NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ALEXANDRE MIGUEL MANTA DA COSTA

Bachelor of Science in Electrical and Computer Engineering

COLLABORATIVE IMPROVEMENT OF SMART MANUFACTURING USING PRIVACY-PRESERVING FEDERATED LEARNING

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2022

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COLLABORATIVE IMPROVEMENT OF SMART MANUFACTURING USING PRIVACY-PRESERVING FEDERATED LEARNING

ALEXANDRE MIGUEL MANTA DA COSTA

Bachelor of Science in Electrical and Computer Engineering

Adviser: Ricardo Alexandre Fernandes da Silva Peres
Invited Professor, NOVA University Lisbon

Co-adviser: José António Barata de Oliveira
Associate Professor, NOVA University Lisbon

Examination Committee

Chair: Fernando José Vieira do Coito
Associate Professor, NOVA University Lisbon

Rapporteur: André Dionísio B. da Silva Rocha
Assistant Professor, NOVA University Lisbon

Member: Ricardo Alexandre Fernandes da Silva Peres
Invited Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2022

Collaborative Improvement of Smart Manufacturing using Privacy-Preserving Federated Learning

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.8.6) [1].

To my parents and grandparents.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Ricardo Peres, for all the help, support and motivation. I could not have chosen a better person to guide and teach me. Thank you for introducing me to the scientific research world.

I would also like to thank my co-supervisor, Professor José Barata, for welcoming me into his scientifical research group, where I had the opportunity to develop this dissertation and join scientifical research projects, which helped me grow personally and professionally.

To NEEC, AEFCT, CoPe and In-Nova, thank you for the knowledge, adventures, late nights and early mornings. The things that we created together are the proof that with commitment and hard work, everything is possible. Most of all, thank you for the friendships, lessons learned and for helping me grow.

I would also like to thank the friends I developed in these six years. Rafaela Pinheiro, Margarida Grilo, Moisés Tereso, Rui Costa, Leandro Filipe and Rita Rebelo. Thank you for the great moments, laughs, ups and downs, group works and get-togethers, your support, and friendship. I would especially like to address Miguel Arvana, for always being there in the good and the bad, the right and wrong. Thank you for always understanding.

To my high school friends, thank you for being my second family, the ones who where always around all these years, whom I have created some of the best memories. Thank you for your endless support and friendship.

To Filipa, thank you for being who you are. Your love and support makes me grow every day. Thank you for all your love, patience and always being by my side.

Finally, I would like to address a heartfelt thank you to my parents and grandparents. The ones who raised me, taught me and loved me inconditionally. Words cannot describe how much I thank you for all the sacrifices you made for me. If I am what I am today it is because of you. You have my eternal gratitude for all you have done for me.

"The important thing is to not stop questioning.
Curiosity has its own reason for existence."
(Albert Einstein)

Abstract

Nowadays, data sharing among different sources is is very challenging in the manufacturing domain, mainly due to industry competition, complicated bureaucratic processes, and privacy and security concerns. Centralized Machine Learning (ML) poses an essential aspect in several industries, including smart manufacturing. However this approach may lead to several issues regarding security and performance.

In response to these problems, Federated Learning (FL) was created. FL is an innovative and decentralized approach to ML, focused on collaboration and data privacy. In this approach, data is kept in each source where it is trained locally, and only model weights or gradients are shared to create a global model.

Although several works have already been implemented towards this problem, there are still many unresolved issues concerning the application of FL frameworks in smart manufacturing scenarios. Among the several issues found in the analysed works it is important to emphasize the disregard facing industry 4.0 architectures, strategies and the unavailability to improve those frameworks further.

This work aims to build a FL framework for smart manufacturing with specific concerns in privacy and applicability in industrial scenarios. The main focus of this framework is to facilitate a collaborative approach in the application of ML to manufacturing by enabling the knowledge sharing for this purpose and taking privacy as a special concern. In addition, the implementation and testing of privacy-preserving algorithms, while improving the framework for industrial scenarios are emphasized. A modular approach is chosen to create a framework adapted to various industrial cases by implementing several nodes that focus on specific aspects of data collection, data treatment, connection with the FL system, and ML model management.

The results revealed a competitive model performance of the framework compared to the centralized approach while keeping data at each source, protecting its privacy. The implemented framework also proved to be compliant with the IEEE Std 3652.1-2020 standard guidelines, attaining the established requirement levels.

Keywords: Federated Learning, Privacy-Preserving Federated Learning, Smart Manufacturing, Framework

Resumo

Atualmente, a partilha de dados entre diferentes fontes é um grande desafio no domínio da manufatura, principalmente devido à concorrência da indústria, processos burocráticos complicados e preocupações de privacidade e segurança. O *Machine Learning (ML)* impõe-se como um aspeto essencial em várias indústrias, incluindo a manufatura inteligente. Contudo, esta abordagem pode levantar várias questões relativamente à segurança e ao desempenho.

Em resposta a estes problemas, foi criado o *Federated Learning (FL)*. FL é uma abordagem inovadora e descentralizada de ML, centrada na colaboração e privacidade de dados. Nesta abordagem, os dados são mantidos em cada fonte, onde são treinados localmente, e apenas os pesos ou gradientes dos modelos são partilhados para criar um modelo global.

Embora vários trabalhos já tenham sido implementados visando esta temática, ainda existem muitas questões por resolver relativas à aplicação de *frameworks* de FL em cenários de manufatura inteligente. Entre as várias questões encontradas na literatura analisada, é importante enfatizar a desconsideração pelas arquiteturas e estratégias da indústria 4.0 e a indisponibilidade para melhorar essas *frameworks*.

Este trabalho visa construir uma *framework* de FL aplicada à manufatura inteligente com preocupações específicas no que toca a matérias de privacidade e aplicabilidade em cenários industriais. O principal objectivo desta *framework* é facilitar uma abordagem colaborativa na aplicação de ML ao fabrico, permitindo a partilha de conhecimentos para este fim e enfatizando a preocupação na privacidade dos utilizadores. Uma abordagem modular foi escolhida para criar uma *framework* adaptada a vários casos industriais através da implementação de vários nós que se concentram em aspetos específicos da recolha de dados, tratamento de dados, ligação com o sistema de FL e gestão do modelo de ML.

Os resultados revelaram um desempenho competitivo do modelo em relação a uma abordagem centralizada, mantendo os dados em cada fonte e protegendo a sua privacidade. A *framework* implementada também provou estar em conformidade com a norma IEEE Std 3652.1-2020, atingindo os níveis de exigência estabelecidos.

Palavras-chave: *Federated Learning*, *Privacy-Preserving Federated Learning*, Manufatura inteligente, *Framework*

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AI	Artificial Intelligence
AUC	Area under the ROC curve
CPS	Cyber-Physical System
DCN	Data Collection Node
DL	Deep Learning
DP	Differential Privacy
ERP	Enterprise Resource Planning
FedAvg	Federated Averaging
FedOpt	Federated Optimization
FedYOGI	Federated Optimization with the adaptive optimizer YOGI
FL	Federated Learning
FPGA	Field-Programmable Gate Array
FSVRG	Federated Stochastic Variance Reduced Gradient
FTL	Federated Transfer Learning
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GFN	Global Federated Node
GPU	Graphics Processing Unit
gRPC	gRPC Remote Procedure Calls
HFL	Horizontal Federated Learning
IoT	Internet of Things
IT	Information Technology
LFN	Local Federation Node
LIN	Local Intelligence Node

MES	Manufacturing Execution System
MIA	Membership Inference Attack
ML	Machine Learning
NN	Neural Network
PID	Proportional–Integral–Derivative Controller
PLC	Programmable Logic Controller
PPFL	Privacy-Preserving Federated Learning
RAMI 4.0	Reference Architectural Model Industrie 4.0
RDP	Rényi Differential Privacy
SCADA	Supervisory Control and Data Acquisition
SGD	Stochastic Gradient Descent
SMC	Secure Multi-Party Computation
SVRG	Stochastic Variance Reduced Gradient
TPU	Tensor Processing Unit
VFL	Vertical Federated Learning

Chapter 1

Introduction

1.1 Background

Nowadays, the usage of Artificial Intelligence (AI) in the industrial context faces two major challenges: the isolation of data, and the rise of data privacy and security. In the majority of fields, data is largely concentrated, leaving most of the stakeholders with limited or poor-quality data. Breaking barriers that hold data to its origins and sharing it between organisations is almost impossible. This obstacle results from industry competition, long and complicated bureaucratic processes and privacy, and security concerns. Concurrently, there is an exponential growth in awareness relative to data security and privacy. Recent events regarding data leakage by multinational companies have raised the concern for data privacy. It is usual to have data transactions in traditional Artificial Intelligence (AI) methods, but since new regulations like General Data Protection Regulation (GDPR) were created, such transactions may face legal issues [2].

In smart manufacturing, centralised Machine Learning (ML) poses an essential aspect mainly for taking part in, for example, cost optimisation, productivity improvement, error reduction, and quality control. In this context, a centralised approach to ML may lead to several issues, including high latency in real time scenarios, lower resistance to security attacks, and data leakage concerns [3].

To answer these problems, Google created Federated Learning (FL), an innovative and decentralised approach to ML, focused on collaboration while maintaining data privacy. In FL, data is kept in each source where it is trained locally, and only the parameters (model weights or gradients) are shared. The purpose is to create a global model with similar accuracy as a model trained with all the data gathered in the same place [2, 4, 5].

1.2 Research Problem and Motivation

Several studies show progress towards creating FL frameworks. The great majority are general-purpose frameworks which provide the necessary tools to apply FL to any given problem. Most of the general-purpose frameworks can already be implemented, but still do not provide the essential tools to guarantee users privacy, and also need implementation and testing before being applied to a real scenario. In the specific case of smart manufacturing, some frameworks were proposed to tackle difficulties in model accuracy and data heterogeneity, which are frequent problems in industrial context [4]. Although the results are promising, these studies failed to address device heterogeneity problems in smart manufacturing scenarios. The applicability is another problem in smart manufacturing frameworks, since any research frameworks was not applied to an industrial scenario, nor available for further improvement. Considering all of these challenges that arise from the existing FL frameworks, a new solution is necessary. In this sense, the following research question was already considered:

> **In which way can collaborative ML be implemented in complex manufacturing environments whilst ensuring privacy and security?**

To address the presented research question, the following hypothesis was formulated:

> *If a modular Privacy-Preserving Federated Learning (PPFL) framework that leverages existing general-purpose FL frameworks and privacy-preserving algorithms is created, collaborative ML can address several problems found in complex manufacturing environments whilst keeping data private and secure.*

In order to validate the research hypothesis, several tests must be made to guarantee the framework's robustness, and a modular approach will be implemented to simulate a distributed industrial scenario. As a result of this work, it is expected to obtain an open-source framework ready to be implemented in industrial scenarios, with the capability to add new modules and implement new algorithms to it.

1.3 Thesis Outline

Following the introduction, the second chapter presents a review of literature about Industry 4.0, Cyber-Physical Systems, ML, FL, and Privacy-Preserving Federated Learning (PPFL) will be provided to contextualize this work. A comparison between several works reported in the literature and the identification of some implementation challenges will also be presented. The third chapter will describe the proposed framework, mapping it to the reference architecture present in IEEE Std 365.1™-2020 standard [6] and explaining its composing modules.

The fourth chapter elaborates on the framework implementation, starting by explaining the chosen use case. The development of each node and the communication processes

between them are provided, and the implementation of privacy protection mechanisms is depicted.

In the fifth chapter, the testing processes which shall be made to evaluate the framework are described, and the required levels that the framework shall achieve are established.

The sixth chapter covers the tests and results realized according to the fifth chapter. A discussion of the obtained results is also provided with the drawn conclusions of each testing criterion.

Finally, the seventh chapter formulates the conclusions drawn from the carried work and the contributions and future work that may arise from this dissertation.

CHAPTER 2

LITERATURE REVIEW

2.1 Industry 4.0

Nowadays, the complex supply chains require flexible production environments that can quickly and efficiently adapt the goods produced to the market's flexibility and quality requirements. The fourth industrial revolution, or simply Industry 4.0, addresses these challenges by enabling communication between all important production assets in production and Information Technology (IT) [7].

Industry 4.0 is a concept initially introduced in Germany, in 2011, as a strategic initiative to revolutionize the manufacturing process. Currently, the term represents a trend where automation technologies are associated to the manufacturing industry. Among the technologies employed it is important to highlight the enabling technologies such as Internet of Things (IoT), cloud computing and Cyber-Physical System (CPS). Figure 2.1 shows the evolution and the main enablers of every industrial revolution so far [8–10].

The base principal of Industry 4.0 paradigm, and one of its main goals, is integration. Integration can be divided as: Horizontal integration, which regards cooperation and IT systems integration along the value chain; Vertical Integration, where a single company is extensively automated with various IT systems; New work infrastructures; Presence of engineering through the complete production lifecycle; and product customization via small lots or even lot sizes of one [7, 10]. The work of Aceto *et al.* [10] highlights the evidence that the enabling technologies above mentioned are well-established fields with unique characteristics and concerns but also closely interrelated. According to Prisloo *et al.* [11], by integrating this technologies together, several possibilities will be enabled concerning manufacturing capabilities, productivity and efficiency. One of the main aspects concerning the integration of these technologies will be the creation of industrial value. The manufacturers will be able to react faster to market volatility, and create innovative

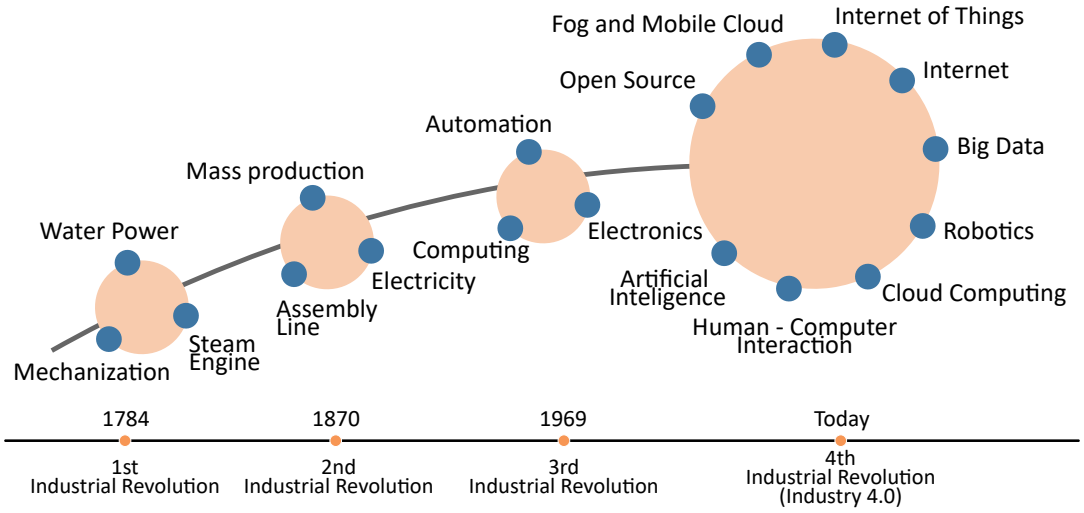


Figure 2.1: Evolution of Industry 4.0 and enabling technologies through time. Adapted from Aceto *et al.* [10].

solutions. Consequently, new trends regarding collaboration, innovation and problem solving will emerge. For example, global partnership achieved through the Internet to perform engineering and industrial design tasks with international, multidisciplinary teams.

Several aspects have lead to the importance of creating of a uniform industry standard and reference architectures. Among these aspects the following should be highlighted: integration of existing proven technologies with new ones; integration of different stakeholders; and the several limitations of Industry 4.0 implementation regarding security, connectivity, standardization and interoperability [7]. A reference architecture is a document which provides knowledge on the development, standardization and evolution of systems in a certain domain. It has the purpose to show the best practices, improve communication, promote reuse and consequently reduce costs and enhance interoperability between systems and/or subsystems.

There are several reference architectures for Industry 4.0 that differ in purpose, content and format. In Industry 4.0, can be found multiple use cases and challenges, each one with different requirements regarding the architecture. To better understand how the architectures frame in an industrial scenario Nakagawa *et al.* [7] compares the industrial automation pyramid to the several reference architectures.Following the same approach, a similar comparison will be made afterwards. Firstly, an overview of the industrial pyramid will be given, followed by a brief explanation of a selected reference architecture. Finally a comparison between them will be presented.

The industrial automantion pyramid (Figure 2.2) is divided in five layers: (i) field

layer, which comprehends physical entities such as machines, devices, sensors and actuators that interact in the production floor; (ii) control layer, representing field devices control and usually is represented by a Programmable Logic Controller (PLC) and/or a Proportional–Integral–Derivative Controller (PID); (iii) system/process layer which aggregates the systems that are responsible for the control of several devices at the industrial floor level. This layer is manly referred to as Supervisory Control and Data Acquisition (SCADA); (iv) operation layer that compiles Manufacturing Execution System (MES), responsible for the monitorization of the complete manufacturing process from raw materials to finished product; and finally (v) the enterprise layer comprised by the systems responsible for the integrated management of enterprises. It can be also called Enterprise Resource Planning (ERP).

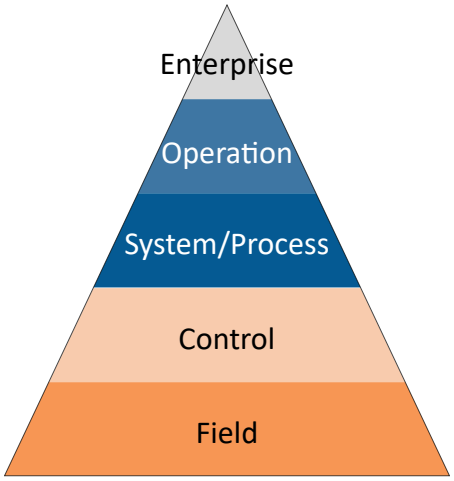


Figure 2.2: Industrial Automantion Pyramid. Adapted from Nakagawa *et al.* [7].

Usually, all architectures cover all layers of the industrial pyramid to a certain extent. Sometimes, each architecture module can cover more than one layer demonstrating that the clear division of such layers is not clear in Industry 4.0. Of the several architectures referenced in Nakagawa *et al.* [7] the Reference Architectural Model Industrie 4.0 (RAMI 4.0) will be highlighted for being one of the most popular, and an international technical specification (IEC PAS 63088:2017).

The RAMI 4.0 is a reference architecture, based on international standards (IEC 62890, IEC 62264, IEC 61512), developed for smart manufacturing, to insure that all stakeholders have a common understanding since they have a generic view of it. To make sure it fulfills the industry needs, it was conceived by a consortia of companies and research institutions who were in charge of design, experiment and adopt it [7]. RAMI 4.0 has a tridimensional architecture (Figure 2.3) whose axes represent Hierarchy Levels, Life-cycle Value Stream and Architecture Layers. The Hierarchy Levels axis intends to spread the idea of flexibility among machines and systems; distribute functions over the network

promoting communication and interaction among all the involved participants; and to include products as part of the architecture. The Product Life-cycle axis, as the name infers, defines assets along the value chain from its ideation through development and maintenance up to its production, usage and further maintenance after it is produced. The assets are defined as goods which have value for an organization as, for instance, a device or equipment. Finally, the axis of the Architectures Layers describes the modeling of physical entities of the industrial network (e.g., devices, equipment and machines) and matches them to their respective virtual representations, describing in detail the properties of CPS [8].

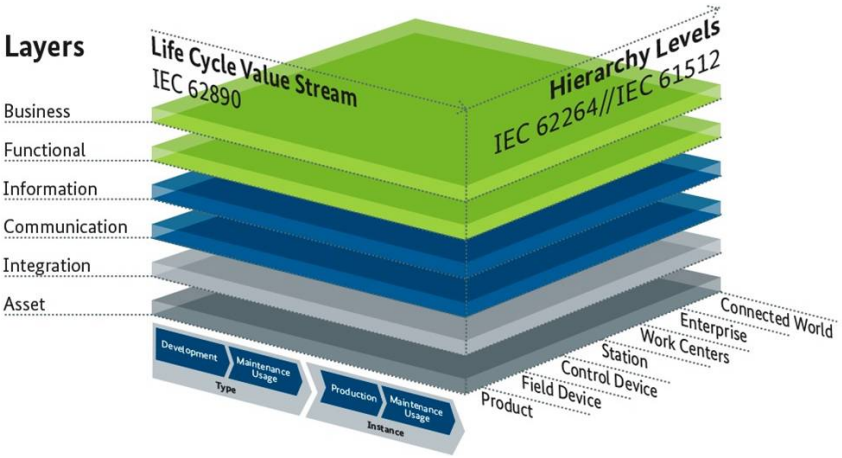


Figure 2.3: Reference Architecture Model Industrie 4.0 (RAMI4.0). Adapted from Contreras *et al.* [12].

Figure 2.4 demonstrates that some of the layers of RAMI 4.0 cover more than one level of the pyramid. It shows that the hierarchy between the different levels of the traditional pyramid are becoming less strict mainly due to the demand for flexibility and complexity by the new industrial paradigm.

2.2 Cyber Physical Systems

CPS are the systems composed by computational entities that connect deeply with the physical world, collaborating among them, and the associated processes. These systems use and provide internet available services, for accessing and processing data which allows them to achieve such connection [9].

In Industry 4.0, networked sensoring devices and software are already integrated with complex physical machinery [13]. This assimilation of not only CPS, but also IoT and Internet of Services with the purpose to control global networks, will result on an emergence where the physical and digital representation cannot be reasonably differentiated [14]. For instance, in the predictive maintenance area, process parameters of mechanical components (e.g., stress and production time) which are subject to wear and

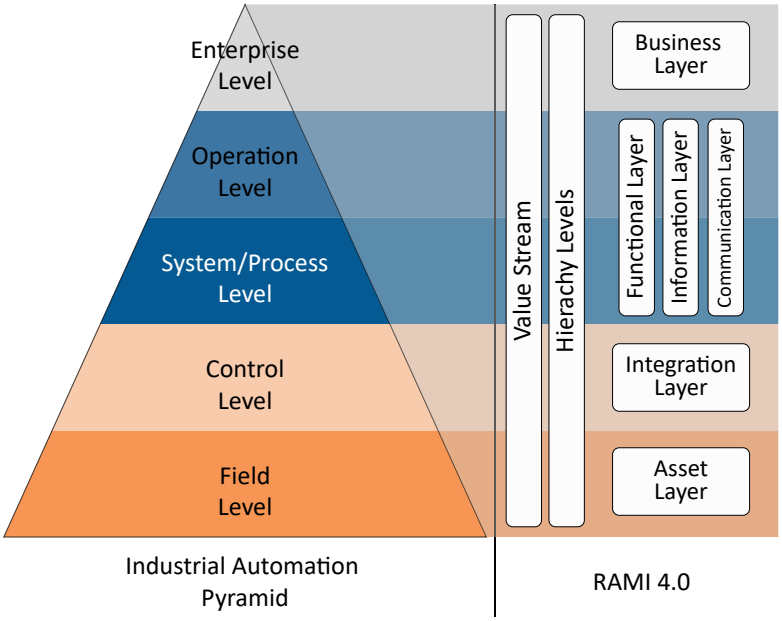


Figure 2.4: Mapping of RAMI 4.0 layers to the Industrial Automation Pyramid. Adapted from Nakagawa *et al.* [7].

tear are recorded digitally. The essential condition for ensuring the operation of these systems results from the integration between the physical element and its digital process parameters [15].

Industry is constantly changing. In the beginning of the 18th century, mechanical systems were the industry's focus. At the start of the 20th century, mass production started to being achieved in order to cope with the demands of a global market. Nowadays, the industry cannot rely on mechanical systems exclusively in order to achieve mass production. According to Industry 4.0, the transition from the linearity of a "value chain"to a "value network", where automation and dynamic are key aspects, relies heavily on CPS to interconnect product systems, infraestructures, and customers [10]. In Table 2.1, Lee *et al.* [16] provides a general overview of the differences between a factory in today's perspective versus an Industry 4.0 factory.

Table 2.1: Comparison between today's factory and an Industry 4.0 factory [16].

	Data source	Today's factory		Industry 4.0	
		Attributes	Technologies	Atributes	Tecnologies
Component	Sensor	Precision	Smart sensors and fault detection	Self -aware Self-predict	Degradation monitoring & remaining useful life prediction
Machine	Controller	Productibility & performance	Condition-based monitoring & diagnostics	Self-aware Self-predict Self-compare	Up time with predictive health monitoring
Production system	Networked system	Productivity & OEE	Lean operations: work and waste reduction	Self-configure Self-maintain Self-organize	Worry-free productivity

Usually a CPS is a centralized embedded computing system that contains a great

quantity of physical systems composed by wireless sensor networks. Therefore CPS can be characterized by four main components [17]:

i) **Physical System:** It englobates the hardware related aspects including, for example, hardware design, energy management, connectivity encapsulation and system testing;

ii) **Cyber and Information System:** Responsible for transforming the information provinient from the physical world into rules and models that can be interpreted by a software system. One of its requirements is the balance between a variety of factors such as concurrent design and formal verification, hierarchical storage systems, file systems, memory management, modular software design, network systems, real-time systems;

iii) **Product of Integration of Heterogeneous Systems:** Deals with time synchronization and space location issues of different parts;

iv) **Security requirements, real-time capability and predictability:** CPS shall be able to overcome the uncertainties of a system, be scalable and toletant to threats. Part of the goals of CPS are to attain a certain security level, be able to solve credibility issues in monitoring and controlling processes and real-time capability

2.3 Machine Learning

Algorithms are a fundamental part of solving computer problems. An algorithm is a set of instructions that transforms a given input into an output. Nevertheless, some tasks are very difficult or even impossible to be solved by a traditional algorithm. In many problems, only the input and the output are known to the user. For such problems, Machine Learning (ML) comes as a solution.

ML is a subset of Artificial Intelligence (AI) where computational methods can make predictions and decisions by learning from past information available, even without being explicitly programmed to do so. In machine learning, the process of learning is done by an algorithm, where parameters in a model are optimized using training data [18, 19].

Data is a crucial aspect of ML and it can take many forms. Quality and size are two key aspects of data that should always be considered for being vital to the success of the prediction algorithm [20]. One of the reasons for the ML popularity are the many petabytes of data that are generated nowadays, mainly due to the Internet phenomenon and how it has become part of our daily life. [21] The application of ML methods to large quantities of data is commonly called data mining, a process where large amounts of data are processed to construct a model. In these cases, the stored data reveals its usefulness when analyzed, and knowledge is inferred from it [19].

ML has a very intimate relation with data analysis and statistics. Statistical theory is fundamental to make the ML models infer from training data, evaluate the model's performance and filter the noise from data [22, 23].

A system is considered intelligent when has the ability to learn and adapt to change. If a system is able to do so when created, it does not need to have a solution for all possible situations it encounters. With the previous statement in mind, it is possible to find a vast

Table 2.2: Machine Learning techniques, adapted from Mohri *et al.* [20].

Machine Learning technique	**Description**
Classification	Assigns a category to each item.
Regression	Predicts the real value of each item.
Ranking	Learns to order a specific set of items according to certain criteria
Clustering	Partitions a set of items into subgroups.
Dimensionality reduction	Transforms a initial set of items into a lower-dimensional set without losing most of its properties

range of applications for ML including computer vision, unassisted control of vehicles (e.g., robots or cars, speech processing, analysis of gene and protein networks, network intrusions, medical diagnosis, design of recommendation systems, search engines, fraud detection) [20]. A large number of areas can benefit from ML as far as the correct dataset is provided to the algorithm.

Such a number of applications cannot be solved with a single ML technique, and quite a few learning techniques were developed. In the Table 2.2, it is possible to see some of the standard ML techniques which have been the subject of extensive studies.

To fully understand how a ML algorithm works, it is necessary to break down the learning process into its several stages and comprehend the vocabulary often used in the literature.

A typical learning process starts with a dataset of examples, instances of data necessary for learning and previous evaluation, which can be labelled or not. The dataset is then broken into training sample, validation sample and test sample. The training sample is used to train the model. The validation sample is necessary to tune the learning algorithm's parameters when labelled data is used. And, finally, the test sample is crucial to evaluate the algorithm's performance.

Before the training, the features (relevant attributes) are associated with the examples. After that, the training process will tune the values of the hyperparameters(Θ), specific inputs of the learning algorithm(A), creating a hypothesis which is a function that maps the features to a set of labels. The best hyperparameters(Θ_0) are chosen by calculating the algorithm's performance with the validation sample. At last, the hypothesis will be used in the test sample and evaluated using a loss function, which is a function that calculates the difference between a predicted label and a true label. If the hypothesis chosen fits the training examples perfectly but other examples are missclassified, it is called overfitting. Overfitting occurs when the model fails to generalize a problem and, consequently, fails to predict other examples [20]. The typical learning process is illustrated in Figure 6.1.

Usually, ML is divided into three types based on the learning "task"characteristics: Supervised Learning, Unsupervised Learning and Reinforcement Learning [22]. In Supervised Learning, labelled examples are used to predict their relationship, mainly for classification and regression problems. On the other hand, in Unsupervised Learning, unlabeled examples are used to learn about their distribution and focus on tasks like clustering, compression and feature extraction [21]. Both Supervised and Unsupervised

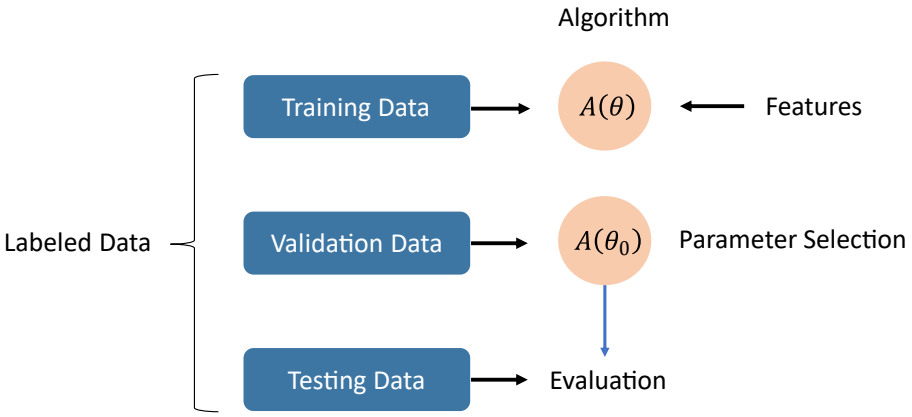


Figure 2.5: Typical training process. Adapted from Mohri *et al.* [20].

Learning will be the subject of the following subsections.

Reinforcement Learning is another type of ML, where the training and the testing phases are deeply interconnected. Every action has a reward, positive or negative. The objective is the maximization of the reward by interacting with the environment [20].

2.3.1 Supervised and Unsupervised machine learning

Supervised Learning maps the input space, constituted by a dataset of labeled examples, to an output space. Most of the Supervised Learning falls into two categories: regression or classification. In a regression problem, the output is real-valued. Otherwise, it is a classification problem. When trained, the model learns a classification rule, allowing to make predictions based on past knowledge [22]. In order to effectively test the rule's accuracy, unseen data is presented to the model and an unbiased evaluation is performed. The accuracy of the predictions is related with a probability so the model can classify with a degree of certainty [18, 22].

In some instances, where a probability calculation is wanted, the rule becomes an association between input and output. The classification rule is used as a describer of data, which gives an insight of the data and allows knowledge to be extracted. Supervised learning can also be used to detect outliers, as long as, after the rule is learned, it is possible to detect the instances that do not obey to such rule. Such data points can represent anomalies in a system that can be further analyzed [22].

In contrast to Supervised Learning, Unsupervised Learning uses unlabeled data (Figure 2.6). The objective of this method is to find patterns in the input data. Generally, a certain structure can be found in the input space, where certain occurrences and patterns can be found more often than others. In statistics, this occurrence is called density estimation. One method that uncovers such hidden patterns and groups similar data together is called clustering [19].

Clustering can be used to explore data and understand its structure or even to preprocess it. In preprocessing, clustering will map data to a new dimensional space where the new dimensionality can be greater than the original dimensionality.

Some advantages of Unsupervised Learning are its less cost, since it doesn't require labelled data, and it is more beneficial to detect anomalies and features possibly hidden in the data structure [22].

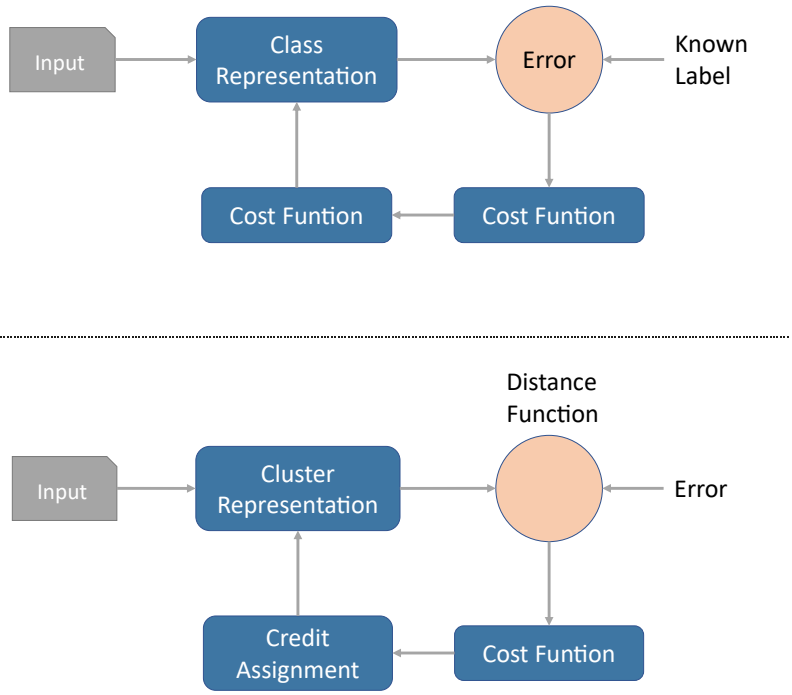


Figure 2.6: Supervised Learning model (up) and Unsupervised Learning model (down). Adapted from Zincir-Heywood *et al.* [22].

2.4 Deep Learning

One of the main challenges of AI is to solve tasks that the human being considers easy to solve but difficult to describe. Problems like recognizing elements on a picture, or giving meaning to written sentences are complicated for a machine. Deep Learning (DL) comes as a solution for those problems. In DL, a computer learns from experience, gathering knowledge without the need of a human specification. The name Deep Learning comes from the quantity of layers that computers use to learn complex concepts by hierarchically building them on top of simple ones [24]. Figure 2.7 illustrates the general differences in architecture between Neural Network and a Deep Neural Network.

DL is based on the concept of artificial neural networks, which began to be the subject of research in the 1940s. Artificial neural networks, or just Neural Network (NN), are

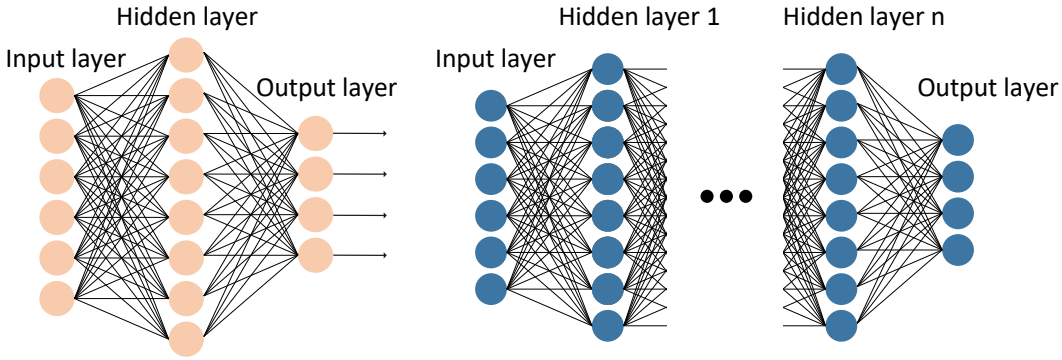


Figure 2.7: Difference between a Neural Network (left) and a Deep Neural Network (right). Adapted from Lee *et al.* [21].

based on the structure and characteristics of human neurons. They adopt a hierarchical structure of simply connected processors ("neurons") gathered in layers where each neuron produces a sequence of activations. The higher layer neurons' input is directly connected to the proceeding lower layer neurons' output through simple linear or non-linear calculations. An "epoch" or "episode" is a passage through all the NN. It occurs when an environment's input activates the first layer, and then information is paired to a respective weight. All the pairs connected to a neuron are summed or multiplied and passed through an activation function originating the output. As previously mentioned, the neurons' output is directly connected to the next layer neurons' input until it reaches the final layer [25, 26]. Figure 2.8 demonstrates the architecture of a simple neural network composed by three neurons.

The algorithm presented before is an example of a NN. Over the years, various DL models have been introduced and the more typical include the Autoencoder, Deep Belief Network, Convolutional Neural Networks and Recurrent Neural Networks [26].

Applying DL techniques to a given set can take a significant amount of time, depending on the computing and storage specifications of the machine running the algorithm. Nowadays, a plethora of accelerators with the capability to do specific tasks have been invented, such as arithmetic co-processors, sound cards and Graphics Processing Unit (GPU). Even more recently, the rise of new hardware optimized for ML and DL has allowed scientists to improve the performance of their models. Among those, the more popular are GPUs, Field-Programmable Gate Array (FPGA) and more recently, Tensor Processing Unit (TPU). The main advantage of this hardware, specifically GPUs is their parallel architecture, which allows increasingly fast computation involving matrix-based operations, typically present in many DL/ML implementations. To enhance this hardware, and take the full capability of its features for DL/ML, manufacturers offer accelerated libraries as a way to access the parallel power of GPUs. Among these, the most used are the NVIDIA CUDA®, NVIDIA CUDA® Deep Neural Network (cuDNN), Intel® MKL

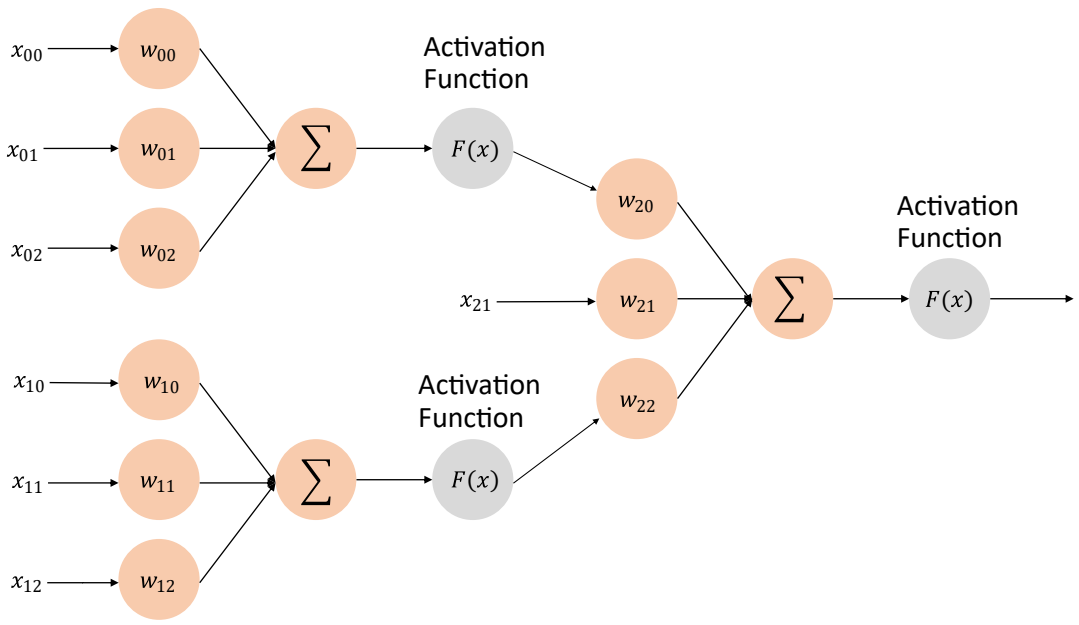


Figure 2.8: Architecture of a Neural Network composed by three neurons.

and OpenCL™[27].

The scientific community has quickly realized the advantages of DL and rapidly this type of learning [27]. With the purpose to create an easier way to design, train and validate DNNs with high-level programming interfaces, DL frameworks were developed [28]. These frameworks work on top of the previously mentioned libraries producing a significant advantage for the scientific community, both academia and industry. Some of the most popular frameworks are Tensorflow, CNTK, Torch, Caffe, MXNet and Theano. The fact that all the previously mentioned frameworks are open source, allows the community to access all resources to assist their work [27].

DL approaches to enable better performance in a wide range of applications. The applications are vast, from computer vision, including object detection, object tracking and image segmentation, to Natural Language Processing [26]. More and more areas are adopting DL techniques to improve the quality of their systems. Among them is worth noting the following: Financial Services, Financial Time Series Forecasting, Prognostics and Health monitoring, Medical Image Processing, Power Systems and Recommender Systems [29].

2.5 Federated Learning

Nowadays, the interest in AI is partially driven by the large quantity of existing data. With the Big Data phenomenon is expected that, by 2025, dozens of zetta-bytes will be

created only by IoT devices [30]. However, the availability of this data for AI purposes faces two significant challenges. Firstly, in the majority of industries, data exists in isolated form, and secondly, due to several cases of data compromisation, data security and privacy have become a worldwide concern. One of the major issues in most fields is data scarcity or poor-quality data. In addition, complicated administrative procedures related to data privacy and security and industry competition, makes it almost impossible to break barriers between data sources in different companies and even between different departments in the same company.

FL comes as a possible solution for the problems mentioned above. Unlike other ML solutions, in FL data is not centralized and not exposed. The idea instead is to each data source train their models locally, and only then those models are exchanged without the risk of data leakage, to form an aggregated global model (Figure 3.2). The accuracy of this final global model shall be very similar to the performance of a model trained with the same data centralized [2].

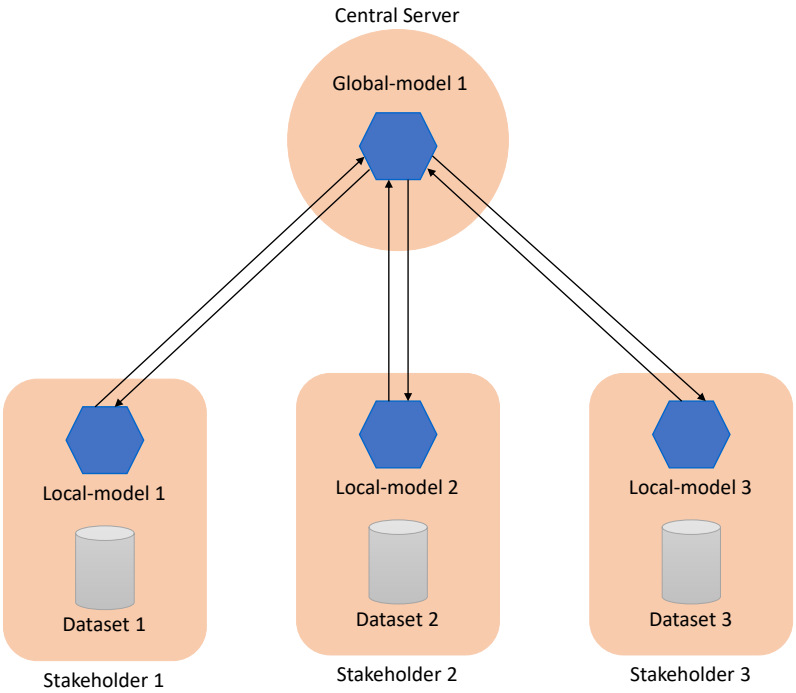


Figure 2.9: Federated Learning reference architecture. Adapted from IEEE Guide for Architectural Framework and Application of Federated Machine Learning [6].

2.5.1 Taxonomy

As a recent technology, FL is under active development with various techniques. In order to better understand FL approaches, and to guide the design of Federated Learning systems, it is important to understand some aspects of its taxonomy, as it is shown in

Figure 2.10. FL will be classified in the following paragraphs regarding data partition, network topology, and data availability.

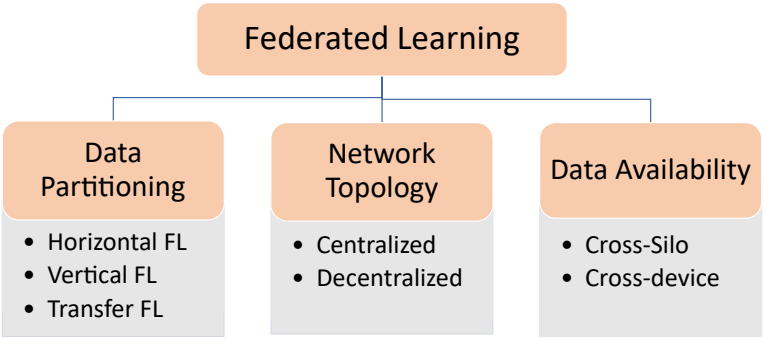


Figure 2.10: Federated Learning Taxonomy. Adapted from Li *et al.* [31].

Data Partition

When data is spread across several owners, its characteristics are also distributed, and the FL approach should be adapted based on the distribution characteristics of the data. Aspects like features and samples may be nonidentical. Based on how data is distributed across various entities. FL approach can be classified as Horizontal, Vertical or Federated Transfer Learning.

Horizontal Federated Learning (HFL), also called sample-based FL, is applied in cases where data shares the same features but vary in samples. In Yang *et al.* [32] is given an example of two regional banks where the intersection of costumers may be very small due geographic distance, albeit the same features are collected. These characteristics make HFL more likely to be used in situations where a large number of devices is involved.

In Vertical Federated Learning (VFL) (or feature-based FL), datasets have the same samples, but differ in features. In the example given by Yang *et al.* [33], a bank and an e-commerce company from the same city share a large propotion of customers, since they correspond to the residents of the city. However, the bank is interested in a certain number of features different from the e-commerce business, and vice-versa, only containing a small number of features in common.

In practical cases, it is usual to find situations where the features and the samples are very limited, for example, in cases related to finance and health care. In order to solve these situations, where each client can have its own specific feature space, FL and Transfer Learning can be combined, originating Federated Transfer Learning (FTL). In this approach, a model is trained to resolve a particular problem. However, by taking advantage of Transfer Learning properties, the trained model can be applied to different problems that may share similar properties. The performance of FTL is strongly correlated with the interrelation between the problem's domain, making it a suitable approach for many

FL scenarios where most of the stakeholders are correlated not only by the nature of their industry, but also by the nature of their problems [4, 6].

Network Topology

Regarding network topology, FL can be categorized as centralized or fully decentralized. Centralized FL, differs from a traditional ML approach mainly because data is not transferred to a server where the model is trained. Instead, all the stakeholders train locally their models with their respective data and only the parameters/weights are shared with the central server, protecting each of the stakeholders data from being compromised.

On the other hand, in a fully decentralized FL approach, the necessity of a centralized server and the global model concept are discarded. A third party authority is only needed to establish the protocols to be followed during the FL process. To substitute the necessity of a central server, peer-2-peer communication is implemented, and a set of algorithms are used to ensure reliability and trust among stakeholders [34].

Data Availability

Initially, FL was only applied in mobile and edge devices. Since FL has gained visibility, an interest has been developed to involve groups of fewer stakeholders, essentially focused on organizations, to collaborate in training a model. To face the differences in data availability between both approaches, two data settings for FL where minted: cross-device FL and cross-silo FL [35].

In cross-device FL, a large quantity of edge devices are comprehended, including IoT or even mobile phones. Mainly due to the nature of the devices, most of the data share the same features, being most appropriate for the use HFL. In scenarios where there is a great amount of spread devices, only a fraction of the clients are available in a given time frame making it impossible to identify each individually. Although the density of devices, in the majority of cases unreliability is a problem, mainly due to battery, network, or idleness issues. The resource allocation strategies were the found solution to overcome these problems. In particular, client selection and device scheduling should be employed to maximize the contributions to the final model.

In contrast to cross-device FL, cross-silo FL comprehends fewer clients, typically organizations with distributed data centers. Although there are fewer clients, they are more reliable and able to be addressed individually, and consequently there is more data available to train. In cross-device FL cases, where large amount of data from fewer clients is concerned, the employment of encryption techniques is preferred as a way do address data security and privacy issues. [34].

2.5.2 Algorithms

Algorithms play a key role in integration, optimization, aggregation, and achievement of consensus in the different approaches of FL. These algorithms may vary depending

on the architecture used in specific FL scenarios. In centralized FL, aggregation algorithms are fundamental to gather all local model updates and optimize them. Some algorithms already account for privacy issues, preserve communication bandwidth or support asynchronous updates from clients [34]. This section is aimed to review some of those algorithms in order to understand how they work and the advantages they can bring to the FL system. In particular, the following algorithms will be briefly presented: Federated Averaging (FedAvg), Federated Stochastic Variance Reduced Gradient (FSVRG), Co-Op and FedOpt.

Federated Averaging (FedAvg)

The FedAvg algorithm begins with the initialization of the global model and its distribution throw a subgroup of clients. The clients who received the model update perform the updates using, usually, the Stochastic Gradient Descent (SGD) algorithm for optimization. Finally, the clients transfer every updated local model to the central server, which will be create the global model by computing the weighted sum of all the received models.

Federated Stochastic Variance Reduced Gradient (FSVRG)

FSVRG is another algorithm with synchronized model updates. The FSVRG starts with the download of the global model by all the clients and respective training with each local data. After that, the clients upload the gradients to the central server, where the full gradient is formed. The full gradient is downloaded by the clients, and each client initializes their local model and their local step-size. The process is followed by the creation of a permutation of the local data. Subsequently, the clients will perform as many Stochastic Variance Reduced Gradient (SVRG) updates as the number of training examples held by each client. This process is interactive and use, not only the local step size, but also the full gradient. At last, when all the local models are computed, the central server forms a new global model similarly to FedAvg.

Co-Op

In contrast to the algorithms mentioned above, the Co-Op algorithm proposes an asynchronous approach. Each client starts optimizing their algorithm with its training data. Afterwards, the client requests the global model from the central server. When having the global model, the client makes a comparison between the ages of both models. In case the local model is outdated, the client fetches the global model. On the other hand, if the client is overactive, the training continues. Only if the client's model's age is between a certain threshold this model is merged with the global model [36].

Federated optimization (FedOpt)

The FedOpt algorithm works similarly to the FedAvg algorithm, with the difference that the server and the clients are "gradient-based" optimizers. With the FedOpt algorithm, the client aims to minimize the loss function with its local data, while the server allows for the usage of adaptive optimizers for aggregation. The usage of these optimizers, for example, YOGI, ADAM and ADAGRAD, provides for the integration of adaptivity to the FL process, improving FL's convergence [37].

2.6 Privacy Preserving Federated Learning

In order to get a distributed FL approach, it is mandatory the protection of the involved data. Each stakeholder must be assured that their data is secured and their privacy concerns are met. Privacy Preserving Federated Learning, as the name indicates, is a combination of privacy preserving techniques with FL. In order to guarantee this requirements, it is necessary to understand the threats to privacy and security and find solutions to mitigate these threats.In the implementation of privacy preserving techniques to FL, a challenge is the balance between data privacy and model accuracy, as privacy mechanisms might affect the performance of FL models [5].

2.6.1 Data Security

To better understand and tackle security issues in FL, it is crucial to identify its key elements, including vulnerabilities, threats, and the defense techniques that may be used to counter such threats.

2.6.1.1 Vulnerabilities

Vulnerabilities are weaknesses in a system that can be exploited unauthorizedly. To build a more secure system, it is essential to identify vulnerabilities and implement defense mechanisms. According to Mothukuri *et al.* [34], there are five primary sources that can be considered as weak points to exploitation:

1. FL involves a significant amount of client-client and server-client communications. As so, insecure communication channels are an open vulnerabilities.

2. In FL, the possibility of having numerous clients translates into numerous opportunities for data and model parameters manipulation. Each client can be a vulnerability in the system if it is itself vulnerable.

3. The central server is responsible for aggregation and sharing models and updates for clients. An attack on these server is a major vulnerability has it connects directly to all the clients.

4. The aggregation algorithm should also have the capability of identifying abnormal updates from clients. A weaker aggregation algorithm which disregards abnormal or even malicious updates presents itself as a vulnerability to data manipulation from the clients and consequently as vulnerability to the system.

5. Finally, the architecture of a FL framework is a weak point, since it can be attacked, promoting data exposure. Examples of possible failures are the cases when the architecture is not correctly implemented, or when the privacy and security concerns are not completely fulfilled.

2.6.1.2 Security Threats

A threat is a possibility to exploit a weakness in a system, violating the security and the privacy of a system. In FL, a threat usually intends to access sensible data or even the training procedure, which can compromise the FL process. Some of the main threats to FL are analyzed in this section.

Poisoning

Poisoning occurs when an element of the FL process is tampered at critical points, and this attack can occur at different levels. For example, data poisoning occurs where malicious data is injected into a client and affects his model. On the other hand, model poisoning is where a malicious model is sent for aggregation, poisoning the global model. Finally, the data modification occurs where the clients' data are changed. For example, the label swapping in a client is a case of data modification.

Poisoning attacks become more relevant in FL due to the high number of stakeholders involved in the process, increasing the probability of success of this attack [34].

Backdoor attacks

In backdoor attacks, a participant in the FL process may replace the global model with another, keeping the accuracy, but controlling the model's performance on a specific subtask. The attacker has a direct influence on the weights of the global model for being part of the FL process. He can modify the weights of his local model or even incorporate the evasion of defences into the local model loss function. Since the model's accuracy is kept, this threat is relatively difficult to identify [38].

Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN) are primarily used in image classification algorithms. In a FL environment, the attacker creates a replica of the global model which classifies fake computer-generated images. While the attacker continues receiving the global model updates, the generator algorithm will improve until high-quality is reached,

and the fake images are able to interfere with the global model. These images can poison the global model, affecting the performance on specific classification targets [39].

System disruption IT downtime

System disruption is an unavoidable threat that can be exploited to steal information from the FL environment. Although the training can resume after the system disruption without any malicious interference, as it is an non planned disruption, it is a threat to be considered and specific measures should be applied [34].

Malicious Server

A central server takes a key role in several FL architectures. Compromising this server, would be nefarious, since an attacker could extract the client's private data, manipulate the global model, or even take advantage of the shared computational power to do malicious tasks [34].

Communication Bottlenecks

Communication is a key point of FL, and in an environment with heterogeneous data, one big challenge is to preserve communication bandwidth. The usage of complex ML algorithms, or model uploads of large quantities of data by numerous users, can lead to a severe bottleneck in communication [40].

Free-riding attacks

Free-riding attacks are carried by stakeholders, who do not contribute to the update of the global model. In FL, every stakeholder contributes with its local training to update and further enhance the global model. The free-riders are clients who generate fake weights which are uploaded to the global model (free-riding attack). This attack can occur for several reasons: the lack of the required data, concerns about stakeholders privacy privacy and shortage of computing power sometimes by non-using it for the FL process [41].

Eaves Dropping

An eavesdropping attack occurs in the communication round of FL. Ineffective communication between clients and the server can be created when the proper safety measures are not met. This attack is highly improbable to happen once the system architects verify the right configurations [34].

Interplay with data protection laws

This threat can happen due to misconfiguration of the FL environment. The occurrence of this threat is almost improbable to happen, since the global model is verified adequately before being deployed [34].

2.6.1.3 Defensive Techniques

Defensive techniques are applied to mitigate the possibility of a FL system suffering from an attack, or even lower the risk of being attacked. These techniques which prevent model corruption cannot necessarily avoid training the model with malicious data, but ensure that the model does not overfit that data minimizing the damage taken by the model. Two types of defences can be identified: proactive, which identifies threats and risks prematurely, and reactive, which identify attacks and takes defensive measures. Several defensive techniques can be employed in a FL system to prevent specific attacks. Table 2.3 presents a summary of the most popular techniques.

Table 2.3: FL security defensive techniques adapted from Bouacida *et al.* and Mothukuri *et al.* [34, 42].

Defensive techiques	Description
Sniper	Based on euclidean distance, it excludes adversarial updates
Federated Destillation	Transfers knowledge from a fully trained model to another model. It saves computational cost and enhances robustness by transfering knowledge instead of the weights
Anomaly Detection	Monitors suspicious client updates in order to prevent malicious attacks
Moving Target Defense	Reduces the likelyhood of a successful attack by randomizing the FL modules.
Trusted Execution Environment	A trusted ecosystem which provides integrity and confidentiality by atesting anf verifying the code.
Pruning	Reduces the size of the Neural Network model by droping some of its neurons. It reduces complexity, improves acuracy and prevents backdoors
Federated Multi-Task Learning	Trains models for multiple related tasks in simultaneous to enhance fault tolerance

2.6.2 Data Privacy

One of the major concerns of FL is the privacy. Although FL maintains the client's privacy by utilizing model parameters instead of data, it is still possible to suffer some privacy attacks due to training data revelation based on the parameters. For a better understanding of how privacy concerns should be addressed it is important to know the threats faced and the technique that can mitigate those threats. In this section threats and defense techniques will be addressed as well as the cost of implementing privacy techniques in a FL system.

2.6.2.1 Threats

Although all the security threats should also be addressed to privacy it is important to denote two significant threats that affect mainly privacy, as they can be used to reconstruct users' data. Those threats are Membership Inference Attack (MIA) and GANs & Reconstruction Through Inference which will be addressed in this section

Membership Inference Attacks

A MIA infers the aspects of data through misusing the trained model. In MIA,an attacker deliberately misuses the global model to obtain the model output. By analyzing several inputs and the respective outputs, the attacker can infer and predict the original training data, risking the system's privacy [43].

GANs & Reconstruction Through Inference

GAN were mentioned before as a security attack. In a privacy scenario, GANs can be used to generate data from the global model, retrieving sensitive information that was originally used to train it. This attack allows the user to take advantage of FL global model to infer the training data used by other FL stakeholders [34].

2.6.2.2 Techniques to mitigate threats and enhance privacy

The FL system should employ privacy-preserving techniques and algorithms to prevent attacks on privacy and maintain data integrity. The most mentioned techniques in the literature are presented in this section.

Secure Multi-Party Computation

Secure Multi-Party Computation (SMC) is used in models where multiple clients are involved, preventing that each client does not know anything except the input given, and the respective output. The main objective to not leak any knowledge, which it would require complicated computing protocols, and may not be efficient. Although attainable, in some scenarios, partial data leakage is preferred if security is guaranteed [2].

Differential Privacy

It is a method that adds noise into data and/or uses generalization algorithms to hide private and sensitive attributes, making it impossible to be restored. In FL, this method hides a stakeholder contribution to the global model. However, this method still requires data transferring which may be a liability [2].

According to [35, 44, 45], we can summarize the definition of differential privacy as follows: An algorithm A is differentially private if for all $S \subseteq Range(A)$, and for all the

adjacent datasets D and D':

$$P(A(D) \in S) \leq e^{\varepsilon} P(A(D') \in S) + \delta \tag{2.1}$$

Where $P[.]$ represents the probability and D and D' are decentralized datasets called adjacents if D' differs from D by the addition or subtraction of a record. The privacy loss (ε, δ) is the quantifier of DP, where the smaller the (ε, δ), the greater the privacy of the algorithm.

Applying DP to any given dataset to enhance data privacy often recurs upon adding a certain amount of noise to data using noise generation mechanisms such as Gaussian or Laplace mechanisms [44].

In FL, it is possible to distinguish between two definitions of DP, global differential privacy (GDP) and local differential privacy (LDP). In Liu et al. [44], the distinction is made by dividing the perspective of who adds noise to data. In FL approaches where the central server is responsible for applying the DP mechanism to the global model are called GPD approaches. On the other hand, when each client is responsible for adding DP to their models and protecting their privacy, we call it an LDP approach.

While GDP follows Eq. (1), an LDP approach is held by the following:

$$\frac{P\left[A_n(D_n) \in S_n\right]}{P\left[A_n(D_n') \in S_n\right]} \leq e^{\varepsilon_n} \tag{2.2}$$

Where n means the n-th participant.

Homomorphic Encryption

It allows every client to compute functions in encrypted data by using parameter exchange and encryption mechanisms. Homomorphic encryption lets mathematical operations be done in encrypted ciphertexts. This encryption does not transmit either data or model, protecting these details from malicious clients [2, 35].

Adversarial Training

Adversarial training protects FL models from evasion attacks that try to inject fake data and disturb the models. This protection is done by trying all permutations of an attack at the beginning of a training phase. This technique improves the privacy of users' data, and makes the ML model more robust to attacks, even minimizing the threat of data inference [34].

2.6.2.3 Costs of privacy

To boost privacy preservation in FL, the cost in accuracy and efficiency needs to be accounted. A study mentioned in [34] shows that FL with Differential Privacy (DP) has a cost in accuracy in heterogeneous environments. DP adds noise to the parameters to ensure privacy, which inevitably affects accuracy. In conclusion, privacy directly impacts

accuracy and efficiency, the level of privacy shall always be adjusted to the models, to achieve the desired accuracy, privacy, and security requirements.

2.7 Challenges and Gaps

Being a relatively recent technology, FL is in current development. Its application to an industrial scenario is still conceptual, and several challenges are yet to be mitigated.

As an effort to create a FL solution to a smart industry scenario, in 2021 Franco *et al.* [46] proposed a self-adaptive divided framework to cope with the existing industrial automation systems architecture as well as the ML procedures used in the industry. The focus of this work was the improvement of accuracy and efficiency of a FL framework. In order to achieve these improvements, the authors proposed modelling a multi-assignment optimization problem and, subsetting the datasets through the involved devices.

Model accuracy is also the focus of the work of Liu *et al.* [47]. However, the effort to achieve the desired accuracy relies on a Gradient Compression Mechanism where local gradients are compressed to reduce the number of gradients exchanged between the clients and the central server.

In 2021 I-Kai Wang *et al.* [48] created a framework for an industrial scenario based on Federated Transfer Learning. This framework was focused on adapting the base industrial knowledge to each smart device's needs using FTL to local specific data. The data division was made through every device, making each one a client of the federated learning. This approach leads to every device's specific adaptation of the global model suited to its particular needs.

The works mentioned above, propose a Federated Learning framework applied to an Industrial scenario, focusing only on efficiency and adaptability to smart manufacturing. Data privacy problems are only addressed by using a FL system. In addition, with the information presented in the previous section, it is safe to assume that only the FL system is not enough to protect data privacy it is also necessary further measures to safeguard each FL client's data.

With the focus on privacy preservation, Jiang et al.[49] proposed the use of membership proof in the FL system. Such membership proof is generated by cryptographic accumulators and is issued as a smart contract by the server on the blockchain. This technique ensures robustness to failures, verifiability of the clients and resistance against active adversaries.

Another similar approach is the work of Hao *et al.* [50] which also uses encryption mechanisms to ensure privacy, and the communication between all the elements of the framework is encrypted. Although the high level of encryption and the computational cost, the framework still achieves a decent accuracy in training the MNIST dataset.

In Liu *et al.* [51], privacy and security of FL users were the primary concern. A blockchain-based FL framework was proposed, where attackers are recognized by the execution of smart contracts, defending the system against poisoning attacks. This study

proved the trade-off between model performance and client protection, which although ensures better protection against attacks, also underperforms compared to a simpler FL model.

The previous studies focus primarily on the security and privacy of FL models. Although an improvement in privacy is noticeable, a deprecation on accuracy is evident. In addition, all the above-mentioned frameworks create a FL system from scratch. However, applicability to a real case scenarios is not a concern as data collection and preparation is not mentioned. In the testing of these frameworks, the used datasets were manly MNIST and CIFAR, which do not translate the challenges of an industrial scenario where data is more complex and even may be untreated. Finally, beyond the literature, none of the frameworks were available for further improvement or testing, making it difficult to reproduce and implement.

CHAPTER 3

Proposed Framework

The present chapter will provide a full description of the proposed framework. The framework's primary goal is to face industrial scenarios while facilitating cooperation between stakeholders and accounting for privacy preservation and security. Not only to ensure the proposed capabilities but also the minimum requirements of quality, safety and measurability, this framework will follow the guidelines of the standard IEEE Std 365.1™-2020 [6] on the creation of Federated Learning (FL) frameworks.

This chapter will start with the framework requirements, emphasising functional and non-functional requirements. Then the fundamental taxonomy aspects that must be considered in the framework building will be explained. An overview of the framework will be provided along with its mapping to the reference architecture and correspondent system layers. Finally, the framework's composing modules and interactions will be described.

3.1 Requirements

This section will briefly explain the functional and non-functional requirements necessary to achieve the proposed work. According to the standard ISO/IEC/IEEE 24765:2017 [52], the functional requirements state the results that a particular process should produce. Therefore, the functional requirements of the proposed framework are the following:

- **FR-01**: The framework shall include data collection nodes and the ability to communicate with a database system to store the collected data safely;
- **FR-02**: Include data preprocessing capabilities to fit the Machine Learning (ML) training requirements. The framework needs preprocessing capabilities to fit the

raw data present in the database to the ML algorithm requirements;

- **FR-03**: Support privacy-preserving techniques. The framework shall implement privacy-preserving algorithms to secure each stakeholder's data during the FL process;
- **FR-04**: Optimise each stakeholder's model performance. The use of the framework shall serve each stakeholder in improving its ML models allowing for decentralised training;
- **FR-05**: Allow metric monitoring for model improvement. The framework shall implement model evaluation on the trained models to guarantee that the previous requirement is met;
- **FR-06**: Provide the collaborative improvement of ML models through the usage of FL capabilities. The framework must implement a FL system that collaboratively improves each stakeholder's ML models using FL techniques.

Additionally, non-functional requirements establish constraints on how the system will execute its processes [52]. The establishment of the non-functional requirements was adapted from the guidelines of the IEEE Std 3652.1-2020 [6]. Considering these guidelines and based on the procedures explained in the previous section and the functional requirements mentioned above, the following list of non-functional requirements is presented.

- **NFR-01** (Efficiency): The framework should be efficient concerning time and memory consumption;
- **NFR-02** (Security): The framework should guarantee the stakeholders' data security;
- **NFR-03** (Privacy): The framework should provide means to perform collaborative ML while keeping each stakeholder's data private;
- **NFR-04** (Model performance): The framework should provide a competitive ML model performance compared to a centralized approach.

3.2 Taxonomy

For the proposed FL framework to achieve the capabilities mentioned at the beginning of this chapter it will require a foundational architecture that provides a starting point to be adapted to several scenarios. According to the standard IEEE Std 365.1™-2020 [6], the reference architecture (Figure 3.2) presents a guideline which sits on three main classes of entities, that should be addressed when creating a FL framework: data distribution, user roles, and system modules.

3.2.1 Data

When data distribution is concerned, as explained in section 2.5.1, it can be classified as Horizontal Federated Learning (HFL), Vertical Federated Learning (VFL) or Federated Transfer Learning (FTL). Each type of data distribution carries unique features that the framework's FL setting shall overcome. The data differentiators and colliders across heterogeneous clients must be considered when federating the ML training.

3.2.2 Users

Regarding user participation, the guideline for creating a FL framework involves four essential roles that users may play. It is important to mention that each user can play more than one role, depending on the implementation scenario. As explained in the standard IEEE Std 365.1™-2020 [6], the four roles are the following:

Data Owners

The data owners are responsible for collecting, preparing, and training data. It is also its responsibility to maintain the privacy and security of the local model, which will be provided for aggregation, by recurring to privacy-preserving techniques and algorithms. Finally, the data owner is responsible for participating in the training and evaluation rounds and communicating the intermediate results to other parties and the server during the inference phase.

Model Users

The model users have particular interest in the federated learning process. In the great majority of cases, they are the ones who will benefit from the use of the trained model and the FL services.

Coordinator

The coordinator works as an administrative entity in the federated learning process. It is responsible for various procedures, including: developing the algorithm, the infrastructure, and the service; coordination of the train and test of the model; privacy-preserving and secure computing through, for instance, security protocol determination, key generation, and data decryption; and model management including training and testing.

Auditor

The auditor is responsible for verifying the legitimacy of the data providers, requesting explanations from the coordinators on their data management, model management and economic incentive activities and monitoring the federated ML model building process. In this dissertation, the auditor role will be played during the development and use of the proposed framework.

3.2.3 System

According to the standard IEEE Std 365.1™-2020 [6], the composing modules of the reference architecture are grouped by layers, according to their functional relevance. It is important to note that the use of specific modules may depend on the requirements of each use case, so that some modules may be included or omitted from a framework. In the particular case of this dissertation, only the modules required for a framework in an industrial scenario will be discussed. The reference architecture is divided into five layers: Service layer, Operator layer, Algorithm layer, Infrastructure layer, and Cross-layer.

In the **Service layer**, the business logic is implemented. All the management of participants, tasks and services are managed in this layer, providing every user access to the functionalities needed for each role they may play in the federated learning system.

In the **Operator layer**, the responsibility is to implement the characteristic operations of a federated learning algorithm. This layer shall implement the machine learning algorithm and the optimisation and aggregation algorithms. Also, it is in this layer where the encryption and privacy preservation algorithms are implemented.

In the **Algorithm layer**, the algorithmic logic for federated learning is implemented. It includes data structuring, model selection for the specific scenario, and evaluation metrics for performance, efficiency, privacy preservation and security. The outcome of the capabilities of this layer shall assist in the decision-making and troubleshooting process as it provides essential data for both tasks.

The **Infrastructure layer** is responsible for supporting the operations of the FL framework. It provides a computing component, a storage component, a communication component and an interface between these components, this layer, and the operation layer.

In the **Cross-layer**, the interaction between layers is realised, offering supporting capabilities such as security functionalities, service and strategy management, and regulation and auditing of the system.

3.3 Overview

After presenting the guidelines in Section 2.5.1, and the functional and non-functional requirements, the developed framework will be composed of four modules. Three modules will be associated with each stakeholder as the base operation flow within them. In contrast, one of the modules will have a similar role as a FL aggregation server.

As the framework was built following the guidelines from the IEEE Std 365.1™-2020 [6], it can be mapped to the reference architecture system's layers, and a clear user distinction of roles can be identified. Figure 3.2 depicts the framework's architecture and a brief description of the interactions between the modules. In addition, it can be visualized the system layers and role mapping of the proposed framework to the reference architecture.

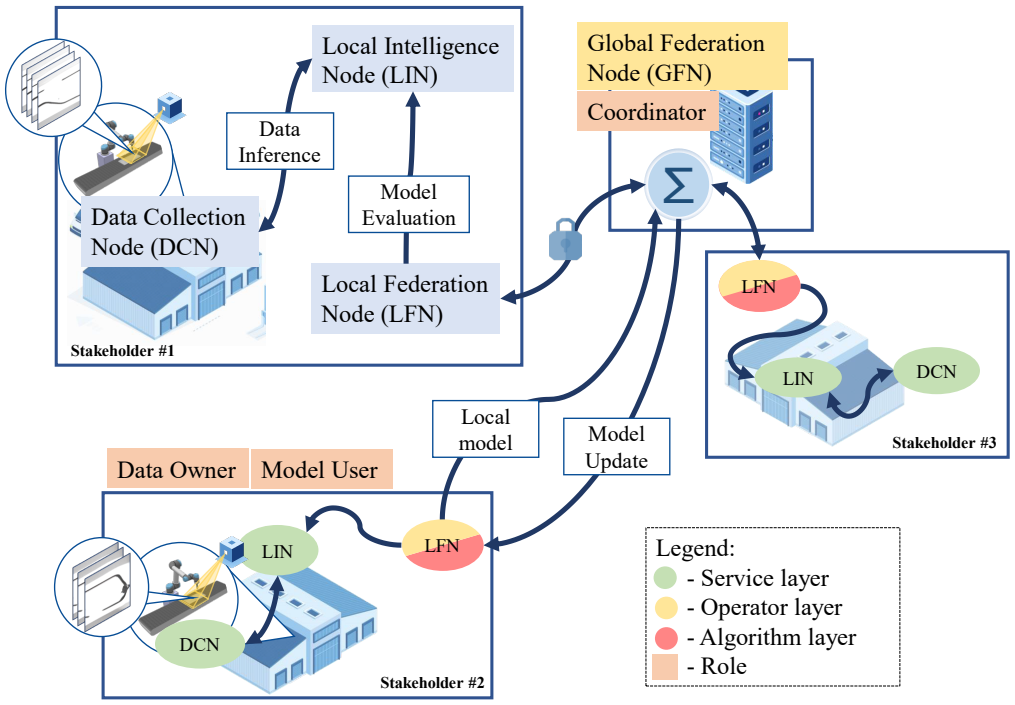


Figure 3.1: Proposed framework with system layers and role mapping to the reference architecture.

3.4 Component Specification

As shown in Figure 3.2, each stakeholder contains at least three different nodes: DCN, LIN and LFN. The Global Federated Node (GFN) acts as an aggregator for every stakeholder in the FL context. In this section, each module will be further explored.

Global Federation Node (GFN)

The GFN is a comprehensive implementation of the aggregation server in a FL context. Its role is to manage the stakeholders involved in the FL process and provide a secure mechanism where each local model, trained by its respective stakeholder, is aggregated by an aggregation algorithm. Consequently, it does not have direct access to the stakeholders' data

The GFN shall be agnostic to the used algorithm and the type of machine learning algorithm. Also, the GFN is responsible for initialising the parameters used by the stakeholders providing a common ground for the model training. These functionalities map the GFN component to the operator layer.

Local Federation Node (LFN)

The Local Federation Node acts as an extension of the FL client. It is responsible for training the local dataset and participating in the FL training and evaluation rounds.

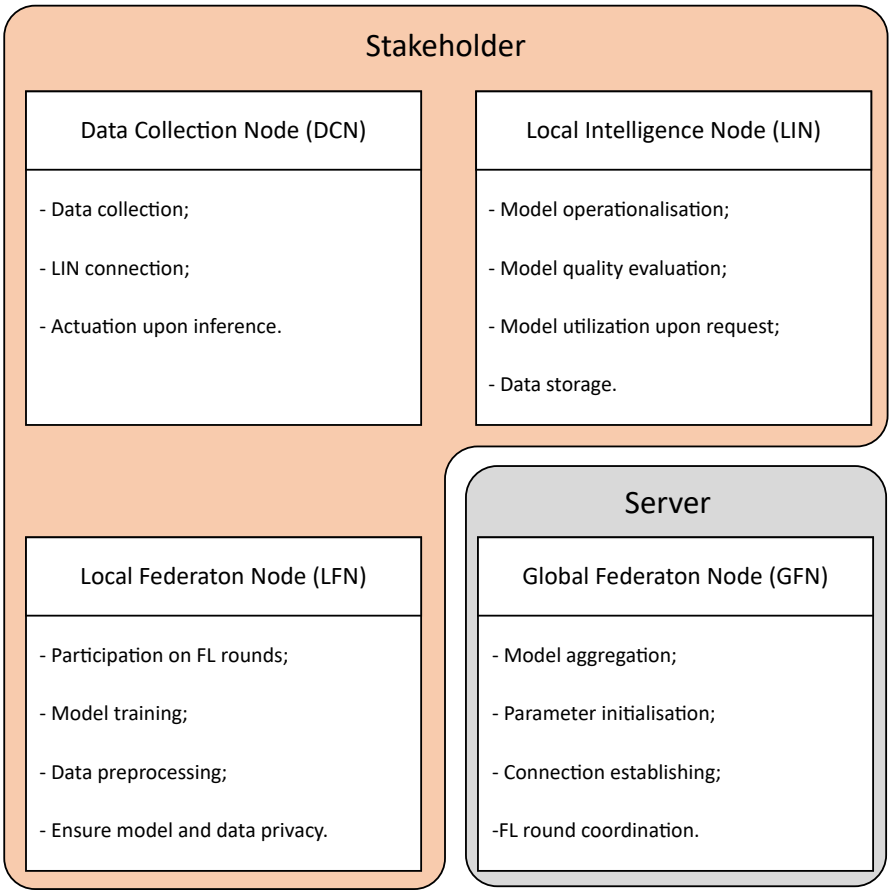


Figure 3.2: Composing nodes of the proposed framework and correspondent main functionalities.

It is also responsible for delivering the model to the respective LIN nodes. This node provides the functionality to make secure connections to the database for data retrieval and preprocessing it to be used in the FL process. The LFN also uses evaluation metrics to verify the resulting model's performance and privacy intake according to the stakeholders' quality parameters.

However, when considering a straightforward approach to LFN-GFN interaction, a problem arises as it is required that each stakeholder relies on the GFN. To address this problem, the LFN implements privacy algorithms such as Homomorphic Encryption or Differential Privacy.

As mentioned in Section 2.6.2.2 with HE, data can be computed while still encrypted. The LIN node can homomorphically encrypt the respective model updates before sending them to the GFN. By taking advantage of the properties of HE, the GFN may aggregate the encrypted updates without accessing the raw values from the stakeholders. After that, the GFN send the. updated global model back to the stakeholders, where it will be decrypted and used in the next training round.

On the other hand, by using differential privacy, as explained in the Section 2.6.2.2, it

is possible to add an amount of noise to data while keeping its usability for ML training. This process introduces plausible deniability to data, safeguarding each stakeholder from being associated with it.

The LFN can be mapped to the operator and algorithm layers when considering the privacy-preserving algorithms, the data preprocessing functionalities, and the model selection techniques it implements.

Local Intelligence Node (LIN)

The main functionality of LIN is the operationalisation of the FL model. The LIN node receives the updated model and is responsible for checking its quality compared to the existing model, employing the best-suited model for inference. Also, this node is responsible for fusing data from multiple associated DCNs and storing the data in a secure database. When considering its data evaluation and the provision of the trained ML models for inference, this node can be mapped to the algorithm and service layers.

This node's functionality allows multiple implementations, including agent-based or service-based approaches, which will be later discussed in Section 4.3.

Data Collection Node (DCN)

Finally, DCN is a key part of digitalising a physical component. It provides the means for run-time data to be collected and then used for training, evaluation and inference.

Upon collecting data, the DCN runs a data cleaning method following an established data structure each stakeholder defines. When the collected data may present missing values, the DCN relies on a mechanism of data imputation to standardise the data saved in the database. This methodology allows a generic procedure for structured and unstructured data to be stored and used in future applications when suitable.

Data collection and cleaning functionalities map the DCN to the service layer.

3.5 Interaction Specification

After establishing the framework requirements and presenting the composing components, it is essential to tackle the interactions aggregated with each module. Regarding the component interaction, it is possible to divide them into three main flows: data flow, ML model flow, and FL flow.

3.5.1 Data Flow

Data flow sums up the interaction between the DCN and the LIN. As shown in Figure 3.3, in this process, data is collected in the DCN model, sent to the LIN node, where the ML model is located, and the inference is made. After the inference, the collected data is stored in a secure database that stays until needed. This interaction process is

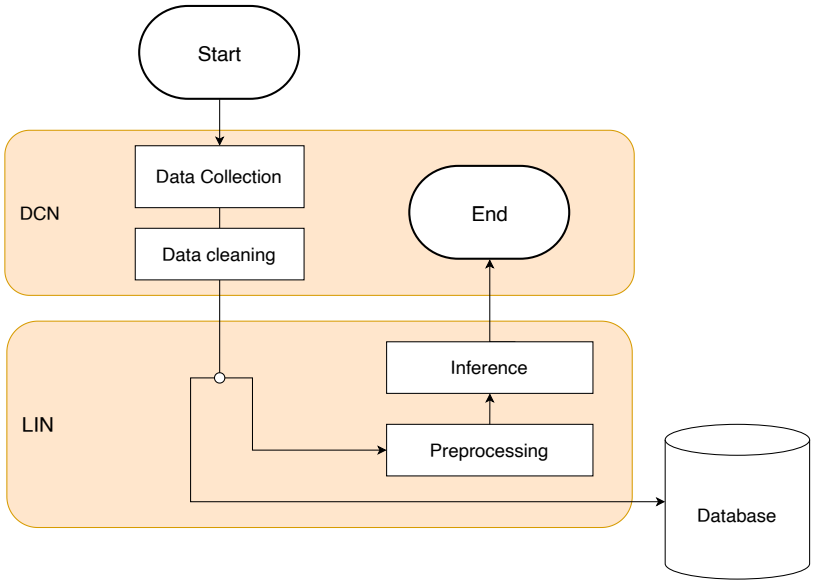


Figure 3.3: Data flow between the DCN and the LIN.

deliberately simplified and agnostic to the implementation technology, making it possible to implement, for instance, agent-based or service-based approaches or another approach that the stakeholders might find suitable for their specific industrial scenario.

3.5.2 Model Flow

The Model flow is one of the most important interactions on which the proposed framework is based. As shown in Figure 3.4, the LFN node starts by accessing the necessary data to train the ML model and preprocesses it to match the ML training model criteria. After being ready to begin the Federated training of the model, the LFN signs into the FL training by acknowledging the GFN of its availability. When all the necessary stakeholders are ready to begin the FL training, the GFN sends the initial model parameters, and each stakeholder trains the model with its data. Upon finishing the training round, each stakeholder sends the parameters to the GFN, where the chosen aggregation algorithm aggregates all the parameters, creating the global model. This global model is then sent to each stakeholder as the new starting parameters, where the cycle continues until the agreed number of global training rounds finishes.

When the last global model is sent to each stakeholder, the LIN node verifies it by evaluating and comparing its performance against the already employed model. If it outperforms the current model, the new ML model is saved and employed in the LIN nodes that will use it.

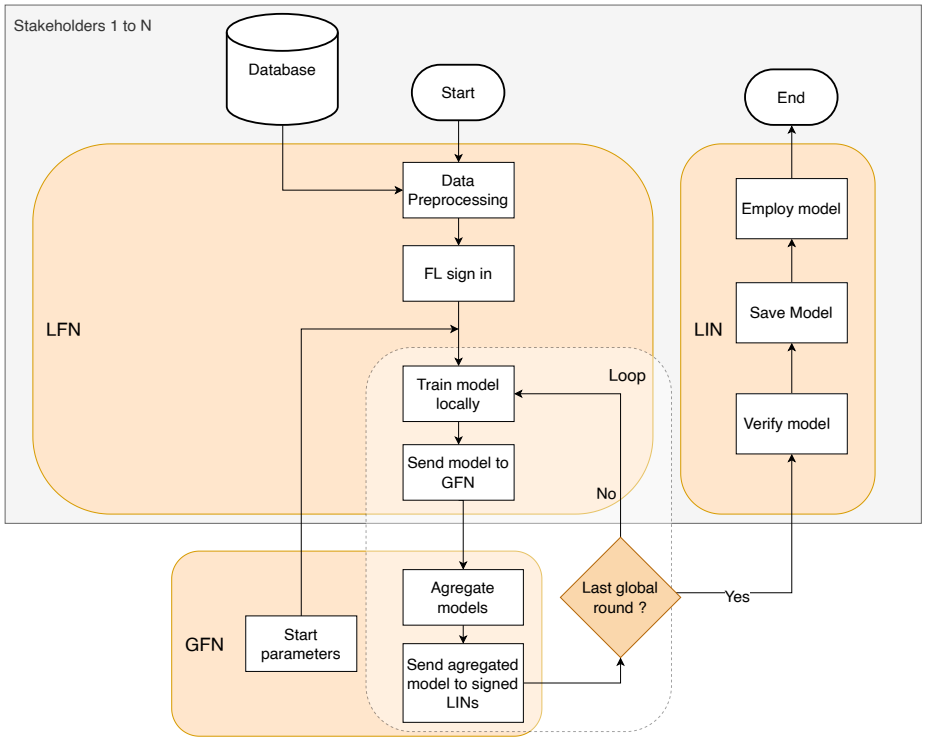


Figure 3.4: Model training and evaluation flow.

3.5.3 Federated Learning Flow

Finally, the FL interaction characterises the FL process associated with the decentralised training of the ML models. This process allows the collaborative improvement of ML learning models by federating the ML train. This data flow follows the guideline of the reference model presented in the standard [6].

Figure 3.5 presents an example of the FL system applied in this framework. It is essential to mention that the shown example only assumes two stakeholders and two training rounds to reduce the reading complexity. Both these values can be escalated accordingly.

The FL flow is applied when several stakeholders desire to train a common ML model without needing to share data and take advantage of the FL benefits . To do so, each stakeholder's LFN sign in to the GFN, demonstrating its availability to initiate the FL training. After each sign-in, the GFN responds with the initialisation parameters, which provide a common starting ground for the local training. Upon receiving the starting parameters, a global round is initiated where each stakeholder trains the algorithm with its data. After finishing the local training, each stakeholder sends the respective model's parameters to the GFN, where the aggregation occurs according to the chosen algorithm. When the aggregation finishes, the new global model parameters are sent to the stakeholders, and

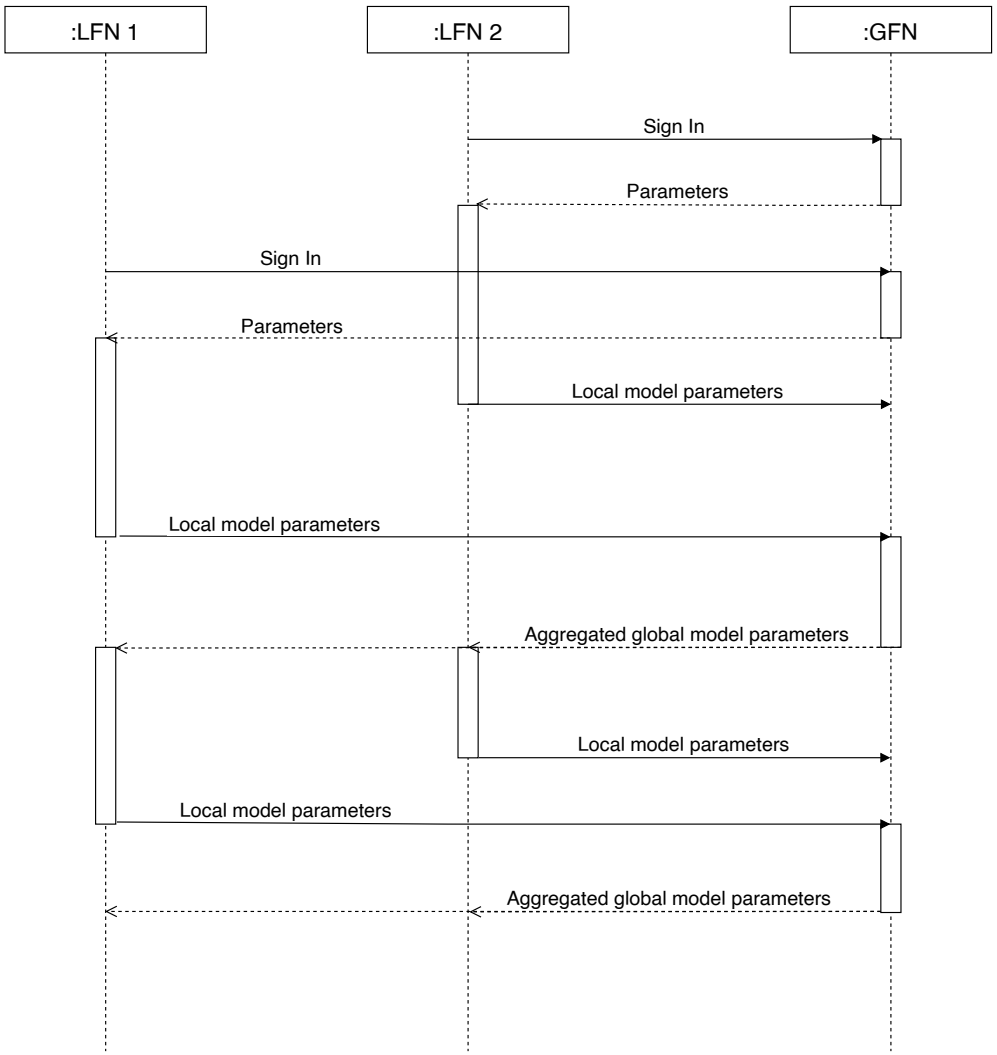


Figure 3.5: FL interaction between the GFN and the connected LFNs.

a new round begins. When all the rounds have finished, the stakeholders keep the last global model parameters which is the final trained model.

Although not mentioned in the Figure 3.5, an evaluation round can be included at the end of each training round to evaluate the global model training progression.

It is relevant to mention that the training parameters (e.g., number of global rounds, ML model, type of dataset, aggregation algorithm, number of stakeholders which will participate) are previously agreed upon, following an economic system.

Although creating an FL economic system sits out of this dissertation's scope, its presence must be accounted for. The importance of an economic system relies on the presumption of fairness between all the stakeholders, even though they might not bring the same value to the FL process. Economic incentives (whether monetary or influential) should be attributed to the stakeholders proportionally to their contribution to the FL

process to tackle this issue.

Chapter 4

Implementation

Recalling the framework architecture presented in Figure 3.2 and the system interactions, it is possible to divide the framework into three action stages: data collection, model update and Federated Learning (FL) training. Each of these stages involves specific functionalities and interactions.

This Chapter will present the implementation of each component mentioned in the previous chapter, the architectural decisions and the tools adopted throughout the development of this framework.

4.1 Case study

The case study, which will provide a starting point for building the framework, is based on a real industrial application (Figure 4.1) adapted from Peres *et al.* [53]. In this scenario, different defects can occur in the adhesive cord, which an automated quality inspection system should detect. The classes are depicted in Figure 4.2 and represent two types of defects: discontinuity and excess.

This case scenario aims to serve the built framework with a real scenario to demonstrate the impact that a multi-stakeholder collaborative approach based on FL can have when compared to the performance (in terms of accuracy) of local, single-stakeholder solutions.

For this purpose, the chosen dataset[1] comprised 372 images, 207 images of the *Discontinuity* class, and 165 belonging to the *Excess* class. It was split into three subsets attributed to each client considered in this case study. Two subsets were artificially manipulated to predominate one of the defects and the third subset was built with a minimal amount of data. This division pretends to emulate a scenario where each manufacturer

[1] https://github.com/RicardoSPeres/federated_learning_iros_2022

has limited access to data for a specific type of defect and an abundance of data from the other or limited data for both defects, enhancing the likelihood that each manufacturer will benefit from a collaborative approach.

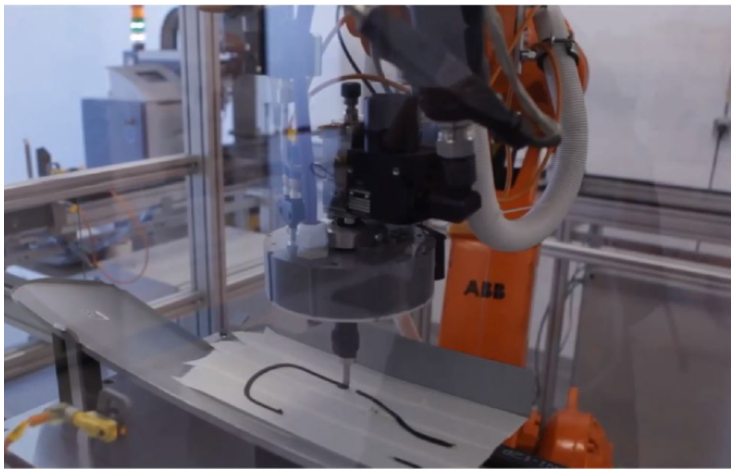


Figure 4.1: Reference industrial application for the case study [53].

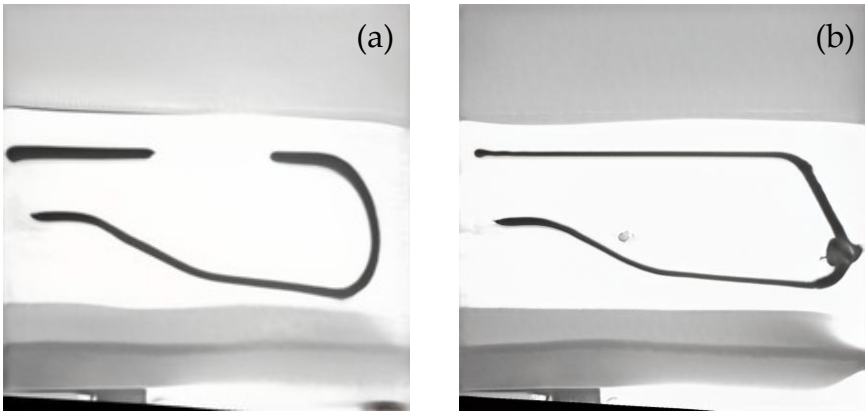


Figure 4.2: The two classes of images contained in the study case dataset. Discontinuity (a) and excess (b).

4.2 Technologies

Implementing the proposed framework to validate the case study requires hardware and software components. This section discusses the devices and the main tools chosen in developing the framework nodes, the FL component, and the network layer that integrates the components.

4.2.1 Hardware devices

The hardware devices used to run the framework were two Personal Computers (PC). A PC with an Intel® Core™ i7-9750H CPU @ 2.60GHz, 16GB of RAM (15.9 GB usable) and an NVIDIA® GeForce RTX 2060 with 6.0 GB of dedicated memory graphics card was used to run the Global Federated Node (GFN). Furthermore, a PC with an Intel® Core™ i7-9700K CPU @ 3.60GHz, 16.0 GB (15.8 GB usable) and an NVIDIA® GeForce RTX 2070 with 8.0 GB of dedicated memory graphics card was used to run three stakeholders, each with one Data Collection Node (DCN), one Local Intelligence Node (LIN) and one Local Federation Node (LFN). Both PCs were connected in the same network environment through an ethernet cable.

4.2.2 Software tools

Python: The programming language chosen to implement the framework was Python, a general-purpose, high-level, interpreted programming language. The choice of Python over other programming languages was due to its simplicity, great range of use due to its many community-made libraries, support of major machine learning and federated learning frameworks and an active developer community who can help solve problems that arise during implementation.

SQLite: SQLite was the chosen database engine to implement each stakeholder's database mainly due to its accessibility through Python language being the data condensate in a file, reduced complexity and versatility.

Flower: Flower is the FL framework chosen to implement the FL system. This open-source, end-to-end, federated learning framework offers the tools and functionalities to perform experimentation with a high level of customization. Flower presents several advantages compared to similar frameworks, which include scalability, client-agnosticism, privacy-agnosticism and flexibility. Also, it is possible to implement the client side of this framework in heterogeneous environments, which presents an advantage for a manufacturing setting. Finally, one major advantage of this framework is the capability to integrate different Machine Learning (ML) frameworks in the same workload, allowing different clients to work on different training frameworks while using the same FL environment [54].

TensorFlow: TensorFlow presents itself as an end-to-end open-source platform for implementing machine-learning algorithms. It offers different levels of abstraction and flexibility to build and train ML models. Also, it presents an API (Keras API[2]) which makes the implementation of ML solutions intuitive. The user-friendliness and the gradual learning curve were the fundamental aspects of the choice of this framework in the implementation.

[2] https://keras.io/

4.3 DCN – LIN interaction

The implementation of the DCN – LIN interaction was made recurring to a client-server model, where the chosen communication protocol was HTTP. For that purpose, in the LIN script, a Flask server was created with a simple API to take requests from the associated DCN, which will act as a client in this process. The HTTP protocol was chosen as a proof of concept of the interaction between the DCN and the LIN. This choice is not exclusive and does not prevent the possibility of another choice of implementations as long as the base architecture is respected.

4.4 The data collection node (DCN)

As mentioned in the Section 3.4, the Data Collection Node is responsible for collecting and cleaning data and sending it to the connected LIN.

In this implementation, the data collection process was abstracted, therefore the collection method was substituted by a folder containing the respective dataset where the respective images were saved. The DCN script collects the image and converts it into a data stream. The data stream is then added to a JSON file, and an HTTP POST request is built and sent to the LIN. After sending the HTTP POST request, a response is collected to be sent for the respective actuation function. Denote that the data cleaning process was skipped due to the type of collected data. A cleaning process should be implemented if a different type of data is collected.

4.5 The local inference node (LIN)

In the LIN, the data collected is fed to the ML model, which classifies it and saves it into the database, sending back the result of the classification to the DCN. Also, the ML model is evaluated and updated in this node.

The LIN script was implemented as an HTTP server recurring to the Flask framework. For the development of the API, the REST architectural style was followed to promote interoperability among the developed nodes.

As demonstrated in Table A.1 the LIN API provides two POST methods. An inference method is associated with the DCN and the data flow, and a model evaluation method is associated with the LIN and the model flow. Further detail on the API endpoints can be found in Appendix A.

Table 4.1: LIN API methods.

Method	**Route**	Requirements
POST	<server_url>/img_inf	image file
POST	<server_url>/model_eval	.h5 model file

In the inference method (Figure 4.3), the DCN provides a JSON message with an image string encoded in base64. The JSON format and the existence of the image are vital elements for the acceptance of the request, being verified at the beginning of this method. If the elements are confirmed, the base64 string is converted to bytes and back as a PIL image. Before being fed to the model for prediction, the image needs to be preprocessed and converted into an acceptable format. This process is made available through TensorFlow, which offers functions for preprocessing images by leveraging the Keras API. After the prediction, the result is assessed, and the image and respective label are encrypted and saved into the database in the expected format with the corresponding label. Finally, the prediction results are returned to the DCN.

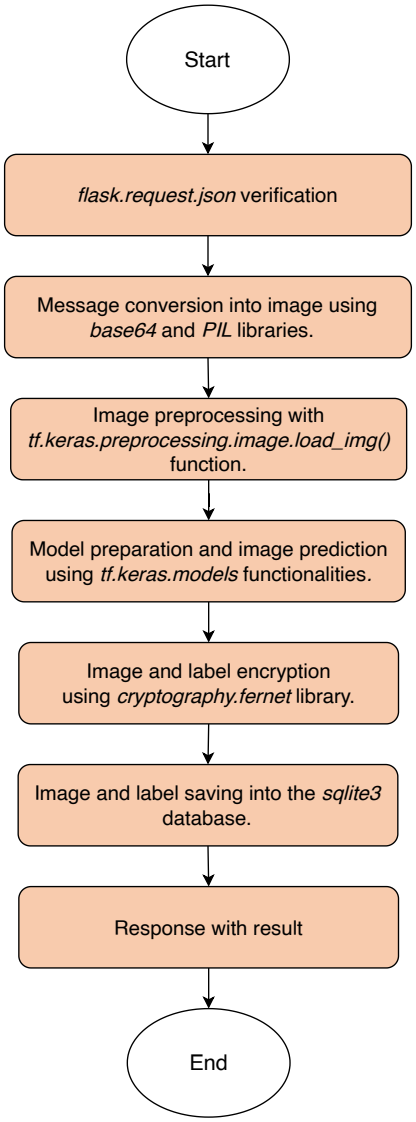


Figure 4.3: Inference behaviour.

On the other side, in the model evaluation method (Figure 4.4), the LFN, after participating in the FL training rounds, gathers a final ML model and sends it to the LIN through the *model_eval* method. In this method, the LIN receives an ML model and sets it for testing, where a pre-determined number of images are selected to evaluate the new model's accuracy. Then the results are compared to the existing model present in the LIN. If the new model outperforms the currently employed one, the LIN will substitute it, and the performance parameters are updated for further evaluations

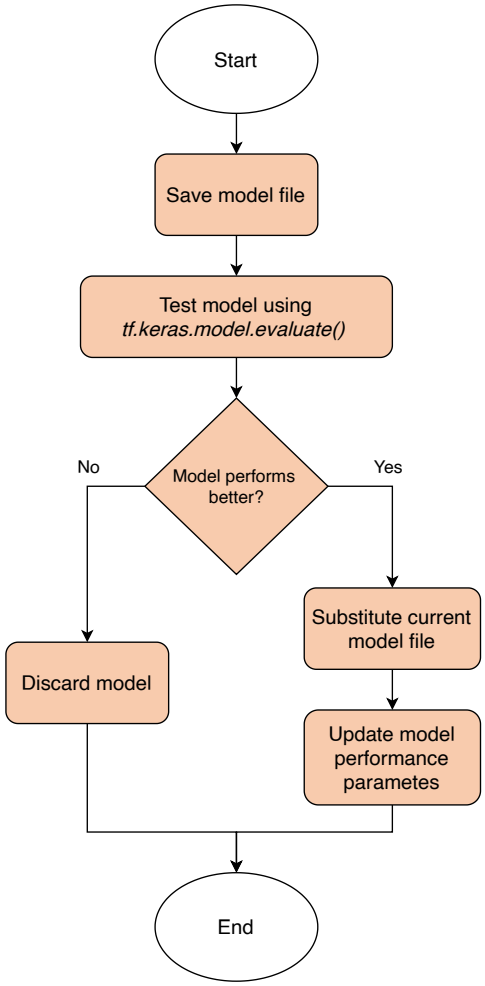


Figure 4.4: Model evaluation behaviour.

4.6 The local federation node (LFN)

As mentioned in Section 3.4, the LFN is the node responsible for participating in the FL process and implementing the privacy algorithms to prevent data leakage from the client side of the FL process.

However, the FL process is a rather complex process to implement on its own, so by leveraging the Flower framework, it is possible to implement a robust FL system customizable for the needs of this project.

4.6.1 Data preprocessing

To initiate the FL process, each involved stakeholder needs to preprocess its data for training the ML model. The LFN starts by connecting the respective stakeholder's database and collecting the data necessary for the training. To preprocess the data, the Keras API provides functionalities for image preprocessing. As the optimization of the ML model is not the main focus of this project, the preprocessing consisted of the following steps:

- Image resizing into a shape of (224, 224, 3) which is the required input image size for the used model;
- Image rescaling in a factor of 1/255 to lower the image's RGB values to a 0-1 scale allowing the model to process these values;
- Division into a training set and validation set fit and evaluate the ML model, respectively;
- Shuffling the training set to minimize the training loss.

4.6.2 Deep Learning model construction

The purpose of this case study, as mentioned in Section 3.4, is to make an image classification for two classes (discontinuity and excess) in the provided training set. In order to provide the same baseline in this case study, a deep learning classification model architecture must be implemented for every stakeholder.

TensorFlow provides a plethora of pre-trained deep learning model architectures through the Keras API [3]. From the provided architectures, the MobileNetV2[4] was chosen due to being a lightweight, high-performing visual recognition model with low computational cost.

The creation of the Deep Learning (DL) model for the case study started with the application of the MobileNetV2 as a base model, with the parameters described in the table 4.2, followed by the addition of two layers, a *GlobalAveragePooling2D* layer and a *Dense* layer with a dimensional output space of two (corresponding to the two classification classes) and a softmax activation function.

[3] https://keras.io/api/applications/

[4] https://keras.io/api/applications/mobilenet/#mobilenetv2-function

Table 4.2: MobileNetV2 utilized parameters.

Paremeter	Input	Description
input_shape	(224, 224, 3)	MobileNetV2 required input shape that must have three input channels, width and height
include_top	False	Boolean input which tells wheter to include the fully-connected layer at the top of the network
weights	'imagenet'	Weights initialization file.

4.6.3 Flower client architecture

Creating the FL framework using Flower involves two steps, the definition of a Flower client, the definition of a Flower server and its corresponding strategy (the latest will be described in further detail in Section 4.7). In the LFN, each stakeholder defines its Flower client, which will provide the functionalities to participate in the FL training. This section describes the details of the Flower client's creation in the LFN.

4.6.3.1 Interface

The interaction between the Flower clients and the server is made through an interface implemented in the clients called *Client*. This interface executes necessary methods each client needs for FL training; whenever each method is needed, the server calls for it to each client.

Flower provides a class named *NumPyClient* to implement the *Client* interface. As shown in Figure 4.5, the *NumPyClient* already provides several methods that each client can build upon and customize according to its specific requirements.

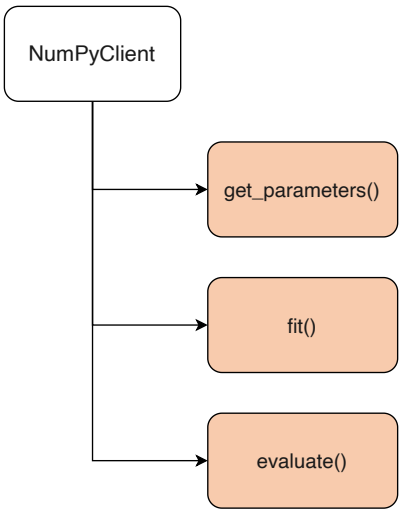


Figure 4.5: Customizable methods provided by the *NumPyClient* class.

4.6.3.2 Client Sign in

Upon server readiness, each client must sign in for federated training. Flower abstracts this process from the user by providing a gRPC Remote Procedure Calls (gRPC) connection between the server and respective clients. When a stakeholder wants to connect to the server to enter the FL training, it must initiate its *Client* interface by calling the method *start_numpy_client*() and passing, as parameters, the server's IP address and its *NumPyClient* class with the DL model, the training and validation data.

When the *NumPyClient* class is instantiated, it starts by adding the model configurations necessary for training in the *__init__*() method. These configurations include a stochastic gradient descent optimizer function, a categorical cross-entropy loss function and the evaluation metrics. The chosen evaluation metrics were Accuracy, AUC, Recall and Precision.

4.6.3.3 Training

The local training of each stakeholder is implemented in the *fit()* method of the *NumPyClient* class. As depicted in Figure 4.6, the *fit()* method starts by obtaining the parameters sent by the server. The first time this method is called, each stakeholder receives the initialization parameters from the server, creating a common baseline among all the FL clients. The other times the *fit()* method is called, each stakeholder starts with the aggregated model and trains the new local model on top of it.

To proceed with the DL model training, each stakeholder can choose to get the hyperparameters from the server or use their own hyperparameters. If the first option is chosen, the server passes the hyperparameters in a configuration dictionary (explained in Section 4.7) available to every client.

The DL model training is then started using the TensorFlow fitting function, where the model, dataset and hyperparameters are passed.

For evaluation of the model training, it was chosen to retrieve the resulting metrics of the fitting function and save them into an Excel file.

Finally, the parameters of the newly trained model, the number of training samples used and the collected results are returned from the *fit()* method to the server for aggregation and evaluation.

4.6.3.4 Evaluation

The Flower framework allows the model evaluation process to be done in the server or the clients. The choice of the model evaluation in the server would imply that the GFN had access to all clients' data to perform an independent evaluation. The scope of this dissertation is to create a framework that prioritizes FL clients privacy while leveraging the benefits of FL, so the access to data by the GFN would represent a major privacy breach that would invalidate this project's purpose.

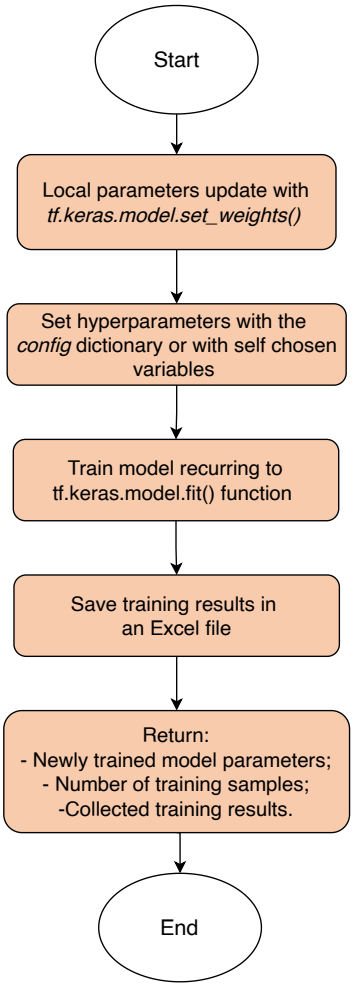


Figure 4.6: *fit()* method behaviour.

To perform a decentralized evaluation of the global model in the *NumPyClient* class, Flower offers the *evaluate()* method where each client can use their own test set to perform the model evaluation. Although the evaluation is done locally, each stakeholder can choose from using the server configuration or their own.

In the *evaluate()* method, the global model parameters are received and passed to the local model. As shown in Figure 4.7, the global parameters are then assigned to the local model, which in turn is used by the Tensorflow model's evaluation function. The resulting metrics are saved in an Excel file for each user and returned by the *evaluate()* function.

4.6.3.5 Differential Privacy

Although a FL learning approach introduces an extra level of privacy to a distributed ML approach, it still presents some vulnerability points. Even though the proposed framework addresses the implementation of both the FL client and the aggregation server,

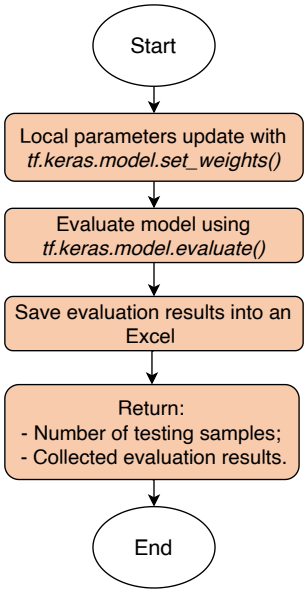


Figure 4.7: *evaluate()* method behaviour.

it is plausible that a stakeholder still does not feel comfortable participating in the FL process. In order to add an extra layer of privacy to the stakeholders, this framework allows the use of differential privacy in the local model training.

Recalling differential privacy described in Section 2.6.2.2, it introduces a certain amount of noise to data to enhance its privacy, which can be done locally or globally. In this framework, the choice of local differential privacy was made by considering that the server might not be trustworthy so that each stakeholder may choose if it wants to train its model with Differential Privacy (DP) or not.

The implementation of DP in each stakeholder in the Flower framework involves the readjustment of the *NumPyClient* class for accepting the training with DP and providing the necessary extra functionalities needed in the differentially private training of the DL model. The choice between differentially private training and regular training is made by a boolean variable called *DPSGD*. As seen in Figure 4.8, this variable controls the training flow by changing the model parameters and using or not the additional methods for the differentially private training.

Figure 4.8 shows both the regular FL flow and the differentially private FL flow. As the regular FL flow functions on the LFN side were explained above, now the focus is redirected to the differentially private FL flow and the additional methods it brings.

In the DP training, each stakeholder needs to provide a set of additional custom configuration parameters specific to this training, namely the number of *MICROBATCHES*, the value of $L2\_NORM\_CLIP$ and the $NOISE\_MULTIPLIER$ value.

When the DP train flow starts, data verification is necessary, precisely the quantity of examples used by each stakeholder. The chosen DP optimizer depends on an even choice

of three factors, the *MICROBATCHES*, the *BATCH_SIZE* and the number of training samples, which each stakeholder can configure. The *MICROBATCHES* must divide evenly the *BATCH_SIZE*, which must be a multiple of the chosen number of samples in the dataset. In the differentially private mode, for the *NumPyClient* to accept the training dataset, the training batches must equally distribute the number of examples used in the local training. Figure 4.9 shows an example configuration a stakeholder must consider when training its local model with DP.

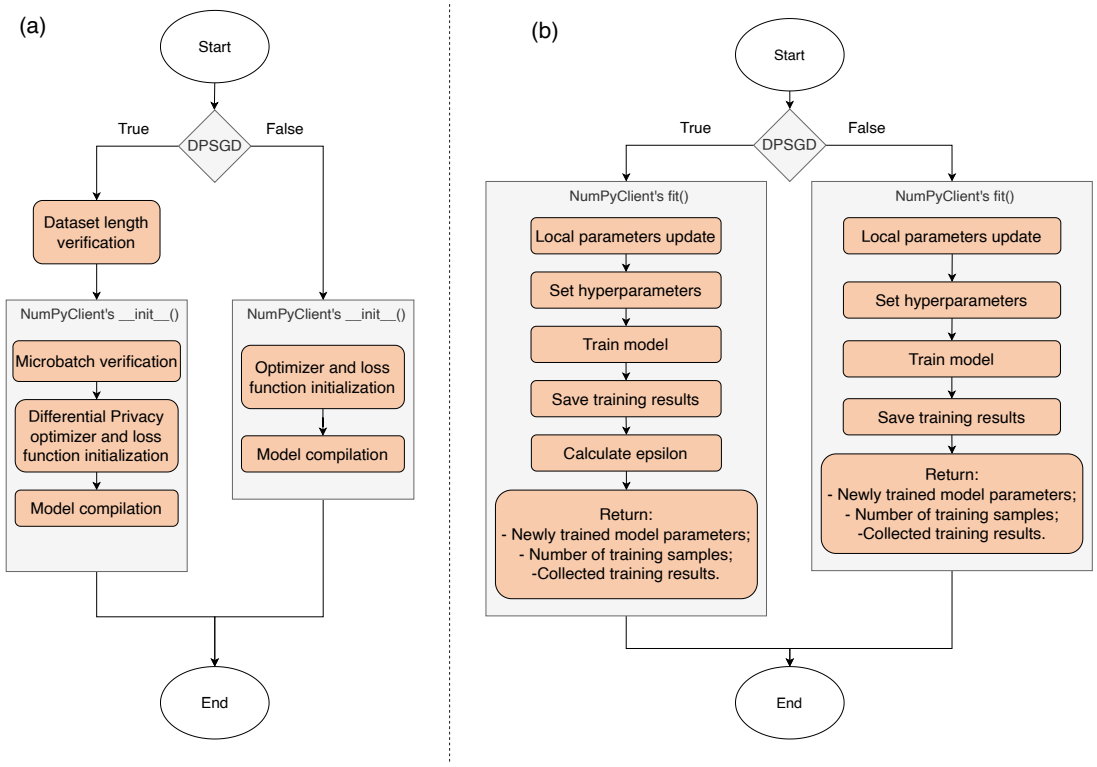


Figure 4.8: Comparison between the addition or not of differencial privacy in the *NumPyClient's* $\_\_init\_\_()$ method (a) and *fit()* method (b).

Once the number of *MICROBATCHES* is verified, the *NumPyClient* class initializes the model compilation. Equally to regular FL training, the model compilation requires an optimizer function and a loss function.

DP optimizer and loss function

For differentially private training, the chosen optimizer was the *VectorizedDPKerasSGDOptimizer*. This optimizer is the key to adding DP to the model training. However, it is fundamental to understand firstly how an Stochastic Gradient Descent (SGD) optimizer works before diving into the addition of DP to it.

SGD is an iterative method used to optimize a loss function. The SGD process, pictured in Figure 4.10, starts by selecting a random training data batch with known inputs

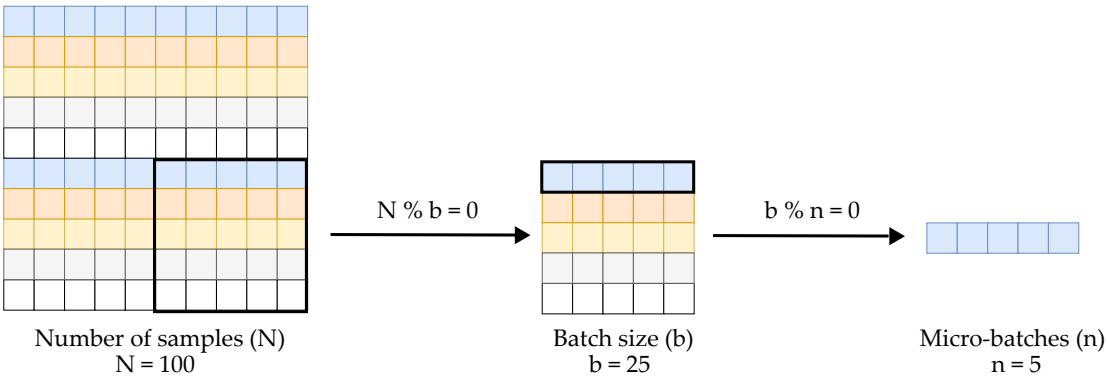


Figure 4.9: Example choice of the data related variables.

x and the respective labels y. Then the loss function computes the loss between the model prediction and the associated label. After that, the gradient of the loss function concerning the model parameters is computed. Finally, the resulting gradients are multiplied by the learning rate, which is then used to update the model parameters. The calculated gradients explain how each parameter should be updated to predict the correct labels better [55].

After understanding the fundamental functionality of the SGD, it is needed to modify it to become a differentially private optimizer and introduce noise to the gradients. The *VectorizedDPKerasSGDOptimizer* implements two significant changes to the SGD algorithm (Figure 4.11).

Firstly it limits the influence of each training point sampled from a micro-batch on the resulting gradient computation, bounding the sensitivity of each gradient. This limitation is achieved by clipping each computed gradient on each training point, bounding how much each training point impacts the model parameters.

Secondly, the algorithm needs to be randomized, so it is impossible to know whether or not a specific point is included in the training set. This randomness is achieved by adding samples of random gaussian noise to the clipped gradients. This technique makes it difficult to compare the updates applied by the SGD with and without a specific point in the training set.

In Figure 4.11, it is possible to see the usage of the configuration hyperparameters defined in advance by the stakeholder in the DP optimizer. Two of them, $L2\_NORM\_CLIP$ and $NOISE\_MULTIPLIER$, are of particular note. These two variables are significant in the steps mentioned above on introducing DP to the SGD algorithm.

The $L2\_NORM\_CLIP$ variable represents the maximum Euclidean distance of each gradient computed on an individual training example of a minibatch. This parameter serves to bound the optimizer's sensitivity to individual training points. The optimizer

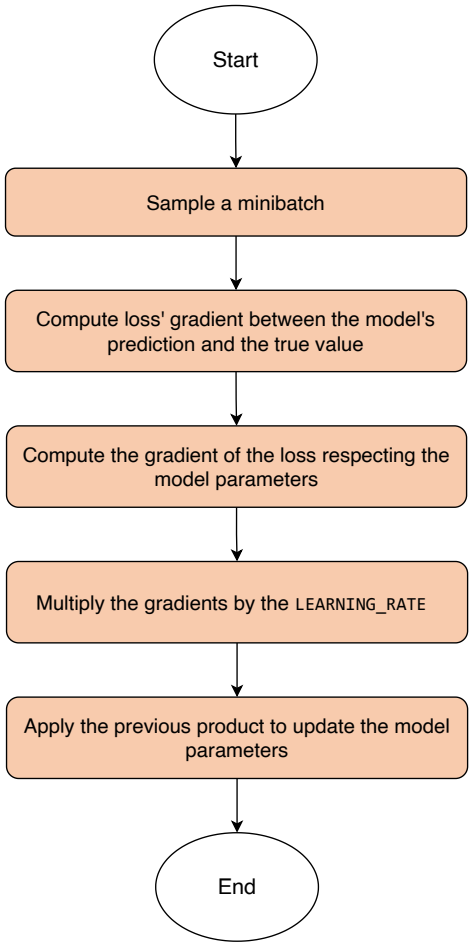


Figure 4.10: SGD behaviour.

needs to be able to compute per example gradients, so the loss function needs to output a vector loss instead of the averaged loss of the entire *MINIBATCH*.

On the other hand, the $NOISE\_MULTIPLIER$ variable controls the amount of noise sampled and added to the gradients before they are applied to the optimizer. It is important to note that the amount of noise is generally correlated to better privacy results and, often, lower utility in data.

The *VectorizedDPKerasSGDOptimizer* requires a loss function that computes the vector of per-example loss instead of its mean (set by default). The *CategoricalCrossentropy* loss function was chosen to fulfil this requirement, considering that the reduction parameter should be set to *NONE* to output the required vector.

Epsilon calculation

Lastly, after implementing the necessary changes for the DP training, it is necessary to measure the privacy budget spent to train the DL model so each stakeholder can

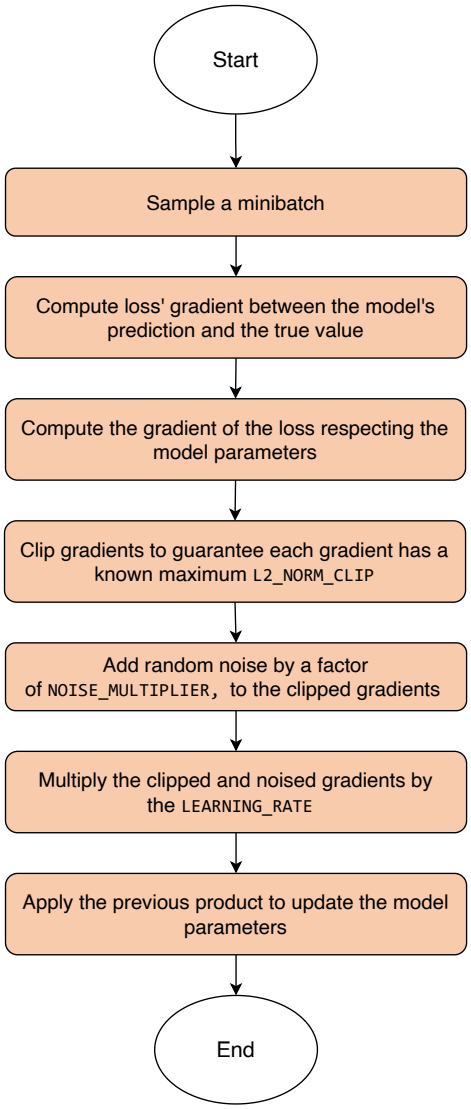


Figure 4.11: Diferentially private SGD behaviour.

clearly understand the tradeoff between the accuracy drop and the privacy optimization. The function *compute_epsilon*() was developed for the stakeholder to have a measuring reference for its privacy loss each time the *fit()* method is called.

As explained in Section 2.6.2.2, the (ε, δ) values provide the privacy loss value for each training round. The delta value represents the bounding probability that an information point is disclosed, and ε is the privacy budget value, or in other words, the strength of guaranteed privacy. As a bounding value, ε is usually a small value, although the existence of a significant value of ε does not necessarily mean lousy privacy protection.

As the Figure 4.12 demonstrates, the *compute_epsilon*() function starts by calculating the steps the optimizer takes over the training data and the *sampling_probability*, which is the probability of a single training point being included in a training batch.

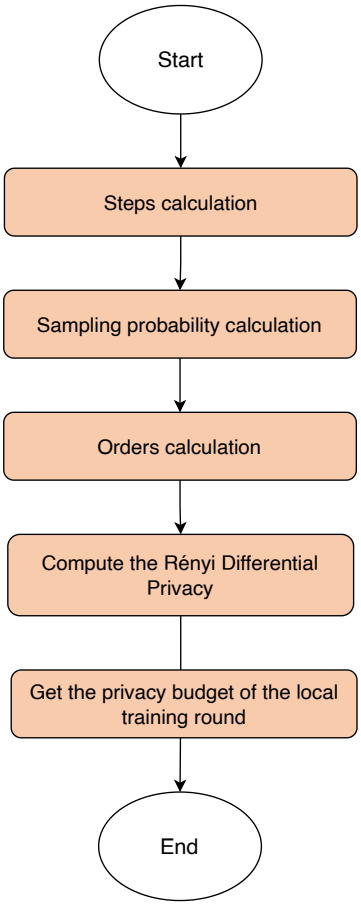


Figure 4.12: *compute_epsilon*() method behaviour.

To calculate the epsilon value, it is still necessary to calculate the Rényi Differential Privacy (RDP). The RDP is used to analyze the DP guarantees given by the sampling process and following the addition of Gaussian noise tasks performed by the *VectorizedDPKerasSGDOptimizer*. The *Tensorflow_Privacy* library provides a function *compute_rdp*() which takes as parameters the steps, the *sampling_probality*, the *noise_multiplier* and the *orders*. The *orders* variable defines a list of orders on which the Rényi divergence will be calculated. This list is already provided in a TensorFlow privacy example on GitHub[5] The RDP function and the list's creation transcend the scope of the present dissertation. In case the reader has an interest in delving deeper into these subjects, more explanation can be found in the work of Mironov et al. [56].

After getting the RDP value, it is finally possible to calculate the privacy budget associated with the DL model training. Tensorflow privacy offers a function that calculates epsilon value by giving the delta value beforehand. This function takes as parameters the list of orders, the calculated RDP and the target delta. Concerning the TensorFlow

[5] `https://github.com/tensorflow/privacy/blob/89de03e0dbc5ebf32835ca00a2426ea607bf6516/research/audit_2020/mean_audit.py`.

privacy documentation, the value of the target delta was set to be the inverse of the value of the training data size.

Diferentially private training

Finally, the value of epsilon is then presented to each stakeholder in each training round so it can be analyzed to discover if there is privacy leakage in the DL training.

4.7 The global federation node (GFN)

The last node that constitutes the framework is the GFN. This node is an extension of the FL server, connecting each client to the FL. Like the LFN, the GFN is also implemented recurring to the Flower framework as part of the FL system. The Flower framework allows for a high level of customization for the FL training by passing to the *fl.server()* function a set of custom configurations that are passed to the clients and altered in the chosen strategy. Taking into account this level of customization GFN script has been divided into four parts which constitute the flow of this node (Figure 4.13).

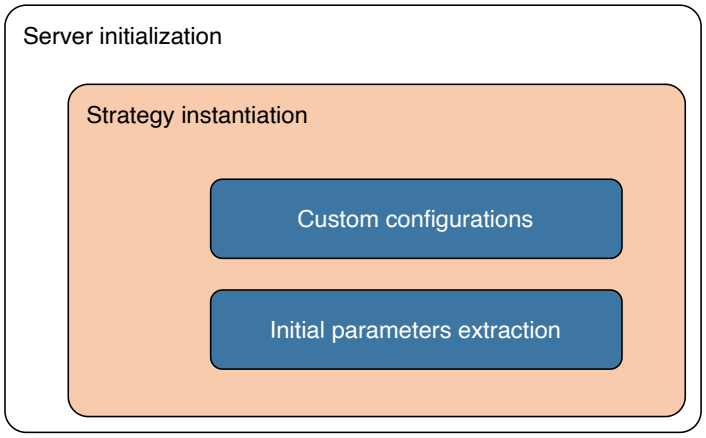


Figure 4.13: GFN script construction.

4.7.1 Custom configurations

In the Flower server, it is possible to customize the aggregation strategy with functions regarding several steps of the FL training. Each aggregation strategy to be implemented in Flower follows a standard class which may take five optional functions (Table 4.3) as parameters to customize the functionality of the strategy.

For the implementation of the GFN, it was chosen to implement the $on\_fit\_config\_fn$ parameter with a function called $fit\_config()$. This function passes a Python dictionary that tells the FL clients the size of each batch of data which should be used in the model training and the number of epochs that each client should train its data locally.

Table 4.3: Strategy customization functions.

Customization function	Description
on_fit_config_fn	Function which may customize the clients' local training hyperparameters.
on_evaluate_config_fn	Function which may customize the clients' local evaluation hyperparameters.
fit_metrics_aggregation_fn	Function that passes the custom metrics from the fit() function which the clients want to be aggregated.
evaluate_metrics_aggregation_fn	Function that passes the custom metrics from the evaluate() function which the clients want to be aggregated.
evaluate_fn	Function that performs the server side evaluation if called.

4.7.2 Model initialization

The GFN is responsible for providing the model parameters at the beginning of the FL training. This provision is made by instantiating the DL model in the same way as done in Section 4.6.2 and compiling it with the optimizer function, the loss function and the evaluation metrics. In the GFN implementation, the optimizer used was SGD, and the loss function was categorical cross-entropy. After the model compilation, its parameters are extracted by a function provided by Flower and saved into a variable.

The choice of model, optimizer and loss functions in a real-case scenario should be agreed upon between the stakeholders through an economic system. However, this dissertation is merely based on a real scenario, so the choice was made without this system. Also, the choice of optimizer was made accordingly with the use or not of DP, so whether the stakeholder chooses or not to use DP, the initial parameter can serve both scenarios.

4.7.3 Strategies

One of the significant advantages of the Flower framework is the use of a standard model for strategy development which allows a high customization level. In Flower, the strategies are responsible for the FL computation tasks at the server level. In each strategy Flower provides, a specific set of methods are expected. These methods are essential for the FL algorithm, as they control the FL flow. Table 4.4 depicts a general overview of the constituent methods of the strategies.

Table 4.4: Methods present in Flower strategies.

Method	Description
initialize_parameters()	Initial method responsible for providing the initial global parameters.
configure_fit()	Method responsible for selecting the clients which may participate in the training round and passing the instructions to send to those clients.
aggregate_fit()	Method responsible for aggregating the results of the successfully trained local models. The implementation of this method will depend on the aggregation algorithm.
configure_evaluate()	Method responsible for selecting the clients which may participate in the evaluation round and passing the instructions to send to those clients.
aggregate_evaluate()	Method responsible for aggregating the results of the chosen evaluation clients which successfully performed the local evaluation.
evaluate()	Method responsible for the server side evaluation.

Each of the methods described above can be customized to fit the needs of the FL system. In the GFN script, a Strategy customization class, *SaveModelStrategy()*, was developed to exemplify this process. This class aims to save the aggregated model in each global round on the server side for result collection. The development of the *SaveModelStrategy()* class starts with its definition taking a FL strategy as a parameter. The chosen strategy was *fl.server.strategy.FedAvg*, which implements federated averaging as an aggregation algorithm. To save the model in each global round, it was necessary to redefine the $aggregate\_fit()$ method to gather the aggregated weights from the Federated averaging algorithm and save them in a specific directory.

Although the development of this class facilitates the analysis of the global model results, it should not be implemented in a production environment as the GFN should save any data related to the clients in any circumstances.

To conclude the strategy creation, a *SaveModelStrategy()* object is created with the chosen configuration functions, the initial model parameters and five additional variables related to the FL clients. These additional variables (Table 4.5) aim to inform the server of how the task distribution should be done by informing it of the quantity or percentage of clients allocated to each task.

Table 4.5: Customization variables of the *SaveModelStrategy()* class.

Variable	Description
fraction_fit	Percentage of avaliable clients used in the training.
fraction_evaluate	Percentage of avaliable clients used in the evaluation.
min_fit_clients	Minimum number of necessary clients to perform the training round.
min_evaluate_clients	Minimum number of necessary clients to perform the evaluation round.
min_available_clients	Minimum number of available clients in the system.
accept_failures	Boolean variable responsible for allowing or not rounds that contain failures.

4.7.4 Server initialization

The final step in the GFN script is the server initialization. After defining the custom functions, the personalized strategy and its parameters, the server initialization is provided by the Flower framework by the server class. This class calls a method named $start\_server()$ which takes the server address, the strategy object and a config dictionary which takes the number of global rounds the training should last.

4.7.5 LFN – GFN communication

Lastly, after implementing all the nodes, it is essential to describe how the GFN and the LFN communicate and how the FL training flow works in the Flower framework. This explanation allows a better understanding of the FL process implemented on both the GFN and the LFN. In Flower, the communication between the server and the clients follows a specific data flow. This flow is aligned with the proposed FL flow described in

Section 3.5. It presents significant changes that do not compromise the proposed flow but are relevant to mention.

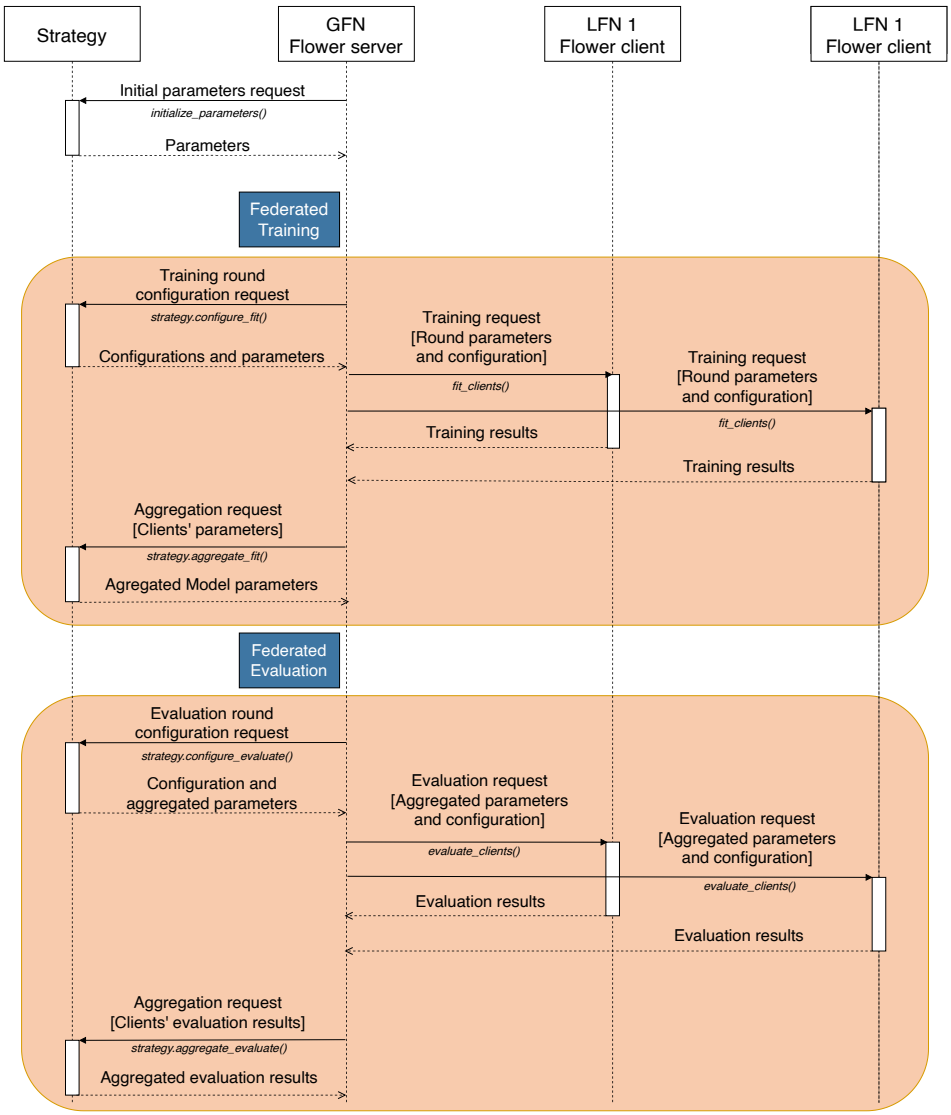


Figure 4.14: FL flow implemented by the Flower framework.

In Figure 4.14, it is possible to see how Flower implements the flow of the first FL round. The following FL rounds follow the same flow except for the initialization of the model parameters. In each round, the FL server asks the strategy script to select the clients participating in the training round and provide the necessary configurations. Then the server provides the round's base parameters and configurations and asks the clients to train their local models. After each client trains its model, they send their results which are then provided to the strategy to aggregate them with the chosen algorithm.

Finally, the most significant change that Flower provides relating to the proposed FL flow is the local evaluation in each round. This process is similar to the training process

as it starts with the strategy providing the client with the evaluation configurations. Each client makes its evaluation of the current global model, and lastly, the evaluation results are sent for aggregation.

4.7.6 SSL

The Flower framework communication system works on top of gRPC Remote Procedure Calls (gRPC) streams, providing an efficient binary serialization format [54]. In this communication system, raw bytes of data are serialized and transmitted through the network between the server and the clients.

The data streams transmitted between the aggregation server and the clients are by default not encrypted, which presents a significant vulnerability. Even though differential privacy adds an extra protection layer to each stakeholder, there is still a significant security flaw in having unencrypted private data between the FL clients and the server. By having an unencrypted communication route, the framework is prone to man-in-the-middle attacks where the data parameters of each client can be gathered for a membership inference attack, revealing critical points in the training data.

This vulnerability is mitigated by using a secure communication bridge between the GFN and the LFNs. Flower offers the possibility to use the SSL protocol to provide secure communication between the FL server and the clients. SSL provides a private connection through a Symmetric-key algorithm to encrypt the transmitted data.

Adding this protocol to the implemented framework required creating and using SSL certificates passed to the GFN and the connected LFNs forming a secure network. A shell script provided by the Flower framework developers was used to generate self-signed certificates whose path was passed as an additional parameter called *certificates* in the *start_server*() method present in the GFN script and equally in the *start_numpy_client*() present in the LFN script.

It is important to note that the usage of self-signed certificates should not be implemented in a production environment and only serve as a proof of concept in developing this project.

Chapter 5

Tests

This chapter illustrates the testing scenarios and performance evaluation metrics used to assess the conformity of the framework according to the guidelines of the performance evaluation scheme presented in IEEE Std 3652.1-2020 [6].

The implementation presented in Chapter 4 based on the proposed framework will be tested based on four criteria: model performance, privacy and security, computation efficiency and data sets. An additional criterion regarding economic viability is mentioned, though it sits outside this dissertation's scope.

The first sections will specify the tests that should be made and the evaluation parameters used in each testing criteria.

The last section will explain the chosen test case scenario, its suitability for the context of this dissertation and the requirements that should be met regarding all the test criteria.

5.1 Model performance

Federated Learning models are expected to perform similarly to a centralized Machine Learning (ML) model. Each stakeholder expects to obtain a model capable of suiting its needs and an advantage to be taken from the Federated Learning (FL) process. As mentioned in Section 4.6.2, the presented framework works with a Deep Learning (DL) algorithm for image classification. According to IEEE Std 3652.1-2020 [6], the verification process for image classification passes by comparing the accuracies of the centralized, the local, and the federated training. The evaluation should follow the following equation:

$$Acc_{disc} = Acc_{cent} - Acc_{fed} \tag{5.1}$$

The Acc_{disc} value provides the distance between an FL model accuracy and a centralized training accuracy. The greater the Acc_{disc}, the higher the discrepancy between the

FL model's and centralized model's accuracy. Table 5.1 should be followed as a reference to attribute a quantitative level to the model performance with reference to the Acc_{disc}.

Table 5.1: Model accuracy requirements [6].

Model accuracy requirement level	Requirements
0	Achieving an equivalent or more competitive performance to that of a single-data-node model
1	Achieving a performance with noticeable deterioration compared to that of a data-centralized model
2	Achieving a performance with insignificant deterioration compared to that of a data-centralized model
3	Achieving an equivalent or more competitive performance to that of a data-centralized model

5.2 Computation efficiency

Regarding computation efficiency, to be considered efficient, a FL framework should be tested regarding time and memory consumption. The time and memory an algorithm consumes will depend on the environment in which a framework is entered. It is necessary to consider and describe several aspects to contextualize the framework before analyzing time and memory. In the Table 5.2 can be found a list of the aspects that should be considered before testing a framework efficiency.

As time is concerned, the evaluation of a FL framework should consider two steps of the FL flow, the model training time and the model evaluation time. In model training and evaluation time, T_{te} is measured between the starting and end of the respective session, as the Equation 5.2 explains.

Table 5.2: Factors concerning efficiency [6].

Factor	Evaluation aspect
Roles and structure	When evaluating the FML algorithm, the entire framework structure should be explained. The evaluation of an algorithm should be organized by roles or tasks.
Data set	For each client, the data set it owns should be described. For horizontal FML models, the sample size of the data set, or the batch sizes, should be recorded. For vertical models, the number of features and the feature data types in each platform should be specified.
Hardware	The hardware information, including the number of servers, CPUs, GPUs, memory, and storage, should be elaborated For edge devices including phones, home terminal devices, remote cameras, etc., the cost of battery energy and network resources should be specified as well.
Implementation	The choice of programming language, the choice of a compiler, the compilation options, and the operating system used should be specified.
Encryption and decryption	Encryption and decryption, significantly, increase computational complexity and the required memory space. The choice of encryption/decryption algorithms, as well as the parameter settings should be specified, when reporting the training/testing time and required memory.
Communication efficiency	The communication efficiency depends on both the hardware and network capability. Specifically, data size, bandwidth, network topology, firewall, switch, and network types should be specified when evaluating the communication efficiency in FML.

$$T_{te} = T_{end} - T_{start} \tag{5.2}$$

However, as the training and the evaluation time increases proportionally to the training data size, it is necessary to discount the influence of the dataset size by normalizing the training time according to the Equation 5.3, where:

$$T_{te\_norm} = \frac{T_{te}}{Num_{te}} \tag{5.3}$$

In Table 5.3, it is possible to assess the level requirements which will be used when evaluating the time efficiency of the code.

Table 5.3: Time efficiency requirements [6].

Time efficiency requirement level	Requirements
0	Supporting completion in the order of weeks
1	Supporting completion in the order of days
2	Supporting completion in the order of hours
3	Supporting completion in minutes or seconds

Concerning memory usage, the FL standard addresses the evaluation of two types of usage: Intrinsic and Auxiliary. Intrinsic memory usage M_{intr} evaluates the quantity of memory needed for the data on which the code will operate. As for auxiliary memory usage M_{aux}, it evaluates the quantity of memory needed by the code. Table 5.4 depicts the evaluation requirements for which the memory efficiency shall be tested.

Table 5.4: Supporting computations on edge devices [6].

Memory efficiency requirement level	Requirements
0	Supporting computations on super-computing clusters
1	Supporting computations on high-performance single-node machines
2	Supporting computations on ordinary computing power servers
3	Supporting computations on edge devices

5.3 Privacy and security

A FL framework to be considered privacy-preserving and secure must effectively defend against leakage and manipulation attacks. According to the FL standard, the attacks' extension and influence should be considered when evaluating the privacy-preserving and security mechanisms. Regarding privacy preservation, to evaluate the capabilities of the framework, the tests presented in Table 5.5 should be followed.

Table 5.5: Privacy requirements [6].

Privacy requirement level	Requirements
0	No defending ability
1	Successfully defending leakage during transferring
2	Successfully defending database and aggregator leakage

As seen in Table 5.5, the attribution of each level presumes the successful defence against specific attacks. The criteria chosen for a successful defence are failure of the attack, or if the attack is successful, the data acquired is not viable to cause any damage to the defender. On the other hand, when considering the evaluation of the security of a FL framework, the tests depicted in Table 5.6 should be considered.

Table 5.6: Security requirements [6].

Security requirement level	Requirements
0	No corresponding plans for potential attacks
1	Successfully defending read-write attack for a model on a central server
2	Successfully defending data recovery for channel monitoring
3	Successfully defending data recovery for read-write attacks database and channel monitoring
4	Successfully defending model controlling based on 1–3 attacks

Similarly to the privacy testing, in Table 5.6, the security level presumes the defence with success against specific attacks. The criteria to decide if the defence against a security attack is successful are similar to privacy testing. For a level of security to be achieved, the correspondent attack should fail, or in the case of a successful attack, it does not cause damage to the FL framework or its participants.

5.4 Datasets

The used datasets are the final evaluation topic to be considered in a FL framework. Although the FL standard does not propose evaluation metrics for analyzing the used datasets, it provides qualitative measures to be examined in the context in which the framework is implemented. The quality of data is essential to machine learning. Whether a centralized or a distributed approach is concerned, the data used is crucial for the outcome of the machine learning model. Four measures should be followed to properly evaluate the quality of the data used: Sparsity, Skewness, Distribution and Data number [6]. Verifying these measures in the context where the framework is inserted contributes to preventing anomalies in the FL training.

5.5 Test scenario

The IEEE Std 3652.1-2020 [6], provides a diverse and representative list of application domains of FL, which are divided into three application areas: business-to-consumer (B2C), business-to-business (B2B) and business-to-government (B2G). It would be expected to place the framework developed in the business-to-business category. However, after analyzing the provided test cases (Finance, Health and Marketing), it was decided that neither case would fit the framework objectives. Alternatively, a test scenario for IoT was chosen in the business-to-consumer case, which is the most approximate scenario for testing this dissertation's work. This choice was made by evaluating several similarities between the presented work and the IoT scenario, from which it is important to highlight the following:

- Local data processing and Machine Learning (ML) model built with privacy preservation;
- More control in each client site to prevent privacy leakage;
- Lower communication costs, as FL emphasize principles such as data localization, low latency, and lower power consumption.

Although the most approximate testing scenario, the IoT test case was followed only in the required levels for each test criteria mentioned above. The role design and main activities were developed considering an industrial case scenario as presented in Section 4.2.

For an FL framework to conform to the standard, it must meet the requirements relating to each criterion. These requirements, as seen in the previous sections, are measured quantitatively on a variable scale for each criterion. The following table presents the acceptable levels for a framework to comply with the IoT test case.

Table 5.7: Requirements for IoT application [6].

Use case type	**Use case**	**Efficiency requirements**				**Security requirements**	**Privacy requirements**	**Model performance requirements**
		Training time	**Testing time**	**Intrinsic memory usage**	**Auxiliary memory usage**			
B2C	Internet of Things (IoT)	0	1	2	2	3	2	2

The levels presented in the Table 5.7 correspond to test scenarios to which the framework must be submitted regarding the criteria above mentioned.

Chapter 6

Results and Validation

In this Chapter, the testing results are presented and discussed in order to evaluate the level achieved by the proposed work in each evaluation criterion.

The first section describes the dataset used to test the framework and evaluate its suitability for the testing scenario. The following sections will provide the results and analysis of the tests realized for the three remaining criteria: Model Performance, Computation Efficiency, and Privacy and Security.

6.1 Dataset

The used dataset, as mentioned in Chapter 4.2, is composed by 372 images and derives from a real industrial scenario of two defects presented in a glue application case. With the intention to apply federated learning to this scenario, the training subset containing 256 images was divided into three subsets, each representing a stakeholder. This division is depicted in Table 6.1, where it is possible to identify the predominance of one class over the other in the first two stakeholders and a scarce dataset in the third stakeholder. A validation subset of 116 images was used in all the model evaluations. In a real scenario, a validation dataset should be attributed to each stakeholder, though to have a common evaluation baseline, the same validation dataset was used to evaluate all stakeholder's data.

Table 6.1: Quantity of data entries in each stakeholder by label.

Client	**Dataset**	
	Discontinuity	**Excess**
Stakeholder D	120	8
Stakeholder E	4	92
Stakeholder F	16	16

Each stakeholder trained its model locally with its corresponding dataset. Moreover, with the data of all the three stakeholders, a fully centralized model was trained and evaluated with the same validation subset. These trains provide a baseline comparison for the Federated Learning (FL) model.

Recalling Section 5.4, the dataset used for a FL training should be examined concerning four measures: sparsity, skewness, distribution and data number [6]. These measures evaluate the suitability of the dataset for a FL train. However, it is essential to consider the scenario where the FL framework sits, as the dataset evaluation is a subjective topic.

Sparsity

The sparsity of data verifies if a large quantity of data is divided through many stakeholders, where each stakeholder only contains a small amount of data. In these cases, the dataset should not be suited for FL training as it may cause communication overheads. In this dissertation's scenario, there are only three stakeholders, each with a significant amount of data. Although one of the stakeholders purposely contained a smaller dataset, it did not cause a communication overhead.

Skewness

Data skewness verifies if the number of samples between each stakeholder differs so that the imbalance of the training time affects the FL system. The most significant difference between stakeholders' data is verified between stakeholders E and F. This difference was not considered relevant in this scope as the difference in the average training time was not relevant.

Distribution

The data distribution measure verifies the fairness of the quantity of data each stakeholder provides to the FL training process. In the studied scenario, data was distributed evenly except for stakeholder F, in which a deliberate small amount of data was attributed to simulate a data scarcity scenario.

Data number

Finally, the data number measure verifies if the total data is enough for the FL training. Since the dataset was obtained indirectly, the results produced by applying it to the proposed framework generated reasonable statistical value.

After the previously mentioned measures, it was considered that the dataset is fit for a FL training and the results produced are relevant for applying FL in an industrial scenario.

6.2 Model performance

The evaluation of the FL model consists of comparing its accuracy to a central baseline model with the decentralized data gathered in one sight. Although this evaluation works as a comparison term where FL might stand as an advantage, several additional results might be relevant to demonstrate the FL model's performance.

In this section, the analysis of the FL model performance will be discussed regarding the following aspects: choice aggregation algorithm; choice of combination between global and local training rounds; evaluation metrics of the best performing models of each aggregation algorithm; and the algorithm comparison with the best performance to the centralized and local models.

6.2.1 Central and local models

First, to evaluate the federated learning model, it is essential to set a baseline for comparison. This baseline is set by training the model locally in each stakeholder, emulating a scenario where each manufacturer has limited access to a specific type of defect, or a general lack of data. Another baseline setting is made by training a centralized model, where all the data from the three stakeholders is gathered in one sight. For the baseline trainings, the configurations depicted in Table 6.2 were followed.

Table 6.2: Baseline models' configurations.

Configuration	**Value**
Batch size:	32
Learning rate:	0.0001
Epochs:	200

The primary metric to evaluate the Deep Learning (DL) classification algorithm was the model's accuracy (Equation 6.1). This metric provides insight into the relationship between the correct predictions and the total of predictions made by the algorithm.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Number\ of\ total\ predictions} \tag{6.1}$$

After training each stakeholder with its algorithm, it was possible to verify that the lack of an even dataset does not allow the algorithms to learn from their features but instead memorize the training algorithm.

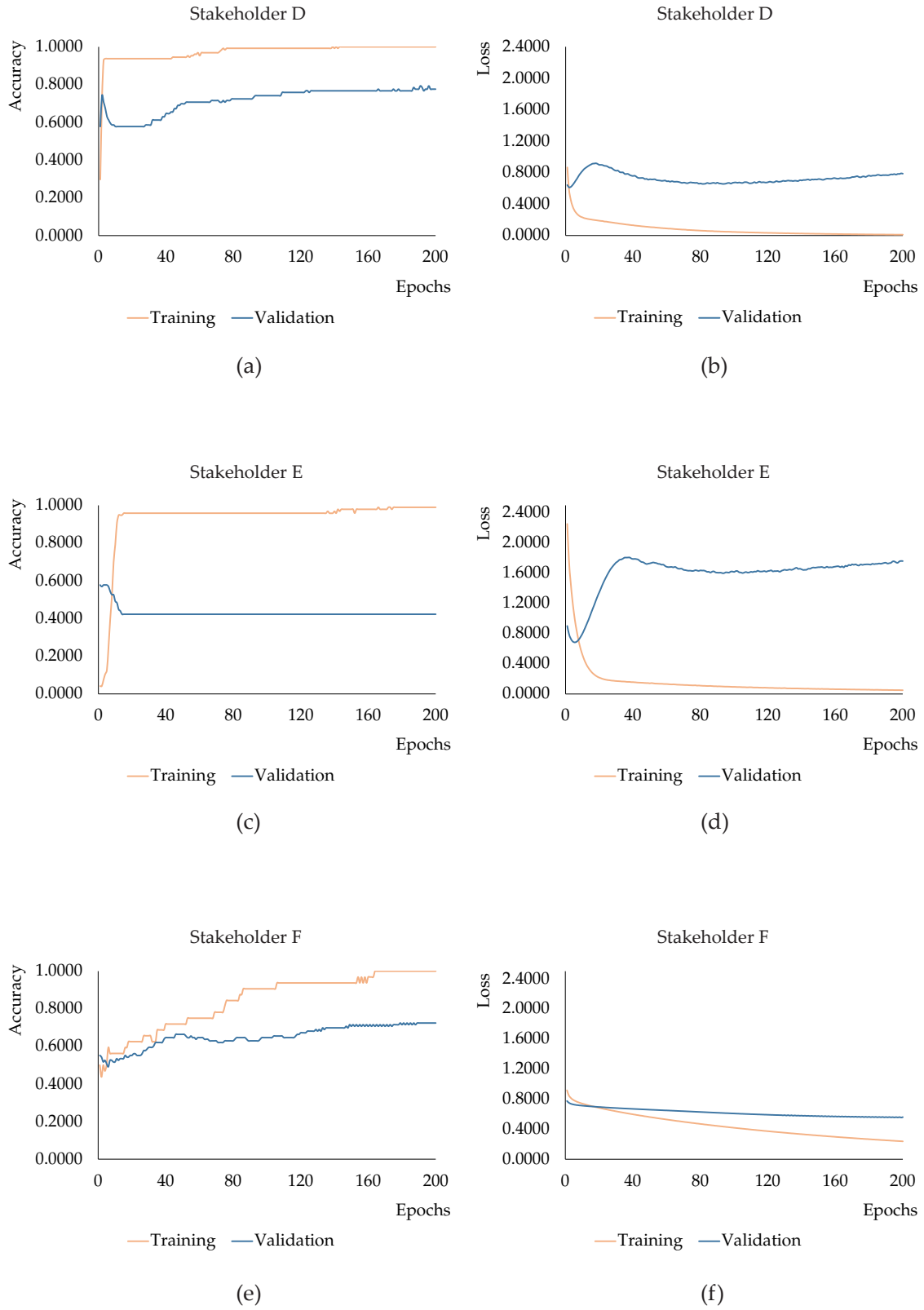


Figure 6.1: Local training accuracy and loss graphics of stakeholder D (a, b), stakeholder E (c, d) and stakeholder F (e, f).

Figure 6.1 depicts the accuracy and loss graphics per stakeholder. By analysing these plots, it is evident the lack of data leads to the general underperformance of the DL models and the overfitting of the training data scenario. This underperformance is notorious by comparing the training and validation accuracies. In the accuracy graphic of stakeholder D (Figure 6.1-a) the model improves slightly all over the epochs with a slight decrease in the first training epochs. This accuracy slope might occur due to the lack of a significant subset of data with one of the labels.

In contrary, stakeholder E's model does not learn. As the validation accuracy of the model decreases (Figure 6.1-c), it is possible to infer that the data does not allow for the model to learn and is overfitting the training data. Stakeholder F also had a slight learning curve with a small decrease in the middle of the training and an underperformance relatively to stakeholder D which is depreended to result from the lack of data.

In the loss graphics (Figure 6.1-b, d, f), the losses show that the models fit the training data as the losses tend to zero. However, only stakeholder F has a decreasing validation loss, when comparing with the other two stakeholders. A high validation loss means a low ability to fit new data, indicating that a model with a high validation loss and a low training loss has most probably overfitted the training data.

Another noteworthy aspect is that the stakeholder D, the one with more data, performed better than the others, with an accuracy of 0.77. In contrast, stakeholder E had the worst performance, which is assumed to derive from the lack of data samples with the discontinuity label. Figure 6.2 confirms the previous statement, as it is possible to verify the influence of the shortage of one label in the model's performance.

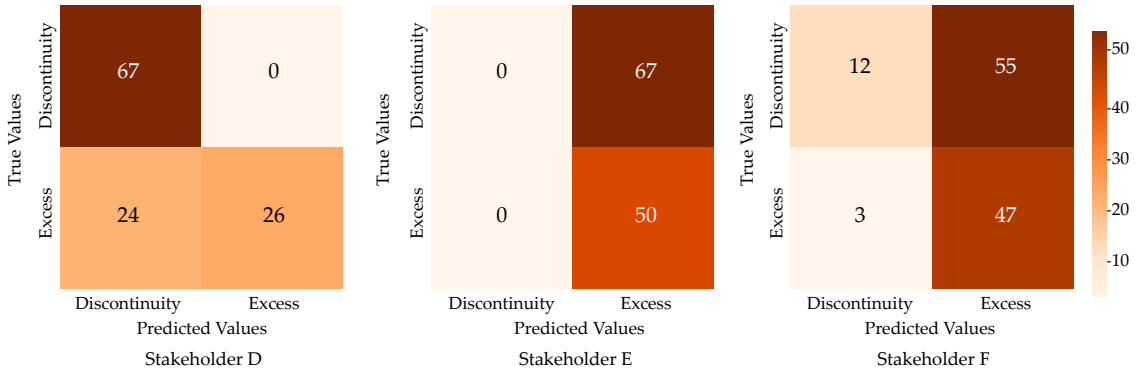


Figure 6.2: Confusion matrixes of the local models of each stakeholder when tested with the validation data.

As expected, the centralized training (Figure 6.3) provided generally better results than the local trains. With an accuracy of 0.77, the centralized model outperformed two of the three stakeholders. The similarity of performance found between stakeholder D and the centralized model might be due to the fact this stakeholder holds half of the entire dataset. However, in contrast to the centralized model, stakeholder D overfitted

the training data, presenting both low training and validation loss.

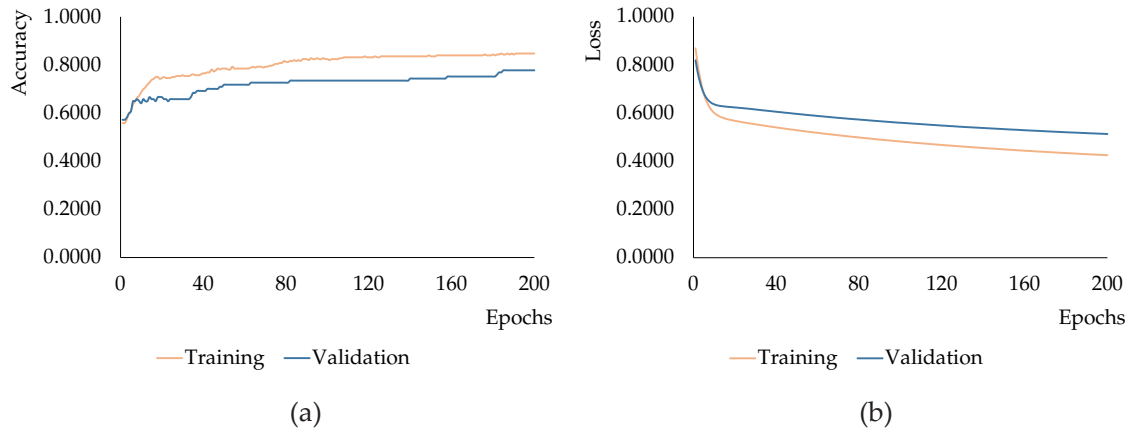


Figure 6.3: Centralized training accuracy (a) and loss (b) graphics

Aggregation algorithms

FL training was performed with three algorithms: Federated Averaging (FedAvg), Federated Optimization (FedOpt) and Federated Optimization with the adaptive optimizer YOGI (FedYOGI). All the algorithms were tested under the same conditions. As mentioned in Chapter 2.5, FL training comprises local and global aggregation rounds, influencing the federated model's final accuracy. In order to test and optimize the result of each algorithm, the batch size and learning rate parameters were kept constant, while the number of local and global rounds was variable. Table 6.3 clarifies which test values were used in the stakeholders' configuration.

Table 6.3: Stakeholders' configuration values.

Stakeholders' configurations	
Batch size:	8
Learning rate:	0.0001
Global rounds:	10, 20, 50
Local rounds:	5, 10, 20

Table 6.4 presents the accuracy results generated by trains. It is worth noting that each algorithm achieved its maximum accuracy at different combinations of global and local rounds. The FedAvg algorithm attained its maximum accuracy with 50 global and 5 local rounds. The increase of local training rounds was proven ineffective, as the accuracy value decreased. In the FedOpt scenario, the utmost accuracy was achieved with the use of 50 global rounds and 20 local rounds. This aggregation algorithm showed improvement, as the total number of rounds increased. Federated Optimization with

Table 6.4: Accuracy values resulting from the FL trains with different combinations of local and global rounds.

		Algorithm								
		FedAvg			FedOpt			FedYogi		
Global Rounds		10	25	50	10	25	50	10	25	50
Local Rounds	5	0.4655	0.5603	**0.8017**	0.5948	0.6034	0.7155	0.7069	**0.7845**	0.7759
	10	0.5776	0.6293	0.7672	0.5776	0.6896	0.7155	0.7240	0.7759	0.7759
	20	0.7759	0.7845	0.7759	0.6466	0.6810	**0.8017**	**0.7845**	0.7672	0.7672

the adaptive optimizer YOGI (FedYOGI), the last aggregation algorithm, achieved the maximum accuracy with two training combinations: 10 global rounds with 20 local rounds; and 25 global rounds with 5 local rounds. In this case, increasing the number of global or local rounds negatively affects performance, beyond those that achieved the top accuracy.

When taking a closer look at the best accuracy training of FedOpt algorithm (Figure 6.4), there is possible an accuracy drop between global rounds in the stakeholders D and E. This drop is more accentuated in stakeholder E, although the stakeholder D has more training data. However, all the model's accuracy increases over the epochs, which means that the overall algorithm is learning over time. On the other hand, Figure 6.5 denotes a general loss decrease across all stakeholders, translating into a growing capability of the model to learn from new data. The best performing accuracy and loss evolution throughout the training and validation epochs of FedAvg and FedYOGI algorithms can be found in Appendix A.

Evaluation metrics

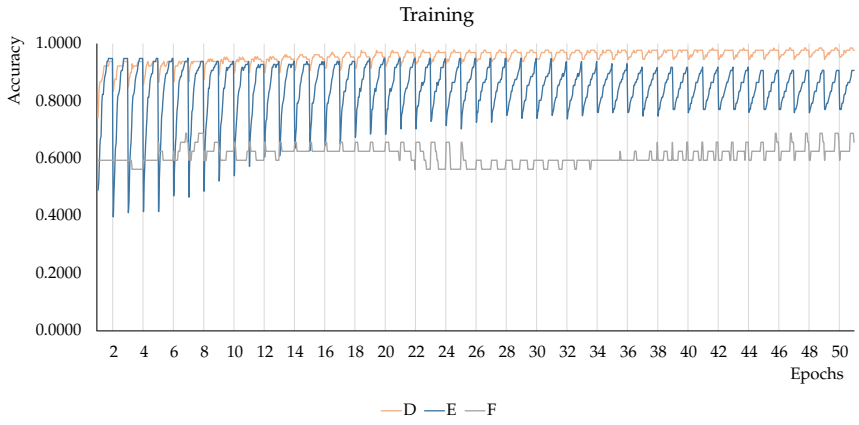
The final step of the model evaluation is the assessment of the classification Machine Learning (ML) models through the additional metrics. For this purpose, the following metrics were chosen: validation loss, Area under the ROC curve (AUC) and F1 score.

As mentioned before, validation loss measures how well the model fits new data. In its turn, AUC corresponds to the area under the receiver operating characteristic curve. This metric shows how well the classifier distinguishes the different classes. Finally, the F1 score gives a comparison value when several classifiers are evaluated.

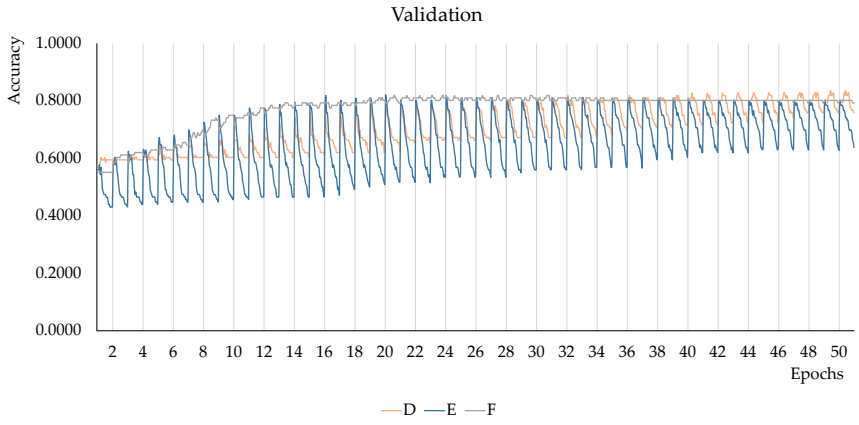
Table 6.5 compares the models with the best accuracy from each aggregation algorithm, standing out the FedAvg algorithm and the FedOpt algorithm as the models with better validation accuracy. However, the model aggregated with the FedOpt algorithm has lower validation loss and slightly higher AUC. These differences mean that the FedOpt model better fits new data, and better distinguishes the two classes provided for classification. Thus, FedOpt is the tested FL algorithm with the best performance.

Discussion

Table 6.6 compares the FL algorithm with the best overall performance, and the centralized model. The values suggest that there is a performance advantage in training



(a)



(b)

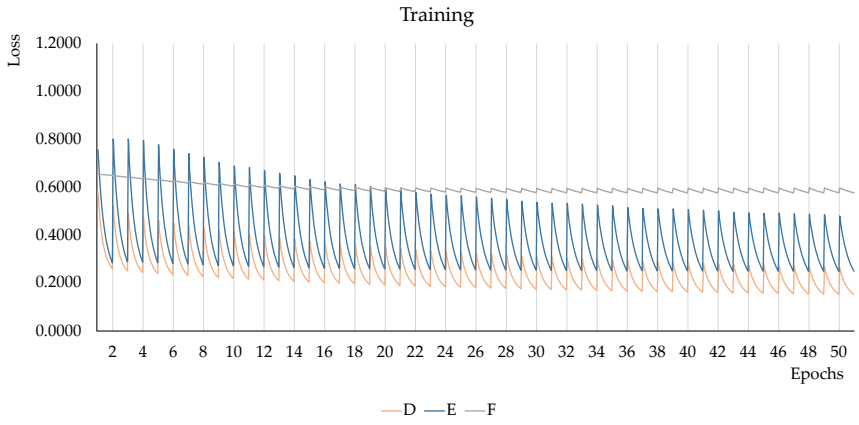
Figure 6.4: Accuracy (a) and validation accuracy (b) graphics of the three stakeholders in FedOpt training with fifty global epochs and twenty local epochs.

the model recurring to FL. The accuracy slightly outperforms the centralized model's accuracy, even thought the higher values of AUC and F1 score.

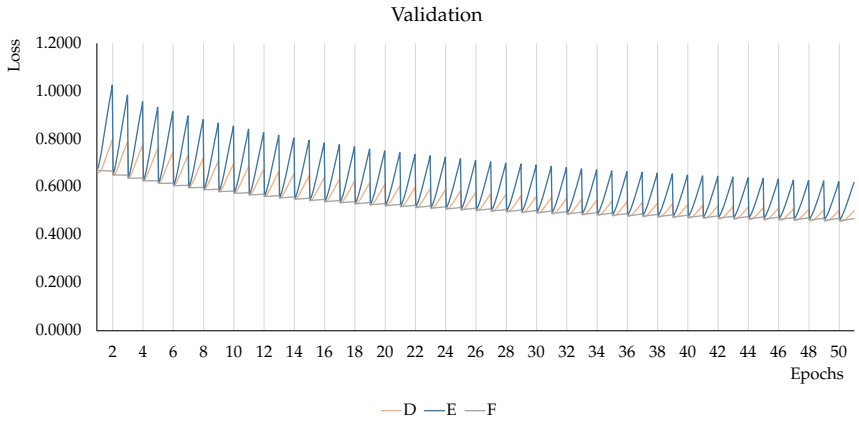
Upon applying Equation 6.1 the resulting ACC_{disc} is -0.0239. Although the similar value with the FL model, ACC_{disc} evidences a more competitive performance than the centralized model.

The comparison between the centralized and FL models reveals one of the advantages of applying FL in an industrial scenario. The proposed work suggests a competitive performance compared to the centralized model, while keeping the datasets in the sources without having data transference.

According to Table 5.1, it can be attributed the "Model accuracy requirement level



(a)



(b)

Figure 6.5: Loss (a) and validation loss (b) graphics of the three stakeholders in FedOpt training with fifty global epochs and twenty local epochs.

3", which results on an equivalent or more competitive performance than a model with centralized data.

6.3 Computation Efficiency

A FL framework has several aspects that must be considered about computation efficiency. Throughout this document, most of these evaluation aspects have been explained, such as the roles, dataset, hardware used, software implemented, and used encryption protocols. The network, also an evaluation aspect, will be covered below in more detail.

The proposed framework was tested in a Local Area Network (LAN) environment,

Table 6.5: Evaluation metrics of the models with the best accuracy from each aggregation algorithm.

Algorithm		Accuracy	Loss	AUC	F1 Score
FedAvg	50 global rounds 5 local rounds	0.8017	0.5354	0.8345	0.8017
FedOpt	50 global rounds 20 local rounds	0.8017	0.4585	0.8801	0.8017
FedYogi	25 global rounds 5 local rounds	0.7845	0.4865	0.8496	0.7845
FedYogi	10 global rounds 20 local rounds	0.7845	0.4880	0.8425	0.7845

Table 6.6: Comparison between the FL train with the best performance and the centralized training.

Model	Loss	Accuracy	AUC	F1 score
Centralized	0.5137	0.7778	0.8299	0.7778
FedOpt	0.4585	0.8017	0.8801	0.8017

where two computers communicated through Wi-Fi. During the training, the communication bandwidth was collected with a mean of 2.83 ms of ping, 66.14 Mbps of download speed and 79.75 Mbps upload speed.

To evaluate the computation efficiency of a FL framework, two main aspects must be taken into account: time consumption and memory efficiency. The evaluation of these aspects is needed to assess the framework efficiency, and the viability of the FL use as an alternative to centralized ML.

6.3.1 Time consumption

The time consumption of a FL learning framework must be verified in the most time-consuming subprocesses, which are the model training and the model evaluation.

Training time

In order to assess the time consumption during the training of a model, the Equation 5.2 was applied. The gathered times per stakeholders were divided by the number of local training epochs. As the model aggregation algorithm does not influence each stakeholder local training time, the distinction between aggregation algorithms was not considered in this evaluation.

Table 6.7 shows the average training time of each stakeholder sorted by the number of local epochs. In order to calculate each average time, all the training times of each local round were included, disregarding the number of global epochs and the aggregation algorithm. The collected values demonstrate no significant time consumption variations

Table 6.7: Average training time for each stakeholder.

Local Rounds	**Time**		
	Stk D	Stk E	Stk F
5	14.3309	14.7096	12.2448
10	25.6122	26.9116	22.4272
20	49.0445	50.4290	42.4213

between stakeholders, probably because of all them were executed on the same hardware and software conditions. Naturally, higher local training rounds end up in higher training times. However, the influence of each stakeholder's dataset size is still not considered. To verify how the dataset size might influence the training time, is necessary to normalize the training times according to the Equation 5.3.

Table 6.8: Normalized training time for each stakeholder.

Local Rounds	**Normalized time**		
	Stk D	Stk E	Stk F
5	0.1088	0.1493	0.3711
10	0.1974	0.2767	0.6950
20	0.3846	0.5265	1.3235

Usually, the training time should increase with the amount of data used to train the dataset. In this way, Table 6.8 presents unexpected results, since denotes the opposite event. This occurrence might be due to all three stakeholders were trained in the same machine, without dedicated hardware, or without the batch size standardization. In particular, a standardization for all stakeholders might not be suitable for the dataset size of stakeholder F, since its batch size and dataset size are the same.

Evaluation time

When concerning the evaluation time, all the stakeholders executed the evaluation process once in each global round. In this framework, the global model was locally evaluated for privacy reasons, providing three evaluation times per global round. The mean of each stakeholder evaluation time is depicted in the Table 6.9.

Table 6.9: Average evaluation time for each stakeholder.

Stakeholder	**Average evaluation time**
D	1.0539
E	1.6881
F	0.4342

The results from Table 6.9 demonstrate that the evaluation round is done in a short period of time, for each stakeholder, presenting a low time consumption in each global round. However, similarly to the average training time, these results do not provide information about the relation between the evaluation time and the number of samples. In order to enlighten this relation, it is necessary to normalize these evaluation times.

Table 6.10: Normalized average evaluation time for each stakeholder.

Stakeholder	Normalized average evaluation time
D	0.0083
E	0.0177
F	0.0136

Table 6.11: Time consumed in the tested algorithms with different training round combinations.

		Algorithm								
		FedAvg			FedOpt			FedYogi		
Global Rounds		10	25	50	10	25	50	10	25	50
Local Rounds	5	389.818	761.491	1390.930	474.748	425.024	2738.085	393.124	364.475	643.027
	10	485.296	620.377	2159.300	299.078	575.915	2108.503	521.773	611.349	1121.014
	20	476.325	971.225	1644.940	810.449	1801.977	3514.829	817.608	1019.637	3538.451

Table 6.10 shows the evaluation times normalized by the validation data. As mentioned before, the validation dataset used to evaluate the global model was the same in all stakeholders to provide an uniform global model evaluation. Its is possible to conclude that the usage of the same dataset for evaluation in all threes stakeholders was the cause of the similar evaluation time results.

Total time

As previously mentioned, the most time-consuming subprocesses in a FL framework are the training time and evaluation time. However, only evaluating these two subprocesses does not provide a general insight into the time consumption of the whole FL process. For a better understanding of FL's time consumption, the whole process was timed for every aggregation algorithm with every combination of local and global rounds.

Table 6.11 highlights the FL processes duration with higher accuracy for each algorithm. FedOpt and FedAvg algorithms provided the models with the best accuracy, whereas the latter finished in less than half the time of FedOpt. As expected, FL processes' duration increases with both the number of local or global rounds.

Discussion

Upon evaluating the time taken by the FL framework during the FL process and subprocesses, the framework performs efficiently. The most consuming FL processes did not reach an hour. The average time consumed was 1136.25 ± 0.01 seconds, corresponding to 18.94 minutes in all global and local rounds combinations. It is notable a significant time difference between the two most accurate algorithms (FedAvg and FedOpt). FedOpt took more than double the time of FedAvg, even though FedOpt provided better validation loss and AUC and the resulting accuracy was equal for both.

For completing the FL process in minutes, it is possible to attribute the "Time efficiency requirement level 3" to this framework, for completing the FL process in minutes.

However, it is relevant to denote that the realized tests were made in a research environment with a good network connection. To properly test the FL framework's speed in a real scenario, it should be applied in an industrial network or even on a global network between several industrial sights.

6.3.2 Memory consumption

The memory usage of a FL framework is naturally higher than a standard ML training algorithm, which may present a setback on the used hardware depending on the use case. The evaluation of the memory usage of the proposed framework was divided into two subsets, the memory used by the training and evaluation data (M_{intr}), and the memory used by the framework's code (M_{aux}). Each stakeholder's training dataset and the validation dataset were contemplated to evaluate the memory of the data. The latter was included in all three stakeholders as they used it in the evaluation rounds.

As for the training and evaluation data corresponding to M_{intr}, stakeholder D had a dataset with 1.98 Mb, stakeholder E had 1.97 Mb, and stakeholder F's dataset had 1.3 Mb.

On the other hand, the memory used for the framework's code M_{aux} was equally divided as no stakeholder possesses more functionalities than the other. However, in a real-world scenario where the stakeholders might not be associated, the memory consumption might vary between them. As the framework was programmed in Python, to evaluate the M_{aux} more accurately, it is necessary to consider the virtual environment and the libraries used.

Table 6.12: Auxiliar memory consumed by each entity, measured in Mb.

Memory usage	**Entity**			
	Stakeholder D	Stakeholder E	Stakeholder F	Server
Code	26.73	26.73	26.73	0.0150
Virtual environment	5457.92	5457.92	5457.92	5457.92
Total	5484.65	5484.65	5484.65	5457.93

Table 6.12 depicts the consumed memory by each stakeholder and the server. The virtual environment memory consumed was common to all the entities as the used virtual environment possessed the same dependencies and libraries. Also, it is essential to note that the virtual environment was common to all the stakeholders as they were instanced in the same machine. The choice to include the virtual environment memory in three stakeholders was made to simulate a real scenario where all three stakeholders must include the same dependencies in their respective machines.

Discussion

From the memory consumption results collected and considering the choice of hardware made, it is possible to affirm that the proposed framework is suitable for execution in standard personal computers. As the application of this framework aims at an industrial

scenario where the computation power is typically more significant, it is safe to affirm that the proposed work is suitable for a manufacturing scenario. Also, according to Table 5.5 is possible to frame the memory consumed in the "memory efficiency requirement level 2", corresponding to the framework being able to be computed in ordinary computer power servers. The usability in edge devices were not tested due to not being a use case requirement. Therefore, the framework was not qualified for the "memory efficiency requirement level 3".

6.4 Privacy and Security

The privacy of the proposed framework was one of the important topics approached in this dissertation. Also, as mentioned in the Chapter 2.6.1, the framework's security influences its ability to keep the FL client's data private.

6.4.1 Security

The security testing was made regarding the security requirements necessary for the test scenario. According to Table 5.7, the security requirements level 3 is necessary to achieve the desired security. Therefore it is necessary to defend successfully against a read-write attack on the database and channel monitoring.

Read-write attacks

The security measure taken to defend the database was the encryption of the data upon its storage. Data recovery then depends on a decryption key to reveal the database's contents. Assuming a successful breach of the database, the attacker to access the content must pass the SQL command "*SELECT * FROM image_table*". However, the retrieved content is encrypted, as shown in Figure 6.6. The retrieved data is unusable without the decryption key, and its security is still kept.

Channel monitoring

The SSL cryptographic protocol was implemented regarding channel monitoring to prevent man-in-the-middle attacks. For that purpose, certificates were generated for the Global Federated Node (GFN) and the stakeholder's Global Federated Nodes (GFNs) creating an encrypted channel among them. The secure channel was tested with Wireshark, a network analyzer which allowed to follow the communication between the stakeholders and the GFN.

In Listing 6.1, it is possible to see part of an unencrypted communication stream between the GFN and the LFNs. Although the gRPC communication is done throw byte streams which, without proper conversion to the proper format, are not understandable to the naked eye, it is still possible to identify some information by looking through the

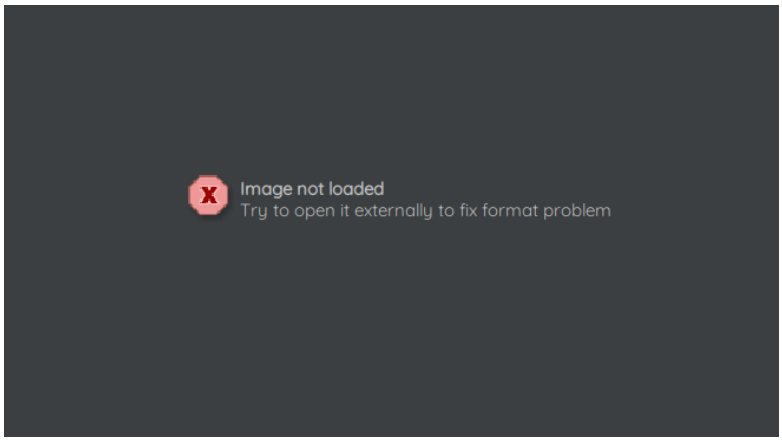


Figure 6.6: Example of an encrypted data point.

message stream. A knowledgeable attacker with the proper tools and knowledge would easily convert the data stream and retrieve the information contained in it.

Listing 6.1: Unencrypted communication stream.

```
1  PRI * HTTP/2.0
2
3  SM
4
5  ..$.....................?.....?..... ..................?.....................@
6  :authority.localhost:8080@.:path$/flower.transport.FlowerService/Join@.content-type.
       ↪ application/grpc@.grpc-accept-encoding.identity,deflate,gzip@.accept-encoding
7  identity,gzip@.te.trailers@
8  user-agent2grpc-python/1.43.0 grpc-c/21.0.0 (windows; chttp2)
       ↪ .....................................0............0.............................V
       ↪ .....V............D............[.~............................u.....u
       ↪ ..?..................
9  ....
10 ...NUMPY..v.{'descr': '<f4', 'fortran_order': False, 'shape': (3, 3, 3, 32), }
```

On the other hand, as Listing 6.2 depicts, the same part of the data stream is encrypted. Even with knowledge of the transferred data type and the proper tools, it would be challenging to decrypt the transmitted data without the proper decryption key. This transmission prevents data retrieval during its transfer between stakeholders and the GFN, adding a security layer to it.

Discussion

Upon testing, the framework's security was demonstrated to be effective against the performed attacks. Although the security breaches in the whole system were assumed beforehand in order to evaluate a more profound attack, the framework proved to have implemented a security system that allows the data to remain secure within the framework's

Listing 6.2: Encrypted data stream.

```
1  ..............q..>....!
2  ...o...%....,)...c. "<..[]0..{'.R=.........m.x
3  q..
4
5  .........+.,./.0.............localhost..........
6  ..............#.........grpc-exp.h2.
7  ........................3t...3.G.E...A.)r.\....8..#Zv..Q.W..X..DO.=.Kz...{.....t.I/$.u.-..B
       ↪ ..%..I....Ta.-.....+.................
8  ..............................................................................,...
       ↪ A.......e..*.2.;.M.b.w`.u..."<..[]0..{'.R=.........m.x
9  q..
10 ....0.3.E...A.......!.|../.9.5.).....<x{...
```

Table 6.13: Resulting accuracy of the differentially private FL train with the FedAvg and the FedOpt algorithms, when applied different learning rates and levels of noise.

		Algorithm							
		FedAvg				**FedOpt**			
Noise Value		1.2	2	5	10	1.2	2	5	10
Learning rate	0.0015	0.7759	0.5776	0.5776	0.5776	0.7845	0.7931	0.7672	0.7500
	0.1500	0.8017	0.7500	0.7500	0.6034	0.8017	0.7414	0.7069	0.6983

processes. When evaluating the framework's security level taking into account Table 5.6, it is possible to verify that the proposed work defended against a read-write attack on the database and a channel monitoring attack successfully, not leaking any viable data to the attacker. With these defences considered, it is possible to affirm that the proposed framework fits the "security requirement level 3" corresponding to successful defences in the database and the communication channels. The "security requirement level 4" was not considered as the test case scenario requires only the "security requirement level 3".

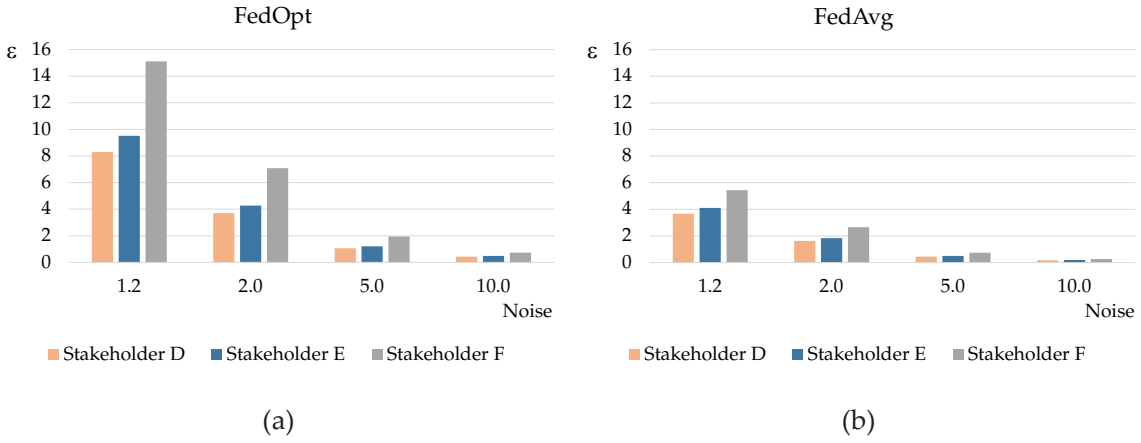
6.4.2 Privacy

The chosen approach to improve the framework's privacy was the addition of differential privacy to the clients' training. However, Differential Privacy (DP) presents itself with a setback. Adding noise to data increases its privacy and decreases the training accuracy. The FedAvg with fifty global rounds and five local rounds and FedOpt with fifty global and twenty local rounds were chosen to test the model's accuracy depending on the amount of noise to be added and the variation of the learning rate. These algorithms were chosen for being the ones that provided the best accuracy in the model evaluation.

In Table 6.13, the influence of the noise addition on the model accuracy becomes clear. Four noise levels were tested in both algorithms, and the results show an evident decrease in accuracy as the amount of noise increased. It is also possible to verify the influence of changing the learning rate in each algorithm for the different noise values. In the FedAvg algorithm, the increased learning rate showed improvements in the model accuracy in constant noise values. On the other hand, in the FedOpt algorithm increasing

the learning rate proved, in general, to worsen the accuracies when the noise value was constant.

Regarding the privacy budget spent on the differentially private training (ε), Figure 6.7 compares the two chosen algorithms and the values of added noise. Naturally, the value of ε decreases with the increased addition of noise. In other words, the training guarantees more privacy for the stakeholder. However, both algorithms have a visible difference in the ε values. As the aggregation algorithm does not influence the privacy budget, this difference is attributed to the number of local epochs applied to each study case. The FedOpt algorithm used twenty local epochs, and the FedAvg algorithm only used five, which provoked the evident discrepancy in the epsilon value computation. It is also important to denote that the learning rate does not influence the ε, so it was not considered for this evaluation.



(a) (b)

Figure 6.7: Resulting ε from the diferentially private trains with different noise levels.

Upon comparing the epsilon values of Figure 6.7, it is evident that the FedAvg algorithm with DP provided better ε results than the FedOpt algorithm. In addition, when verifying the accuracy values of Table 6.13 concerning the ε value of each train, the FedAvg train with the noise value of five and the learning rate of 0.15 stands out for being the train with a lower ε and with lower accuracy loss.

Influence of DP in the training

When DP is applied to the FL training process, it is expected that the training process is affected by the generated loss and the accuracy loss between global rounds to be higher than in non-differentially private training. However, the value of noise applied cannot be in a quantity that profoundly affects the absolute accuracy of the model.

As it is possible to verify in Figure 6.8, the FL training with the FedAvg aggregation algorithm with fifty global rounds, five local rounds, a learning rate of 0.05 and a noise

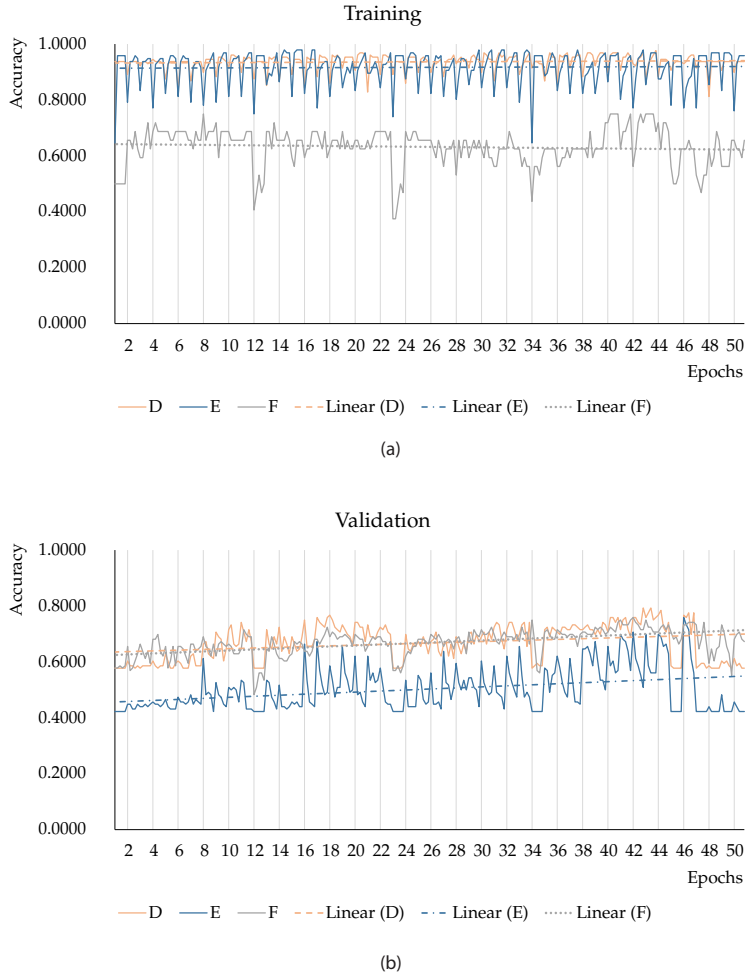


Figure 6.8: Accuracy (a) and validation accuracy (b) graphics of the three stakeholders in FedAvg training with fifty global epochs and five local epochs when differential privacy is added with a noise value of two.

value of 5 presents to be affected by the introduction of DP in its training. In the validation accuracy evolution graphic (Figure 6.8 (b)), the accuracy loss between global rounds is evident and even challenging to perceive the accuracy evolution. Only by following the trend lines in the graphic is it possible to visualize an accuracy evolution in the FL training.

When comparing the performance of the FedAvg in the same training conditions as in Figure 6.8, though, without DP (Figure 6.9), it is possible to understand the correlation between DP and model performance and the tradeoff existent between these two factors.

The tradeoff becomes clearer when evaluating both models' losses (Figure 6.9 (b)). In the train with added DP, it is evident that the loss gain might constitute a severe problem when applied new data to the model for inference. As an alternative, the choice of a lower level of noise will reduce the loss. For example, the choice of a noise factor of two provided an equal accuracy and less than half the loss, as it is possible to verify in Figure

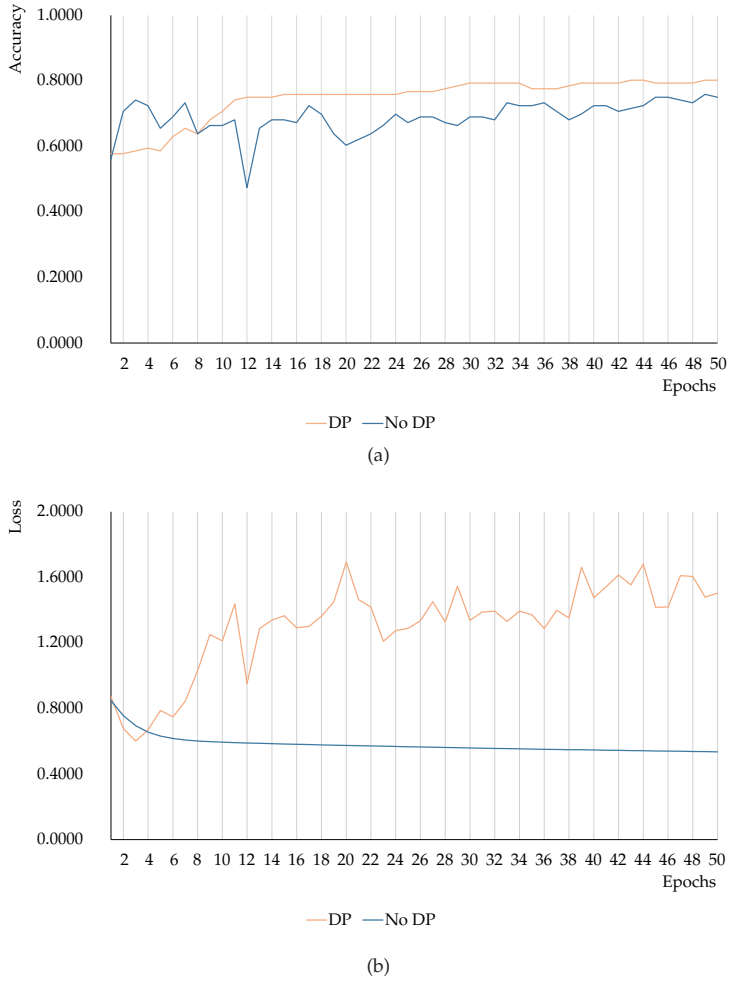
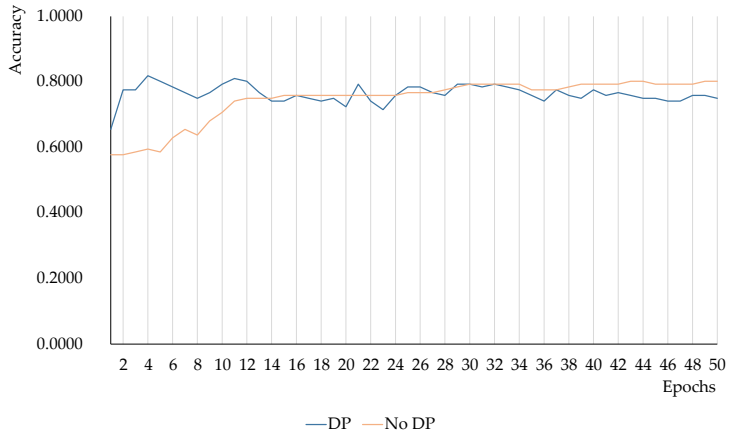


Figure 6.9: Comparison of the training accuracy (a) and loss (b) of the FEDAVG algorithm with fifty global epochs, five local epochs with and without the usage of DP with a noise value of 5.

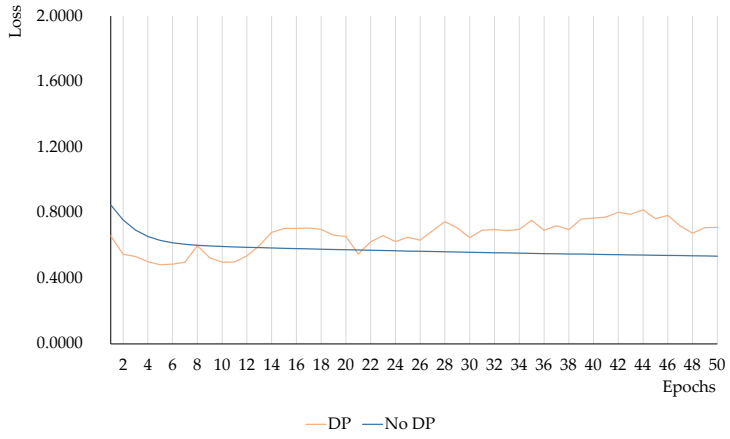
6.10.

Discussion

Differential privacy introduces an arbitrary quantity of noise to the DL training process, which translates into an additional layer of privacy for the data used in each local training. As it was verified, the addition of noise also presents a setback in the model performance, lowering its accuracy and increasing its loss. The existing tradeoff between model performance and privacy depends on several factors, from which two were highlighted, the noise volume and the learning rate of the model. Although the learning rate does not directly influence the differential privacy process, it influences the training process and, consequently, the final model's performance. It was later verified that the model with the best accuracy/privacy ratio was not the most viable for usage. From this verification, it can be inferred that a thorough validation is needed for differential privacy



(a)



(b)

Figure 6.10: Comparison of the training accuracy (a) and loss (b) of the FEDAVG algorithm with fifty global epochs, five local epochs with and with the usage of DP with a noise value of 2.

as it may be deeply involved with the model performance.

It can be concluded that the usage of DP in FL brings an additional privacy layer to the framework with a performance cost. As the framework is agnostic to the study case implemented, an absolute noise value cannot be recommended as it depends more on the ML scenario than on the framework configuration. However, the framework successfully supported the usage of DP in its trains. The ability to configure the training and DP parameters presents itself as an advantage in the protection of privacy of the framework's clients.

Privacy testing

According to Table 5.5, privacy testing is similar to the security tests already performed on the proposed work. However, it is interesting to put such tests into a privacy perspective.

Discussion

When viewing the performed security tests from a privacy perspective, data privacy would still be secure if assuming a security breach of the database. Although data retrieval would be possible, it would still possess an encryption barrier. The encrypted data does not reveal any information the attacker might use to infer information from each stakeholder. Regarding the transmission of data between the stakeholders and the GFN when encrypted, the SSL protocol adds a robust layer of encryption to the whole communication. An attempt to infer the model parameters would be most improbable without the correct decryption key. In addition, through the gRPC communication, the Flower framework does not transfer the data in a format easily convertible without knowledge of the data types used by this framework.

Finally, the prevention of leakage would be challenging regarding aggregator leakage without an algorithm that allows the aggregation of the encrypted local models' weights, such as homomorphic encryption. However, the addition of differential privacy to the aggregated models partially secures data privacy, complicating the inference of data from the model's weights.

Considering the tests that must be done in Table 5.6 for achieving the "privacy requirement level 2", the defences against leakage during transference and database leakage were considered successful. Although the defence against aggregator leakage was not considered successful, the implementation of differential privacy during the training contributed profoundly to this test. Finally, the framework was considered to achieve the "privacy requirement level 2" in a research environment, but if applied to a real industrial scenario, the privacy requirement level achieved would be one.

Chapter 7

Conclusions

The work developed throughout this dissertation resulted in a functional prototype of a federated learning framework, suitable for industrial study case scenarios. There is a plethora of industrial scenarios in which the employment of a Federated Learning (FL) framework can be an advantage. Therefore, a defect classification scenario was chosen to establish a research guideline, where three stakeholders were simulated with data deficiency. Based on the experiments carried out in [57], this type of approach shows promise regarding the generalization beyond the bead shape included in the training set.

The application of FL to the case study was proven to be an advantage for the tested stakeholders' performance. When comparing the FL results with the centralized training results, slightly superior performance was achieved, proving FL to be a solution to reach larger data quantities in a distributed setting while not compromising the privacy of each data owner.

The computation resources taken by the framework to complete the FL task were deemed very efficient, considering the scenario where the proposed framework should be included. Although the time consumption was dependent on the quantity of data, the FL process was completed with low computational power and in less than an hour.

Regarding the security and privacy of the framework, the application of encryption algorithms in the database and in the communication process was revealed to be fruitful in defending against possible attackers. In the considered scenario of a security breach, the framework was still capable of maintaining data confidentiality, whether in the database or communication process. The addition of differential privacy to the local training was also considered fruitful, enabling the data to remain confidential when training the models. However, this privacy technique also proved to affect the model performance negatively. With this factor in consideration, the framework allowed the stakeholders to configure the DP parameters and get the best performance/privacy ratio. The ability of

the stakeholder to use (or not) DP in its training process gives the framework's clients the choice to increase their privacy with the setback of decreased accuracy.

Overall the results obtained from the proposed framework's performance were in line with the requirements established in Table 5.1. The proposed work successfully implemented a framework which improved a manufacturing process using privacy-preserving federated learning. The datasets were kept in each stakeholder without information sharing among them.

In conclusion, this dissertation's work fulfilled the objective of creating a FL framework capable of facing industrial scenarios where data sharing is still a problem. The results produced in Chapter 6 confirm the formulated hypothesis, from which can be concluded that for the insurance of privacy and security, collaborative Machine Learning (ML) can be implemented in complex manufacturing environments with the resource of a federated learning framework, allowing decentralized training and the addition of security measures and privacy-preserving algorithms.

Neverwithstanding, some improvements to achieve even better privacy results and improve the economic system in FL are not excluded. A description of work contributions and potential ways to pursue future work are detailed in the following sections.

7.1 Contributions

This dissertation provided the following contributions:

- A novel approach to the application of federated learning in an industrial context that provides access to ML without privacy leakage. The developed framework allows collaborative ML through the process of FL with additional privacy and security measures. The framework was constructed with a modular design adding to the advantages of the already existing general purpose FL frameworks, namely Flower (Figure 7.1), allowing it to be adapted to various industrial scenarios.

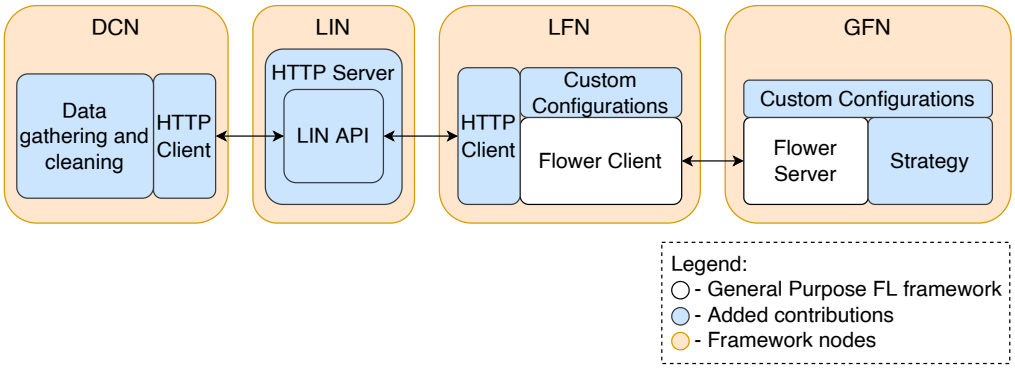


Figure 7.1: Framework contribution.

- A prototype software that allows the implementation and customization of the framework. This prototype presents itself as a tangible contribution. Since it is open-source, the community can freely use, customize and improve it. The developed framework is available online at the following link: `https://github.com/AlexCosta157/CISP_FL`.
- An article was also prepared and submitted as an output of the developed work. Until the submission of this thesis, the manuscript is under review.

7.2 Future Work

Despite the satisfactory results in this dissertation, as an initial prototype framework, further work is required to improve the framework's robustness and fairness for all its clients. Some potential solutions are given below:

- Implement different AI problems to improve the framework's customization and adaptability to new challenges;
- Realize scalability tests with a more significant number of modules and clients with a more robust dataset to verify the performance scalability;
- Deploy the framework in a real scenario to verify its behaviour in a real industrial scenario. Also, it would be interesting to realize a test between stakeholders from different parts of the world to test the communication efficiency in such a scenario;
- Implement an economic system of rewards to attract stakeholders into using the framework. The economic system must also address the scenario of one stakeholder possessing a more considerable amount of data than the others and contributing more to the FL training;
- Apply homomorphic encryption to the FL training. In this dissertation, it was assumed that the Global Federated Node (GFN) is a trustworthy framework node. However, in a real scenario, that might not be the case. With homomorphic encryption, the aggregator never accesses the raw data, thus protecting each client's privacy.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[2] Q. Yang et al. "Federated machine learning: Concept and applications". In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (Jan. 2019). ISSN: 21576912. DOI: 10.1145/3298981. arXiv: 1902.04885 (cit. on pp. 1, 15, 23, 24).

[3] L. U. Khan et al. "Resource optimized federated learning-enabled cognitive internet of things for smart industries". In: *IEEE Access* 8 (2020), pp. 168854–168864. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3023940 (cit. on p. 1).

[4] M. Aledhari et al. "Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications". In: *IEEE Access* 8 (2020), pp. 140699–140725. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3013541 (cit. on pp. 1, 2, 17).

[5] X. Yin, Y. Zhu, and J. Hu. "A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions". In: *ACM Computing Surveys* 54.6 (July 2021). ISSN: 15577341. DOI: 10.1145/3460427 (cit. on pp. 1, 19).

[6] L. T. S. Committee. *3652.1-2020 - IEEE Guide for Architectural Framework and Application of Federated Machine Learning*. Tech. rep. 2020. DOI: 10.1109/IEEESTD.2021.9382202 (cit. on pp. 2, 15, 17, 27–30, 35, 59–63, 65).

[7] E. Y. Nakagawa et al. "Industry 4.0 reference architectures: State of the art and future trends". In: *Computers and Industrial Engineering* 156 (June 2021), p. 107241. ISSN: 03608352. DOI: 10.1016/j.cie.2021.107241 (cit. on pp. 4–6, 8).

[8] D. G. Pivoto et al. "Cyber-physical systems architectures for industrial internet of things applications in Industry 4.0: A literature review". In: *Journal of Manufacturing Systems* 58 (Jan. 2021), pp. 176–192. ISSN: 02786125. DOI: 10.1016/j.jmsy.2020.11.017 (cit. on pp. 4, 7).

[9] L. D. Xu, E. L. Xu, and L. Li. "Industry 4.0: State of the art and future trends". In: *International Journal of Production Research* 56.8 (2018), pp. 2941–2962. ISSN: 1366588X. DOI: 10.1080/00207543.2018.1444806 (cit. on pp. 4, 7).

[10] G. Aceto, V. Persico, and A. Pescapé. "A Survey on Information and Communication Technologies for Industry 4.0: State-of-the-Art, Taxonomies, Perspectives, and Challenges". In: *IEEE Communications Surveys and Tutorials* 21.4 (Oct. 2019), pp. 3467–3510. ISSN: 1553877X. DOI: 10.1109/COMST.2019.2938259 (cit. on pp. 4, 5, 8).

[11] J. Prinsloo, S. Sinha, and B. von Solms. "A review of industry 4.0 manufacturing process security risks". In: *Applied Sciences (Switzerland)* 9.23 (Dec. 2019), p. 5105. ISSN: 20763417. DOI: 10.3390/app9235105 (cit. on p. 4).

[12] J. D. Contreras, J. I. Garcia, and J. D. Díaz Pastrana. "Developing of industry 4.0 applications". In: *International Journal of Online Engineering* 13.10 (Jan. 2017), pp. 30–47. ISSN: 18612121. DOI: 10.3991/ijoe.v13i10.7331 (cit. on p. 7).

[13] S. I. Shafiq et al. "Virtual engineering object/virtual engineering process: A specialized form of cyber physical system for industrie 4.0". In: *Procedia Computer Science*. Vol. 60. 1. Elsevier B.V., 2015, pp. 1146–1155. DOI: 10.1016/j.procs.2015.08.166 (cit. on p. 7).

[14] J. Jamaludin, Rohani, and Jemmy Mohd. "Cyber-Physical System (CPS): State of the Art". In: *2018 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube)*. 2018, pp. 1–5. ISBN: 9781538679395. DOI: 10.1109/ICECUBE.2018.8610996 (cit. on p. 7).

[15] H. Lasi et al. "Industry 4.0". In: *Business and Information Systems Engineering* 6.4 (Aug. 2014), pp. 239–242. ISSN: 18670202. DOI: 10.1007/s12599-014-0334-4 (cit. on p. 8).

[16] J. Lee, B. Bagheri, and H. A. Kao. "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems". In: *Manufacturing Letters* 3 (Jan. 2015), pp. 18–23. ISSN: 22138463. DOI: 10.1016/j.mfglet.2014.12.001 (cit. on p. 8).

[17] Y. Liu et al. "Review on cyber-physical systems". In: *IEEE/CAA Journal of Automatica Sinica* 4.1 (Jan. 2017), pp. 27–40. ISSN: 23299274. DOI: 10.1109/JAS.2017.7510349 (cit. on p. 9).

[18] C. Wang et al. "Machine learning in additive manufacturing: State-of-the-art and perspectives". In: *Additive Manufacturing* 36 (Dec. 2020), p. 101538. ISSN: 22148604. DOI: 10.1016/j.addma.2020.101538 (cit. on pp. 9, 11).

[19] Alpaydın Ethem. "Introduction". In: *Introduction to Machine Learning*. Ed. by T. Dietterich et al. Second. Cambridge, Massachusetts: The MIT Press, 2010. Chap. 1, pp. 1–20. ISBN: 978-0-262-01243-0. (Cit. on pp. 9, 11).

[20] Mohri Mehryar, Rostamizadeh Afshin, and Talwalkar Ameet. "Introduction". In: *Foundations of Machine Learning*. Ed. by Bach Francis. second. Cambridge, Massachusetts: MIT Press, 2018. Chap. 1, pp. 1–9. ISBN: 9780262039406 (cit. on pp. 9–11).

[21] J. H. Lee, J. Shin, and M. J. Realff. "Machine learning: Overview of the recent progresses and implications for the process systems engineering field". In: *Computers and Chemical Engineering* 114 (June 2018), pp. 111–121. ISSN: 00981354. DOI: 10.1016/j.compchemeng.2017.10.008 (cit. on pp. 9, 10, 13).

[22] N. Zincir-Heywood, M. Mellia, and Y. Diao. "Overview of Artificial Intelligence and Machine Learning". In: *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning*. Ed. by N. Zincir-Heywood, M. Mellia, and Y. Diao. 1st ed. Piscataway, New Jersey: Wiley-IEEE Press, 2021, pp. 19–32. DOI: 10.1002/9781119675525.ch2 (cit. on pp. 9–12).

[23] J. Wang and Q. Tao. "Machine Learning: The State of the Art". In: *IEEE Intelligent Systems* 23.6 (2008), pp. 49–55. DOI: 10.1109/MIS.2008.107. URL: www.computer.org/intelligent (cit. on p. 9).

[24] I. Goodfellow, Y. Bengio, and A. Courville. "Introduction". In: *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, pp. 1–27. ISBN: 9780262035613 (cit. on p. 12).

[25] J. Schmidhuber. "Deep Learning in neural networks: An overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 18792782. DOI: 10.1016/j.neunet.2014.09.003. arXiv: 1404.7828 (cit. on p. 13).

[26] X. Du et al. "Overview of deep learning". In: *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. Wuhan, China: IEEE, 2016, pp. 159–164. DOI: 10.1109/YAC.2016.7804882 (cit. on pp. 13, 14).

[27] G. Nguyen et al. "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey". In: *Artificial Intelligence Review* 52.1 (June 2019), pp. 77–124. ISSN: 15737462. DOI: 10.1007/s10462-018-09679-z (cit. on p. 14).

[28] Nvidia Corporation. *Deep Learning Frameworks*. 2021. URL: https://developer.nvidia.com/deep-learning-frameworks (visited on 11/09/2021) (cit. on p. 14).

[29] S. Sengupta et al. "A review of deep learning with special emphasis on architectures, applications and recent trends ". In: *Knowledge-Based Systems* 194 (2020), p. 105596. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2020.105596. URL: https://www.sciencedirect.com/science/article/pii/S095070512030071X (cit. on p. 14).

[30] O. A. Wahab et al. "Federated Machine Learning: Survey, Multi-Level Classification, Desirable Criteria and Future Directions in Communication and Networking Systems". In: *IEEE Communications Surveys and Tutorials* 23.2 (Apr. 2021), pp. 1342–1397. ISSN: 1553877X. DOI: 10.1109/COMST.2021.3058573 (cit. on p. 15).

[31] Q. Li et al. "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection". In: *IEEE Transactions on Knowledge and Data Engineering* (2021). ISSN: 15582191. DOI: 10.1109/TKDE.2021.3124599. arXiv: 1907.09693 (cit. on p. 16).

[32] Q. Yang et al. "Horizontal Federated Learning". In: *Federated Learning*. Ed. by Brachman Ronald J., Rossi Francesca, and Stone Peter. Morgan & Claypool, 2020, pp. 49–68. DOI: 10.2200/S00960ED2V01Y201910AIM043 (cit. on p. 16).

[33] Q. Yang et al. "Vertical Federated Learning". In: *Federated Learning*. Ed. by Brachman Ronald J., Rossi Francesca, and Stone Peter. Morgan & Claypool, 2020, pp. 69–82. DOI: 10.2200/S00960ED2V01Y201910AIM043 (cit. on p. 16).

[34] V. Mothukuri et al. "A survey on security and privacy of federated learning". In: *Future Generation Computer Systems* 115 (Feb. 2021), pp. 619–640. ISSN: 0167739X. DOI: 10.1016/j.future.2020.10.007 (cit. on pp. 17–24).

[35] P. Kairouz et al. "Advances and Open Problems in Federated Learning". In: *Foundations and Trends® in Machine Learning* 14.1–2 (Dec. 2021), pp. 1–210. DOI: 10.1561/2200000083. URL: http://arxiv.org/abs/1912.04977 (cit. on pp. 17, 23, 24).

[36] A. Nilsson et al. "A performance evaluation of federated learning algorithms". In: *DIDL 2018 - Proceedings of the 2nd Workshop on Distributed Infrastructures for Deep Learning, Part of Middleware 2018*. Association for Computing Machinery, Inc, Dec. 2018, pp. 1–8. ISBN: 9781450361194. DOI: 10.1145/3286490.3286559 (cit. on p. 18).

[37] S. Reddi et al. "Adaptive Federated Optimization". In: *ICLR 2021*. Feb. 2021. DOI: https://doi.org/10.48550/arXiv.2003.00295. arXiv: 2003.00295. URL: http://arxiv.org/abs/2003.00295 (cit. on p. 19).

[38] E. Bagdasaryan et al. "How To Backdoor Federated Learning". In: *International Conference on Artificial Intelligence and Statistics*. abs/1807.0 (July 2018), pp. 2938–2948. arXiv: 1807.00459. URL: http://arxiv.org/abs/1807.00459 (cit. on p. 20).

[39] J. Zhang et al. "Poisoning attack in federated learning using generative adversarial nets". In: *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019*. Institute of Electrical and Electronics Engineers Inc., Aug. 2019, pp. 374–380. ISBN: 9781728127767. DOI: 10.1109/TrustCom/BigDataSE.2019.00057 (cit. on p. 21).

[40] L. Wang, W. Wang, and B. Li. "CMFL: Mitigating communication overhead for federated learning". In: *Proceedings - International Conference on Distributed Computing Systems*. Vol. 2019-July. Institute of Electrical and Electronics Engineers Inc., July 2019, pp. 954–964. ISBN: 9781728125190. DOI: 10.1109/ICDCS.2019.00099 (cit. on p. 21).

[41] J. Lin, M. Du, and J. Liu. "Free-riders in Federated Learning: Attacks and Defenses". In: (Nov. 2019). arXiv: 1911.12560. URL: http://arxiv.org/abs/1911.12560 (cit. on p. 21).

[42] N. Bouacida and P. Mohapatra. "Vulnerabilities in Federated Learning". In: *IEEE Access* 9 (2021), pp. 63229–63249. ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3075203 (cit. on p. 22).

[43] S. Truex et al. "Demystifying Membership Inference Attacks in Machine Learning as a Service". In: *IEEE Transactions on Services Computing* 14.6 (Feb. 2019), pp. 2073–2089. ISSN: 19391374. DOI: 10.1109/tsc.2019.2897554 (cit. on p. 23).

[44] Z. Liu et al. "Privacy-Preserving Aggregation in Federated Learning: A Survey". In: *IEEE Transactions on Big Data* (2022), pp. 1–20. ISSN: 2332-7790. DOI: 10.1109/TBDATA.2022.3190835. URL: https://ieeexplore.ieee.org/document/9830997/ (cit. on pp. 23, 24).

[45] M. Akter et al. "Edge Intelligence: Federated Learning-based Privacy Protection Framework for Smart Healthcare Systems". In: *IEEE Journal of Biomedical and Health Informatics* (2022), pp. 1–11. ISSN: 2168-2194. DOI: 10.1109/JBHI.2022.3192648. URL: https://ieeexplore.ieee.org/document/9834059/ (cit. on p. 23).

[46] N. Franco et al. "Towards a Self-Adaptive Architecture for Federated Learning of Industrial Automation Systems". In: *Proceedings - 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2021*. Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 210–216. ISBN: 9781665402897. DOI: 10.1109/SEAMS51251.2021.00035 (cit. on p. 25).

[47] Y. Liu et al. "Communication-Efficient Federated Learning for Anomaly Detection in Industrial Internet of Things". In: *2020 IEEE Global Communications Conference, GLOBECOM 2020 - Proceedings*. Vol. 2020-Janua. Institute of Electrical and Electronics Engineers Inc., Dec. 2020. ISBN: 9781728182988. DOI: 10.1109/GLOBECOM42002.2020.9348249 (cit. on p. 25).

[48] K. I-Kai Wang et al. "Federated Transfer Learning Based Cross-Domain Prediction for Smart Manufacturing". In: *IEEE Transactions on Industrial Informatics* (2021). ISSN: 19410050. DOI: 10.1109/TII.2021.3088057 (cit. on p. 25).

[49] C. Jiang, C. Xu, and Y. Zhang. "PFLM: Privacy-preserving federated learning with membership proof". In: *Information Sciences* 576 (Oct. 2021), pp. 288–311. ISSN: 00200255. DOI: 10.1016/j.ins.2021.05.077 (cit. on p. 25).

[50] M. Hao et al. "Efficient and Privacy-Enhanced Federated Learning for Industrial Artificial Intelligence". In: *IEEE Transactions on Industrial Informatics* 16.10 (Oct. 2020), pp. 6532–6542. ISSN: 19410050. DOI: 10.1109/TII.2019.2945367 (cit. on p. 25).

[51] Y. Liu et al. "A Secure Federated Learning Framework for 5G Networks". In: *IEEE Wireless Communications* 27.4 (Aug. 2020), pp. 24–31. ISSN: 15580687. DOI: 10.1109/MWC.01.1900525. arXiv: 2005.05752 (cit. on p. 25).

[52] ISO, IEC, and IEEE. *ISO/IEC/IEEE 24765:2017(E)*. Tech. rep. New York: Institute of Electrical and Electronics Engineers, Inc, 2017, pp. 1–541. DOI: 10.1109/IEEESTD.2017.8016712. URL: www.iso.orgwww.ieee.org (cit. on pp. 27, 28).

[53] R. S. Peres et al. "Generative adversarial networks for data augmentation in structural adhesive inspection". In: *Applied Sciences (Switzerland)* 11.7 (Apr. 2021). ISSN: 20763417. DOI: 10.3390/app11073086 (cit. on pp. 38, 39).

[54] D. J. Beutel et al. "Flower: A Friendly Federated Learning Research Framework". In: *arXiv preprint arXiv:2007.14390* (July 2020). arXiv: 2007.14390. URL: http://arxiv.org/abs/2007.14390 (cit. on pp. 40, 58).

[55] R. G. Wijnhoven and P. H. De With. "Fast training of object detection using stochastic gradient descent". In: *Proceedings - International Conference on Pattern Recognition*. 2010, pp. 424–427. ISBN: 9780769541099. DOI: 10.1109/ICPR.2010.112 (cit. on p. 50).

[56] I. Mironov. "Renyi Differential Privacy". In: *Proceedings of IEEE 30th Computer Security Foundations Symposium CSF 2017* (Feb. 2017), pp. 263–275. DOI: 10.1109/CSF.2017.11. arXiv: 1702.07476. URL: http://arxiv.org/abs/1702.07476%20http://dx.doi.org/10.1109/CSF.2017.11 (cit. on p. 53).

[57] R. S. Peres et al. "Simulation-Based Data Augmentation for the Quality Inspection of Structural Adhesive with Deep Learning". In: *IEEE Access* 9 (2021), pp. 76532–76541. ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3082690 (cit. on p. 85).

APPENDIX

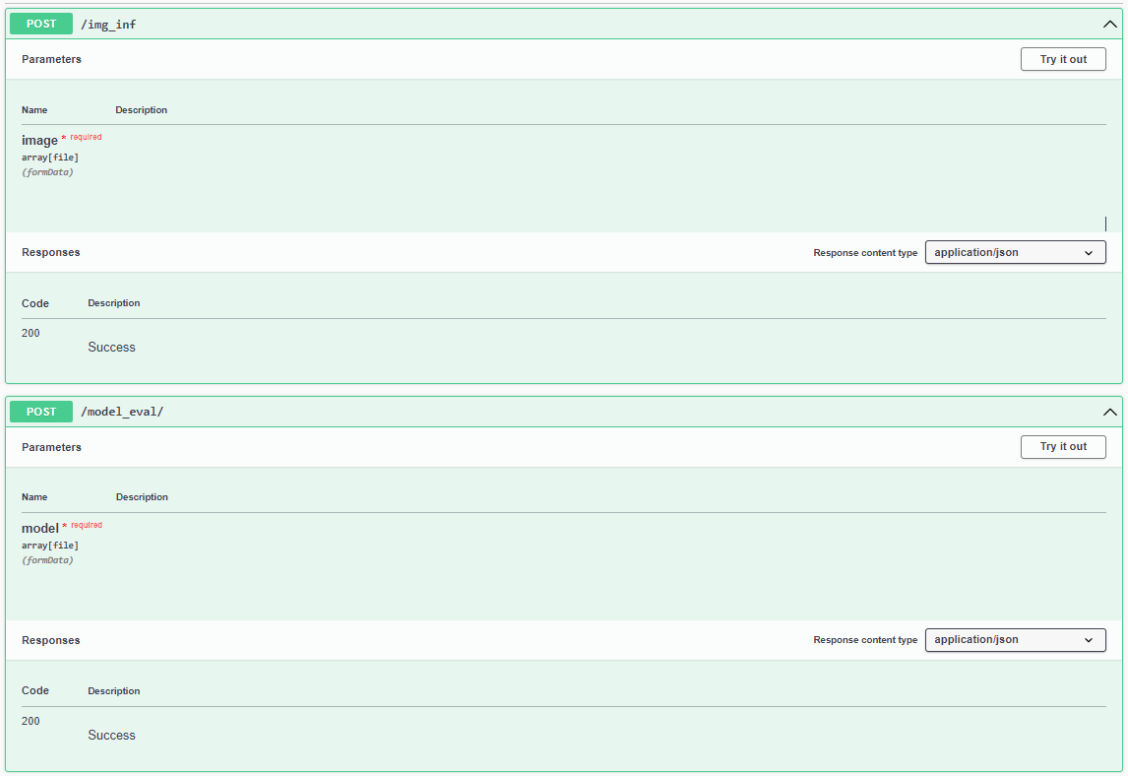
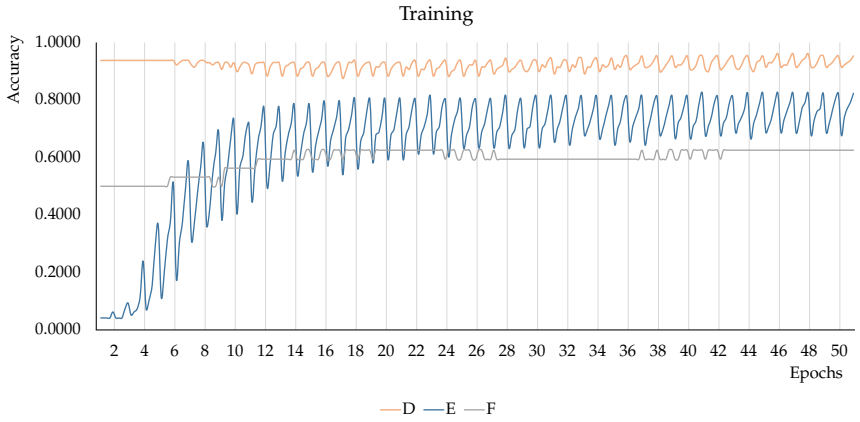
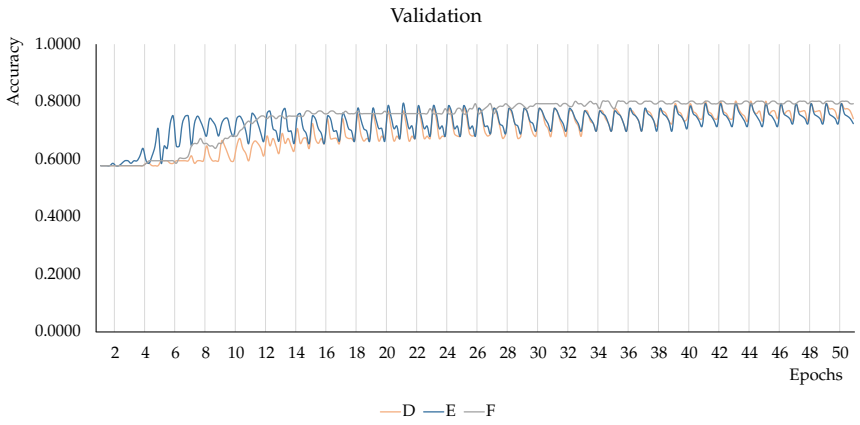


Figure A.1: LIN API endpoints.

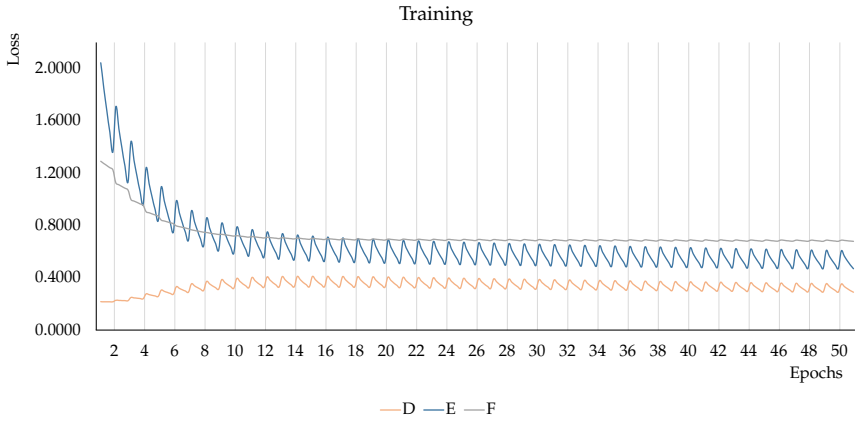


(a)

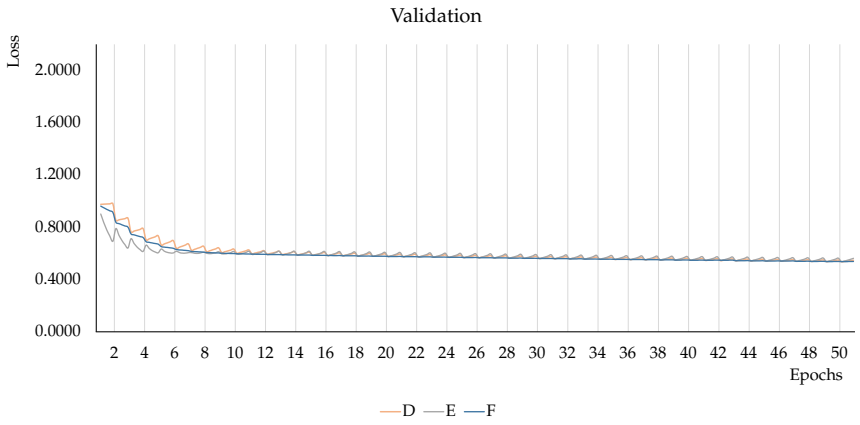


(b)

Figure A.2: Accuracy (a) and validation accuracy (b) graphics of the three stakeholders in FedAvg training with fifty global epochs and five local epochs.

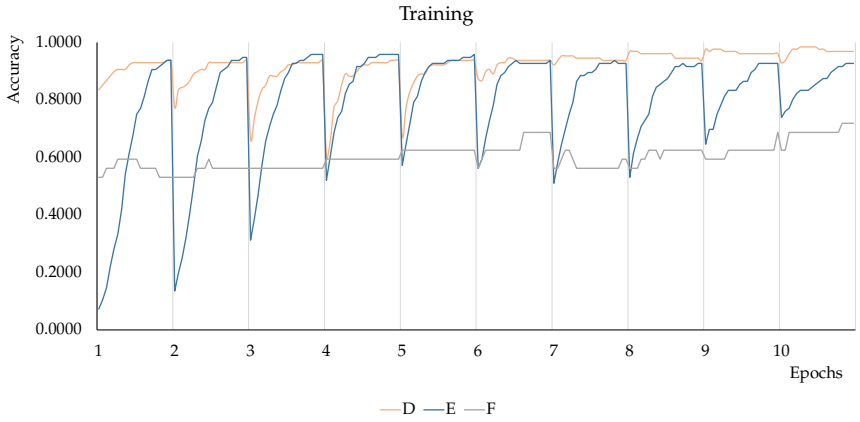


(a)

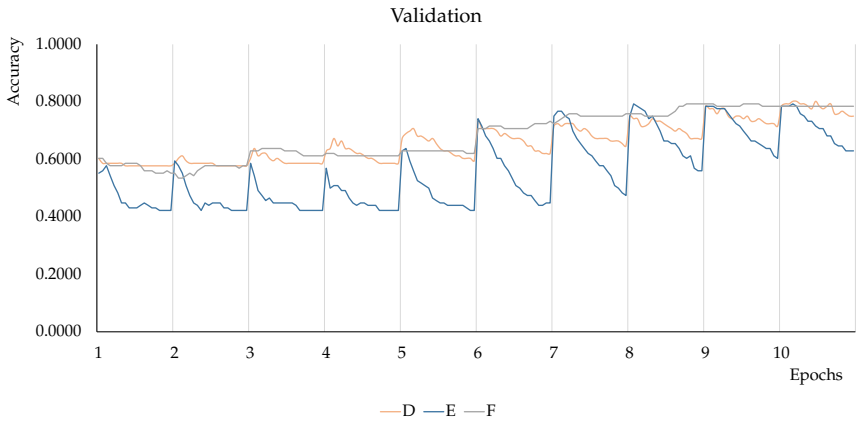


(b)

Figure A.3: Loss (a) and validation loss (b) graphics of the three stakeholders in FedAvg training with fifty global epochs and five local epochs.

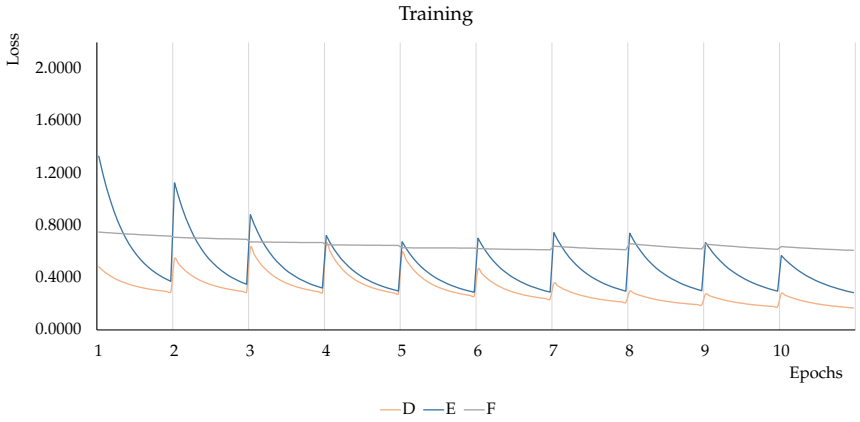


(a)

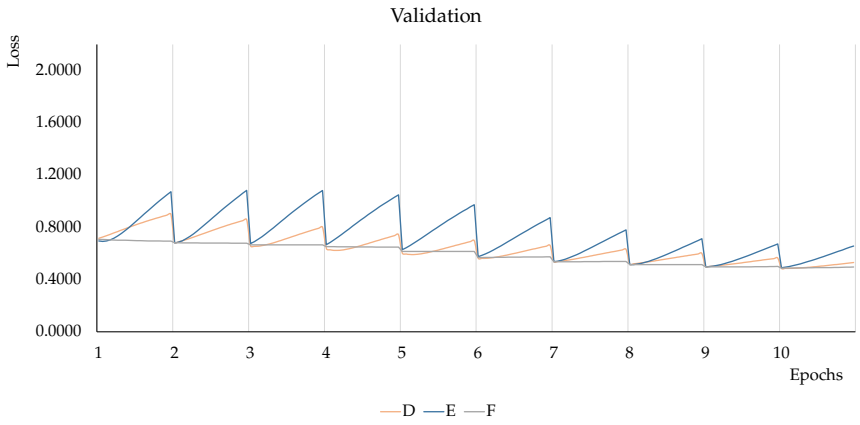


(b)

Figure A.4: Accuracy (a) and validation accuracy (b) graphics of the three stakeholders in FedYOGI training with ten global epochs and twenty local epochs.

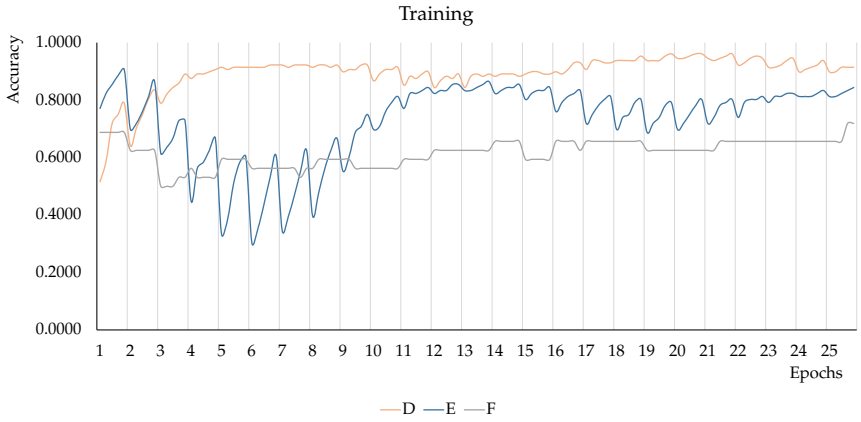


(a)

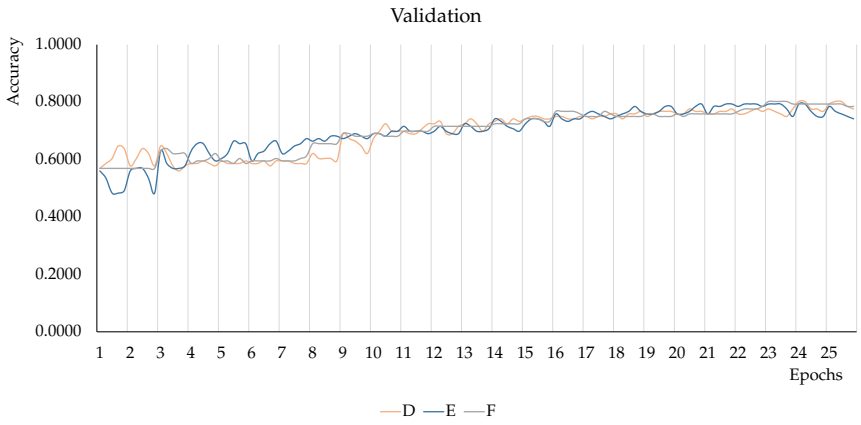


(b)

Figure A.5: Loss (a) and validation loss (b) graphics of the three stakeholders in FedYOGI training with ten global epochs and twenty local epochs.

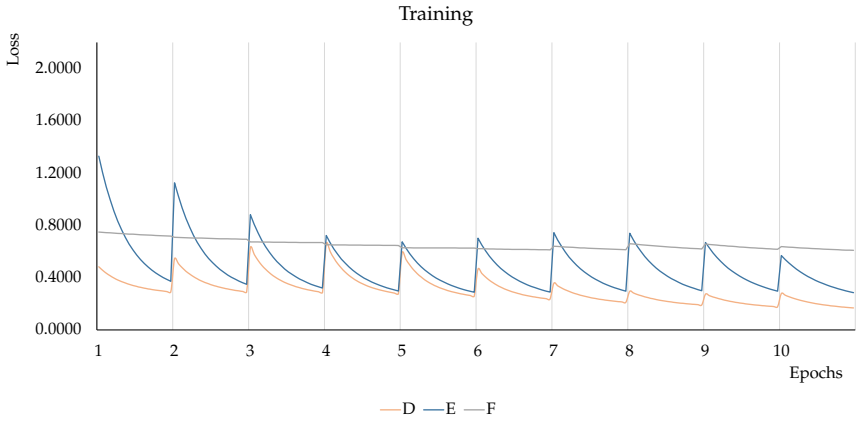


(a)

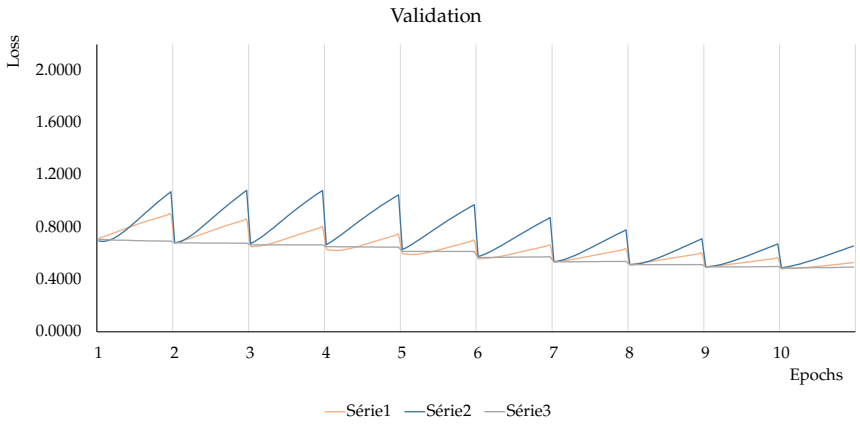


(b)

Figure A.6: Accuracy (a) and validation accuracy (b) graphics of the three stakeholders in FedYOGI training with twenty five global epochs and five local epochs.



(a)



(b)

Figure A.7: Loss(a) and validation loss (b) graphics of the three stakeholders in FedYOGI training with twenty five global epochs and five local epochs.