning terminology," Sep 2020.

[38] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018.

[39] K. Xu, H. Mi, D. Feng, H. Wang, C. Chen, Z. Zheng, and X. Lan, "Collaborative deep learning across multiple data centers," *CoRR*, vol. abs/1810.06877, 2018.

[40] I. Cano, M. Weimer, D. Mahajan, C. Curino, and G. M. Fumarola, "Towards geo-distributed machine learning," *CoRR*, vol. abs/1603.09035, 2016.

[41] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Expert Systems with Applications*, vol. 39, no. 1, pp. 424–430, 2012.

[42] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 90–98, SIAM, 2017.

[43] D. Liu, T. A. Miller, R. Sayeed, and K. D. Mandl, "FADL: federated-autonomous deep learning for distributed electronic health record," *CoRR*, vol. abs/1811.11400, 2018.

[44] K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," *CoRR*, vol. abs/1902.01046, 2019.

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[46] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, DIDL '18, (New York, NY, USA), p. 1–8, Association for Computing Machinery, 2018.

[47] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, vol. abs/1610.02527, 2016.

[48] Y. Wang, "Co-op: Cooperative machine learning from mobile devices.," Master's thesis, Dept. Elect. and Comput. Eng., 2017.

[49] Commscope, "Microwave communication basics," 2017.

[50] L. Reggiani, "Lecture notes in wireless communication," 2018.

[51] "Multipath fading." `https://www.electronics-notes.com/articles/antennas-propagation/propagation-overview/multipath-fading.php`. Accessed : 2019-08-29.

[52] F. Coenning, "Understanding itu-t error performance recommendations." https://www.julesbartow.com/Pictures/ITS/ITU-T_Errors_ApplicationNote2.pdf.

[53] L. Magni, "Machine-learning-assisted failure management in microwave networks," Master's thesis, Telecommunications engineering, 2019.

[54] M. Asad, A. Moustafa, and T. Ito, "Federated learning versus classical machine learning: A convergence comparison," 10 2020.

[55] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *CoRR*, vol. abs/1610.03295, 2016.

[56] B. Ravi, J. Thangaraj, and S. Petale, "Data traffic forwarding for inter-vehicular communication in vanets using stochastic method," *Wireless Personal Communications*, vol. 106, pp. 1591–1607, Jun 2019.

[57] G. Psychou, D. Rodopoulos, M. M. Sabry, T. Gemmeke, D. Atienza, T. G. Noll, and F. Catthoor, "Classification of resilience techniques against functional errors at higher abstraction layers of digital systems," *ACM Comput. Surv.*, vol. 50, Oct. 2017.

[58] D. Yogatama, P. Blunsom, C. Dyer, E. Grefenstette, and W. Ling, "Learning to compose words into sentences with reinforcement learning," *CoRR*, vol. abs/1611.09100, 2016.

[59] M. Ferdman, A. Almutaz, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, and A. Ailamaki, "Clearing the clouds: A study of emerging scale-out workloads on modern hardware," in *17th International Conference on Architectural Support for Programming Languages and Operating Systems*, March 2012.

[60] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *CoRR*, vol. abs/1912.09789, 2019.

[61] G. E. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied artificial intelligence*, vol. 17, no. 5-6, pp. 519–533, 2003.

[62] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*,

DIDL '18, (New York, NY, USA), p. 1–8, Association for Computing Machinery, 2018.

[63] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[64] A. Kanekar, "Optimization and convergence of machine learning algorithms," Jun 2018.

[65] D. Loiacono, "Model evaluation, selection and ensembles machine learning," 2020.