INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Aprendizagem federada para comunidades de agentes que permita a mobilidade dinâmica em ambientes de proteção de privacidade

BRUNO RAFAEL GONÇALVES RIBEIRO
Abril de 2024

P.PORTO

Federated learning for agent communities to enable dynamic mobility in privacy-preserving environments

Bruno Rafael Gonçalves Ribeiro

Student n.: 1180573

Dissertation to obtain the master's degree in Artificial Intelligence Engineering

Supervisor: Doctor Luis Gomes, Polytechnic of Porto - School of Engineering

Co-supervisor: Doctor Zita Vale, Principal Coordinator Professor, Polytechnic of Porto - School of Engineering

Jury:

President:

[Nome do Presidente, Categoria, Escola]

Members:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, April 2024

«Commit your works to the LORD
And your plans will be established.»

Proverbs 16:3

Resumo

A privacidade e a segurança dos dados têm sido uma das principais preocupações dos utilizadores da Internet nos últimos anos. Com o aumento do uso de inteligência artificial e dos dados existentes na Internet, os utilizadores começaram a ficar preocupados com a forma como as empresas utilizavam os seus dados. Com isto, vários países começaram a desenvolver os seus próprios regulamentos de proteção de dados. Para cumprir estes regulamentos e continuar a oferecer serviços baseados na inteligência artificial, as empresas e os investigadores criaram uma técnica chamada aprendizagem federada.

A aprendizagem federada distingue-se da aprendizagem automática tradicional ao deslocar o processo de treino do servidor central para os clientes. Os clientes treinam o modelo com os seus dados locais e, em seguida, o modelo é enviado de volta para o servidor, onde os agregará num único modelo. Esta abordagem ajuda a distribuir o poder computacional e a evitar a exposição dos dados dos clientes. No entanto, há problemas ainda inexplorados pela comunidade de investigação, que dizem respeito ao lado dinâmico da aprendizagem federada. A investigação atual parte do princípio de que os clientes estão fixos numa federação durante toda a execução, que os seus dados não mudam ao longo do tempo e que não existem outras opções para além da federação em que se encontram a treinar.

Os sistemas multiagentes, por outro lado, são um conceito que se baseia fundamentalmente em ambientes dinâmicos e que mudam com o tempo. Estes sistemas são utilizados para conceber e simular cenários realistas, incorporando características humanas, como a inteligência, a proactividade e a adaptabilidade, em entidades chamadas agentes. A sua capacidade adaptativa e comportamental que os permite seguir um objetivo pre-estabelecido faz com que se torne relevante para a aprendizagem federada, no sentido em que os clientes podem ver o seu ambiente a ser alterado, como os seus objetivos iniciais, a distribuição e a quantidade dos seus dados, entre outros. Esta abordagem de treino colaborativo com as capacidades adaptativas e dinâmicas dos agentes pode contribuir para tornar os sistemas de aprendizagem federados mais robustos e realistas.

Esta dissertação tem como objetivo combinar ambos os conceitos através de uma ferramenta de desenvolvimento dedicada, validando a sua eficácia num cenário em que os clientes podem escolher qual o melhor modelo de federação a utilizar. A framework proposta chama-se PEAK-FL e permite a criação de sistemas de aprendizagem federados baseados em agentes. Esta solução visa ajudar os utilizadores a concetualizar, a construir e a analisar as vantagens e desvantagens que este tipo de sistemas apresentam. Foram realizados vários casos de estudos para validar as suas várias componentes e os resultados mostram que, quando corretamente aplicada, esta combinação pode melhorar os resultados.

Palavras-Chave: Aprendizagem Federada, Sistemas Multi-Agente, Aprendizagem de máquina, Mobilidade Dinâmica de Clientes, Programação Genética

Abstract

Data privacy and security have been a main concern of internet users for the last few years. With the rise of artificial intelligence applications and the amount of data available on the internet, users started to become worried about the companies using their data. Several countries started to develop their own data protection regulations to protect their users. To abide by the regulations and still offer artificial intelligence-based services, companies and researchers come up with a technique called federated learning.

Federated learning distinguishes itself from traditional machine learning by moving the training process from the central server to the clients. The clients train the model with their local data and then they send the model back to the server where it will aggregate them into a single model. This process is repeated until the desired performance is reached. This approach helps to distribute the computational power and to avoid exposing the clients' data. However, some problems were not yet fully explored by the research community, regarding the dynamic side of federated learning. Current research assumes that the clients are fixed in one federation for the entire run and that their data does not change throughout time and that there are no other options besides the federation they are in.

Multi-agent systems, on the other hand, is a concept that is fundamentally based on dynamic and time-changing environments. These systems are used to design and simulate realistic scenarios by incorporating human characteristics, like, intelligence, proactiveness and goal-oriented behaviours, in these entities called agents. They can work together to achieve a common goal and adapt to the circumstances they are in. This is relevant to federated learning in the sense that the clients have a common goal, and the environment can change over time, like the data type and distribution the client has, and better federations might appear, creating the need for clients to change their behaviour throughout time, so the model performance is not reduced, and they can still achieve their desired goal. This collaborative training approach with the adaptive and dynamic agent capacities to work in unstable environments can work together to make federated learning systems more robust and realistic.

This dissertation aims to combine both concepts through a dedicated development framework and its validation in a scenario where the clients can choose which federation model is better for them to use. The novel framework proposed is called PEAK-FL and allows the creation of federated learning systems based on agents. This solution aims to help the users conceptualize and build this kind of system and explore more deeply the benefits and drawbacks that these systems have. Several case studies were conducted to validate several components of the framework and the results show that, when properly applied, the integration of agents can help the clients to achieve better results.

Keywords: Federated Learning, Multi-Agent Systems, Machine Learning, Dynamic Client Mobility, Genetic Programming

Acknowledgements

Firstly, and foremost, I would like to thank personally the greatest scientist of all time, who is great in wisdom, knowledge, understanding and all kinds of craftsmanship, the one who dominates the art of teaching, the art of love and the art of life. The one who is undoubtedly the greatest being that ever lived and is still living, as you are the life itself, Jesus Christ of Nazareth. I am grateful for the life you gave me, the family and friends, the professors and colleagues, and the people that surround me in my daily life. I am grateful for this world that you created with wisdom, full of value, meaning and purpose, full of adventures and knowledge to uncover. I am grateful for the spiritual support and guidance you gave me throughout the development of the dissertation, the revelations and the ups and downs it had that helped me build my vision towards life and the future. I am grateful to you God as you have always been with me.

I cannot thank enough my family, my father and mother, my brothers and sister, and my friends, Pedro Girão, Marcelo, Ricardo and all the others that make part of my life for all the important moments we had. Thank you for the emotional and spiritual support you gave me, for the endless conversations we had about my academic journey, the dissertation, my life and my future and the endless moments of pure joy and happiness.

I could not also forget the intense effort that my professors Luis Gomes and Zita Vale put into the success of my academic path directly and indirectly. Thank you for the help, ideas, and insights you gave me about the problems and decisions I had to make throughout these years. Luis Gomes, I would like to thank you for the numerous conversations and discussions we had. Thank you for the trust and responsibility you gave throughout these years, and the patience you had with me. Professor Zita Vale, I would like to thank you for the opportunity you gave me in this research centre I work in and for providing me with the tools and environment to develop my academic skills and to build my career path. I deeply appreciate it.

I thank you all and everyone else who in one way or another became part of this journey, helped me get new ideas and help get through the difficult moments of this journey. Thank you all, and God bless you eternally. Amen.

Table of Contents

List of Figures

List of Tables

Acronyms

ACL Agent Communication Language

AI Artificial Intelligence

API Application Program Interface

ANN Artificial Neural Network

BDI Beliefs Desires Intentions

BOID Beliefs Desires Intentions Obligations

CNN Convolutional Neural Network

DF Directory Facilitator

DFL Decentralized Federated Learning

DL Deep Learning

DNN Deep Neural Network

ESS Energy Storage System

FEMNIST Federated Extended Modified National Institute of Standards and Technology

FedAvg Federated Averaging

FedTour Federated Tournament

FGP Flexible Genetic Programming

FIPA Foundation for Intelligent Physical Agents

FL Federated Learning

GAN Generative Adversarial Network

GDPR General Data Protection Regulation

GECAD Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development

GP Genetic Programming

gRPC Google Remote Procedure Call

GUI Graphical User Interface

HTTP	Hypertext Transfer Protocol
ICL	Inter-Agent Communication Language
IID	Independent and Identically Distributed data
IoT	Internet of Things
JADE	Java Agent Development Framework
JID	Jabber Identification
JSON	JavaScript Object Notation
KIF	Knowledge Interchange Format
KQML	Knowledge Query Manipulation Query
MARL	Multi-Agent Reinforcement Learning
MAS	Multi-Agent Systems
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
MNIST-ROT	Rotated Modified National Institute of Standards and Technology
MUC	Multi-User Chat
NLP	Natural Language Processing
OAA	Open Agent Architecture
PEAK	Python-based framework for heterogeneous agent communities
PEAK-FL	Python-based framework for heterogeneous agent communities for federated learning
REST	Representational State Transfer
RNN	Recurrent Neural Network
SL	Supervised Learning
SPADE	Smart Python-based Agent Development Environment
XMPP	Extensible Messaging and Presence Protocol

1 Introduction

This chapter provides the reader with the main motivations of this dissertation, the targeted objectives, and the scientific contributions accomplished, as well as a description of the structure of this document. The chapter is divided into four sections. In the Contextualization section, the background, motivations, and reasons for the development of the dissertation are given. In the Research Questions and Objectives section, the research questions and the objectives that drove the development process of the dissertation are presented. The Scientific Contributions section describes the research projects that supported the development of the dissertation, the scientific papers that were developed and published as well as some awards that were received. Finally, the Document Structure section gives a light description of what each chapter in this dissertation is about and what is expected to be found in them.

1.1 Contextualization

Nowadays, the world is moved by technology, where everything is becoming smarter and more connected with the digital world, therefore, increasing the rate at which the data is produced and stored (Petroc Taylor, 2023). As this rises some concerns involving the security and privacy of the data, the European Union, and other countries, created regulations to defend internet users and their data. The European Union called their regulation General Data Protection Regulation (GDPR), and it states guidelines on how the companies and services available on the internet ought to manage users' data (European Union, 2018). This regulation caused some preoccupation on the industry side as it constrained the accessibility of the users' data, especially in small and medium-sized enterprises (Presidente & Frey, 2022).

With this problem at hand, researchers tried to come up with a solution that could overcome this data accessibility limitation. One promising solution found was federated learning (Q. Li et al., 2023). In federated learning, the main idea is to change the way machine learning models are trained by shifting the training process from the central server to the clients. In this process,

the clients do not have to share the data with the central server, instead, they will receive the model from the server and train it with their data locally. After the training, the clients send the model back to the server and the server will aggregate all the clients' models into one aggregated model. This way, federated learning prevents the clients' data from being exposed to other entities. This training process is then repeated until the model reaches the desired performance.

Despite this intrinsic capability of federated learning to avoid exposing the clients' data to other entities while training a machine learning model collaboratively, it does not ensure the good performance of the model or even the safety of the client's data (Kairouz et al., 2021). The unpredictability of the quality and amount of data that each client has can make the training of the model more difficult. In addition to that, malicious actors can still have access to the original client's data if the necessary security measures are not taken (P. Liu et al., 2022a). These problems have been the main concern of the scientific community in this field. However, there is a need to tackle some other challenges that are not so much explored in the literature. One of them is the lack of dynamic mobility of the clients between federations during runtime (E. Rizk et al., 2020). In a real and competitive world, these situations will occur frequently, where new federations might appear that serve better the purpose of the client, for example. Another challenge that needs to be considered is the assumption made by the researchers that the entities participating in the federation know each other previously and that they have access to the same technology. In situations where the federated learning is maintained between a company and its clients its sufficient, but that limits the applicability of this technology. So, a framework that can enable the creation of a federated learning platform that can manage the demand and response between servers and clients and that allows interoperability between systems is needed.

The lack of dynamism regarding clients' mobility among federations, self-improvement, autonomy, and interoperability in federated learning could be improved if combined with another mature concept called multi-agent systems (Falco & Robiolo, 2019). These systems are founded in individual entities called agents which can work together to achieve a common goal. They can represent and replicate real-life entities, like clients and servers in federated learning. A lot of research in this field has also been done towards system interoperability, self-adaptation, and decision-making (Rodrigues et al., 2014). Multi-agent systems can also take advantage of federated learning to share knowledge between agents without sharing personal data. The benefits and drawbacks of multi-agent systems in federated learning have yet to be explored thoroughly and at the moment no development framework allows this exploration to take place.

For that reason, this dissertation aims to achieve the integration of these two concepts through the development of a dedicated framework, and the testing and validation of such systems in energy community-related case studies. The framework proposed is called Python-based framework for heterogeneous agent communities for federated learning (PEAK-FL). PEAK-FL enables the conceptualization, design, development, testing, and analysis of federated learning solutions based on autonomous and intelligent agents by providing the basic structure to

implement these systems. The framework allows the monitorization of the agents and the data being published in the system in real-time, including the evolution of the models in the federations. PEAK-FL was tested in two different case studies and the results show that solutions based on multi-agent systems can indeed improve the performance of federated learning models.

1.2 Research Questions and Objectives

For this dissertation, a list of research questions and objectives was formulated to help guide the research and its development process. Regarding the research questions, there is one main question that is the focus of the dissertation, and it was further divided to help identify better the problems to be answered by the dissertation. The list of the research questions is the following:

- RQ1 – What benefits and drawbacks can multi-agent systems, a dynamic self-improvement distributed approach, have in automating and dynamizing federated learning, a private and collaborative learning environment approach?
    - RQ1.1 - What open-source state-of-the-art federated learning frameworks are flexible enough to allow the integration of different machine learning models (i.e. genetic programming) and other frameworks?
    - RQ1.2 – How can federated learning be integrated with multi-agent systems to provide a dynamic distributed learning environment?
    - RQ1.3 - How can federated learning help the agents achieve their goals more effectively and securely?
    - RQ1.4 – What characteristics inherent in agent-based systems make federated learning scenarios more realistic?
    - RQ1.5 - To what extent can the dynamism of intelligent goal-oriented agents improve model accuracy in federated learning contexts?

The objectives were formulated according to the research questions to be answered and to sequence the steps to be made throughout the dissertation development. The list of the formulated objectives is the following:

- O1 – Explore the state of the art of multi-agent systems, federated learning, and machine learning.
- O2 - Explore the state of the art about previous integrations of federated learning with multi-agent systems regarding dynamic distributed learning environments.

- O3 - Explore the open-source federated learning frameworks available and their flexibility in integrating different models and other frameworks.
- O4 - Explore the use of machine learning alternatives, like genetic programming, in federated learning settings.
- O5 – Conceptualize, implement, and validate a solution to integrate genetic programming as the machine learning model to train in federated learning.
- O6 - Propose new agent architecture models for client and server participation in federated learning environments.
- O7 - Conceptualize, implement, and validate a federated learning development framework based on intelligent, autonomous, and goal-oriented agents.

Table 1 shows the relationship between each research question and the objectives formulated.

Table 1 – Association between the research question and the proposed objectives

Research Questions	Objectives
RQ1	O1, O2, O7
RQ1.1	O3, O4, O5
RQ1.2	O2, O6, O7
RQ1.3	O1, O7
RQ1.4	O1, O2, O7
RQ1.5	O2, O7

1.3 Scientific Contributions

The present dissertation was developed in collaboration with the Research Group on Intelligence Engineering and Computing for Advanced Innovation and Development (GECAD). GECAD is a research unit located in the Polytechnic of Porto–School of Engineering (ISEP) that uses artificial intelligence systems to solve interdisciplinary problems in different domains.

The development of the dissertation was supported by project RETINA[1] - Real-time support infrastructure and energy management for Intelligent carbon-neutral smart cities, funded by the Portuguese Foundation of Science and Technology (FCT). The work of the current dissertation also contributed to the following projects:

- IoTalentum[2] – EU Horizon 2020 Marie Skłodowska-Curie European Training Network on Internet of Things.

[1] RETINA (NORTE-01-0145-FEDER-000062) - https://www.gecad.isep.ipp.pt/RETINA/
[2] IoTalentum - https://iotalentum.eu/

- PRECISE[3] – Power and Energy Cyber-Physical Solutions with Explainable Semantic Learning.

The work of the dissertation contributed also to the publishing of 10 scientific research papers. The following list shows 5 papers published as first author where the author was involved in all the processes of the work:

- [Conference] **Bruno Ribeiro**, Luis Gomes, Zita Vale (2023) "A novel federated learning approach to enable distributed and collaborative genetic programming", presented and published in 22nd Portuguese Conference on Artificial Intelligence (EPIA 2023), DOI: 10.1007/978-3-031-49011-8_16

- [Conference] **Bruno Ribeiro**, Luis Gomes, Ricardo Faia, Zita Vale (2023) "Federated Genetic Programming: A Study About the Effects of Non-IID and Federation Size", presented and published in 20th International Conference on Distributed Computing and Artificial Intelligence (DCAI 2023), DOI: 10.1007/978-3-031-38333-5_20

- [Conference] **Bruno Ribeiro**, Luis Gomes, Rafael Barbarroxa, Zita Vale (2023) "A Novel Framework for Multiagent Knowledge-based Federated Learning Systems", presented and published in The PAAMS Collection: Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. (PAAMS 2023), DOI: 10.1007/978-3-031-37616-0_25

- [Conference] **Bruno Ribeiro**, Ricardo Faia, Luis Gomes, Zita Vale (2023) "Energy Community Integration of a Smart Home Based on an Open Source Multiagent System", presented and published in The PAAMS Collection: Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. (PAAMS Demonstrations 2023), DOI: 10.1007/978-3-031-37616-0_35

- [Conference] **Bruno Ribeiro**, Helder Pereira, Luis Gomes, Zita Vale (2022) "Python-based Ecosystem for Agent Communities simulation", published in 17th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2022), DOI: 10.1007/978-3-031-18050-7_7

In addition to the previously listed papers, there were also contributions made to 6 additional scientific papers, mainly in terms of designing, modelling, and implementing systems using multi-agent systems and Internet of Things (IoT) devices, and helping to analyse the data gathered by the systems:

- [Journal] Ricardo Faia, **Bruno Ribeiro**, Calvin Gonçalves, Luis Gomes, Zita Vale (2023) "Multi-agent based energy community cost optimization considering high electric vehicles penetration" published in Sustainable Energy Technologies and Assessments (SETA), DOI: 10.1016/j.seta.2023.103402

[3] PRECISE (PTDC/EEI-EEE/6277/2020) - https://www.gecad.isep.ipp.pt/precise/

- [Journal] Helder Pereira, **Bruno Ribeiro**, Luis Gomes, Zita Vale (2022) "Smart Grid Ecosystem modelling using a novel framework for heterogeneous agent communities" published in Sustainability, DOI: 10.3390/su142315983

- [Journal] Cátia Silva, Pedro Faria, **Bruno Ribeiro**, Luis Gomes, Zita Vale (2022) "Demand Response Contextual Remuneration of Prosumers with Distributed Storage" published in Sensors, DOI: 10.3390/s22228877

- [Conference] Luis Gomes, **Bruno Ribeiro**, Fernando Lezama, Zita Vale (2023) "A Multi-Agent System Empowered by Federated Learning and Genetic Programming", published in 31st Signal Processing and Communications Applications Conference (SIU 2023), DOI: 10.1109/SIU59756.2023.10223778

- [Conference] Pedro Carvalho, **Bruno Ribeiro**, Nuno Rodrigues, João Batista, Leonardo Vanneschi, Sara Silva (2023) "Feature Selection on Epistatic Problems Using Genetic Algorithms with Nested Classifiers", published in Applications of Evolutionary Computation (Evo* 2023), DOI: 10.1007/978-3-031-30229-9_42

- [Conference] Rafael Barbarroxa, **Bruno Ribeiro**, Luis Gomes, Zita Vale (2024) "Benchmarking AutoGen with different large language models", accepted for submission in IEEE Conference on Artificial Intelligence (IEEE CAI 2024)

The author of this dissertation also presented 2 of the listed papers at international conferences, namely:

- 21st International Conference on Practical applications of Agents and Multi-Agent System (PAAMS 2023), Guimarães, Portugal

- 21st International Conference on Distributed Computing and Artificial Intelligence (DCAI 2023), Guimarães, Portugal

- 22nd Portuguese Conference on Artificial Intelligence (EPIA 2023), Faial, Açores, Portugal

As a result of the in-person demonstration of the paper "Energy Community Integration of a Smart Home Based on an Open Source Multiagent System" an award was won, contemplating the 3rd prize of the AIR Institute Award of Scientific Excellence to the Best Demonstration presented during the 21st International Conference on Practical applications of Agents and Multi-Agent (PAAMS 2023).

Last, but not least, 5 dissemination activities were also conducted to reach a wider audience with the work produced during the development of this dissertation. The list of dissemination activities is the following:

- Presented 2 Advanced Training Sessions in the RETINA project on how to model smart grids and energy communities using the multi-agent system framework PEAK.
- Conducted a workshop in GECAD about building multi-agent systems with the framework PEAK.
- Presented the PEAK framework at IoTalentum and gave a tutorial on how to use PEAK to simulate real scenarios by integrating real smart devices in multi-agent systems.
- Presented a session on how to model energy problems with multi-agent systems using the PEAK framework at GECAD's Energy Day.

1.4 Document Organization

This dissertation is structured around the following chapters:

Chapter 1 is the Introduction. In this chapter, the vision of the dissertation is described in detail by the contextualization of the problem and its solution. The research questions and objectives that guided the development process are also identified. The list of scientific contributions achieved during the execution of the dissertation is presented. Finally, the document organization, the current section, clarifies and helps the visualization of how this dissertation is constructed and articulated to answer the problem at hand as well as to clarify and transmit a clear message to the reader.

Chapter 2 talks about the State of the Art. In this chapter, the research and investigation that was done to understand better the open problems and possible solutions is done descriptively. It guides the reader through the concepts of federated learning and multi-agent systems, what advantages these systems have, what problems they currently face, and ultimately if they can be combined to deliver a good efficient, effective, and concrete solution to the problem at hand.

Chapter 3 describes the Materials, Methods, and Technological challenges of this dissertation. Here it describes the materials, methods, frameworks, libraries, and datasets the were used in the development phase. The context and the reasons why this material was chosen are given, and due credit is given to its authors. In the end, the technological challenges are described by exploring the problems of privacy and security of data and artificial intelligence challenges the engineers and companies face these days to make the use of AI models possible.

Chapter 4 describes the Novel development framework for dynamic federated learning systems proposed in this dissertation. Here are described all the details about the proposed solution and its components. Firstly, is given a description of PEAK and its base functionalities. Then the functionalities that were added to enable the integration of the federated learning in PEAK are described. Finally, a description of the federated learning framework, Flower, and the design decisions made to integrate it in PEAK.

Chapter 5 describes the Case Studies conducted during the development of the dissertation. In this chapter, the author describes the case studies that were developed to explore and test the functionalities of the proposed solution and what benefits the solution shows. It also shows how each case study contributed to the publishing of scientific papers.

Chapter 6 has the Conclusions of the dissertation. Here the main conclusions about the results of the dissertation are given and it is explained how the research questions were answered and how the objectives were accomplished. Further, possible future paths regarding the development of the proposed solution and the continuation of this research area are discussed.

2 State of the Art

This chapter describes the state of the art of the topics addressed by this dissertation, which are multi-agent systems, federated learning, and supervised machine learning. The topics are explored individually but the applications of these three concepts together are also analysed. Throughout the chapter, several different gaps are identified and analysed.

2.1 Multi-Agent Systems

In Artificial Intelligence (AI) several subdomains work together to build intelligent systems. However, there is one subdomain that instead of focusing on the mechanisms of how to make software intelligent focuses on how several intelligent entities can work together and be organized to reach a specific goal. That subdomain is the Multi-Agent Systems (MAS), and as the name suggests, these systems are composed of multiple individual entities called agents (Falco & Robiolo, 2019). To understand better how they work, it is important to describe and define what is an agent. In the literature, there are multiple different definitions depending on the domain and the problem that is being applied (Russell & Norvig, 2009; Wooldridge, 2009). However, a widely accepted definition comes from Wooldridge (2009) and he defines it as anything that can understand the environment around it using sensors and can act upon it with actuators. A hardware agent could be a robot with infrared cameras as sensors and motors as actuators. A software agent is intelligent software that uses the digital environment it is currently in to achieve its goal. The sensors of the agent can be the keys pressed, or file contents and actuators can be the monitor to print some information or write in files. The agent must have a goal that it must fulfil during its lifecycle. To do that, the agent will gather information through its sensors, so it can interpret that information to make its decisions. An efficient agent will make the best decisions possible to maximize its probability of achieving its goals.

In a MAS the agents can be either homogeneous or heterogeneous (Mahela et al., 2020). Homogeneous MAS are MAS where every agent is of the same type and has the same

architecture and characteristics. On the other hand, heterogeneous MAS, which are more common in real-life applications, are MAS that consist of several different types of agents with different architectures and characteristics (Ruiz de Gauna et al., 2020). The main objective of MAS is to solve problems where the knowledge and information are distributed across time and space (Jubair et al., 2018) and can normally be deconstructed into several smaller tasks which single agents can solve. The only way that MAS can do it is by relying on the agents' communication and negotiation skills (Owoputi & Ray, 2022). These skills can be configured differently depending on the environment they are in (i.e., cooperative, or competitive).

The environments in which MAS are deployed are very important since they will dictate which architecture the agents will have. There are at least three different environments, namely, cooperative, collaborative, and competitive (Hoen et al., 2006). A cooperative environment is when the agents have different goals, but they can work together to achieve them individually. (J. Wang et al., 2022). In collaborative environments, the agents will have a common shared goal and they will work together to achieve it. In competitive environments, the agents will try to outrun the other to achieve their own goals. In addition to these MAS environments, it is possible to create hybrid environments (Dorri et al., 2018), for example, robots playing soccer show a collaborative and competitive environment (Duan et al., 2012).

One of the advantages of using MAS in resolving distributed problems is increasing the speed and efficiency of the system by using parallel computing and asynchronous operations (Rousset et al., 2016). Scalability and flexibility are two other advantages given that agents can be added as and when necessary and can adapt to their environment more easily (Charbonnier et al., 2022). In case of same parts of the system fail, MAS provide graceful degradation, avoiding catastrophic failure, by maintaining the minimum required functionalities (Moradi et al., 2017). This increases the robustness and reliability of the system. An individual agent costs less to maintain and implement than a centralized architecture (Balaji & Srinivasan, 2010). Another advantage is the reusability of the several components of the system. The agents in MAS follow modular structures which allows their replacement and upgrade more easily compared to monolithic systems.

2.1.1 Agent Architectures and Characteristics

Given the vast utility of MAS, the agents can be used to solve a variety of different problems (Gronauer & Diepold, 2022; Sharma et al., 2020). For the agent to fulfil its objectives more efficiently, it must have a proper architecture. Every agent must have at least the following three components (Gouws, 2021): (i) the sensors, that will be used to gather the information needed about the environment, (ii) the actuators, that are used by the agent to perform actions in the environment, and (iii) a reasoning engine, where the information gathered though the sensors are processed, generating an action to be performed using the actuators. The reasoning engine can be further divided into several subcomponents. Normally one of the components is a knowledge system that works like a human memory where it can persist the information (W. Shen et al., 2019), and a decision-making system, where the decisions of the agent are

calculated and are normally driven by a goal or a task that the agent was designed to accomplish (Y. Yang & Wang, 2020). There are a lot of variations in agent architecture design, but these are the essential parts of building an agent.

Depending on the complexity of the agent's task or goal, the architecture can vary from a very simple architecture with only the basic components to a very complex architecture by including more advanced abilities (Y. Rizk et al., 2020). The fundamental abilities of an agent are autonomy, social ability, reactivity, and pro-activeness (Wooldridge, 2009). Additional abilities can be integrated into the agents to help them achieve their goals in better ways, however, they are not required. These additional abilities are mobility, veracity, benevolence, believability, rationality, inferential capability, temporal continuity, situatedness, locality, openness, intelligence, learning, adaptation capabilities, flexibility, and more (Wooldridge, 2009). A detailed description of these characteristics is given in Table 2.

Table 2 - List of extra characteristics an agent can have. Adapted from Wooldridge (2009).

Characteristics	Description
Adaptation capabilities	Adapt to unforeseen situations
Autonomy	Have control over its internal state and its behaviours
Social ability	Can interact with other entities
Believability	The actions are based on its beliefs
Benevolence	Does not have conflicting goals and will perform what is asked for
Flexibility	Can dynamically change its behaviour according to the changing environment
Inferential Capability	The ability to make conclusions based on previously gathered information
Intelligence	Makes the most out of its knowledge
Learning	Learning from past experiences
Locality	Can act on its local environment without full knowledge of the environment
Mobility	Move around the environment (virtual or physical)
Openness	Can deal with entities leaving and joining the system
Proactiveness	Takes initiative by taking actions
Rationality	The actions are goal-oriented
Reactivity	Perceives the environment and acts accordingly
Situatedness	The way the agent interacts with its environment through its sensors and actuators
Temporal Continuity	Runs continuously
Veracity	The shared information will not be knowingly false

Figure 1 shows the categorization of agent architectures and their relationship to each other. Based on the different characteristics, agents can have reactive architectures, proactive architectures, or hybrid architectures. The reactive architectures are normally rule-based and are most suited for simple environments where the agent has the full view of the environment and knows the effects of each action it can make (Mualla et al., 2019). Normally they have a

mechanism that maps the different states of the environment to specific actions that must be performed. Reactive architectures can be further divided into simple reflex agents and model reflex agents. The simple reflex agent is the simplest an agent can get as it will only react to inputs made by the environment and will act accordingly (Chudy et al., 2020). The model reflex agent, also called the deliberative reasoning agent, has an internal model that models not only the agent's partial view but the environment as well (Belani et al., 2019).

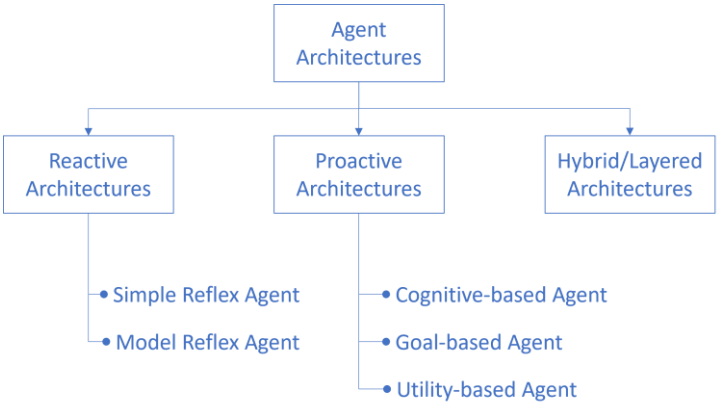


Figure 1 - Agent architectures.

In proactive architectures, the main idea is that the agent will have at least one goal t, and it will act proactively to achieve it (Bremner et al., 2019). This architecture can be further divided into the following architectures: cognitive architecture, goal-based architecture, and utility-based architecture. The cognitive-based agents are inspired and mainly developed in cognitive science and aim to model how the human brain works (Bordini et al., 2020). The goal-based architecture defines one or more goals to be achieved by the agent and those will guide the agent's actions (Belani et al., 2019). The utility-based architecture is also goal-oriented, but it also adds a preference mechanism that selects the preferred sequence of actions to be performed to achieve a certain goal (Haki et al., 2020).

Lastly, the layered or hybrid architectures have both reactive and proactive capabilities integrated into the agent (Barenji et al., 2019). The idea is that the agent becomes more flexible and adaptable to a variety of different situations, always choosing the best way to act. The downside of this approach is its complexity in implementing it. The layers normally are executed in parallel to each other.

Given the versatility of agent architectures, there are several applications where agents can be used (Xie & Liu, 2017). The most recent and trending applications of agents are chatbots and smart assistants (King, 2023). They are single-agent systems, that use natural language models to be able to chat with human beings and other agents using natural language. Examples of this are home assistants like Siri and Google Home (Qammar et al., 2023), chatbots for customer service in the websites (Vergaray et al., 2023), and chatbots for multitasking like ChatGPT (Choudhury & Shamszare, 2023). These systems can normally be extended to more complex environments like controlling smart houses (Jabeur et al., 2022), where the agent will control

the smart devices in a home and learn with the humans to predict their preferences and, in some applications, even reduce energy consumption and costs (Gherairi, 2023). Another area that MAS is highly researched is robotics, being the research more recently focusing on unmanned vehicles (Dinelli et al., 2023). These vehicles represent autonomous agents and can work in groups, and they can be used, for example, for search and rescue operations (Drew, 2021) and combat scenarios (C. Liu et al., 2021).

2.1.2 Agent's Decision Making

One important agent characteristic is decision-making (Esterle, 2022). Intelligent agents can make good decisions at the right time and their decisions can be as simple as turning on the lights or as complex as biding money in an auction at the energy market. The decisions should be always associated with a specific goal and depending on it the design of the reasoning engine can have different approaches and implementations. Good decisions are made when the necessary information about the problem and the environment is known, and the lack of it can lead to bad decisions and outcomes (DeAngelis & Diaz, 2019). This idea is reflected in MAS scenarios where the environment has a lot of entities trying to accomplish different, and sometimes opposite, objectives. More than that, if the agents find themselves in a competitive environment the agents will do their best to not reveal their knowledge and make the decision-making of other agents harder. These problems make the design of decision-making mechanisms challenging.

Decision-making is needed in both collaborative and competitive scenarios, where the agent will have to communicate with the others to get the most amount of information possible. Therefore, communication is a core part of MAS. If the agents are not capable of exchanging information or knowledge it becomes very difficult, or impossible for the agents to work together to accomplish their goals (Dorri et al., 2018). For two agents to be able to communicate, they must use the same language, such as the Knowledge Query and Manipulation Language (KQML) or the Foundation for Intelligent Physical Agents – Agent Communication Language (FIPA-ACL) (Berna-Koes et al., 2004). But only defining the language does not solve all the problems. A dictionary of concepts of the domain that will be discussed between the agents must, also, be defined. This is called ontology, and it becomes more important if the agents work with more than one domain at the same time (Ma et al., 2019). It is not easy to find an ontology that suits all cases, but it is possible to combine more than one ontology to increase the range of concepts to be used.

Most of the communication between agents will be them negotiating about common problems they need to solve, especially in competitive environments (Bulling, 2014). Agents can have different goals and they must act strategically to achieve them, and part of that strategy involves negotiating with other entities (Lewicki et al., 2020). The main goal in negotiation is to achieve an equilibrium point which maximizes the reward of every agent participating in the negotiation. There are several approaches to implementing the decision-making mechanism being one of those approaches is game theory (Sousa & Rocha, 2021). Game theory models the

problem like a game and all the rules are normally known by the agents involved. The agents will expect the other agents to be rational and that they will try to always choose the best option for them. It is assumed that the set of possible actions is known for every agent. One example of this kind of approach is the prisoner dilemma (Kastampolidou et al., 2020). One concept that originated from this type of model was the Nash equilibrium (Kreps, 1989). Nash equilibrium is the optimal state of a negotiation or a game when a given action made by a player is the best considering the actions made by the other players.

Apart from game theory, one of the most used models for general decision-making is the Belief Desire Intention (BDI) (Georgeff et al., 1999). In this model, the agent has a mental state, and this mental state is defined by these three concepts: beliefs, desires, and intentions. Beliefs are the information gathered by the agent through its sensors and it is known as the knowledge database. The beliefs can be based on outdated information that does not correspond to reality. The desires are all the states that the agent wants to be in. The existence of several desires does not mean that the agent will have to go to each one of those states, but these states will influence its decisions. The desires can also be seen as the goals that the agent wants to achieve. Intention is the agent's commitment towards a plan, defined by a sequence of actions, that aims to satisfy a specific desire. The plan is chosen from a library of plans that are predefined and each one of them is meant to achieve different goals (Gavigan & Esfandiari, 2021). One aspect of the BDI model is that the agent does not have to stick with its original beliefs, desires, and intentions, instead, it can update and manipulate them according to the changes in the environment, making it flexible and versatile. There are several variations of this model, for example, the emotional BDI which adds emotions (Puica & Florea, 2013) to the agent's mental state and includes a representation of the agents' resources and capabilities and the BOID, which stands for Beliefs-Desires-Intentions-Obligations and adds a normative approach to the agent in what regards to each social inclination (Broersen et al., 2001).

2.1.3 Agent's Learning and Intelligence Abilities

Learning is the active process of the agent using its observational capacities to gather information about the environment and use it to recreate the belief system that can better represent the environment (Asaad et al., 2021). For an agent to be considered intelligent it must have learning capabilities. But why does an agent need to learn? In complex domains, some problems cannot be solved by monolithic systems or even multi-agent planning, which requires a complete model of the environment (Verbraeken et al., 2021). The only solution is to design agents that can learn from their interactions and experiences with the environment they are in. The agent will only have access to a partial view of the environment and its knowledge is very limited. The agents should be able to learn how and when to act, how the other agents make their decisions and what are the other agents' goals, plans, and beliefs (Wooldridge, 2009). This creates a problem known as the moving target problem that happens when in an environment of intelligent agents, all the agents are learning and changing their performances over time, making it difficult to track and update the internal profiles of the other agents (Gronauer & Diepold, 2022). However, agents that learn and adapt to social environments and can adjust

the level of cooperation with other agents perform better than the ones that don't (Y. Gal et al., 2010).

The agent can learn by using deductive inference, inductive inference, probabilistic reasoning, and reinforcement learning (Michalski, 1993). Deductive learning is when the agent uses its current knowledge to infer how the environment works around it and explain the reason for some action-state sequence. In inductive inference, the agent will use its observations to try to explain the relation between the action-state sequences. The probabilistic reason approach uses statistical models and machine learning models to learn the relationship between action-state sequences and try to predict future events that might occur. In reinforcement learning the agent will learn using the try-and-error method.

Of all the different learning approaches an agent can use, the most researched recently is reinforcement learning (Wong et al., 2023). The subdomain is called multi-agent reinforcement learning (MARL), and it studies specifically the application of reinforcement learning techniques in MAS. This approach allows the agent to adapt to different environments, while in other types of learning (i.e., deductive, inductive, and probabilistic reasoning) the agents are designed for specific environments (B. Zhang et al., 2023). In reinforcement learning, the agent will take an action which will consequently make the environment change into a new state (Wong et al., 2023). The quality of that state transition is then evaluated by a reward signal. The objective of the agent is to maximize the rewards gained throughout the time by transitioning the environments to the right state. A simple diagram can be seen in Figure 2. One advantage of MARL over the other approaches is the ability to exchange learning experiences directly between agents, diminishing the learning time (Nguyen et al., 2020). Another advantage is the robustness that MARL can give to the system by being able to replace an agent that fails with another one with similar learning experiences. However, MARL faces two main challenges currently (Hernandez-Leal et al., 2019). One challenge is the difficulty regarding the definition of an appropriate formal goal and the other is the need to manually design quality features on which the agents must learn.

Intelligence in agents has been researched since the beginning of artificial intelligence itself (Wooldridge, 2009). More recently the logic-based approach has been abandoned because of the emergence of machine learning models. However, there is still little research being done (Calegari et al., 2021). Inductive learning, for example, has been recently applied in the ethics of autonomous intelligent agents, given that it helps clarify and explain the reasoning behind agents' actions, contrary to standard machine learning models, and allows more ethical decision-making by agents (Dyoub et al., 2020). Probabilistic reasoning has also been studied by Mekuria et al. (2019) and Nguyen & Rakib (2019). Mekuria et al. (2019) use probabilistic reasoning to enable a MAS to manage a smart home and deal with inaccurate data and unobserved variables. Nguyen & Rakib (2019) proposed a probabilistic-based reasoning MAS solution for resource-bounded agents in coalition scenarios. Regarding MARL there is a lot of research going on, especially in the robotics and energy systems front. Hu et al. (2023) proposed a MARL solution for the anti-conflict path planning problem of automated guided vehicles. D. Chen et al. (2023) apply MARL to autonomous vehicles so that they can learn how to perform

on-ramp merging on both merge lanes and through lanes in mixed-traffic environments, with human and agent drivers. May & Huang (2023) propose a MARL solution that helps the prosumers of an energy community to optimize the energy costs by learning the fluctuations of the energy market prices. Y. Wang et al. (2023) use MARL to allow cooperative energy management of an energy community with plug-in hybrid electric vehicles. The type of reasoning applied to the agents will depend on the complexity of the scenario and the task that the agent must resolve. These works, however, do not consider the privacy, and data security aspects of the scenarios studied, making the solutions unsuitable for privacy-constrained environments.

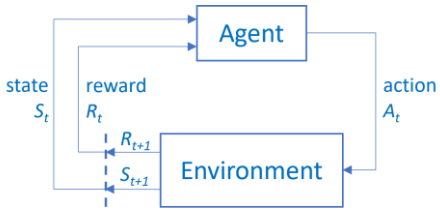


Figure 2 – Example of a reinforcement learning model. Adapted from Wong et al. (2023).

2.1.4 Organizational Structures

MAS can be shortly described as a group of agents. They have a lot of advantages as opposed to a group of disorganized agents that can lead to worse outcomes than expected (Dorri et al., 2018). When developing a MAS, it is necessary to develop a structure that is appropriate to the problem that is trying to be solved. This will help the agents to drive the flux of the information to the right places in an efficient manner.

There are two types of MAS engineering, agent-centred MAS, and organization-centred MAS (Ahmed Abbas, 2015). The agent-centred MAS can be seen as a bottom-up approach where the designer will focus its design primarily on the individual agents. The designer will design the agent interactions with the environment without concern with the global structure of the system. The idea is that given a good implementation of individual agents the MAS can somehow organize itself to achieve an optimal solution. This type of approach can lead to unpredictability and uncertainty because of the chaos that can be installed if no structure or coordination is designed (Ferber et al., 2004). In an organization-centred approach, the idea is that the designer will start from the whole MAS structure first by dividing the system into several parts and creating different roles with different objectives and responsibilities to enable the coordination of the agents. The role defines how an agent must interact with other roles and what is the set of actions available for that role. Each agent is not limited to only one role. The organization-centred MAS engineering can be seen as a top-down approach. Its adoption and the implementation of “organizations, societies, communities and groups of agents can reduce system complexity increase its efficiency and improve system scalability” (Ahmed Abbas, 2015). The organization-centred engineering is the approach of choice for designing complex and large-scale software systems.

Many organizational structures have been used and studied as shown in (Horling & Lesser, 2004). Each structure has its unique characteristics, strengths, and weaknesses (see Table 3). That is why an analysis must be made when choosing the best structure for a problem. There are problems where more than one structure satisfies the needs for the solution, and there are cases where neither of them alone can solve the problem at hand. In those cases, a hybrid structure must be developed. And there are more complex cases where structure adaptation must happen in real-time which is called self-organization. The most used structures are hierarchies, holarchies, coalitions, teams, congregations, societies, federations, markets, and matrix organizations according to (Horling & Lesser, 2004).

Table 3 - MAS organizational structures. Adapted from Horling & Lesser (2004)

Paradigm	Key Characteristics	Benefits	Drawbacks
Hierarchy	Decomposition	Can be used in many domains and is scalable	Can lead to bottlenecks in some cases
Holarchy	Decomposition with autonomy	Allows functional units to be autonomous	It has low performance, and it is necessary to organize the system in holons
Coalition	Dynamic, goal-directed	Exploit strength in numbers	The drawbacks from the long term do not compensate for the short-term gains
Team	Group level cohesion	Are centred in teams and can address large-grained problems	Communication overflow
Congregation	Long-lived, utility-directed	It is easy to discover agents	Sets may be overly restrictive
Society	Open System	It has well-defined conventions and public services	It is very complex, and agents may require additional social capabilities
Federation	Middle agents	Facilitates dynamic agent pool and is good in matchmaking, brokering and translation	The intermediaries can become a bottleneck
Market	Competition through pricing	It is good at allocation, has a centralized approach and increases the fairness	Adds the potential for malicious behaviour and a need to increase the agent's sophistication

2.1.5 Development Frameworks for Multi-Agent-Systems

Development frameworks are crucial to make the development process of solutions and prototypes more efficient and less time-consuming. A small survey was conducted to analyse which MAS frameworks were best suited for the proposed solution. Because MAS is a field being researched since almost the beginning of artificial intelligence itself, there are already great

amounts of frameworks to choose from as testified by these two reviews (Pal et al., 2020; Wrona et al., 2023).

For this survey, some search constraints were considered to help narrow the search results. One of the constraints was the type of framework searched. In MAS there are two types of frameworks, frameworks for Agent-based modelling and frameworks for deployment of agent-based systems. In this survey, only frameworks for the deployment of agent-based systems were considered. The frameworks must be open-source so that there is no need to spend resources on using closed-source frameworks, and the internal code must be known so that it can be adjusted and adapted to the author's needs. To simplify the search for frameworks and given that there are too many MAS frameworks to count only the most used frameworks nowadays and the easiest to find in the search engines were chosen for the survey. The programming language was not a restriction, however, the author had a preference for Python-based frameworks. In Python, there are thousands of libraries available in the domain of artificial intelligence and its coding simplicity is a good reason to be chosen. The frameworks surveyed are listed in Table 4 and further described in the following paragraphs.

Table 4 – Descriptive list of the surveyed multi-agent system development frameworks.

Frameworks	Version	Programming Language	License
JADE (Bellifemine et al., 2007)	4.5.3	Java	LGPL
OAA (Martin et al., 1999)	2.3.2	Java	LGPL
PEAK (Ribeiro, Pereira, Gomes, et al., 2023)	1.0.13	Python	GPL-3.0
SPADE (Palanca et al., 2020)	3.3.2	Python	MIT

Java Agent Development Framework (JADE) is an industrial-grade Java-based framework which aims to help the user build MAS effectively and efficiently by providing him with all the necessary tools (Bellifemine et al., 2007). It is an open-source framework and fully compliant with the FIPA standards to allow extensibility and compatibility. For the communication between agents an asynchronous peer-to-peer approach is used. The communication language used between agents is FIPA-ACL. JADE enables the creation of multiple containers in different hosts. This allows the full mobility of agents in real time from one container to another. JADE has also other features like agent abstraction, task execution and composition model and yellow page service that allows the publishing, subscribing, and discovering of services.

Open Agent Architecture (OAA) is a MAS framework that aims to help users develop heterogeneous agent communities in distributed and distant environments (Martin et al., 1999). It provides blackboard features for agents to share resources and knowledge in a centralized manner. OAA provides an original agent communication language called Inter Agent Communication Language (ICL). ICL is a symbolic declarative language that supports the exchange of natural language expressions and complicated tasks through high-level vocabulary. The ICL communication layer is like KQML as it provides a list of the different types of events and their related parameters. ICL's content layer is like the Knowledge Interchange Format (KIF) and specifies specific goals, triggers, and data items that can be integrated into the events. In addition to that, ICL permits direct human interaction with the distributed agents using several

different forms of communication like pointing, drawing, speaking, handwriting, and using a standard graphical user interface.

Python-based framework for heterogeneous agent communities (PEAK) is a Python-based MAS framework that was originated from the SPADE framework and developed by this dissertation author (Ribeiro, Pereira, Gomes, et al., 2023). PEAK was made to add more functionalities on top of SPADE. SPADE has a lot of fundamental functionalities, but it lacks some important ones. The key concept of PEAK is that it allows the creation of heterogenous MAS by allowing the coexistence and interaction between different MAS. To facilitate the development of MAS, PEAK enables the organizational-centred engineering approach for MAS because of the tools it provides to construct and structure MAS. Because it is based on SPADE it also runs on top of the XMPP. In addition to the core XMPP functionalities PEAK also added the Multi-User Chat, the Service Discovery, and the Publisher Subscriber. PEAK provides the tools necessary for agents to share a common knowledge database providing ways to implement MAS for different problems and scenarios. In addition to SPADE's asynchronous environment, PEAK also adds the possibility to run agents using multiprocessing mechanisms. PEAK also adds three more features namely the debugging tool, the simulation environment and the PEAK Dashboard that enables the monitorization of the PEAK ecosystem.

Smart Python Agent Development Environment (SPADE) is a Python-based framework for developing and deploying MAS (Palanca et al., 2020). It uses the Extensible Messaging and Presence Protocol (XMPP) as a communication protocol and architecture (Saint-Andre, 2011). XMPP follows a server-client architecture and is mainly used for instant messaging applications like social media apps. SPADE has an asynchronous structure which makes the execution of agents quick and efficient requiring less computational resources. Both XMPP and SPADE are flexible and extensible enabling the integration of additional features without impacting the rest of the system. Because of XMPP modularity, SPADE can use any XMPP server to deploy its agents making it very versatile. Some key features of SPADE are instant messaging, presence notification, and a contact list. FIPA specifications can be integrated with SPADE's communication protocol.

The surveyed frameworks have functionalities that are needed when designing and developing MAS and help to reduce the boilerplate. Some of the frameworks have modern solutions that can be integrated with nowadays systems. However, something noticed in the survey is that none of the frameworks has any data privacy or security measures or tools concerning the training of machine learning models collaboratively.

2.2 Federated Learning

Inside the domain of Distributed Learning, there are several kinds of solutions one can use to train a machine learning model in a distributed way. One of those ways is using federated learning (FL) which was first introduced by (McMahan et al., 2016). A good and straightforward definition of FL is from (Kairouz et al., 2021) which defines it as:

"Federated learning is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective."

The main idea behind FL is to enable the training of ML models without the clients sharing their private data with other entities. In standard ML settings, the users usually need to share their data with a central entity, i.e., a cloud-based service or an organizational entity, so that the central entity can aggregate, prepare, and clean the data so it can then be used to train a model. Instead of focusing the training of the model in a centralized entity and transmitting the data from the clients to the server, the reverse is done in FL. The model is shared from the central entity to the clients and the clients train the model with their local data, without ever sharing their private data (Jin et al., 2023).

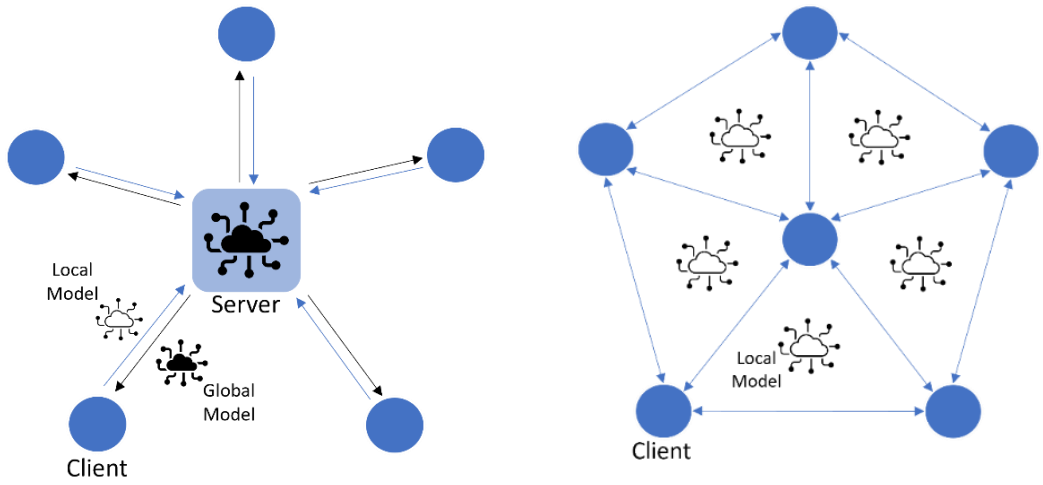
FL should be used in situations where several individuals want to solve a similar ML problem and each one of them does not have the necessary data to train it individually (T. Wang et al., 2023). For that reason, a federation can be created to train a model collaboratively. This also can make the model more robust since it is being trained and tested against a variety of data. Another advantage of this approach is the computational power and resources that end up being divided by more than one entity, increasing the speed of training, and alleviating the effort in each device.

2.2.1 Federated Learning Architectures

Federated Learning (FL) is intrinsically related to the physical architecture of the system (Kairouz et al., 2021). A regular FL system follows a server-client architecture, in a star topology, where the server provides a service, in this case FL, and the clients can participate and use that service, as shown in Figure 3.a. Another architecture approach is using a graph architecture, as shown in Figure 3.b, where the clients are connected to their neighbours. In the server-client architecture, the entities that participate in the system are the server and the clients, which both have different roles. In a FL system, a group of a server and its clients is called a federation. The federation must share the same ML problem so the model can converge to the ideal solution. The server is the orchestrator of the federation, and it is responsible for synchronizing the entire FL process (T. Li et al., 2020). The server has other responsibilities like aggregating the model updates and the model evaluations. The client has a local private dataset for training an ML model. The client will be responsible to train the model and evaluate it by measuring the performance of the model against its local dataset.

In FL, there are at least three phases in each round: (i) the client selection phase, (ii) the training phase, and(iii) the evaluation phase (Wen et al., 2023). The client selection phase is when the server must choose, from the available pool of clients, which clients will train and evaluate the model for the next round. The training phase is when the server shares the global model with the clients for them to train. After the training, the clients will send the updates of the model

to the server so it can aggregate them and update the global model. The evaluation phase is when the server shares the global model so that the clients can evaluate the precision of the model against their local datasets. After the evaluation, the clients send the results to the server so it can aggregate and have the precision of the model federation-wise. A federation can run for several rounds until the global model reaches a satisfying level of accuracy. This is a base architecture for a regular federated learning system, however not every FL system shares this kind of configuration, and there are a lot of variations and adaptations (Kairouz et al., 2021).



(a) Centralized architecture (b) Decentralized architecture

Figure 3 - Centralized and decentralized federated learning architectures.

FL can be further categorized depending on the type of clients the federations have and what kind of data the clients have (Kairouz et al., 2021). Cross-sile FL is a system where the clients are represented by corporations or big dimensional entities and normally the size of the federations is smaller and the computational power each client has is much higher creating the possibility of training larger models (Huang et al., 2022). Cross-device FL is when the federation is composed of edge devices instead of organizations (B. Liu et al., 2023). The edge devices do not have too much computational power, limiting the size of models that can be trained. However, the size of federations is normally much higher than cross-sile FL reaching numbers between 2,000 to 10,000, and sometimes even higher. The communication channels are less resilient in these systems, making clients lose connection during the execution of the federation. In regards to the type of data used in each client, FL can be categorized by being vertical or horizontal. Vertical FL is when the clients have different sets of features that want to train but correspond to the same target data (Wen et al., 2023). For example, two companies of different domains want to train a model that will be used on the same clients. In horizontal FL the clients have different examples, but the set of features is the same (Jin et al., 2023). For example, two companies working on the same domain but are localized in different places geographically, targeting different clients.

FL has received a lot of attention and several applications of it have been found. In the energy systems domain, Lin et al. (Lin et al., 2022) propose an FL system to identify household characteristics while protecting sensitive data from households. Lee and Choi in their work (Lee & Choi, 2022) use an FL system to manage local home energy systems. In the domain of biomedical, Jeon (Jeon et al., 2019) proposes an FL system to protect the patient's data from leakage while training a deep learning model. Regarding new algorithmic approaches for general applications, Liu Z. et al. (Z. Liu et al., 2023) proposed a new secure encryption algorithm called doubly homomorphic secure aggregation to increase the security levels of cross-silo horizontal FL environments. Qiu X. et al. (Qiu et al., 2023) work, on the other hand, proposed also a new secure aggregation algorithm but, in this case, specifically for cross-silo vertical FL systems. In Gao et al. (D. Gao et al., 2023) work a novel open-source cross-device FL platform is proposed to help users deploy real FL systems with heterogeneous devices reducing the boilerplates and some of the implementation burdens found in this kind of system. Another example of cross-device FL systems is the Malan et al. (Malan et al., 2023) work that aims to reduce the communication burden between devices while maintaining high-quality learning capabilities by embedding a mechanism that reduces the size of the model gradually and iteratively used in the training and synchronization phases. These works present new approaches to FL security and even new platforms for implementing FL systems in real scenarios however the authors assume the scenarios are static and do not change over time.

Apart from the regular FL, also known as centralized federated learning, there is also a decentralized FL architecture (Müller et al., 2021). Decentralized FL, also known as peer-to-peer and serverless FL, changes the server-client architecture to a node graph where the clients are connected directly, removing the need for a central entity from the system (Figure 3). The responsibility of aggregating and synchronizing the federation is passed to the clients. This solves some issues regarding the centralized FL approach. One is that centralized FL has a single point of failure, the server. If the server goes down for some reason the whole system becomes inoperable. Another issue is that centralized FL is less scalable than the decentralized one (Witt et al., 2022). When there are a lot of clients the amount of communication traffic rises greatly. Whether in decentralized FL the communication in each client is made to client neighbours, dispersing the amount of traffic from the central server to the whole network. However, despite being better in some ways than centralized FL there are still some challenges to overcome like scalability issues when the number of participants increases, trustworthiness between the federation members and the lack of modular, efficient, and scalable frameworks (Beltrán et al., 2022).

Decentralized FL (DFL) is still a very recent topic and much less explored than centralized FL. There are a lot of different approaches to decentralized FL, the blockchain-based approaches are the most used (Witt et al., 2022). Chen et al. (Chen et al., 2021) propose a decentralized validation mechanism and a proof-of-stake consensus mechanism where honesty by the clients is rewarded in blockchain-based FL systems. Desai et al. (Desai et al., 2020) propose a hybrid blockchain-based FL framework that discourages backdoor attacks by detecting and punishing attackers. And in J. Li et al. work (J. Li et al., 2021), a novel aggregation and synchronization algorithm is described as blockchain-assisted DFL. Outside blockchain, there are also other

works. One of the first works to be entitled DFL is Mäenpää's work (Mäenpää, 2021) where he proposes a peer-to-peer FL system with an adaptation of the Federated Averaging aggregation algorithm called Federated Averaging for Peer-to-Peer. The way it synchronizes the clients' models is by using their neighbours' updates. Another work by Lalitha et al. (Lalitha et al., 2019) proposes a graph-like network structure and uses a Bayesian approach for the model updates. Chou et al. in their work (Chou et al., 2021) propose a cross-device DFL system that works by pairing devices with each other to update the local models. Another example is in Behera et al. work (Behera et al., 2021) where at each round a leader is selected to be responsible for the aggregation of the model. There are a variety of approaches in DFL that the research community is addressing. However, there is a lack of adaptation and dynamism between federation members that allow the active participation and flux of members during the execution of the system. In addition to that, the lack of efficient frameworks that enable the deployment of this kind of system is also an issue.

2.2.2 Federated Learning Known Challenges

FL can solve different problems and be applied to a variety of different domains and scenarios. This makes FL a great candidate solution for several distributed learning scenarios, but, as with any good technology, it has its challenges (Wen et al., 2023). Given that FL is an ML setting, several ML challenges are inherited from it. For example, the data quality regarding the data heterogeneity and distribution can affect the model's precision and reliability, and the difficulty that comes from trying to train precise and efficient models (Priestley et al., 2023). Even though FL's focus is to prevent clients from sharing their data while still being able to train and use the ML models, this does not make it flawless. FL assumes that every participant, including the server, is trustworthy, which may not be entirely true, especially when the participants are competitors in the same business (i.e., as in horizontal cross-silo environments) (Kairouz et al., 2021). FL on its own is still open to attacks and failures and it requires extra steps to make the training of ML models more secure to the clients. In addition to that, FL federations are known to be unreliable, especially in cross-device scenarios, making the internet connections between clients a challenge as well (Huang et al., 2022). A big effort is being put towards FL deployed in real scenarios, so more of the research can be tested against realistic scenarios, embodying the several limitations that exist and are not normally simulated in research case studies. Figure 4 shows an overview of the known challenges in FL systems.

Despite FL being made specifically with privacy and security in mind, its raw implementation cannot protect the data entirely by just using the system architecture. Several attacks and threats can be made to these systems (P. Liu et al., 2022a). When in an FL system the clients and server share their model's update, it is possible, to use reverse engineering to recreate the data used to train that specific model (C. Zhao et al., 2021). So, a malicious client could infer the client's data by doing that. There are different approaches to this problem. One of them is the use of cryptographic mechanisms and algorithms that have difficult access to the original clients' model updates (Sah & Singh, 2022). The downside is that increases computation effort and time and is normally dependent on the type of model to be used. Another solution is split

learning, which allows the deep neural networks to be split into at least two parts so that the server and the clients have different layers of the model (Abedi & Khan, 2020). This approach has still limitations regarding the type of models to use, as for now it only allows deep neural networks and the unbalanced data in clients can significantly reduce the accuracy level of the model (Joshi et al., 2022). Normally privacy and security solutions are highly dependent on the type of problem and system that is being designed, however, a lot of effort is being put towards this in the research community (T. Nguyen & Thai, 2022).

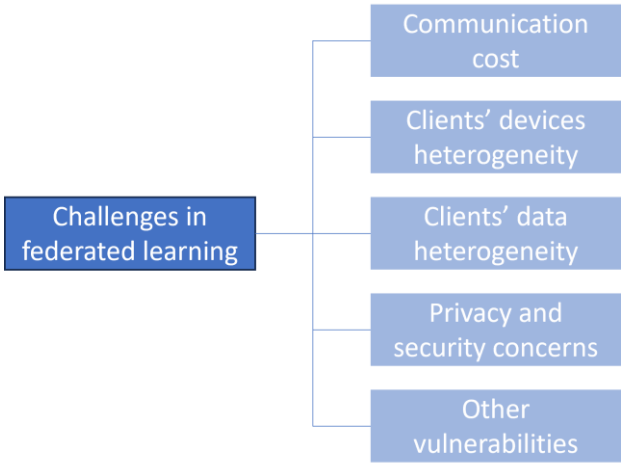


Figure 4 - Challenges in federated learning. Adapted from K. Zhang et al. (2022).

Another thing to have in mind when developing an FL system is its heterogeneity (Q. Yang et al., 2020). In FL we can have heterogeneous clients and heterogeneous data. Client heterogeneity is related to the characteristics of the computational devices that are operating on the client side (Cho et al., 2020). Especially in cross-device FL, it is normal to have a wide range of devices, like edge devices, the Internet of Things (IoT), home computers, and cloud services, all of which differ a lot. The characteristics of the clients' devices can vary from computational power, memory sizes, communication bandwidths, connection stability, etc. One way to approach this heterogeneity is by using more refined client selection algorithms (S. Q. Zhang et al., 2022) and FL solutions that reduce the communication effort in the model aggregation phase (Gogineni et al., 2022). On the other hand, when we are referring to heterogeneous data we are talking about, specifically, non-independent and identically distributed data (non-IID) (Zhu et al., 2021).

Independent and identically distributed data (IID) is easy to be characterized, however, non-IID can be characterized in different ways (Kairouz et al., 2021). Like in traditional ML settings, data quality is very important. Even in centralized data, there are a lot of challenges which increase in complexity when the data is decentralized and kept private (Jain et al., 2020). The quality of the clients' datasets is not known and for that reason, it can become challenging to mitigate these problems. When the data is generated in different devices through different users, it is natural that the data will be unbalanced and non-identical. Some problems related to non-IID are (Zhu et al., 2021): covariate shift, which happens when the statistical distribution of the data

is different in each client; prior probability shift, which happens when the statistical distribution of a label is different in each client; concept drift, that happens when clients share the same labels but use different features to identify does labels; concept shift, that happens when clients use the same features to identify different labels; and, unbalanced data is when the data quantity and speed of growth are very different in each client. To solve some problems related to non-IID, Gao et al. (L. Gao et al., 2022) propose a solution to reduce the impact of non-IID data as a FL model by using local drift decoupling and correction. Tian et al. work (Tian et al., 2022) proposed a stateless FL algorithm that allows faster model conversion while not restricting the number of clients participating.

Some of these challenges, like non-IID, communication efficiency, and client heterogeneity**,** are also issues that DFL faces. In addition to those challenges, the network topology can be a great influence in deciding which ML model to use (Kavalionak et al., 2021). He et al. (He et al., 2018) proposed a new decentralised training algorithm that is resilient to changes in the network topology and client participation, however, there is a lack of solutions that can be applied to more variety of scenarios (Kairouz et al., 2021). The communication patterns and efficiency can also be a reason for the difficult implementation of DFL in real scenarios (Shahid et al., 2021). One tentative of trying to solve this problem is in Lui et al. (W. Liu et al., 2021) work, which proposes a solution that balances the communication efficiency with computation costs in DFL. The frequency of the local updates of the clients in DFL is also another constraint to have in mind, as the excessive updates can overload the network and make it less efficient and not operable at all. Liao et al. (Liao et al., 2022) tried to solve this issue by creating an optimization algorithm that calculates the frequency needed for model updates based on the network topology.

2.2.3 Multi-Agent-Based Federated Learning Systems

FL can be viewed as a solution that aims to solve distributed learning problems where the participants try to collaboratively solve a common ML problem by training a model in a distributed way (Wen et al., 2023). One field that has been researched for a long time and that tries to achieve identical goals is the Distributed Artificial Intelligence field, more specifically multi-agent systems (MAS) (Fourez et al., 2022). In MAS the goal is to solve a space and timely distributed problem with several autonomous and independent entities that have limited knowledge and can cooperate to solve the problem. Though is not entirely the same thing, both techniques can work together to achieve better results. Several things that MAS can provide to FL are the adaptability, flexibility, resilience, dynamism, efficiency, and scalability that are inherent in MAS (S. Q. Zhang et al., 2022). MAS can also benefit from FL as the agents could implement distributed learning with the advantage of not needing to exchange data directly with each other. In centralized FL, MAS could even help with the single point of failure problem, by replacing and adapting the agents to the losses of the system (Javadpour et al., 2023). In collaborative environments in MAS, the agents could use decentralized FL to learn from their neighbours, or even learn from more experienced agents and achieve results more efficiently and effectively (Oroojlooy & Hajinezhad, 2023).

Although the benefits that can arise from the combination of these two technologies have not yet been deeply researched, some works are starting to guide the research towards that direction. Rincon et al. (Rincon et al., 2022) used MAS to give flexibility and dynamism to the FL system, by allowing agents to join the federation dynamically, during the system runtime. Znaidi et al. (Znaidi et al., 2022) analysed the usage of MAS in FL in terms of prediction-privacy trade-off. Wei et al. (Wei et al., 2019) integrated MAS with FL to help human participants visualize and understand better how the FL system works. X. Li et al (X. Li et al., 2022) use an FL system with multi-agent reinforcement learning to better allocate resources in vehicle-to-vehicle communications. Zhang S. et al. (S. Q. Zhang et al., 2022) use agents to dynamically optimize communication efficiency in an FL environment with the help of a client selection algorithm based on reinforcement learning. Gholami A. et al. (Gholami et al., 2022) propose a confidence metric to enable agents to train using a decentralized FL while preventing poisoning attacks. Han C. et al. (Han & Yang, 2021) used MAS to solve a communication efficiency problem in a fleet of ships to help maximize the results and minimize the resources while using FL to train an ML model. These papers explore different aspects of the MAS and FL together, however, other aspects should be explored like the conceptualization of a framework for developing FL systems based on agents that can standardize the integration of such systems, the different ways MAS can solve the single point of failure in centralized FL, and explore the possibility of a platform for FL that can be managed by MAS. In addition to this, the impacts of dynamically having agents joining and leaving federations have not yet been deeply studied.

Another domain that aims to solve a slightly similar problem is distributed learning in MAS (Barrientos & Luevano, 2022). In this domain, the privacy and security of data are not the primary focus. For example, B. Wang et al. (2022) in their work propose a multi-agent reinforcement learning system that leverages the computation power needed to train the agents in a distributed approach. In J. Tan et al.'s (2022) work it is proposed a distributed multi-agent solution for making decentralized offloading decisions. In these examples, the security and privacy of data are not considered and therefore limit their application to more practical scenarios where data privacy should be protected.

2.2.4 Development Frameworks for Federated Learning

Before the implementation of the proposed solution, a survey of FL development frameworks was done to help decide which framework would be used. There are a lot of FL frameworks out there but none of them, at the moment, helps the user to develop dynamic collaborative and distributed learning environments with FL and MAS. A framework was chosen so that the integration of FL functionalities in the MAS framework was facilitated and there would be no need to reinvent the wheel and integrate everything from the bottom up. One requirement for the programming language was Python, so it could be the same language as the MAS framework. In addition to that, the framework needed to be open source since there was a need to adapt and use its functionalities individually. Other characteristics were considered as well, for instance, the present version of each framework, the documentation quality and the

number of issues opened and closed. The frameworks considered are listed in Table 5 and their description are further detailed in the following paragraphs.

Table 5 - List of the surveyed federated learning development frameworks.

Frameworks	Version	Documentation	Issues Open/Closed	License
FATE (*FATE*, 2019)	V1.11.3	Completed	770/1017	Apache-2.0
Flower (Beutel et al., 2020)	V1.5.0	Complete	241/204	Apache-2.0
OpenFL (Foley et al., 2022)	V1.5.0	Incomplete	98/131	Apache-2.0
PySyft (*PySyft*, 2017)	V0.1.7	Incomplete	122/3303	Apache-2.0
Substra (*Substra*, 2022)	V0.41.0	Completed	3/92	Apache-2.0
TensorFlow Federated (The TensorFlow Federated Authors, 2018)	v0.62.0	Completed	77/310	Apache-2.0

FATE stands for Federated AI Technology Enabler and is an industrial-grade framework made by WeBank Co., Ltd, that joined the Linux Foundation for AI and Data as an open-source project (*FATE*, 2019). This framework can handle linear regression models, tree-based models, deep learning models, and transfer learning models as the global model of the FL system. To integrate the models, one can only use TensorFlow or PyTorch. It has integrated methods for feature selection on the client side, and it has security protocols like homomorphic encryption and multi-party computation. Additionally, FATE comes with different components that can be used aside from the core component, like the KubeFATE, which allows the use of Kubernetes, and FATE-Board, which is a dashboard for the clients to manage the FL system. FATE can be deployed using Docker containers to test and debug FL solutions.

Flower is a community-driven framework that was developed by researchers for researchers (Beutel et al., 2020). The framework was built with flexibility, decoupling, and adaptability in mind so that researchers can use, test, substitute, and adapt existing components of the framework with new ideas and techniques. It is possible to integrate ML workloads in the FL system. It is possible to use any ML algorithm with this framework, and it enables the user to configure the FL systems with high detail. Flower also has a large list of algorithms for aggregation and client selection already available in the framework.

OpenFL is a framework made by Intel and it only works with deep learning algorithms from the TensorFlow and PyTorch ML frameworks (Foley et al., 2022). There are two types of workflows available: the director-based workflow and the aggregator-based workflow. The director-based workflow allows several experiments to be executed in sequence. This is more dedicated to research projects where there is a need to test a model using different configurations in sequence. The aggregator-based workflow defines an experiment, and the distribution must be done manually. All the clients have access to the model and the flux of the FL system. In this

framework, the model must be copied manually to each device that will participate in the FL system.

PySyft is a framework made by OpenMined (*PySyft*, 2017). There are, at least, two main roles in PySyft, namely data scientist and data owner. Data scientists can question remote data owners, so they can obtain knowledge from that remote dataset, according to the permissions given by the data owner. This is done without copying the data or transferring the data from its source. Data owners can configure their local FL systems and restrict the type of knowledge that can be gathered from their data. PySyft has all the functions and methods to create the models and send them to the remote machines, so they can be trained. Additionally, there is an external component called PyGrid which is dedicated to managing the data and the communication between the nodes in an interactive manner.

Substra is another FL framework that joined the Linux Foundation for AI and Data as an open-source project (*Substra*, 2022). It was created by Owkin, and it was developed especially for hospitals and biotech companies. The way Substra integrates the global model in the FL system is a bit different from the other frameworks. The global model is created in an external script that is then used by the FL system. This allows the customization of the global model while keeping it decoupled from the overall system. In Substra, the federations can train composite models which are composed of two different models, the trunk model, and the head model. The trunk model is the public global model that is shared among the clients and is always updated on the server. The head model is the private local model that each client has and cannot be shared. With Substra is also possible to develop decentralized FL systems. The framework is divided into 3 different components: the web app, the substra, and the substrafl. The web app monitors and manages the local FL system and is accessed by the clients. The substra API is the core of the framework and it has all the essentials to create datasets, algorithms, and the tasks necessary to run and train the ML models. The substrafl is a high-level API that has the tools for high-scale experiments, with access to Docker and Kubernetes.

TensorFlow Federated was developed by Google, and it was the first FL framework being created (The TensorFlow Federated Authors, 2018). The framework is open source, and it works with common ML algorithms, like deep neural networks (DNN), and algorithms without learning characteristics, like mathematic and statistic algorithms. TensorFlow Federated is limited to algorithms available in the TensorFlow and Kera's frameworks. Because TensorFlow allows the creation of personalized algorithms, it is possible to build personalized algorithms in TensorFlow Federated to some extent. This framework is divided into two different components: the Federated Learning API and the Federated Core API. The Federated Learning API is a high-level interface that allows the integration of TensorFlow ML algorithms with the FL system. The Federated Core API is the low-level interface that allows the integration of more personalized models and techniques inside the FL system. TensorFlow Federated has also a simulation mechanism to test FL systems in a single computer, called single-machine simulation.

There are more development frameworks that were not reviewed for this survey. Despite being the most popular, these frameworks do not provide the essential tools to develop autonomous

and intelligent clients and servers nor permit an easy dynamic configuration of the roles each entity has. With regards to the monitorization of the federation's performances during execution time, only some of them provide a graphical user interface, however, they do not allow the flexibility and personalization of those interfaces. Also, only Substra, from these frameworks, has the tools to develop centralized and decentralized FL, but those have the tools to create a platform or ecosystem that enables the existence of both systems, where clients and servers can participate and provide and use those services. Despite the frameworks not having these features, some of them can be used as a starting point to implement them.

2.3 Supervised Machine Learning

Machine Learning (ML) is an area of AI that studies new ways of solving problems automatically, without programming explicitly the solution (Jung, 2022). The solution normally called interchangeably model or algorithm, is the mathematical formula that best maps the input variables to the output variables. In other words, it models the relationship between the input data and the output data (Pandey & Rautaray, 2021). Traditionally, the model was designed manually with the help of mathematical and statistical techniques and now, the ML algorithms do that automatically. ML algorithms are never perfectly accurate, but the greater number of examples are used to train the algorithm the better the model will be (Khanal et al., 2020).

ML can be categorized into several different subareas, namely, supervised learning (SL), unsupervised learning, semi-supervised learning, reinforcement learning, and so on. For the scope of this dissertation, it was chosen the most used SL model in federated learning, artificial neural networks, and another SL model that is yet to be seen explored in federated learning contexts, genetic programming. What distinguishes SL from the other categories is the type of data used to train the algorithms (El Mrabet et al., 2021). The data must be labelled which means for each example in the dataset, the expected result is known. This allows the algorithm to know if the outputs it is generating are good approximations or not to the expected result. In the training phase of an SL algorithm, the dataset is divided at least in two, one for training the algorithm and the other for testing it. The testing set allows the algorithm to compare its performance against the expected results and measure how well it models the problem (Sen et al., 2020). For these algorithms to be accurate they need to be trained in very good-quality datasets, which in many cases are not easily available (Vemuri, 2020). In SL, there are two kinds of algorithms, regression algorithms, and classification algorithms (El Mrabet et al., 2021). In regression, the output values are numeric and in classification are descriptive and categorical. Some examples of SL algorithms are artificial neural networks and genetic programming.

2.3.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are one of the most used, explored, and trending algorithms in the last decade (Shao et al., 2022). Its popularity is given by its versatility, ability to be applied in diverse problem types (i.e., classification, regression, and cluster problems), and the high

precision that it can reach, especially when they are trained on large datasets (Abiodun et al., 2018). ANNs try to emulate the human brain anatomy by mimicking how biological neurons signal each other. These neural networks are composed of several neurons interconnected to each other and they are organized in layers (Abiodun et al., 2019). There should be, at least, one input layer, one hidden layer and one output layer. When the ANN receives input data the input neurons will use it to calculate their outputs based on the bias and the weight the neuron has, and, if activated, it passes the output to the next layer of neurons. The neurons are only activated when the output of the neuron reaches a determined threshold. If the threshold is not reached the neuron will not output any value. To train them in a supervised learning setting, the ANN is fed with labelled data. Some calculations are made to compare the predicted result to the expected result, and some adjustments are made to the neurons' weights to approximate its predictions with the expected outcomes.

ANN can have a variety of different architectures depending on the task to be solved (Abiodun et al., 2018). The ANNs can have several layers of neurons, structured in different ways. If it has more than three hidden layers, the ANN is considered a deep neural network (DNN) (O'Mahony et al., 2020). Depending on the connections and the flux of the data between the neurons there can be several approaches as well, being the most known the fully connected feedforward neural networks. These networks have the neurons of each layer connected to all neurons of the next layer, and the data flux goes only in one direction, from the input layer to the output layer. There are other architectures widely used like the convolutional neural network (CNN), recurrent neural networks (RNN) and the generative adversarial networks (GAN). CNNs are feedforward networks that use linear algebra to capture data patterns and are mostly used in image recognition, pattern recognition and computer vision (Z. Li et al., 2022). RNNs are networks that have feedback loops which inject the output of neurons on neurons of the same hidden layer, helping maintain the order of the data, and are mostly used for natural language processing and speech recognition (Yu et al., 2019). GAN is a neural network with two main components, the generator, which creates fake as close as possible to real data, and the discriminator, which will try to identify if the data from the generator is fake or not (Aggarwal et al., 2021). GANs are very useful for generative tasks like text-to-image and image-to-image.

The ANN's versatility allows it to be used in a variety of different tasks. In the medical domain, ANNs are used to help doctors make more efficient and confident analyses of health problems, for example, to detect early evidence of breast cancer by using an ANN to analyse the microcalcification in mammograms (Rehman et al., 2021). ANNs are also used to help doctors to increase their efficiency in analysing reports. For example, Olthof et al. (2021) used an NLP model to classify orthopaedic traumas in radiology reports. Another utility in the medical field is in writing automated reports as McMaster et al. (2023) work shows. Other fields impacted by neural networks are Industry 4.0 and transportation. In Industry 4.0 ANNs are being used for making the use of autonomous robots in human environments more secure (Mahajan et al., 2023). In transportation, it is being used widely for autonomous vehicles but also for safety reasons (Safarov et al., 2023). This is only a rather small demonstration of the applicability of ANNs, however, that does not mean that ANNs are always the right choice for any task. They

are considered black-box models, and in some scenarios, there is a need to understand the reasoning behind some decisions provided by these models.

2.3.2 Genetic Programming

Genetic Programming (GP) is an ML algorithm based on the theory of natural selection used to evolve programs automatically (Ahvanooey et al., 2019). The algorithm creates an initial population of individuals, where each individual corresponds to a solution. The population then evolves through several generations and goes through a process of mutations and crossovers to generate better individuals. The process stops when some constraint is reached, like an individual reaching a predefined performance. GP models are used for ML problems and the structure of the individuals represents the program that will map the dataset to the output (Agapitos et al., 2019). There can be several ways to represent that program in the individual, like the tree-based, stack-based, and grammar-based structures, the most used the tree-based structure (Ahvanooey et al., 2019). The GP algorithm must have a function set, the list of functions to use in the programs, and the terminal set, the list of variables and constants to be used in the programs. It is the combination of these functions and terminals that originate the programs. In a tree-like structure, the functions represent the internal nodes, and the terminals represent the leaves of the tree, as shown in Figure 5. The programs are not required to use all the functions and terminals present in the sets, being able to omit some of them. The GP can be combined with other ML algorithms to help maximise the performance of the solutions.

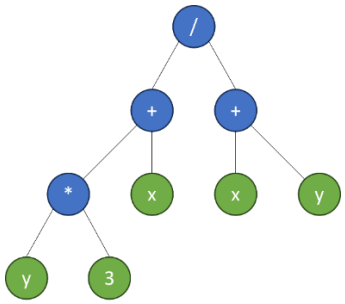


Figure 5 - Example of a tree-based genetic programming individual: $(3y+x)/(x+y)$.

Another trending area of research among the GP community has been computer vision given the ability of GP to allow better interpretability of the solutions, the reduced necessity of highly expert knowledge and the reduced number of parameters and training instances needed to have good solutions (Bi et al., 2021a). The way GP is adapted to computer vision is by using a function set that has computer vision functions (i.e., image filtering, feature extraction and pooling functions) and is normally combined with simple statistical linear models (Fan et al., 2022). Palconit et al. (2021) work proposes the use of a GP variant called multigene GP to better predict the tracking and tagging of fish. Concepcion II et al. (2021) combine hybrid decision trees and GP to detect chemical components in the lettuce leaf canopy, showing high accuracy. Price et al. (2019) propose a novel GP approach for feature extraction in image classification tasks by improving the population diversity, redundancy removal and reduction of the bloating effect of

the solutions generated. Cortacero et al. (2023) propose Kartezio, a cartesian genetic programming-based solution for image analysis in the biomedical field. This solution reaches state-of-the-art precision while reducing the number of training instances needed as well as increasing the interpretability of the solutions generated. Lavinas et al. (2023) extended Kartezio for computer vision tasks by integrating a Lexicase Selection technique to control the size of the solutions and therefore maintain better the interpretability of the solutions generated.

2.3.3 Distributed Learning Models

In machine learning, as the complexity of problems being solved increases it also increases the need for models to be bigger and more robust to handle bigger sets of information (Verbraeken et al., 2021). Not only that but the runtime used to train models also increases with the amount of data being used and the size of the model, especially in DNN and GP algorithms. For this reason, researchers and system designers see themselves obligated to recur to distributed systems to allow the distribution of computation power (parallelization) through several machines and increase the total amount of bandwidth for data transfer. There are other cases where centralizing the data is not ideal since the data can be inherently distributed and too expensive to centralize it, or even too big for a single machine to store it. There are two ways to apply a distributed learning system, one is using data parallelism, and the other is model parallelism (Patil & Chickerur, 2023). The data parallelism is when the data is distributed among several working nodes (it can be machines or single processors) and the complete model is shared with every node so they can train on the data they have. Model parallelism is dividing the model between the worker nodes, and they train their part of the model using the complete dataset, to which they all have access. In this context, FL can be applied as either a data parallelism or model parallelism approach depending on how is implemented (J. Liu et al., 2022).

Despite the distributed learning being intrinsically related to the physical system, the models, either DNN or GP, must be adapted individually to be able to be trained and evolved, respectively using distributed and parallel approaches. In DL most of the research in distributed learning is focused specifically on using several graphical processing units (GPU) and/or cloud-based solutions. Mai et al. (2020) proposed a new framework for enabling adaptive training in distributed training of ML models by allowing the definition of high-level adaptation policies. Soomro et al. (2023) propose a novel approach to quickly optimize scheduling tasks and workload distribution of CNN models among several worker nodes. Nuriyev et al. (2024) also propose a new universal scalable algorithm to accelerate parallel deep learning applications based on an all-reduce algorithm. Other authors developed solutions for distributed learning in systems with edge devices. Srirama & Vemuri (2023) propose an actor model-based distributed learning approach for fog computing that aims to train ANNs in IoT devices. Wu et al. (2020) propose a distributed deep learning-driven task offloading algorithm for making near-optimal decisions about the distribution of tasks between mobile devices, edge cloud servers and

central cloud servers by considering heterogeneous devices in the network and reducing the computational complexity.

Regarding GP, the research aims to increase the speed of the evolution of the solutions and make solutions with more generalization capabilities by evolving different populations based on different fitness landscapes. The parallelization can be done at the individual level, population level and fitness evaluation level. At the individual level, to speed up the evolution process researchers have used GPUs (B. M. P. Silva et al., 2021). Beata et al. (2021), for example, proposed a novel engine that allows the use of TensorFlow to run GP algorithms in GPUs. At population level, the strategies try to attribute different populations to different computational resources. This can be seen in Russo's work (2020) where a parallel distributed Evolutionary approach to optimize parameters on evolutionary computation algorithms distributed through several machines is proposed. Sachindra & Kanae (2019) use a different strategy in this regard. They propose a new strategy where several demes are created and evolved parallelly each with different individual characteristics to maximize the individual variety. A master deme is then created to evolve its population based on the best 25% of all the other demes increasing the generalization capabilities of the individuals. At the fitness evaluation level, Vega et al. (2020) propose a solution to evaluate and control better the bloat problem in GP algorithms by considering not the sizes of the individuals but the time they take to be evaluated. This allows the algorithm to categorize the individuals by different categories and posteriorly run those groups in parallel and distributed systems. In exception to FL, the research in distributed learning, either in deep learning or in GP, does not normally focus on the privacy and security issues of data.

3 Methods, Materials and Technological Challenges

This chapter will describe the methods and materials used in the development of the solution and the reasons for choosing them. After that there is a brief introduction to the ethical issues in artificial intelligence and what approach was made regarding this matter. And lastly, an overview of data privacy and security is given in the context of this dissertation.

3.1 Methods and Materials

For this dissertation, some development frameworks and libraries were chosen to be part of the implementation process of the proposed solution. The following sections describe the multi-agent system development framework chosen, the federated learning development framework chosen, and the machine learning libraries and models used. In addition to that, the datasets that were used in the case studies are also described in detail.

3.1.1 Multi-Agent Systems Development Framework

As the core of the proposed solution, a MAS framework had to be chosen. A good flexible framework should be chosen given that it would have to enable the integration of a FL framework on top of it. A survey of MAS was made in Section 2.1.5 to analyse the most important MAS frameworks that currently exist. After a thorough analysis, the PEAK framework was chosen.

Python-based framework for heterogeneous agent communities (PEAK) is a MAS framework that helps the user build and deploy multi-agent ecosystems (Ribeiro, Pereira, Gomes, et al., 2023). The ecosystem is a prosperous environment for the MASs to share resources, socialize, and live together. PEAK allows the development of a wide range of MAS architectures. The MAS

can be implemented in real-world scenarios, integrating IoT, for example, or used in simulated environments (e.g., smart grid simulation). Because there are a lot of frameworks for MAS development, PEAK was not built from the ground up. Instead, it uses SPADE as its core and adds extra functionalities. Beyond the additional functionalities, PEAK has an external Graphical User Interface (GUI), the Dashboard, for interactively monitoring the PEAK ecosystem. Figure 6 shows the internal structure of PEAK.

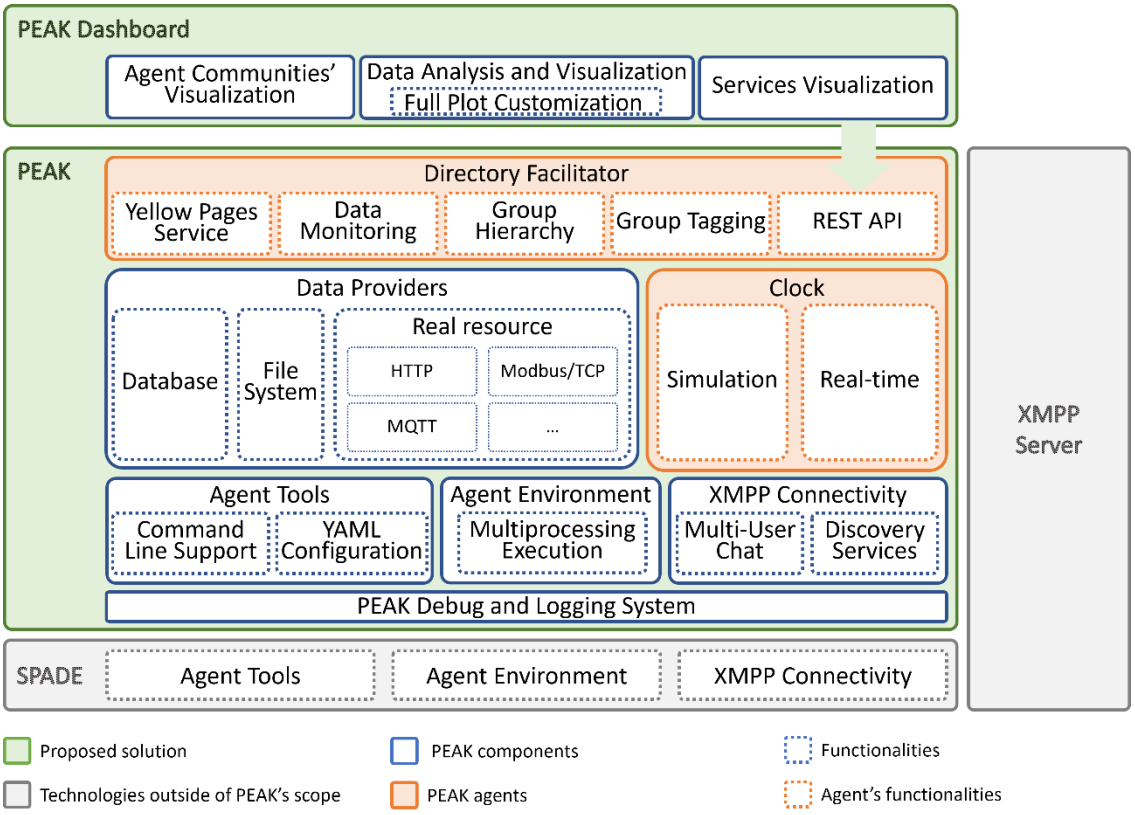


Figure 6 - PEAK internal architecture.

The main reason that led to the decision to choose the PEAK was mostly the familiarity that the author had with the framework. This helped with the integration of the FL framework making it more efficient given that all the internal structure of PEAK and every detail was known. By being a Python framework, it allows for a wider range of AI libraries to be integrated with it. Another advantage that PEAK has, besides its flexibility and expandability, is the ability to monitor and analyse in real-time the MAS Ecosystem and the generated data by the agents in a dashboard-like application.

3.1.2 Federated Learning Development Framework

The proposed solution of the current dissertation had to use an FL framework to facilitate the integration of FL functionalities in the MAS framework. A survey of FL frameworks was made in Section 2.2.4 to analyse all the current FL frameworks that exist. After a thorough analysis, the Flower framework was chosen.

Flower is a community-driven framework that was developed by researchers for researchers (Beutel et al., 2020). The framework was built with flexibility, decoupling, and adaptability in mind so that researchers can use, test, substitute, and adapt existing components of the framework with new ideas and techniques. Flower modularity allows the integration with other frameworks more easily, for example, MAS framework and evolutionary computation libraries. It is possible to integrate ML workloads in the FL system and to use any ML algorithm with this framework. Flower also enables the user to configure the FL systems with high detail and provides a large list of algorithms for aggregation and client selection that are already implemented. This framework is also updated frequently and has easy-to-follow tutorials and examples which helped the initial phase of adaptation to the Flower workflow.

One of the main reasons that led to the choice of Flower and not the others was the flexibility in its architecture that allowed an easy configuration of every component existent in an FL system. The other frameworks were very close or did not follow a standard FL configuration that supported the scenarios that would be used in the case studies of this dissertation. Another thing that made a big difference was the detailed documentation and the number of easy-to-follow examples in the repository of the framework. The other frameworks had lack of documentation and most of them were hard to read. Almost any other had working examples of those which were tested alongside Flower. Finally, Flower was the framework at the moment of choice with the most contributors and with a bigger group of maintainers, making it more reliable and efficient when it comes to bugs and error detections and corrections.

3.1.3 Machine Learning Libraries

Machine Learning (ML) libraries are essential when it comes to implementing AI algorithms. These libraries make the development, test, and validation phases much more efficient by removing the boilerplates. In this dissertation, during the development of the proposed solution, four ML libraries were used to facilitate the operation of the datasets and to implement either DL models or GP models. The libraries are NumPy, TensorFlow, DEAP and Rasa.

NumPy stands for Numerical Python and is an open-source library written in Python and C (Harris et al., 2020). The C language helps the library to calculate efficiently and faster the parts that require more computational power. This library is used for calculations with large, multidimensional matrixes. It handles arrays much better than the regular arrays provided by Python. It has also a lot of mathematical functions and operations to help the developer not to waste time with mathematical boilerplates. NumPy is used by a lot of other libraries, and it has a large scientific community supporting it and contributing to it. This library was mostly used to prepare and operate with the datasets used in the case studies.

TensorFlow is one of the most popular ML libraries out there (Abadi, Agarwal, et al., 2016). It was developed by Google with help from the community. It is a free and open-source library. This library aims to provide the best tools for users to build top-notch ML-based applications. It

can help the user to build ML pipelines and several models can be configured and trained. TensorFlow allows the use of Compute Unified Device Architecture (CUDA) (NVIDIA et al., 2020). CUDA is an application program interface that helps the execution of complex matrix calculations by using graphical processing units available in a computer machine. This makes the training of the ML models much more efficient and less time-consuming.

DEAP stands for Distributed Evolutionary Algorithms in Python and is a prototyping library for testing different evolutionary computation algorithms, ideal to be used in research (De Rainville et al., 2012). It gives access to every component inside the library, and it enables its configuration to suit each different problem. DEAP works well with parallelism and multiprocessing frameworks. It uses Scalable Concurrent Operations in Python (SCOOP), a framework that allows concurrent parallel programming in Python. DEAP allows the development of customized genetic algorithms, regular genetic programming algorithms and string-typed genetic programming algorithms. The big difference between DEAP compared to other evolutionary libraries is that it provides an open interface to the user that enables highly and deeply customizable algorithms without losing practicality. This library was used to execute the open-source GP algorithms in the case studies of this dissertation.

Rasa is an open-source library that aims to help the user develop and deploy generative conversational AI platforms and assistants (Rasa Technologies Inc., 2023). One key difference of this library compared to other NLP libraries is that Rasa is highly customizable. It allows the user to choose any NLP or large language model to operate as the motor of the assistant, in a plug-and-play fashion. The algorithms associated with the intent classification and entity extraction can also be configured and adapted to the different use case scenarios. This library has a tool for chat assistants, voice assistants and tools to help deploy the assistant in several third-party messaging apps and communication platforms. This library was used to implement an assistant in one of the agents of MAS.

3.1.4 Machine Learning Models

In the development of the proposed solution, several scenarios were tested to validate it. For these scenarios, two different models were used against different datasets and settings. One of the models is a DNN made by Ramos et al. (2021), and the other model is a GP made by Bi et al. (2021b).

Ramos et al. (2021) model was made for an energy forecasting contest, and it was used to be tested in an FL environment. The model is a feedforward fully connected neural network with four layers, the input layer that can have five or seven nodes, two hidden layers that have 32 nodes each, and the output layer that has only one node. The input of the model is the energy consumption from the last five or seven days, depending on the number of previous days considered, at a given period of the day. The output of the mode is the forecast of the energy consumption for the next day at that given period of the day. The optimization algorithm used is the stochastic gradient descent and it uses an early stopping condition which will monitor the

evolution of the loss value for the last 10 epochs, and if there is no improvement in the training, it stops. The model runs for 200 epochs. The loss function used is the Mean Squared Error, and the metric Root Mean Squared Error is also calculated. This model was implemented using TensorFlow.

In the case studies it was also used a GP model called Flexible Genetic Programming (FGP). FGP is open-source and is an image classification model (Bi et al., 2021b). It is called flexible because of the way the GP algorithm constructs the structures of a program. Each individual is a program that can have a flexible structure that allows the creation of different kinds of programs. The set of functions used in this GP algorithm are image processing functions. An FGP individual can have four different layers, filtering layer, pooling layer, feature extraction layer and concatenation layer. The output of the individual is then attached to a minmax function to normalize the values and then to a support vector machine algorithm. The support vector machine will receive the normalized output from the minmax function and will try to classify the image using that data. The support vector machine must be trained with the training dataset first and then tested against the test dataset. This model was implemented using DEAP. For further information please read the authors' official work (Bi et al., 2021b).

3.1.5 Datasets

In the development of this dissertation, some datasets were used to train the ML models in Chapter 5. Only public datasets were used, for both image classification and energy modelling. The list of the datasets is shown in Table 6 and a further description is given in the following paragraphs.

Table 6 - Descriptive list of the datasets used in the Case Studies section.

	Type of Problem	Size
Energy Dataset 1	Energy Consumption Forecast	4 years of 5-minute data
Energy Dataset 2	Energy Consumption Forecast	1 year of 15-minute data, 51 clients
FEMNIST	Image Classification	805,263 images
MNIST	Image Classification	72,000 images
MNIST-ROT	Image Classification	72,000 images
Rectangles	Image Classification	62,000 images

The Energy Dataset 1 is a dataset made with the information gathered in the building of the research centre that hosted the author of this dissertation during its development. The building has several smart devices that continuously gather data about the energy consumption of each room, the photovoltaic production and the batteries' charges and discharges. The consumption can be further divided into specific devices like air conditioning, fridges, lights, computers, and laptops. Other data are being monitored like the light intensity outside the building, the wind strength and direction, the humidity levels, and the temperature. The building can be divided into five zones to represent five different households. The dataset has data from four years of energy consumption, from 2019 to 2022.

The Energy Dataset 2 is a pre-processed dataset for energy scenarios that aggregates data from three sources, a building of a research centre in Portugal, a community of households from the United Kingdom and another community of households from France (Goncalves et al., 2022). The dataset represents 51 members, one represents a public library, and the other 50 represent residential households. The data is in periods of 15 minutes and corresponds to a whole leap year, 366 days. The energy community has photovoltaic production, and it has the consumption discriminated in 10 individual commonly used appliances and the data is registered in kilowatts.

FEMNIST stands for Federated Extended MNIST (Caldas et al., 2018). The extended means that the original MNIST dataset was extended with more labels including uppercase and lowercase handwritten letters. This makes a total of 62 classification labels. The federated part of the dataset means that the dataset was adapted for FL scenarios and images were grouped by each different writer. There are in total 3,550 writers, and the dataset is composed of 805,263 images. The distribution of each writer is different in both size and label skewness.

MNIST, the Modified National Institute of Standards and Technology dataset, is one of the most used datasets in the image classification problem domain (Li Deng, 2012). The dataset is composed of images in grayscale of handwritten digits, from 0 to 9. The size of the images is 28x28 pixels and it has 72,000 images in total, originally divided into 12,000 images for training and 60,000 images for testing. The dataset has digits written by several writers, all with different writing styles.

MNIST-ROT is one variation of the MNIST data, it stands for MNIST Rotation (Larochelle et al., 2007). MNIST-ROT is identical to the original dataset, it is also composed of handwritten digits, from 0 to 9, and it has the same number of images in total. The difference is that all the images were rotated using a random degree, making it harder to detect the correct label.

Rectangles is another known image classification dataset (Larochelle et al., 2007). The dataset is composed of black-and-white images with rectangles on it. The rectangles are of different sizes and are in different places. The idea is for the models to identify accurately which rectangle in the image has a larger width or length. The dataset has a total of 62,000 images and the size of each image is 28x28 pixels. Originally the dataset is divided into 12,000 images for training and 50,000 images for testing.

3.2 Technological and Social Challenges

In this section, the ethical issues in artificial intelligence and data privacy and security are described and discussed in the context of the dissertation topics. Regarding the ethical issues the definition of ethics in AI will be given and the main requirements of European guidelines for trustworthy AI are listed. Then the ethical problems still present in the field of federated learning and multi-agent systems are also explored. In regards the data privacy and security, an overview of the concepts is given and explored in the context of machine learning and federated learning.

3.2.1 Ethical Issues in Artificial Intelligence

Ethics in AI is a topic that is very much discussed nowadays, and its importance has been rising since the creation of very powerful AI systems like robot Sophia (Hanson Robotics, 2016), ChatGPT (OpenAI, 2022), and many others. "*With great power comes great responsibility*" as the says goes, and that is very applicable in the AI domain. AI has been increasing its domain applicability and is surpassing the precision of human skills in many problems that a few years back were thought to be impossible to solve. But to understand what the ethics issues in AI are, there should be a definition of what ethics is and what its applicability in AI. Ethics, according to (Siau & Wang, 2020), is a set of moral principles that help us divide the good values from the bad values. In AI, ethics are the set of guidelines that describe ways and principles to design and develop AI systems to minimize negative impacts on the people using the AI as well as the society itself. Because of bad system design and lack of analysis in the development phase of AI systems, problems at different levels can occur in our society (Hagendorff, 2020). For example, when Amazon used a hiring tool based on AI that gave more opportunities to men than to women (Lavanchy, 2018), or the case that Microsoft launched a chatbot on Twitter that learned with the user's interactions, and in less than 24 hours the bot started tweeting racist comments (Vincent, 2016).

Despite these problems not causing too much damage, as they were solved quickly, it is still a concern for AI system developers that these issues must be well analysed and solved even before the issues manifest themselves. That's why the European Commission came up with a set of guidelines on how to build Trustworthy AI (European Commission, 2018). These guidelines were developed by the High-Level Expert Group on Artificial Intelligence in the European Commission. In total, there are seven requirements that the AI systems must follow to be categorized as trustworthy. These guidelines affect everyone around the AI system, e.g., the developers, the deployers, the end-user, and even the society itself. The requirements are the following:

- Human agency and oversight: IA can't harm the physical and psychological integrity of the individuals, but rather protect them. The AI system must support humans and not the other way around.

- Technical robustness and safety: AI systems must be prepared against all attacks possible from malicious actors and have security mechanisms that stop the execution of AI when some action is about to cause any harm to a human. The AI systems must be reliable.

- Privacy and data governance: AI systems must protect the privacy and integrity of the sensitive data that they use, from the training phases to the deployment and execution of the system.

- Transparency: all the lifecycle of the system must be traceable in a way that is possible to know the reasons for any given decision made by the AI, this is possible with the help

of interpretability. In addition, humans have always the right to know that they are interacting with an AI.

- Diversity, non-discrimination, and fairness: AI systems must be accessible to any type of person and the data used to train these systems must not be biased and unbalanced, creating unwanted discrimination scenarios.

- Societal and environmental wellbeing: The environmental and societal impact aspects must be analysed and studied when designing and developing these systems.

- Accountability: The internal and external evaluations of an AI system should be made public so the trustworthiness of the system can be raised. Every negative impact that occurred in the execution of these systems should be made public as well.

These guidelines summarize all the aspects that all the persons involved must be aware of so that it is possible to create trustworthiness around AI. This applies also to the systems involving the combination of FL and MAS.

Regarding this dissertation, ethical problems can come up when using FL systems. These systems require trust between the participants (clients and server) because if the system does not have the security mechanisms available and implemented it is possible to cause harm to the overall system (Lyu et al., 2020). Clients with malicious intentions can modify the global model for their benefit or the prejudice of other clients. Although FL, to some extent, can make the data private, the global model, normally a DL model, can be easily reverse-engineered to recreate the data that was used by a client to train their local model (P. Liu et al., 2022b). Another concern with the FL systems is that they can intensify a problem that already exists in centralized ML systems, which is the presence of non-IID (Zhao et al., 2018a). Because the data is very different from client to client, either the quality, variety, and quantity, can turn the global model biased and unbalanced if not done with the right mechanisms to mitigate these issues (H. Wang et al., 2020a). These and other problems must be dealt with before the implementation of the FL system.

In the MAS field is also possible to commit some unethical actions inside the system. Not only do MAS have the same problems as the single-agent systems, mostly related to the use of ML algorithms, but they also have ethical problems related to the social and distributed aspects of it. According to Hedin & Moradian (2015) and K. Gal & Grosz (2022), possible problems are threats to the communication system of the MAS (e.g., denial of service attack, damage attack, event-triggered attack), threats from hosts to agents (agent authentication), threats from agents to agents (reliability of the information gathered from the internet), altering the code, data, and configuration of an agent and faking services and authorizations. Proper considerations and designs must take place in the development of these systems so that these problems can be minimized at all costs. Especially when combining MAS and FL systems because the ethical issues that can be found in each field add up to each other, increasing the concerns and the impacts and variety of issues that can arise (Cointe et al., 2018).

3.2.2 Data Privacy and Security

Data privacy and security are two topics that are highly related to ethics regarding the AI domain. If we let data privacy and security be trespassed and do not take action to take the necessary precautions, people can act maliciously more easily. If the systems developed are prepared against that type of malicious act, the harm can be reduced or even inexistent (Bertino et al., 2021). But what is data privacy and security? Data privacy is everything that concerns the personal data of the users, more specifically how the data is stored, changed, removed, and shared. Data privacy protects the user from how its data is used giving him control over how its data can be managed (Dilmaghani et al., 2019). Data security, on the other hand, is everything that concerns the protection of data from theft, corruption, and unauthorised access throughout its lifecycle. This lifecycle passes through the creation, storage, usage, sharedness, archiving and destruction of data (Meurisch & Mühlhäuser, 2022). In a summarized way security in AI is about protecting the system in a way that malicious attacks do not compromise the well-being of the system (Fabiano, 2019).

Data privacy and security are major concerns in the field of ML. One concern is the potential for sensitive information, such as personal data, to be accessed or misused by unauthorized parties (X. Liu et al., 2021). Another concern is the possibility of adversarial attacks, where an attacker manipulates the input data to cause an ML algorithm to make incorrect predictions. Additionally, ML algorithms can perpetuate biases present in the training data, potentially leading to discriminatory outcomes. Ensuring the security of ML systems and protecting the privacy of individuals whose data is used for training and testing is crucial to maintaining trust in the technology (Bae et al., 2018). However, measures are already being studied to mitigate these issues (Chen & Babar, 2022). For example, Abadi et al. (2016) propose a technique called differential privacy that makes it difficult for an attacker to infer information about an individual data point by adding noise to the model update parameters. Another example is the work of Steinhardt et al. (2017) where the authors test some attack and defence worst-case situations to study the impact that the attacks may have on the users. As it is possible to see, ML continues to have a lot of challenges regarding the security and privacy of data. For that reason, FL was originated to help tackle some of those problems.

FL is a technology that was created to help mitigate the data privacy problem around ML systems. FL came to decentralize the data and keep it private in a way that it is still possible to train the ML models. Despite helping resolve one of the issues, it didn't directly resolve the others. Most of the data problems described in the ML systems are still applicable to FL systems (S. Shen et al., 2022). One of the core problems in FL is the performance of the systems concerning non-IID data (Kairouz et al., 2021). This issue has been studied and there are already some advancements. For example, Zhao et al. (2018b) find that by sharing a small data subset among all the clients the precision can be increased in a non-IID environment. Another example is H. Wang et al. work (2020b) where the authors propose a control framework that intelligently chooses the clients to participate in each round in a way that counterbalances the bias from non-IID data.

In the process of developing the solutions proposed in this dissertation, the author will use any technology and/or solution that can mitigate privacy and security data problems every time it is possible and necessary. However, because the proposed solution is only a proof-of-concept intention of this dissertation is not to add any advancement regarding these issues, it will not be required to propose any novelty in this matter.

4 A Novel Framework for Dynamic Federated Learning Systems

This chapter describes a novel framework extension called Python-based framework for heterogeneous agent communities for federated learning (PEAK-FL). PEAK-FL is an extension for developing dynamic federated learning systems based on agents with PEAK and it has functionalities related to their conceptualization, implementation, analysis, and validation. To understand how it works and the reasons behind the design decisions, a detailed description of the solution's architecture is given in this chapter. Figure 7 shows the overall architecture of the proposed solution.

In the diagram of the proposed solution, as shown in Figure 7, there are several layers representing the different components that exist. The base layer is PEAK which has all the fundamental functionalities that allow the development, deployment, validation, and debugging of MAS. In this layer, the agents can coexist with each other, create communities, create isolated simulations, and organize themselves inside the ecosystem. Different MASs can interact with each other and use services available to each other. Above PEAK is PEAK-FL, which is an extension of PEAK, and it adds functionalities regarding FL. It adds new behaviours and roles to capacitate the agents with the ability to create and manage FL environments. These new functionalities do not affect the other functionalities in PEAK. On top is the PEAK Dashboard, a web application that interactively shows the PEAK Ecosystem. The PEAK Dashboard is fed by the data generated in the ecosystem and can be organised in different formats (i.e., tables, plots, graphs). With it, it is possible to analyse the data being updated in real time, including the data being produced by the FL systems created with PEAK-FL. To further understand how each of these components works internally and how they interact with each other technically, the following three sections were structured:

- Core functionalities used from PEAK – describes in a more detailed way what the PEAK framework and its basic concepts, the structure and the reasons behind some design decisions, as well as what functionalities were used by the proposed solution;

- PEAK added functionalities - describes which functionalities were added to the PEAK framework that were needed as well as some other additions that do not impact directly the proposed solution but were made for quality, organization and documentation purposes;

- PEAK Federated Learning Extension - describes how Flower framework works behind the scenes, how the integration of Flower was made with PEAK, what client and server agents' models were developed, how the serialization of the model was done so it could be transferred from one agent to the other and, finally, what strategies are available in the framework.

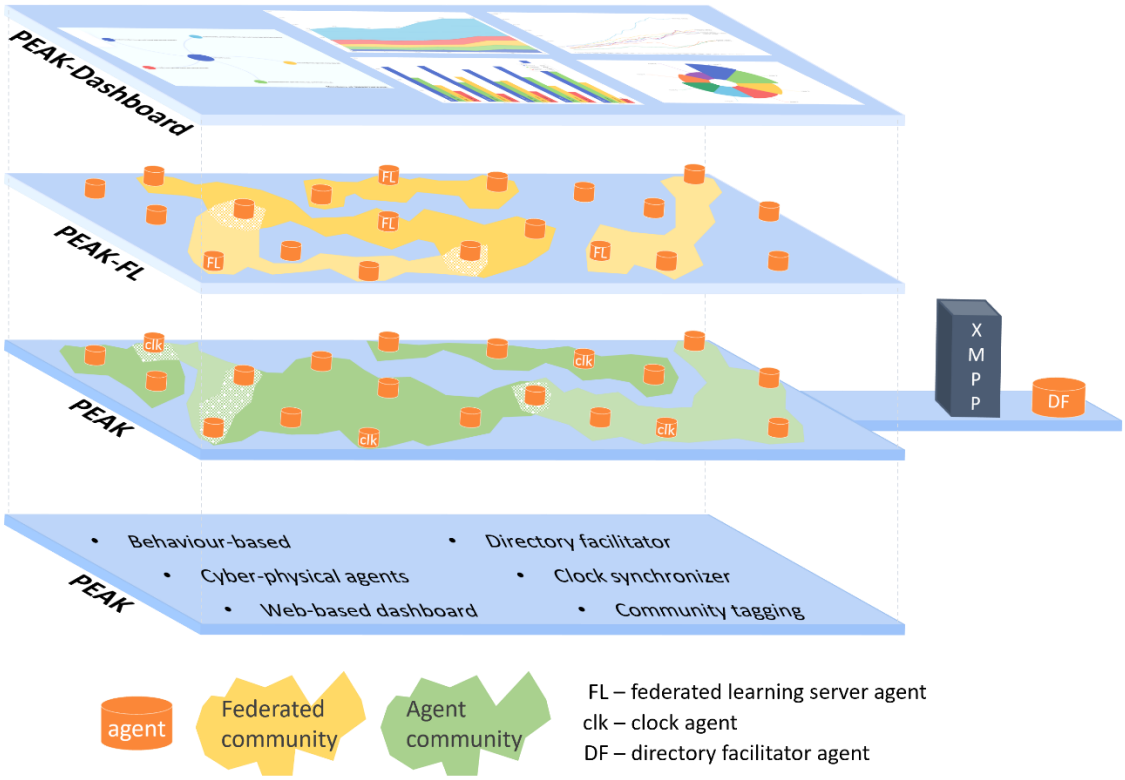


Figure 7 - Overall architecture of PEAK-FL

4.1 PEAK Core Functionalities

PEAK stands for Python-based framework for heterogeneous agent communities and is a framework for designing, implementing, and analysing MAS. The main idea is that different heterogeneous agent communities, with different purposes and goals, can interact, negotiate, and share resources, all in one big ecosystem. This ecosystem also facilitates the integration with other external systems, for example, web dashboard applications.

To understand better how PEAK works and what are the mechanisms that allow the development and deployment of MAS it is necessary to talk about the underlying framework

that provides the basic functionalities. That framework is called Smart Python Agent Development Environment (SPADE), and it provides the tools to design and develop the agents and the barebones of the agent's communication. The protocol used for the communication is the Extensible Message and Presence Protocol (XMPP) and it has the essential functions regarding the authentication, peer-to-peer communication, management of the personal contact list and presence notification system. On top of that, SPADE made the behaviours' basic structure which is inspired by the JADE framework. The behaviours are the following:

- One-shot behaviour: it runs the behaviour only once and immediately.
- Cyclic behaviour: it runs the behaviour cyclically and the user must define how the behaviour will be terminated.
- Periodic behaviour: it runs periodically given the interval defined by the user, the user can also define a specific date and time in which the behaviour should be executed, if the user does not want it to start immediately.
- Finite State Machine behaviour: it is a more complex behaviour that is composed of internal states, each state will execute a single behaviour according to the sequence defined by the user, in a finite state machine manner.

PEAK uses these basic structures to develop more advanced features within the agents and behaviours. The following subsections will describe in more detail the fundamental concepts of the PEAK that one must understand before diving into the actual purpose solution. These concepts are the Ecosystem, the Agent, and the Behaviours.

4.1.1 Ecosystem

PEAK Ecosystem is everything that is related to the environment of agents created by the PEAK framework inside the same physical communication network. Every agent, group and community in the system, the services provided by each agent and even the data that each one of them publishes are all part of the ecosystem. Figure 8 shows an example of it. The PEAK Ecosystem is the largest scope of the system. It is intrinsically related to the communication layer that PEAK was developed upon, XMPP. The Ecosystem is limited by the clusters of XMPP servers and clients (agents and users) that are connected. If, for example, two XMPP servers are not linked through XMPP, then each server represents a different PEAK Ecosystem. Inside the Ecosystem, several different concepts can be identified, namely the MAS, the communities, and the groups.

The Ecosystem can be composed of several MAS that can interact with each other. The MAS in Figure 8 are represented by different colours. Conceptually, a MAS in PEAK is a group of agents that were designed to accomplish a common goal, by either means of cooperation or competition. Practically, a MAS is normally instantiated by the same source (user or device).

Besides that, there can also be single-agent systems, which are normally intended for agents that act as service or resource providers, to help other agents accomplish their goals.

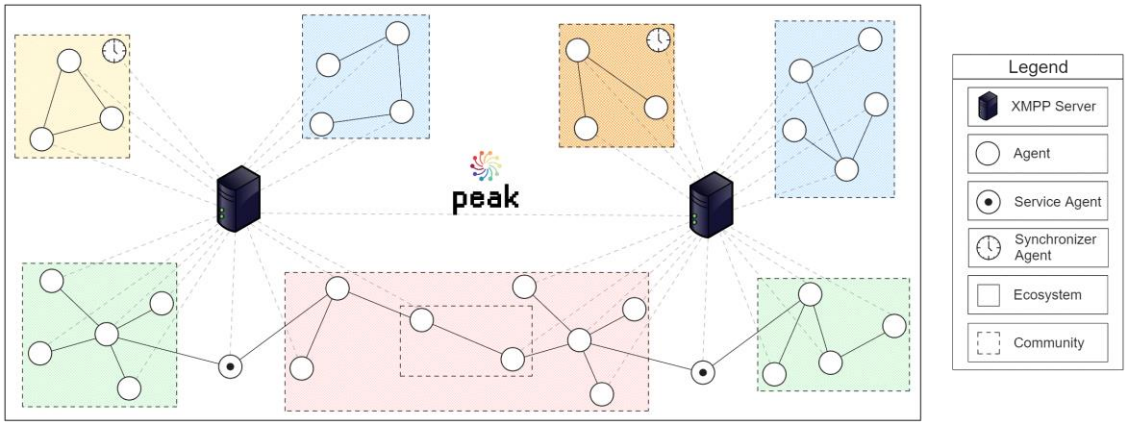


Figure 8 - Detailed representation of the PEAK Ecosystem.

The Ecosystem can also be organized in different communities which are social groups of agents structured to serve a specific purpose. This purpose can be to organize the agents in a structured way, to divide the agents by roles, or to facilitate the availability of a service. These communities can have different structures like societies, hierarchies and markets. The communities are not restricted to a single MAS, which means that there could be communities with agents of several MAS. This is convenient for services that can operate better within a community. Technically speaking, the main objective of the communities is to create and facilitate communication between the agents within the network and between the several groups inside the community. Communities work at the same level as MAS.

To structure the communities there is another important concept to be familiarized with, which is the concept of groups. A group is simply an aggregation of agents with a specific communication channel that facilitates the interactions between the members of that group. Every message sent to the group is broadcast to every member automatically. The communities can be structured by being divided into smaller groups and creating relationships between those groups therefore creating different layers of communication in the same community. This concept is intrinsically related to the functionality of the XMPP called Multi-User Chat and allows the creation of rooms so that users can join in and communicate more effectively with the members of that room. The rooms in the XMPP are what enable the creation of groups inside the PEAK Ecosystem.

4.1.2 Agent

An agent in PEAK is an intelligent and autonomous piece of software that can accomplish an objective that was designated to it by making use of the environment around it. The intelligence and objective of the agent are implemented and defined by the user. PEAK has several utilities that the user can use to help implement and conceptualize the agents for them to achieve their

goals, like the necessary communication functionalities, the functions to create structured communities and predefined agents that enable the creation of simulation environments. In PEAK there are different types of agents already available, like the base Agent, the Directory Facilitator, and the Synchronization agents.

The PEAK Agent is the core agent in the framework, and it adds functionalities to the SPADE Agent. The SPADE Agent has the basic functionalities that the XMPP provides, like the functionalities related to the registration and login of the agents as users, the peer-to-peer communication between the agents, the presence notification functionalities, the management of the agent's contact list, and, apart from the XMPP, it also implemented an internal knowledge system to the agent. The PEAK Agent added the possibility for the agents to use functionalities like Multi-User Chat (MUC), the Service Discovery (Disco) and the Publisher Subscriber service (PubSub).

To register and to log in, the agents need to have a Jabber Identification (JID) and a password. The JID is an identification used in the XMPP to identify each node in the network, including the server itself and some of the services provided. The JID is composed of three parts, namely the local part, the domain, and the resource. The local part is the name of the entity representing it, it can be a user or a service. The domain is the server to which the service or user is registered. The resource can have several purposes depending on the context it is used. In XMPP it is possible to have users with the same local part, but they must be registered in different domains. In a user context, the resource can be used to identify different sessions if the user is logged in more than one place. The only part that is obligatory in a JID is the domain. When used solely it identifies the server itself. The symbols @ and / are used to separate the local part from the domain and the domain from the resource, respectively (i.e., user@example.com/main_session).

The agents can have their list of contacts. This is a functionality inherited from the XMPP. Each agent has its contact list and can add the contacts of other agents to the list. To add a contact, an agent needs to first make a request to another agent and only if it is accepted that the agent can be added to the contact list. In PEAK the agents will accept all the contact list requests automatically, however, this can be reconfigured by the user. This also enables the use of the Presence Notification functionality that allows the agents to see the presence state of the members of their contact lists. The presence state is an indication of what is the availability of a given member, it can be, for example online, away from the keyboard, offline and occupied. The Presence Notification also gives the option to the agents to receive a notification every time some contact changes state (i.e., from offline to online).

MUC was a functionality added by PEAK and it allows agents to create rooms where more than one agent can communicate. They can define topics of conversation in the room and attribute different roles to the agents according to the level of their privileges. The room concept can be seen as a group, like in a social media app, where agents can join the group and chat together. By sending a message to the group the message will be broadcast, by the server, to every member. This allows multi-user communication without the need to send the message

individually to everyone. To create a room the user has to create a JID for that room. In this example, the server should have a MUC component with a unique JID, different from the server (i.e., conference.example.com). To register a room the user must give a unique name, the local part of the room JID, and then choose the domain where he wants to register the room (i.e., room1@conference.example.com). The resource of the room JID is used to identify the user who sent the message (i.e., room1@conference.example.com/agent1). After the room is created the other agents can use the JID of the room to join the conversation.

Disco is an XMPP functionality added by PEAK that allows the users to receive information about the different nodes present in an XMPP network. With this service, agents can discover two types of information, the protocols and description of the node, or the list of items associated with the node. An item is for example a room in the MUC component. This is because the discovery service can be used directly on the servers and the results will be a list of components available in that domain. When the Disco functionality is activated, the nodes can be configured to be publicly available or not. Only the public nodes will be retrieved by the Disco component. This functionality is useful to the agents as they can search and reach out for different services.

PubSub is a functionality that allows the agents to publish and subscribe to different topics available in the network. The agents can create publish nodes where they can publish information. The agents can also subscribe to publish nodes to be notified every time new content is published. The only requirement for an agent to create a publish node is having the administrator role in the server.

Directory Facilitator (DF) was inspired by the Foundation for Intelligent Physical Agents (FIPA) Agent Management specification. In PEAK the DF is an optional component of the ecosystem and is described as a service provider. DF provides its services to the other agents. There should be only one DF per XMPP server. When more than one DF is present in the ecosystem they create a federation of DFs, as stated in the FIPA Agent Management specification, where they work together to provide services to all the agents in the PEAK Ecosystem. DF provides two services to the ecosystem: the agent communities service, and the data management service.

The agent communities service provided by the DF enables the creation and management of the communities inside the PEAK ecosystem. To create a community the agent must request it to the DF and define the structure of the community and its name, which must be unique. The communities can have different types of structures like holarchies, coalitions, societies, and markets. The community is then maintained by its members and every structure update must be sent to the DF. This service also allows the agents to search and join the communities available. The communities are structured using groups and the relationships that those groups have with each other. The DF uses a tree-like data structure, where the nodes represent the different groups inside the community.

The data management service provided by DF enables the persistence and monitoring of the data generated by the agents in real time. This is useful for debugging the ecosystem and analysing the data. To use this functionality, the agents need to send a request to the DF. The

data can be saved and updated periodically and it is persisted in an internal database maintained by the DF. The data format used, for now, is JavaScript Object Notation (JSON). To enable the monitorization of these data in real time, DF provides a representational state transfer application programming interface (REST API). This REST API is public, and it enables the publication of the data, like the communities' structures, the communities' members, and the agents' data, that the DF manages internally to external systems.

The Synchronization agents are two agents that enable the creation of simulation environments inside the PEAK Ecosystem. The agents are called Synchronizer and Synched Agent (SyncAgent) and have two different roles inside the simulation environment. The Synchronizer is responsible for managing the speed of the simulation and notifying the SyncAgents for every tick of the simulation and it has an internal clock for keeping track of the simulation speed. Each tick of the clock represents a different period. There can only be one Synchronizer inside each simulation environment. The SyncAgent is responsible for receiving the notifications from the Synchronizer and acts in each period according to their objectives. The simulation that the Synchronization agents enable is called time-based simulation.

Time-based simulations use time to control the simulation. The user can define the periods between each tick of the clock. There are two different ways to configure the time-based simulation clock. One of the configurations is based on the number of periods. Each tick represents a period, and the agents keep track of the number of the current period. The user must define the number of periods the simulation must have. The other configuration is more complex as it requires the creation of a virtual time frame to be used inside the simulation. For this configuration, the user must define the initial and final date-times of the simulation as well as the time of the simulated period and the real period. The simulated period is the time of the period that each tick has inside the simulation (i.e., 15 minutes). The real period is the speed each tick has outside of the simulation (i.e., 1 second). For example, if the simulated period is 15 minutes and the real period is 1 second, during the execution of the system 1 second will correspond to 15 minutes in the simulation. The simulation stops after the final date-time is reached.

4.1.3 Behaviours

The behaviours in PEAK can be divided into two categories: Core Behaviours and Predefined Behaviours. The core behaviours are the ones that the user will use to customize his agents and the predefined behaviours are some prebuilt behaviours that have specific functionalities.

In PEAK there are four different core behaviours which were adapted from SPADE: the One-Shot Behaviour, which executes only one time, the Cyclic Behaviour, which executes cyclically, the Periodic Behaviour, which runs from time to time defined by the user, and the Finite State Machine Behaviour, that allows the development of more complex behaviours that follow a set of states. These behaviours have inherited the SPADE behaviours functionalities like peer-to-peer communication and the internal knowledge system. This knowledge system can be seen

as an internal database where the agent can persist its knowledge. Each of these behaviours has also an internal message filter so the agent can filter the messages for each specific behaviour. In PEAK, it was added some features to these behaviours that enable the agents to create, join and leave groups in the ecosystem, enable communication through the groups, request the list of members of a given group and retrieve the list of groups existent in the ecosystem. In addition to that a configuration was changed that allowed the behaviour to wait to receive a message without a timeout and the ability to wait for another behaviour to complete was implemented as well.

The predefined behaviours are based on the core behaviours but have specific functionalities and are made to overcome some boilerplates when developing agents. This helps the user to focus more on the actual conceptualization and implementation of the agent instead of repetitive functionalities that the general users would use frequently. The behaviours available in PEAK are the Join Community, Leave Community and Publish Data.

Join Community has two different functionalities. One is to allow the agent to join a group in a specific community and the other is to create a community, if does not exist, and define a structure for it. This second functionality is only possible with the presence of the DF agent in the ecosystem since it is DF's responsibility to maintain the communities' structures in the ecosystem. Leave Community is the opposite of the join community behaviour as it will make the agent leave the community and stop receiving messages from it. If the agent is the last member of the community to leave, the community will be deleted from the ecosystem. Publish Data enables the agent to publish data that is being generated by itself during its execution and allows it to be saved and monitored. The data is sent to the DF and saved in its internal database. The data is then accessible through the REST API service of the DF. To update the data that is already saved in the DF the agent must publish the data with the same identifier.

4.2 PEAK New Functionalities and Improvements

In the previous section, it was described the main concepts of PEAK and what functionalities that already existed. In this section, it is described the new concepts and functionalities that were added posteriorly to PEAK to enable the implementation of the proposed solution. Other improvements that impacted indirectly the development of the proposed solution are described.

4.2.1 New Functionalities

The new functionalities that were added are (i) community tagging, (ii) the yellow page service, (iii) the event-based simulation, and (iv) the Hypertext Transfer Protocol (HTTP) service. For these functionalities to work some of the predefined agents, like the DF and the synchronization agents, had to be updated. In addition to that, new behaviours were also implemented.

Firstly, community tagging is a new feature in PEAK that enables the agent to mark and identify communities and groups with tags. Previously, with the Discovery Service, it was only possible to request the whole list of groups, but now it is possible to narrow the search with the tag mechanism. Tags are identifiers that can be used to facilitate the search for specific communities and groups. For this feature, the DF was updated to keep track of the communities tagged by persisting it in its internal database. In addition to this, two new behaviours were added, the Tag Community and the Search Community. The Tag Community enables an agent to tag a specific community or group with one or more tags. This will send a tag request to the DF, and it will associate the tags with the community identified. The Search Community enables an agent to search for a community or group based on one or more tags. This will send a search request to the DF, and it will search on its internal database for the list of groups and communities that are identified by the tags and send back that list to the agent.

Another functionality added was the yellow page service. According to FIPA Agent Management specification, the DF should have this functionality which must allow the search, registration, modification, and deregistration of services provided by other agents. This functionality will only redirect the agents to the provider of the service they are looking for. For this to be possible the DF was updated to allow the agents to search any service available, register any service, and modify and deregister only the services that the agent is the provider. All this information is stored in the DF database. To enable the agents to perform these actions new behaviours were created, namely, the Search Service, the Register Service, the Modify Service, and the Deregister Service. The Search Service will send a request to the DF to get the list of services available. Keywords can be used to perform a narrower search. The Register Service will send a request to the DF sending the information needed to register a service (i.e. name, identifier, description, and provider) and the DF will respond with a success message if the service does not yet exist. The Modify Service will send a request to the DF to rewrite any information regarding the service. And finally, the Deregister Service will send a request to the DF to remove the service from the list.

Regarding the simulation capabilities of PEAK, a new type of simulation was added called event-based simulation. Previously there was only a time-based simulation which used a predefined time interval to mark different periods in the simulation (i.e., a tick for every second). But for cases where the duration of the agent's actions is not predictable (i.e., the training phase of a FL system), an event-based simulation would be a better approach. The idea of the event-based simulation is to divide the simulation into different phases called events, and the next phase only starts when the previous event has finished. These events are driven by user-configured rules which will say when the event has finished, and which event will start next. The events do not need to be in sequence, they can be structured using a finite-state machine model. For the implementation of this functionality, the previous Synchronizer and Synched Agent were renamed to Time Synchronizer and Time Synched Agent, now responsible for the time-based simulations, and two new agents were created the Event Synchronizer and the Event Synched Agent which are responsible for the event-based simulations. Regarding the process of the event-based simulation, it starts with the Event Synchronizer creating a group for the simulation environment. Then the Event Synched Agent agents will enter the group and the Event

Synchronizer will start the first event. After the Event Synched Agent agents finish their actions for the current event, they will inform the Event Synchronizer, and the next event will start.

Another functionality that was integrated into PEAK is the HTTP service. HTTP service allows the users to exchange files through the XMPP network. For this functionality to work it must be configured a new component in the XMPP server where it will be hosted by the file server. To save a file in the server the agent must request the HTTP service to upload a file with the information about the name of the file, its size, and its file type. The server will then send the two Uniform Resource Locators (URLs) related to the GET and PUT functions in the HTTP. To upload the file the agent must use the PUT URL and should share the GET URL with the agents that can have access to that file.

4.2.2 Extra Improvements

While the new functionalities were being implemented, some improvements were made to PEAK. These improvements are related to the quality and clarity of the code and the interface of PEAK, the documentation, and some other functionalities that help a better execution of the agents as well as their debugging. Following is the list of the improvements:

- Added information to the PEAK documentation to clarify existent content and describe new functionalities;
- Code refactoring to make PEAK more efficient and user-friendly;
- Created examples to explore better each functionality present in PEAK to help the user go through the ideas behind each functionality;
- Published PEAK in the Python Package Index, which makes PEAK available publicly to every Python user;
- Created a docker for PEAK, the XMPP server, the Dashboard and the DF agent, to help facilitate new users to start developing MAS with PEAK more quickly;
- Improved the logging system and added a logging module for the users to handle and write better logs when running their systems;
- Correction of code smells and bugs;
- Added YAML as a configuration file option for the user to be able to configure an entire MAS in a single file and help run it quickly;
- Added functionality in the core behaviours that enable the chaining of behaviours, so it is possible to sequentially run a set of behaviours, one after another.

4.3 PEAK for Federated Learning Extension

PEAK for Federated Learning (PEAK-FL) is a PEAK extension as it aims to bring new functionality related to the creation of FL environments. PEAK-FL added two new agents as well as new behaviours. The agents are the Client and the Server, representing the type of participants that exist in an FL environment, and the behaviours that make agents able to participate, collaborate and make decisions in FL environments. To make the development more efficient a FL framework was used, called Flower. In this section, an overview of Flower's basic concepts is given to understand how it works, then an explanation of how the integration was done and finally, an explanation of how the functionalities work.

4.3.1 Flower

Flower is a FL development framework that allows users to conceptualize, design, implement, test, and analyse FL systems. This is one of the most used FL frameworks and with the most contributors involved. It has a lot of examples with paper implementations. It has two modes of operation, one for simulation purposes which runs the federation in the same computational device, and a deployment mode which uses Google Remote Procedure Calls (gRPC) as the communication protocol, and it can be used in real distributed environments. The gRPC is an open-source framework that enables communication between clients and servers through remote procedure calls. The idea is that the server will have a list of functions with the respective parameters and variable types defined and the client can implement this interface and call these functions remotely. Its overall architecture can be seen in Figure 9.

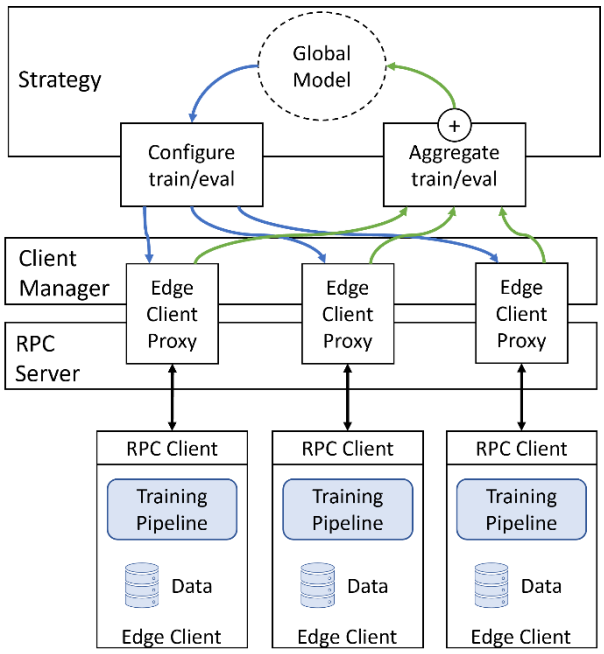


Figure 9 - Flower internal structure, adapted from Beutel et al. (2020).

Flower, and most of the other FL frameworks as well, is based on a client-server architecture. This means that the participants of the network will be either a server or a client. In Flower, the server provides an FL service that allows the clients to join and train collaboratively in an ML model while keeping their data private. Each cluster of a server and client is called a federation. The server is responsible for coordinating the federation and the process of training the model. The coordination strategy can be configured in several different ways depending on the type of clients available in the federation, the type of the model and the type of communication channels the federation will use to communicate. The strategy is the coordination plan that the server will follow to guide the training of the model to the desired outcome. Flower has a list of strategies available with different aggregation algorithms (i.e., Federated Averaging) which the user can make use of in the development of FL environments.

In Flower, the strategy has a predefined way of synchronizing the different phases. The phases are the initialization of the model parameters, the training phase, and the evaluation phase and they are run in this order respectively. A round is made by a training phase followed by an evaluation phase. The initialization of the model parameters can be done on the server side, or it can be done by a random client, chosen by the server. The training phase can be further divided into the fit configuration step, the training model step and the aggregation of the model parameters step. The fit configuration step is when the server selects the clients to train the model based on some heuristic. The evaluation can be configured in two different ways, the server can evaluate the model using a local dataset, or it can have a similar process to the training phase where it will use the clients to evaluate the model. The steps are the configuration of the evaluation process, where the client selection for this phase will be made; the evaluation of the model; and the aggregation of the results.

Two other important concepts used in Flower are the client manager and the client proxy. These concepts are used in the server to help manage and synchronize the agents according to the strategy used. The client manager is where the strategy controls the registration and deregistration of the clients in the federation, provides client sampling and allows the server to wait for a predefined number of clients to join the federation. The client proxy enables the server to execute commands remotely on the client side. These commands are used by the strategy to communicate and synchronize directly with the clients. The commands permit the server to make requests to the clients to initialize the model's parameters, get the model's properties, get the model's parameters, fit, and evaluate the current model.

4.3.2 Flower Integration

To understand how the integration of Flower in PEAK was done, it is important to understand how Python works in asynchronous and synchronous environments. In Python, and most languages, a synchronous environment is the most basic type of programming environment where the program will run the code sequentially, one line at a time. In an asynchronous environment, the code can be divided into different coroutines the coroutines will run concurrently. This means that the coroutines will not run in parallel, instead, they will be put in

a list, at the time of its creation, and the coroutine manager and scheduler will control the flux of the program. Only one coroutine is run at a given moment and every time a coroutine must wait for some external operation, it is switched by another task. In Python, to write asynchronous code a specific syntax needs to be adopted. Both the coroutine management and scheduling are made by the same mechanism, the event loop. In one program there can only exist one event loop, the creation of another will result in the crashing of the program.

Flower and PEAK are synchronous and asynchronous, respectively, and to be able to integrate these two frameworks, Flower required the implementation of two classes that have access to the communication layer, the client manager, and the client proxy. There are two different approaches to integrate these two frameworks: create two different communication layers and have the agents talk to each other through XMPP asynchronously and the clients and servers in FL communicating through gRPC synchronously, or adapt the client manager, the client proxy and the strategy used by the server to an asynchronous environment. The advantages and disadvantages of each approach can be analysed in Table 7.

Table 7 - Advantages and disadvantages of the two approaches approach.

Approaches	Advantages	Disadvantages
Two-way communication	• Strategy interoperability • No limit size on the serialization of the model	• More time is needed to implement • More effort for maintaining the code • More complexity communication-wise • Less flexible approach
Asynchronous adaptation	• Less time to implement • Allows more flexibility and expandability	• Flower strategies must be adapted first before being used • XMPP messages have a size limit

One thing that is more valuable in the two-way communication approach is the interoperability of the strategies already implemented in the Flower framework. The code used in the strategies could be used directly in PEAK-FL. Another advantage is that with gRPC the messages do not have a size limit, however, gRPC uses a serialization method called Protocol Buffers and in its documentation, it says “For data that exceeds a few megabytes, consider a different solution; when working with larger data, you may effectively end up with several copies of the data due to serialized copies, which can cause surprising spikes in memory usage.”. This makes it a problem if the models are too large. This approach would need more time to implement given that the author of the dissertation would need to understand how the gRPC works internally as well as dive into the Flower architecture with even more detail to adapt its internal mechanisms to the second layer of communication. With two different layers of communication, there would also need more effort to maintain the code of the layers, because two different libraries would be used.

On the other hand, the asynchronous adaptation approach takes less time to implement given that the only adaptation needed is to change the syntax. Flexibility and expandability are other advantages over the two-way communication approach, given that it gives more flexibility to the user to construct and implement the FL system as well as new functionalities would be

implemented more easily. The approach has also its disadvantages. One of them is the impossibility of directly using the strategies implementation that Flower has. The other disadvantage is that XMPP messages have a limited size to its content. The only way to overcome this limitation is to implement different ways of serializing and sending the data of the models.

After analysing the advantages and disadvantages of the two different approaches, the asynchronous adaptation was chosen. The two reasons that led to this decision were the time needed to implement each solution and the flexibility that the user should have when structuring an FL system based on agents. The user should see the several functionalities of the PEAK-FL as different pieces of a puzzle that can work together in different ways to be able to test new solutions and systems. To integrate this approach two classes were adopted, namely, the Agent Manager, which represents the client manager in Flower, and the Agent Proxy, which represents the client proxy in Flower. To use the aggregation algorithms already implemented in the Flower framework, the user can adapt the aggregation algorithm on its own or adapt the whole strategy to an asynchronous environment.

4.3.3 Architectural Models of The Client and Server Agents

In PEAK-FL two different architectural models are proposed to allow the agents to participate in FL environments embodying the different types of roles. Those models are the Client and Server agent models. They are just models because they are not limited to the functionalities available by it, but instead, the agent can incorporate a variety of roles and functionalities, depending on its need. In regards to the FL environment, the agents can be at the same time client and server if the context itself justifies. This follows one Python principle called duck typing which the code does not care if the agent is a client or server but instead, it only cares if behaves like a client or a server. However, to facilitate the implementation of both client and server agents with standard behaviours a basic structure of client and server agents is provided.

For PEAK-FL to enable the agents to participate and choose their FL environments it is necessary to equip them with the right behaviours. In Figure 10 is possible to see the behaviours that a client agent will need to participate in FL environments and a Simple Client agent with a basic structure already implemented. There are two main behaviours that the client should have, the Choose Federation and the Run Model Task behaviour. Before the agent chooses a federation to join, the agent must be able to get the list of available federations that there are in the ecosystem. The way to get it will depend on how the user configures the system. One option is to use the DF yellow page service and register the federation as a service so the other agents can find and join the federations easily. Another option would be tagging the communities that are running federations with a specific tag and directly joining the federation. After getting the list of the federations the agent will use the Choose Federation behaviour to apply a decision-making mechanism to perform the selection of the federation. This decision-making mechanism needs to be implemented by the user. In the Choose Federation behaviour, there are two other behaviours available that allow the agent to subscribe (Subscribe Federation) or unsubscribe

(Unsubscribe Federation) to a federation. The selected federation will be saved in the internal knowledge system of the agent. These behaviours are based on the One-Shot Behaviour because they only need to be executed once when needed. This also allows the agent to choose as many federations as it needs at the same time.

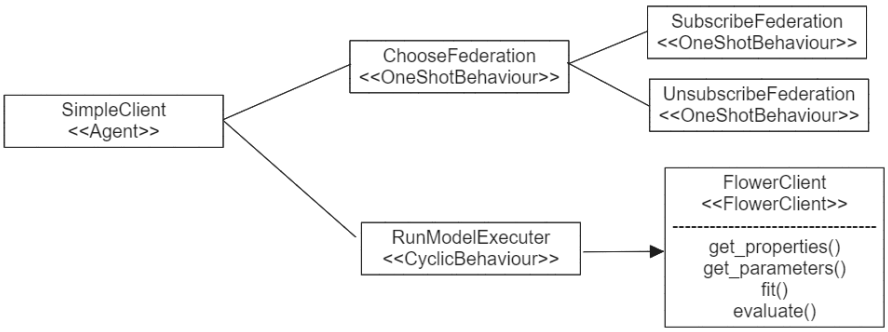


Figure 10 – The PEAK-FL architectural model of the client agent.

After choosing and subscribing to a federation, the agent should have the capacity to manage the model during the federation execution. That is done with the Run Model Task behaviour. The Run Model Task to work it will need an implementation of the model using the Flower Client interface. So, any model implementation done in a Flower client can be directly used in the Run Model Task behaviour. In this behaviour, the agent will wait for the instructions from the subscribed federation so it will be operated according to the server synchronization. These instructions are the same used by the Flower client, namely, get model properties, get model parameters, fit model, and evaluate model. In the instruction “get model properties” the agent will send the model’s properties to the server. In the instruction “get model parameters”, the agent will send the model’s parameters to the server. The instruction “fit model” will train the current model, using the parameters and configurations given by the server and send back the results of the training and the updated model’s parameters. The instruction “evaluate model” will evaluate the model, using the parameters sent by the server, and afterwards, it will send back the results to the server. If the agent is participating in more than one federation it will need a Run Model Trask behaviour for each one of them, as they control only one model at a time.

The Simple Client agent is an agent with a predefined structure already implemented that can be used to prepare a simple FL environment or used as a guideline to develop a customized client. The idea will be to use this to help conceptualize, test, and validate an FL model so then the user can focus on implementing the agent functionalities. This agent will start by choosing one federation given by the user, using the Choose Federation behaviour, and then will start the Run Model Task behaviour to participate in the federation. After the federation ends it will save the results in a file and stop its execution.

In addition to the PEAK-FL client, the framework provides another architectural model, the server agent model. This model allows the agent to participate in a federation with the server

role. For this, several behaviours were conceptualized. Figure 11 shows the behaviours and the Simple Server agent with a basic structure already implemented.

In Figure 11, the first two behaviours that appear are the Register Federation and Unregister Federation. These behaviours will only work in the presence of a DF in the ecosystem given that they depend on the yellow page service to register and unregister the federations. The Register Federation will register the federation as a service in the yellow page service by using a unique identifier and a description of it. The description can have information regarding the type of model the federation has, the average accuracy of the global mode, the type of clients that are allowed, etc. The DF will receive the request and register the service if it is not yet registered. From that point on the service will become publicly available. The Unregister Federation behaviour will remove the service registration from the DF yellow page service by using the service identifier. Only the author of the service can remove it from the list. Both behaviours are based on the One-Shot Behaviour because they only need to be executed when needed. One agent can play the server role for more than one federation.

The next two behaviours are the Receive Subscription and Remove Subscription. These behaviours will handle the subscription requests from the clients. For this, the Agent Manager is needed to control the number of clients participating in the federation. For each client request the Receive Subscription behaviour will register a new client in the Agent Manager and the Remove Subscription will unregister it. Each registration in the Agent Manager will result in the creation of an Agent Proxy. This Agent Proxy allows the server to execute the commands related to the FL on the client side. The Receive Subscription and Remove Subscription need to be operational during the execution of the federation from the beginning to the end, given that more users can join during its execution. This is done using the Cyclic Behaviour which allows the agent to wait for the requests cyclically.

Another behaviour is the Run Federation and as the name suggests the behaviour allows the agent to start the execution of FL systems. The behaviour is based on the Cyclic Behaviour as it will be running for several rounds, with each cycle representing a round. The number of rounds is configured by the user. Inside the Run Federation, there are two other behaviours the Run Fit Phase and the Run Evaluation Phase. The reason behind the division of the federation process into two different behaviours is that each phase needs a different message filter, so the messages are not mixed in case of a system delay, or a client failure. Firstly, the Run Federation will initialize the model parameters. The parameter initialization can be done either by the server or by one client and it's only done once before the first round even starts. After that, the fit phase will start by running the Run Fit Phase behaviour. It’s a One-Shot Behaviour because it only needs to be executed once per round. The Run Fit Phase will first select the clients available to train the model, then will wait for the client model updates and then it will aggregate them and update the global model. Both the client selection and aggregation algorithms can be configured by the user. The aggregation algorithm must be configured mandatorily by the user. Another way to configure these two algorithms is by using an asynchronous adaptation of a Flower Strategies which has these two algorithms implemented. The fit phase ends with the global model already updated and then the evaluation phase starts.

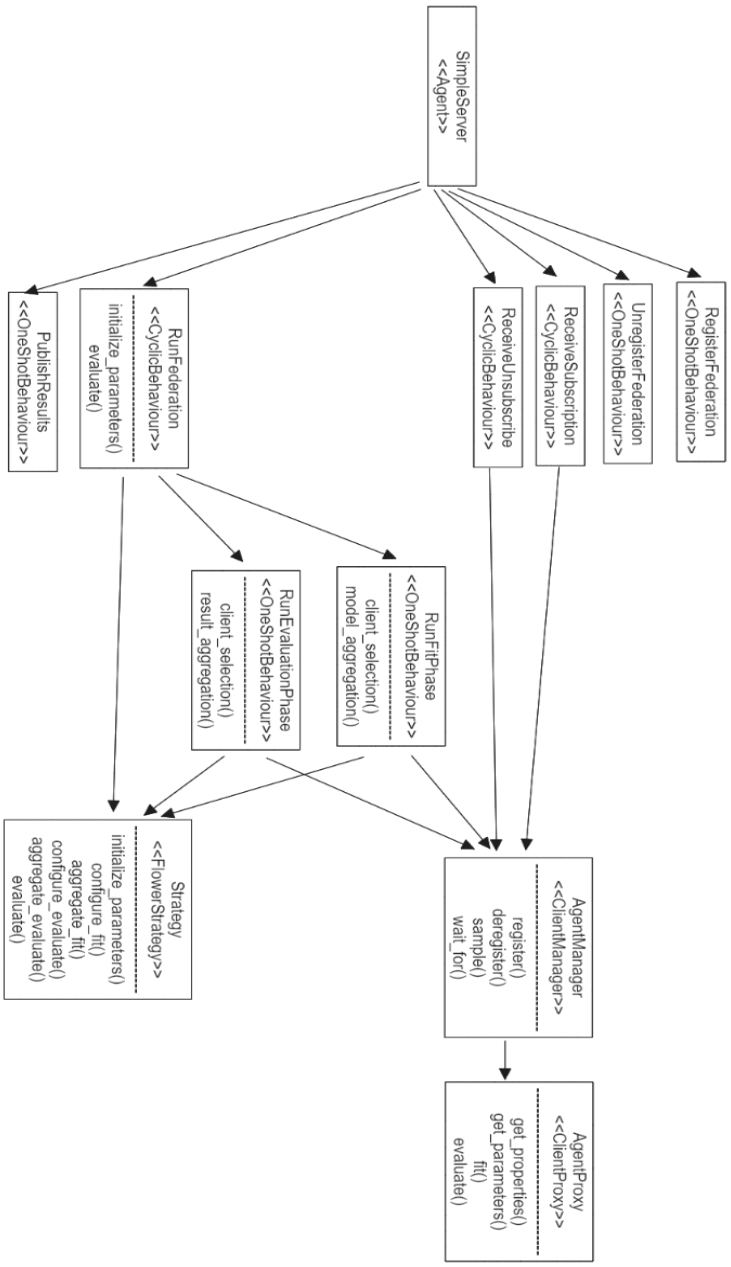


Figure 11 - The PEAK-FL architectural model of the server agent.

The evaluation phase can be done in three different ways. The server can be the one evaluating the model if it has an evaluation function configured, it also can be done by using an asynchronous adaptation of a Flower Strategy, or it can deliver that responsibility to the clients of the federation. For the evaluation function configuration, the user must define the dataset and metrics to be used in the evaluation process. This can be done directly on the Run Federation behaviour or by using an asynchronous adaptation of a Flower Strategy with the evaluation function implemented. If there is no evaluation configured on the server side the responsibility is passed to the clients. In that case, the Run Evaluation Phase behaviour is executed. Firstly, the behaviour will select a set of clients from the pool of available clients, then the model will be shared with the selected clients and the model evaluation will start on the

client side. The evaluation phase ends after the server receives and aggregates the clients' results. The aggregation algorithm must be configured if the evaluation is to be done by the clients, otherwise, the algorithm does not need to be implemented. Similarly to the Run Fit Phase, this behaviour is based on the One-Shot Behaviour.

After the federation has been running for some rounds and the model accuracy has been reaching good results the server agent might want to publish its results. For this, the Publish Results behaviour was developed. In the PEAK-FL context, the publication of the results is done by updating the service description of the federation in the DF yellow page service. The server agent will request the DF to update the service description with the new results. After that, the client agents will be able to see the recent results of the federation and then decide upon that information. The Publish Results is based on the One-Shot Behaviour and can be executed at any time during the execution of the federation.

Like in the client model, the server model also provides a Simple Server agent. This agent has the basic structures needed to create and test a simple FL environment. The Simple Server will first start by registering the federation with the Register Federation behaviour. Then it will execute the Receive Subscription and Remove Subscription behaviours, to let clients subscribe and cancel their subscriptions any time they want. After having the minimum agents needed to start the federation, which is configured by the user, the Simple Server will start the FL process by executing the Run Federation. After each round, the Simple Agent will update the service description of the federation by publishing the results with the Publish Results behaviour.

4.3.4 Approaches to the Serialization of the Learning Model

During the execution of an FL system, the server and the clients will be constantly sending back and forth the model's information needed to train the model collaboratively. For the model to be sent from one agent to another, the model must be serialized. The serialization must happen before an agent sends the model and after the agent receives it. The most straightforward way to do it is to turn the model's information into strings or bytes and send it through the XMPP messages. However, XMPP has a limit on the message size. For some models, it can work very well, but if the federation wants to train a large language model, for example, the XMPP messages could not do it. To solve this problem a software pattern called Strategy (not the same as the Strategy in the Flower framework) was used to facilitate the integration of different ways of sharing the model between agents. The software pattern suggests that it is possible to abstract different solutions that aim to solve the same problem and turn them into different classes. Using the serialization problem, it is possible to solve the serialization by using different methods like using the filesystem of the operating system, using the XMPP messages or using an HTTP service. For now, there are only three options supported in the PEAK-FL, the Filesystem Strategy, the XMPP Strategy and the HTTP Strategy.

For the Filesystem Strategy to be usable it is expected that the agents have access to the same filesystem. This can be done by running the agents in the same operating system on the same

machine, or it can be done by connecting the machines used to run the agents in a remote filesystem service. The way the Filesystem Strategy works is by using a common filesystem between the agents. The strategy will create a file with the model's information, save it on the filesystem and use the path of the file to send an XMPP message to the other agent. On the receiver side the strategy will receive the file path from the XMPP message and will read the file from the filesystem using that path, and in this way restore the original model's information.

The XMPP Strategy will use the XMPP messages to send the model. This strategy will only work with small models that can be fitted into an XMPP message (i.e., genetic programming individuals). In this case, the strategy will transform the model's information in a string format and put it directly in the XMPP message. The model's information is directly sent through the XMPP. The receiver will then use the strategy to transform the string to the original models' information. This technique can be used together with standard data formats like JSON.

The HTTP Strategy uses the HTTP protocol to serialize the model. PEAK has a functionality called the HTTP service which is used in this strategy. The HTTP protocol has several types of requests but for this case only two are required the PUT and the GET requests. The PUT method allows the client to create or substitute resources in the entity providing the service. The GET method is used to obtain a resource from the entity providing the service. In PEAK the XMPP server must have an HTTP service available, and the agent should request the service for two URLs, one for the GET and the other for the PUT. If accepted the server will return both URLs and the agent can use the PUT URL to persist a file in the server, and the GET to retrieve it. If the agent wants to share the file with another agent, the GET method should be sent to the respective agent. Regarding the strategy the agent that wants to send the model will use the strategy to save the model's information in a file and save it in the HTTP service. The strategy will then return the GET URL so that the agent can send it through the XMPP message. On the other side, the agent will receive the GET URL and will use the HTTP service to get the file based on that URL. After receiving the file, it will transform back to the original model's information.

4.3.5 Types of Learning Models Available

Given that to use the Flower implementation of the strategies it is necessary to adapt it to an asynchronous syntax, PEAK-FL already has some examples that can be used to train models in FL environments. For now, PEAK-FL provides the implementation for two different models: (i) artificial neural network, and (ii) genetic programming. They follow the same structure as a Flower Strategy but were written with asynchronous syntax.

Flower has implementations of many ANN Strategies, each one with different aggregation algorithms. The most common and most used is the Federated Averaging (FedAvg). For that reason, its adaptation PEAK-FL was made. In Python, there are only two keywords that are related to the asynchronous environment, which are the await and the async keywords. The async keyword is used in the declaration of functions. It says to the compiler that that specific function will return a coroutine. A coroutine is a function that can be suspended and resumed

several times during its execution, making it an asynchronous function. If in some part of the code, inside the asynchronous environment, there is a need to wait for a specific coroutine to run, the await keyword should be used. This await keyword will tell the program to suspend the current coroutine and start another one, this first coroutine will only be resumed after the coroutine created by the await keyword is finished.

A Flower basic Strategy has the following functions: initialize parameters, evaluate, configure fit, configure evaluate, aggregate fit and aggregate evaluate. PEAK-FL is expecting that these functions coming from the Strategy are coroutines. Because of that, all of them need to be defined as asynchronous functions by adding the async keyword before the function definition. Flower FedAvg, specifically, has two extra functions that do not need to be defined as asynchronous functions given that they do not call any coroutine inside. An additional adaptation was made inside the configure evaluate and the configure fit functions. As already mentioned in Section 4.3.2 Flower Integration, the client manager and the client proxy had to be adapted to the asynchronous syntax, generating the Agent Manager and Agent Proxy entities. These entities have the same functionalities as the original ones, but the difference is that their functions return coroutines. Because the strategy in the configure evaluate and configure fit functions uses those two entities, they need to add the await keyword before the call of the coroutines. In this implementation the logic of the strategy was not touched, only the syntax was adapted to be able to be executed in PEAK-FL. Almost every Strategy available in Flower can be adapted to PEAK-FL in the same way as this one. The key points are putting the async keyword at the beginning of the main functions and using the await keyword every time an asynchronous function is called.

To be able to integrate FL with GP, one strategy was made called Federated Tournament (FedTour). This strategy relies on the aggregation algorithms of the fit phase and the evaluation phase. Because GP is a very different algorithm than ANN, a new interaction between the federation members must be conceptualized. In a GP algorithm, a group of solutions is generated and then improved throughout several generations. Instead of the one solution as in the ANN models, GP has several. In addition to that, in ANN the neurons are represented by weights, linked through statistical formulas and the model structure is the same for every client. This forces the aggregation algorithms to be based on statistical and mathematical formulas to aggregate the values from the clients into one global model. In GP the approach must be different given that the solutions are formulas made by the combination of several functions, the input variables, and some constant values. Because the solutions can apprehend different kinds of structures, the values cannot be directly aggregated like in the ANN strategies. Instead, an evolutionary-inspired solution was developed for the GP aggregation algorithm.

In the GP field, there is a method called tournament selection for selecting individuals from a population (see Figure 12). Individuals from the population are selected to join in tournaments created randomly. The winners of each tournament are selected to perform a crossover for the next generation. Given this concept, an adaptation was used to conceptualize an FL approach to the GP algorithm. In this strategy, there is a concept of a hall of fame, which is an ordered list of the best individuals from a population. They are ordered by their performance as

solutions, which depends on the metric used to evaluate them. The hall of fame can be of any size, and it is configured by the user. In this strategy, there will be the local halls of fame and the global hall of fame. The local halls of fame are the ones maintained by the clients. They will update their halls whenever they generate or receive new individuals. The global hall of fame is in the server, and it will represent the best individuals, federation-wise. Another difference between the global and local halls of fame is the performance measures the individuals have. In the local halls of fame, the individuals will be evaluated only using the client's dataset. The individual's evaluation is from that specific client. The global hall of fame has the aggregation of evaluations from all the clients to each of the individuals.

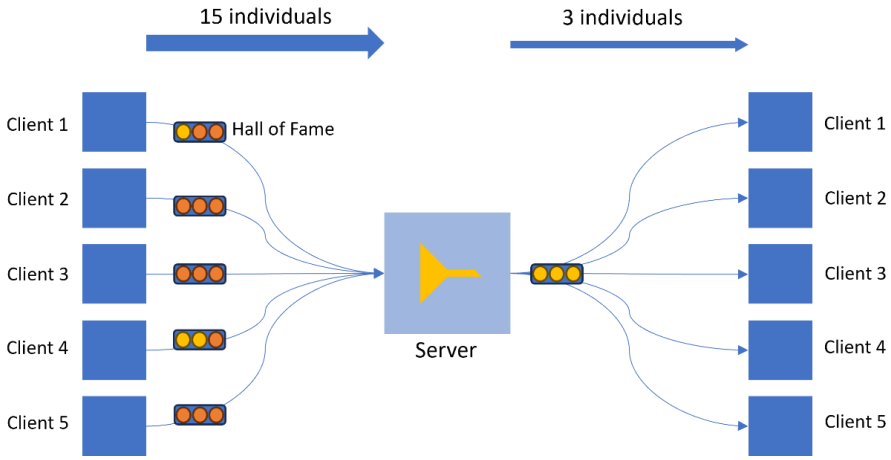


Figure 12 - Genetic programming strategy for the PEAK-FL.

The strategy has two different aggregation algorithms, one for the fit phase and the other for the evaluation phase. The strategy, before the first round, will ask the clients to initialize the first population of individuals locally. Then the first round will start by selecting the number of clients defined by the user. The selected clients will evolve the population using their local datasets. After they reach a predefined number of generations the GP algorithm stops. During the evolution, the local halls of fame will be constantly updated by the new individuals if they have better performance than any individual in the hall of fame. After the GP algorithm execution, the clients will send their local halls of fame to the server. The server after collecting every hall of fame will aggregate them simulating a big tournament, where all the individuals will compete against each other, using their performance calculated on the client side. After the competition, a new hall of fame is generated with the best individuals from them all. To evaluate this new hall of fame (not yet the global one), the server will send this hall of fame to the clients so they can evaluate it. During the evaluation, each client will evaluate the individual locally and can update their local halls of fame with those individuals. After the clients evaluate the individuals, they will send back the individuals with the respective results. In the server, after receiving all the results it will aggregate them. This algorithm will do a weighted average of the results. The weight means that it will have in mind the size of the dataset that each client, meaning that the higher the dataset the more influence will have on the result of the averaging. Having made the aggregation of the results, the server will update the individuals with the weight averaging results and it will update the global hall of fame with them. Only the best will

stay in the hall of fame. After that, a new round starts and this updated global hall of fame is passed to every client so they can add these individuals to their first population. If the individuals are not enough to fill the population new random individuals are created.

5 Case Studies

This chapter describes the case studies that were conducted to test and validate the proposed solution. Each section describes (i) what was studied in the case study, (ii) what was the environment of the test, (iii) which results were obtained after the execution of the case study, and (iv) what insights can be captured by analysing the results. Table 8 has an overview of the case studies conducted in this dissertation.

Table 8 - Description of the case studies.

Case Study	Main Objective	Brief Description
Federated learning with genetic programming model: an image classification case study	Accessing the federated learning mechanisms	It employs a genetic programming model in three federated learning settings and analyses the impact the characteristics of client's data have on their performance
Multi-agent system approach: an energy community modelling case study	Validating the flexibility and the modelling capacities of PEAK	PEAK is used to model three energy communities with different sizes and resources, and it is integrated with reinforcement learning and energy optimization models, and with one smart grid framework
Multi-agent system approach: a smart environment case study	Validating PEAK versatility and applicability	PEAK is used to model three hybrid simulation environments integrated with real smart devices and to develop a smart home application
Federated learning based on agent communications: an energy forecast case study	Validating PEAK-FL in modelling federated learning scenarios	PEAK-FL is tested in two federated learning scenarios, using a genetic programming model and a deep neural network
Federated learning within multi-agent systems: a dynamic mobility case study	Validating a new dynamic approach to federated learning	PEAK-FL is tested in a scenario where decision-making was added to the clients to allow their dynamic transition between federations and see the impact it has on their performance

5.1 Federated Learning with Genetic Programming: an Image Classification Case Study

The main objective of this case study is to explore the internal mechanisms of FL and how the clients work together to train a model by implementing a GP algorithm. Different scenarios were tested to see how the data size and the number of clients can affect the performance of the model. This case study is divided into three different parts The first part tests different numbers of clients and different dataset sizes and analyses their impact on the model. The second part tests the difference between even higher numbers of clients and if the presence of non-training clients impacts the model performance in any way. The third part studies the correlation between the dataset size and the number of good individuals created by the clients. This case study has contributed to the following published works:

- [Conference] **Bruno Ribeiro**, Luis Gomes, Zita Vale (2023) "A novel federated learning approach to enable distributed and collaborative genetic programming", presented and published in 22nd Portuguese Conference on Artificial Intelligence (EPIA 2023), DOI: 10.1007/978-3-031-49011-8_16

- [Conference] **Bruno Ribeiro**, Luis Gomes, Ricardo Faia, Zita Vale (2023) "Federated Genetic Programming: A Study About the Effects of Non-IID and Federation Size", presented and published in 20th International Conference on Distributed Computing and Artificial Intelligence (DCAI 2023), DOI: 10.1007/978-3-031-38333-5_20

- [Conference] Luis Gomes, **Bruno Ribeiro**, Fernando Lezama, Zita Vale (2023) "A Multi-Agent System Empowered by Federated Learning and Genetic Programming", published in 31st Signal Processing and Communications Applications Conference (SIU 2023) DOI: 10.1109/SIU59756.2023.10223778

The GP algorithm used was the FGP (see Section 3.1.4) and the configurations regarding the number of generations (10), and population size (100) are the same in each part of the case study. The rest of the parameters of the FGP were kept the same as in the original paper. The model is evaluated using the classification error rate (%) metric which counts for the percentage of images that were falsely classified. This case study uses three datasets the MNIST (Li Deng, 2012), the MNIST-ROT (Larochelle et al., 2007) and the FEMNIST (Caldas et al., 2018). They are all related to image classification problems and their difficulties are different from each other, respectively.

In the first part of the case study, three scenarios were designed to analyse the effects of different numbers of clients and different dataset sizes (see Table 9). It used MNIST and MNIST-ROT datasets in the three scenarios, making a total of six experiments. In scenario one there are three clients, and the sizes of the train and test datasets are 120 and 600, respectively. In scenario two there are five clients, and the sizes of the train and test datasets are 120 and 600, respectively. In scenario three the number of clients is five and the train and test datasets are

1,000 and 2,000, respectively. Each client has different batches from the original datasets and their data is considered non-IID.

Table 9 – Configuration of the scenarios in the case study.

	Scenario 1	Scenario 2	Scenario 3
Clients	3	5	5
Train set size	120	120	1,000
Test set size	600	600	2,000

The results in Figure 13 show the progression of the classification error rate (%) of the best individuals from the global hall of fame. The green lines correspond to the scenarios that used MNIST, the blue lines correspond to MNIST-ROT, and the red lines correspond to the accuracy of FGP in its original paper. The first thing to notice is that the model classified better the MNIST dataset than the MNIST-ROT expected. In both datasets scenario three has better results than the other scenarios, suggesting that the size of the dataset can have a great impact on the performance of the model in the federation. On the other hand, the change in the number of clients doesn't seem to have an impact on the results. One suggestion is that the small change in the number of clients was not sufficient for it to have an impact on the federation results.

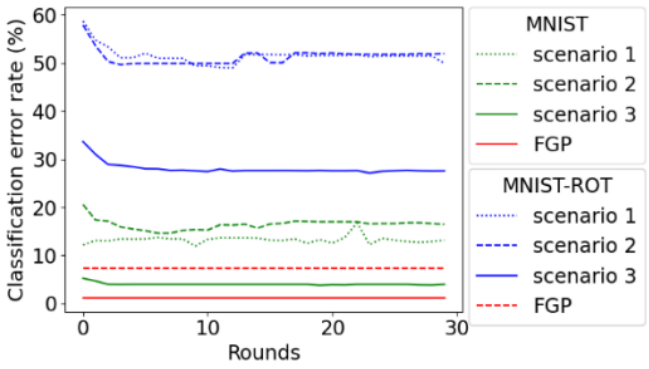


Figure 13 – Evolution of the error of the best individual in the server (Ribeiro, Gomes, & Vale, 2023).

The results in Figure 14 show the progression of the classification error rate (%) of the best individual of the global hall of fame, in the server, and of the local halls of fame, in the clients, in scenario one with the MNIST dataset. The green line corresponds to the server's individuals and the blue lines correspond to each of the clients' individuals. The red line is the accuracy of the FGP in its original paper. The clients have better results than the server. The individuals of the clients have the classification error rate based on their local dataset whereas the server individuals have their classification error rate based on the average of the classification error rate of every client. This indicates that the individuals being created were more suitable locally to the client who created them than to the other clients of the federation. The clients' results are closer to the FGP results. Probably with some adjustments to the solution it is possible to reach the FGP original results or even surpass them. Note that the FGP was tested on the complete datasets and it used the traditional centralized machine learning approach. Another interesting phenomenon is when the server has a peak, the clients have a valley. In the author's

interpretation, when the client produces an individual that has very good performance in the local dataset it is more likely to have worse performance in the datasets of the other clients, because the individual is tuned to that specific dataset, overfitting, and because the datasets are different in a non-IID manner, the individual loses performance on the other clients' datasets, rising the classification error rate in the server.

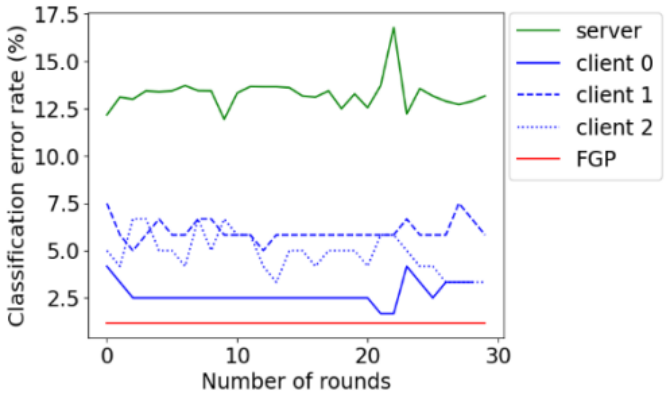


Figure 14 - Error evolution for each node in scenario one with MNIST (Ribeiro, Gomes, & Vale, 2023).

In the second part also three scenarios were elaborated to study the impact that a higher number of agents have on the FL system with a GP model, how the interactions and contributions occur during the federation execution and how the dataset size and the number of individuals shared correlate with each other. The dataset used was the FEMNIST. As shown in Table 10, in the first scenario only one client trained the model. In the second scenario there were ten clients in total but one of them did not train the model, only evaluated it. In the third scenario, there were fifty clients in total but five of them did not train the model, only evaluated it. The clients in scenario one participated in scenario two, and the clients in scenario two participated in scenario three. The clients that train the model have a dataset size higher than 50 instances and lower than 400. The clients that do not train have dataset sizes between 20 and 50. Each client corresponds to a different writer in the original MNIST dataset making the handwritten style unique to the client itself meaning the data is non-IID.

Table 10 - Description of three scenarios.

	Scenario 1	Scenario 2	Scenario 3
Total number of clients	1	10	50
Clients not able to train	0	1	5

The integration of the GP model in the FL system was similar to the first part of the case study but it was added two features, the individuals were signed by the client that created them and the local hall of fame of the client was inserted in the first population of individuals in each round. The sign allowed to track the contributions being made from one client to another through the federation.

Figure 15 presents a Sankey graph used to compare the number of individuals that were shared between the clients from the left (authors of the individuals) and the clients from the right

(receivers of the individuals) from the third scenario. The width of the lines represents the number of individuals that were shared between them, the wider the line the more individuals contributed. Not all clients appear on the left side of the graph. This suggests that not all clients could create sufficiently good individuals that could be used by the other clients. Every client on the left side contributed to more than one client from the right. This means that the individuals produced by the clients on the left had more capacity for generalization, being able to have better performance in the other clients' datasets. It also suggests that the federation is essential for the clients to collaborate and benefit from the solutions from each other. On the right side of the graph, the clients that received more contributions were the clients that did not train. This was expected since the clients who did not train had to depend on the rest of the clients to be able to solve their specific classification problem.

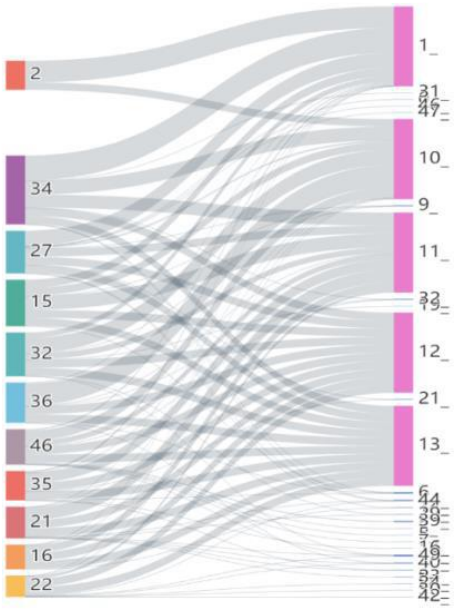


Figure 15 –Contributions made from senders (left) to receivers (right) (Ribeiro, Gomes, Faia, et al., 2023).

To calculate the correlation between the dataset size of the clients and the number of individuals contributed, it was used Spearman's rank correlation coefficient (Figure 16). For these calculations, was used scenario three, which had more clients, and only the clients that trained were considered. In Figure 16.a it was only considered the client-to-client contributions, which are the contributions that ended up being used by other clients. In Figure 16.b it was only considered the client-to-server contributions, which are the contributions that ended up in the global hall of fame. Regarding the client-to-client contributions, the p-value (0.012) is lower than 0.05 meaning that the deviation from the null hypothesis is statistically significant, and the null hypothesis is rejected. In other words, there is a good indication that these two variables have a low correlation, given that the correlation (0.371) is lower than 0.5. Regarding the client-to-server contributions, the p-value (0.001) is lower than 0.05 meaning that the deviation from the null hypothesis is statistically significant, and the null hypothesis is rejected. In other words,

there is a good indication that these two variables are moderately correlated, given that the correlation (0.49) is very close to 0.5.

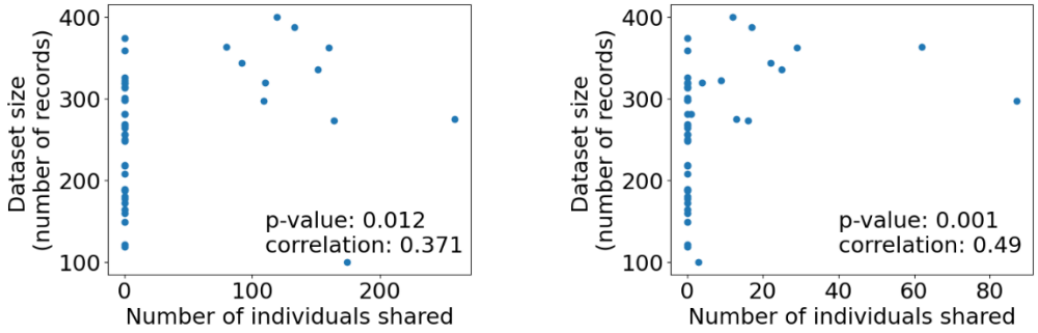


Figure 16 - Dataset size and individual contributions correlation (Ribeiro, Gomes, Faia, et al., 2023).

The client-to-client contributions not only have to pass through the global hall of fame to get to the clients but also must be tested against the local hall of fame of the client that is receiving it. That is the reason why the correlation is lower than the correlation of the server-to-client contributions. When an individual is tested with more variety of instances, and it has good results it becomes more resilient and makes it more likely to be useful for other clients as well. This correlation can be influenced by the type of aggregation that is being made in the server. In this case, the algorithm used to calculate the performance measurements of the individuals in the server is the weighted average of the classification error rate (%).

In this third part of the case study, the idea was to test if adding the local halls of fame as part of the initial population at the beginning of each training phase would allow a better converging of the model. In this part, 10 clients were used. They had access to limited-sized datasets, with the number of instances between 100 and 300. The dataset used was MNIST. In total, the number of instances used by the whole federation was 2954, which corresponds to 0.05% of the original dataset. One of the clients did not train the model and only participated in the evaluation phase. The MNIST dataset is then divided in a non-IID manner between the clients. In this FL environment, the clients trained the model for 10 rounds. The model used for the image classification problem was FGP.

Figure 17 shows the evolution of the error classification rate (%) of the individuals in the hall of fame of each client. In this simulation, every client converged to a better error classification rate (%). Everyone had a decrease in error rate with the help of the federation, showing that all agents benefit from it. We can also see that the results are different for each client, meaning that some clients benefited more than others from the federation. For instance, Client 2 had an error classification rate at the end of almost 15% and at the top, we have clients 0, 5, and 6 that had nearly 25%. The performance of the algorithm has space for improvement given that the original work of the FGP is 10% better. The FL approach can probably achieve the same results by finetuning the GP parameters according to the FL configurations, improving the aggregation

algorithm to allow the individuals with the best features to be used by the federation and increasing the dataset size of the federation.

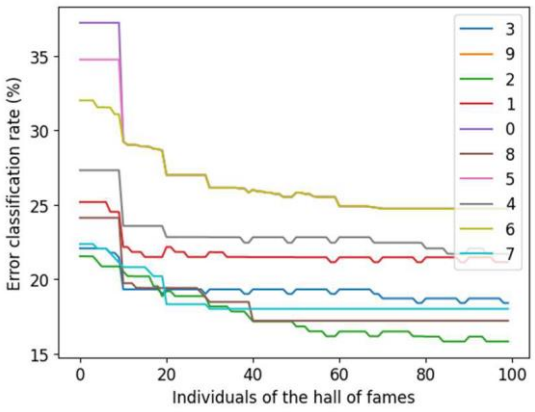


Figure 17 – Error evolution of the halls of fame on the client side (Gomes et al., 2023).

Overall, in this case study it is possible to reach the following conclusions: the local evaluation of the individuals tends to have better results than the global ones; there is evidence that shows the dataset size is moderately correlated with the number of contributions made by the clients; and, the insertion of the local hall of fame in the initial population of each training phase enabled the clients to converge the solutions more effectively throughout the rounds. There is potential in combining federated learning with genetic programming, but further research needs to be done like fine-tuning the FGP better to FL environments and creating new aggregation algorithms to provide better generalization capacities in the individuals.

5.2 Multi-Agent System Approach: an Energy Community Modelling Case Study

In this case study two scenarios were conceptualized to validate the PEAK framework in regards to modelling complex systems. In the first scenario, PEAK was used to model and simulate a large community with 250 members with access to photovoltaic energy generation, batteries, electric vehicles and an optimization algorithm that allowed the community to reduce energy costs. In the second scenario, PEAK was used to model an energy community which adds access to a peer-to-peer (P2P) energy market, forecast services and reinforcement learning algorithms to enable automated decision-making and learning. Also, in the second scenario, PEAK was used as the core of another framework that aims to model smart grid systems based on agents. PEAK Dashboard was used to visualize and monitor the data being gathered in real time in the case study. This case study contributed to the publication of the following two papers:

- [Journal] Ricardo Faia, **Bruno Ribeiro**, Calvin Gonçalves, Luis Gomes, Zita Vale (2023) “Multi-agent based energy community cost optimization considering high electric vehicles penetration” published in Sustainable Energy Technologies and Assessments (SETA), DOI: 10.1016/j.seta.2023.103402

- [Journal] Helder Pereira, **Bruno Ribeiro**, Luis Gomes, Zita Vale (2022) “Smart Grid Ecosystem modelling using a novel framework for heterogeneous agent communities” published in Sustainability, DOI: 10.3390/su142315983

In the first scenario, the main idea was to minimize the energy costs of the day ahead. The energy community has load consumption, photovoltaic energy generation, battery energy storage systems and electric vehicles and has a total of 250 members, 200 of which have photovoltaic generation and 150 have battery energy storage systems. Each community member has two electric vehicles, one is a regular model, and the other is a premium model. All the vehicles have the capability of feeding energy to the grid and the households. The departure and arrival hours of the electric vehicles are between [08:15-11:30] and [19:00-22:00] respectively. The electricity prices for selling photovoltaic energy were determined by the feed-in tariff (45 EUR/MWh) and for buying energy it was considered tri-hourly low voltage tariffs used in Portugal (0.3271; 0.1788; 0.1132).

The multi-agent system contemplates 250 agents representing each member of the energy community, and one agent as community manager. The community members will initially read their respective data regarding the energy consumption and generation, the batteries, and the electric vehicles from an Excel file. After gathering the data, they sent them to the community manager who is going to aggregate all the community’s data and run the two optimization algorithms made by Ricardo Faia. The first one will calculate the best decisions to be made for the day ahead regarding how to spend the energy available. Decisions regarding how much and when to buy and sell energy, how and when to use the batteries and the electric vehicles to reduce the energy costs for the next day. After that scheduling is done, another algorithm is executed to optimize the net load and reduce even further the energy costs by trying to raise energy consumption during lower energy prices and reduce energy consumption during higher energy prices.

In Figure 18 the results generated by the first optimization algorithm are shown. Figure 18.a shows the energy import and export of the overall community and indicates that in the first hours of the day, there is a huge energy import from the grid from consumers' homes. That is related to the charging of the electric vehicles shown in Figure 18.b. The sudden fall of the energy peak of energy consumption must be because that is the hour most people use to go to work, usually by car, that is why that peak matches almost accurately the peak of the electric vehicle charge at home. Another interesting fact is that all the energy discharged from electric vehicles during the day is consumed by the houses. Figure 19 shows the graphs of Figure 18 being produced in the PEAK Dashboard by the agents.

Figure 20 shows the results of the second optimization algorithm which are the forecasted energy usage for the day ahead, the line in blue, and the optimized energy usage, the line in black. The optimization algorithm increased the energy consumption in the time interval [11:00 - 13:30] and reduced it in the time interval [19:30-23:30]. Given that the energy is more expensive during the [19:30-23:30] interval, the optimization algorithm raised the energy when it was cheaper. The energy cost without optimization was 758.433 EUR, and after the

optimization was 711.88 EUR. This allowed the energy community to save a total of 46.553 EUR. Figure 21 shows the same graph but displayed in the PEAK Dashboard.

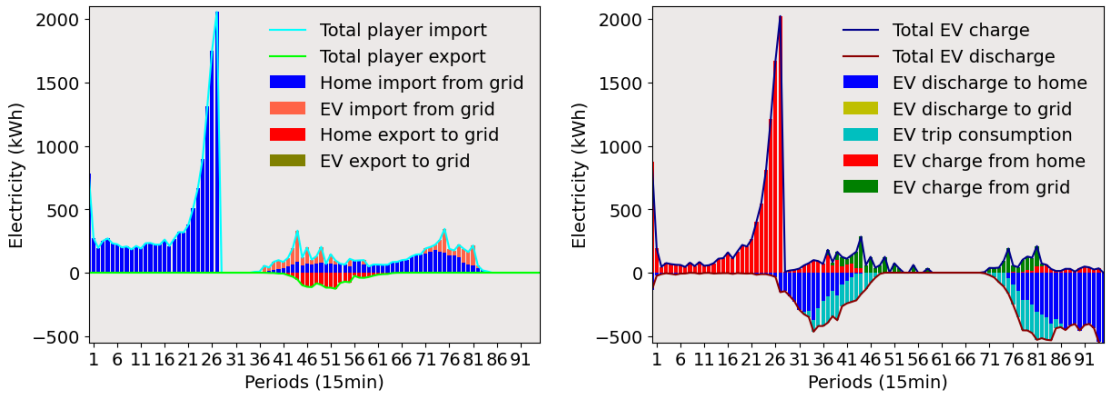


Figure 18 –Results of the optimization for the energy community (Faia et al., 2023).

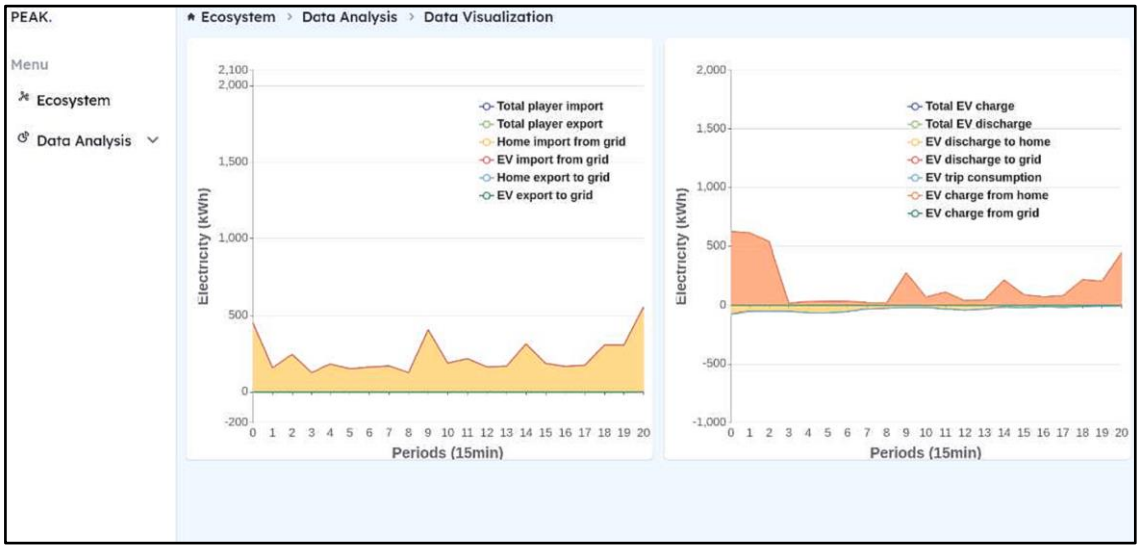


Figure 19 – Print of PEAK Dashboard showing energy optimization (Faia et al., 2023).

For the second scenario, it was added more dynamism and decision-making capabilities in the agents. The energy community has now access to energy forecast services and the energy market, enabling the community members to negotiate energy. The idea of the simulation is for the community members to reduce their energy costs by learning to manage their energy usage, choosing the right services for them, and learning the best way to buy and sell energy in the market. In addition to that, more functionalities were developed on top of PEAK originating another framework called Agent-based ecosystem for Smart-Grid modelling (A4SG). This framework adds virtual and physical agent mobility, to allow a more dedicated resource management, and a branching mechanism to allow agents to pursue their individual goals more intentionally. A4SG focus on modelling smart grid systems based on agents enabling the representation of several entities like groups of aggregators and their customers, energy communities and forecast services. This scenario tests and validates the resilience, flexibility,

and extensibility of PEAK by modelling a more complex scenario and by being used as a base for another framework.

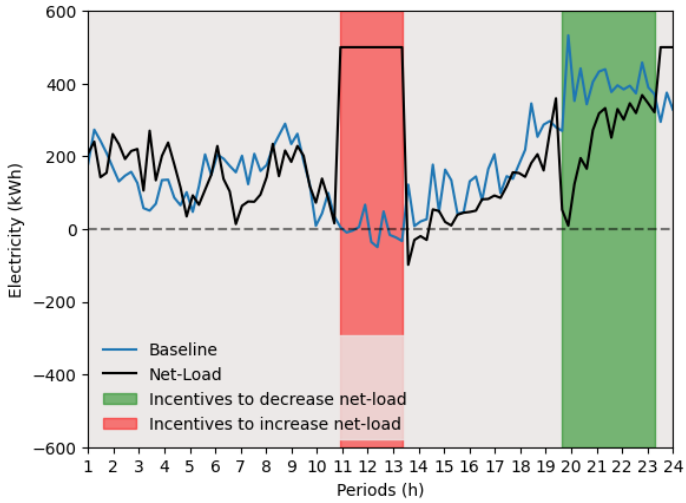


Figure 20 - Energy optimization of scheduling made for the day ahead.

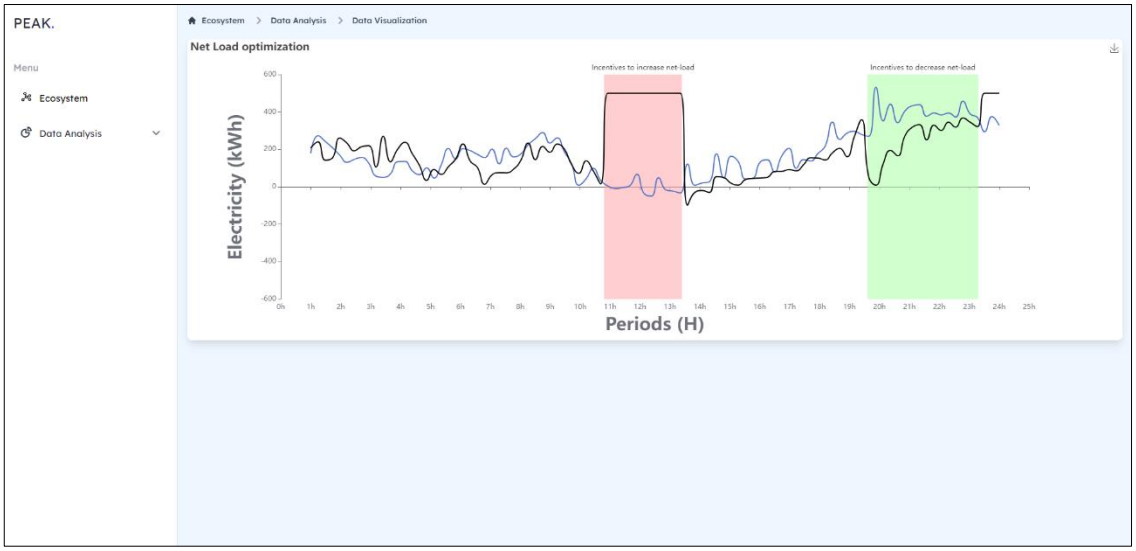


Figure 21 - Print screen of the PEAK Dashboard with the optimization graph.

The energy community model is composed of 20 prosumers who have access to a peer-to-peer energy trading market and weather forecast services. The peer-to-peer market is an auction-type market, where the participants can bid for certain amounts of energy. All the energy transactions were made between the members of the energy community. They could also choose from three weather forecast services with different error profiles. The forecast services are a forecast baseline, based only on mathematical models, a forecast based on a support vector machines algorithm (SVM) and a forecast service based on an ANN. The community members should choose which service best suits their energy usage profile. Each community member represents a regular household with photovoltaic energy generation, and six of them

have access to battery energy storage systems. The battery owners can use the batteries in the post-market phase to reduce costs related to the purchase and sale of energy.

To model the energy community, the A4SG organizes the agents in different MAS and different agent communities (ACOM). In A4SG each MAS represents a different host machine where the agents are running and inside each MAS ACOMs represent the services, communities, and responsibilities of the agents. In this scenario, there are two MASs, one where the energy community is running and another MAS where the forecast services are running. For each MAS and ACOM, there is an agent responsible for managing everything inside it, like allowing other agents to join or not and providing information to outside agents if needed. Each MAS has also a main community, the Main ACOM, where the main agents are. Inside the other ACOMS are the branching agents created by the main agents. The branching agents do not have to be necessarily inside the same MAS, they can be instantiated in ACOMs inside other MAS. The Ecosystem structure of this scenario can be seen in Figure 22. The agent in blue is the mobility agent that manages the virtual and physical mobility of the agents.

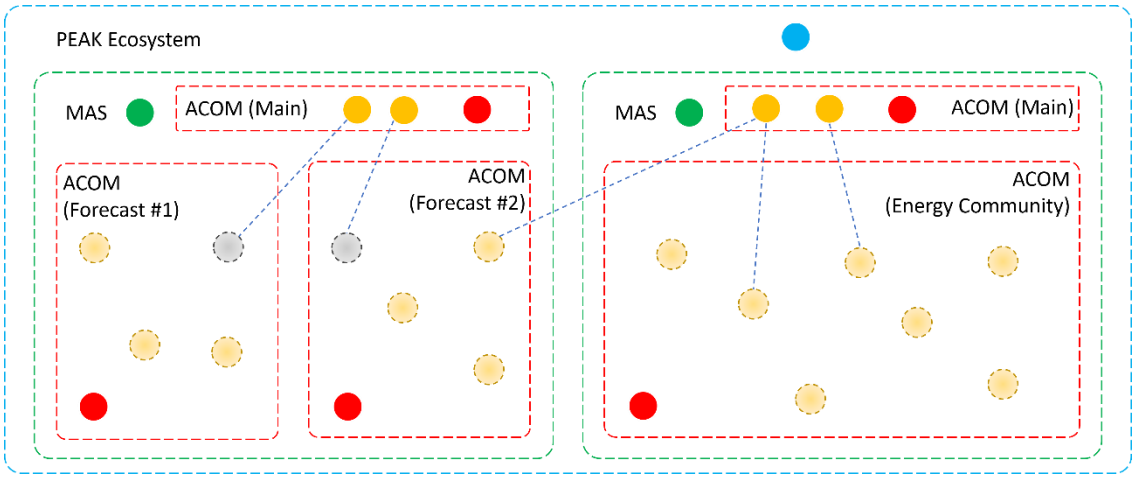


Figure 22 – A4SG ecosystem organization (Pereira et al., 2022).

This scenario simulates three weeks of energy transactions and usage. In the beginning, the members will be distributed randomly for each forecast service. For each week there will be firstly the peer-to-peer simulation of the energy market. After the market phase, the community members will evaluate the performance of the services they are currently using, by calculating the mean absolute error (MAE), and compare it with other services. Then they can decide to change services if they see potential improvement in their performance by doing it. This process is repeated for each week. For the market phase, the agents are provided with two different reinforcement learning algorithms so they can learn how to bid and therefore decrease their energy costs. The algorithms are the Deep Deterministic Policy Gradient and the Twin Delayed Deep Deterministic Policy Gradient. There is also one agent representing the market operator, that is responsible for managing the market operations.

Table 11 shows the distribution of the community members among the forecast services throughout the three weeks of the simulation and the respective average MAEs. The evolution

of the number of customers using each service each week shows that the agents were using their reasoning and virtual mobility to pursue better their energy forecast needs. The baseline forecast service began the simulation with eight customers, losing more than half costumer in the second week, and finishing the simulation with no customers at all. This demonstrates that this service was the worst for forecasting, and it was not useful for any customer. For the other two services it is shown that ANN is better than SVM service given the amount of customers using it, however, for some clients SVM is still better. Regarding the forecast services performance, the MAE weight average for each week reduced from 0.741 to 0.389.

Table 11 - Distribution of the community among the forecast services during the simulation.

	Week 1			**Week 2**			**Week 3**		
	Baseline	SVM	ANN	Baseline	SVM	ANN	Baseline	SVM	ANN
Number of Customers	8	6	6	3	6	11	0	5	15
Average MAE	1.036	0.650	0.438	0.627	0.513	0.401	-	0.459	0.366

Figure 23 shows a visual representation of the customers' distribution among the forecast services. The figure was extracted and adapted from the PEAK Dashboard, and it represents the ecosystem structure throughout the simulation. Figure 23.a is the MAS of the energy community with the two PEAK sub-communities. Figure 23.b is the forecast service MAS showing the distribution of agents among the forecast providers in week one. Lastly, Figure 23.c shows the distribution of agents among the forecast providers in week three.

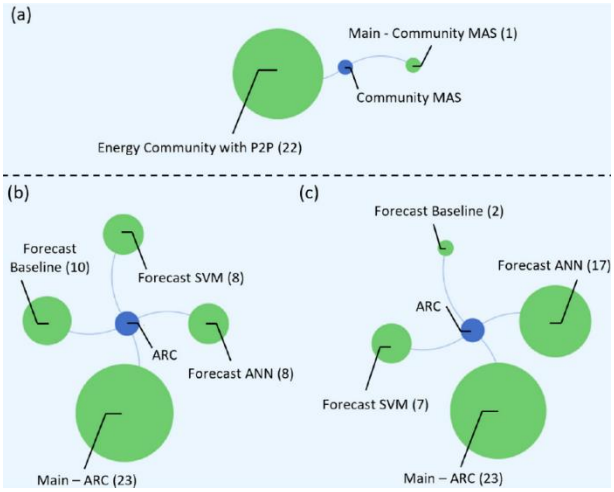


Figure 23 - PEAK Dashboard showing the ecosystem structure (Pereira et al., 2022).

Figure 24 shows the percentage of savings in weeks two and three when compared to the first week. Overall, 90% of the community reduced energy costs by the end of the three weeks by taking advantage of the available forecast services. Only members three and seven had a slight waste of money. In total, the community saved 10.62 EUR in these three weeks. Figure 25 shows the amount of load used from the ESS on each day of each week. The ESS was used only in the

post-market phase to minimize the amount of energy bought and sold, which means that the more accurate the bids in the market were the lower the need to use the ESS. That is reflected in the amount of decrease in ESS from the first week to the third one reaching a 22% decrease.

In this case study PEAK was validated in modelling energy communities with different characteristics and levels of complexity. Its flexibility and adaptation capacities were tested by modelling different roles and integrating mathematical and machine-learning models in agents. Additionally, it shows PEAK's extensibility by using it as a base for another framework. PEAK Dashboard also showed its advantages for developing MAS enabling a live monitorization of the ecosystem and allowing efficient analysis and debugging of errors. This case study demonstrated that PEAK is a reliable and multitasking framework that can be used to implement, conceptualize, and validate complex systems.

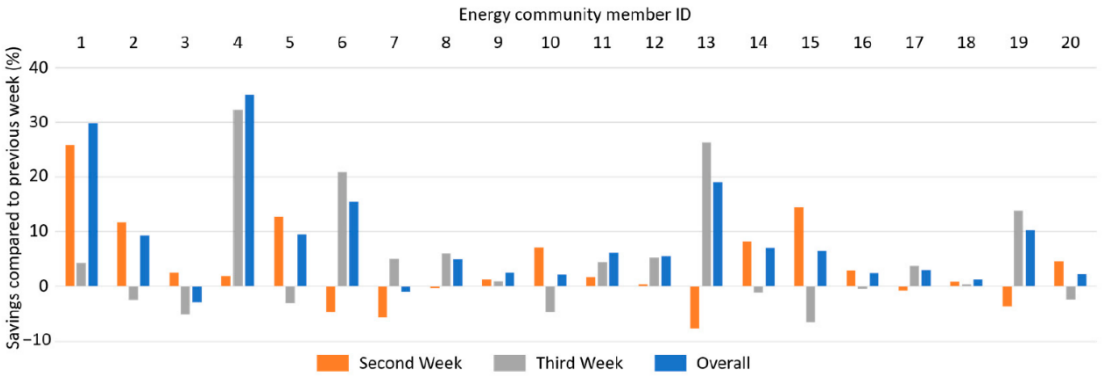


Figure 24 - Energy community savings compared to the first week (Pereira et al., 2022).

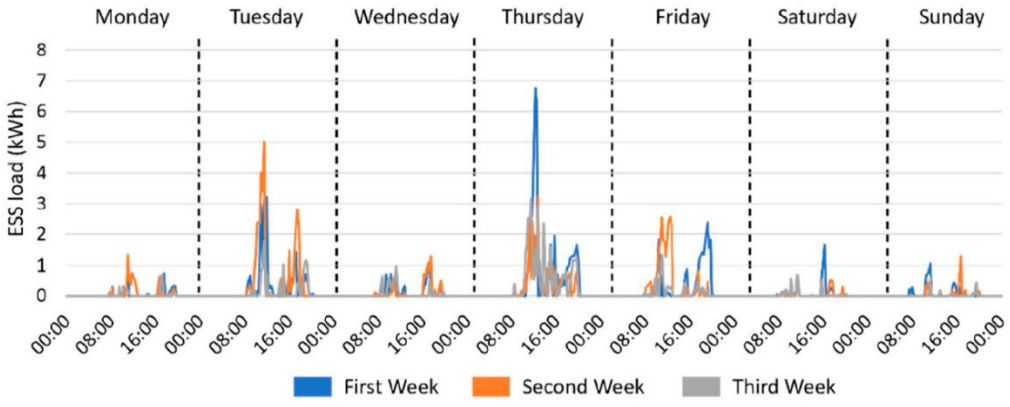


Figure 25 - Energy storage system load comparison between weeks (Pereira et al., 2022).

5.3 Multi-Agent System Approach: a Smart Environment Case Study

The main objective of this case study is to validate the flexibility and extensibility of PEAK and test if it can integrate different smart devices into the system. PEAK was integrated with

batteries, photovoltaic energy generation devices, office lights and consumption and generation monitorisation devices. In addition to that energy models and natural language processing models were also integrated. This case study has three different scenarios. In the first scenario, PEAK was used to implement and simulate an energy community with access to real batteries without any optimization algorithm. The second scenario a more complex energy community was modelled also with access to real batteries, but this time with an optimization algorithm to control them. Finally, in the third scenario, PEAK was used to implement and deploy a smart home application based on agents to manage the energy resources of a real building. This case study contributed to the following published papers:

- [Journal] Cátia Silva, Pedro Faria, **Bruno Ribeiro**, Luis Gomes, Zita Vale (2022) "Demand Response Contextual Remuneration of Prosumers with Distributed Storage" published in Sensors, DOI: 10.3390/s22228877

- [Conference] **Bruno Ribeiro**, Helder Pereira, Luis Gomes, Zita Vale (2022) "Python-based Ecosystem for Agent Communities simulation", published in 17th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2022), DOI: 10.1007/978-3-031-18050-7_7

- [Conference] **Bruno Ribeiro**, Ricardo Faia, Luis Gomes, Zita Vale (2023) "Energy Community Integration of a Smart Home Based on an Open Source Multiagent System", presented and published in The PAAMS Collection: Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. (PAAMS Demonstrations 2023), DOI: 10.1007/978-3-031-37616-0_35

In all three scenarios, it was used physical batteries that are available in the labs of building N in the GECAD office. There are in total six batteries, and it is possible to communicate with them by a protocol called Modbus/TCP. Three of the batteries have a capacity of 2.4 kWh and the other three have 3.6 kWh. The battery system has a security mechanism that does not allow the battery to empty nor allow it to charge or discharge at dangerous rates. Otherwise, the batteries would be damaged, and they would be at risk of creating serious problems.

In the first scenario, a simple energy community was simulated. The idea is to validate PEAK's capacity to simulate hybrid environments where part of the data is gathered from smart devices in real time. The objective of this scenario is to see if the use of batteries, even without any optimization algorithm, can be useful to reduce energy costs.

The energy community has 50 consumers, 15 of whom are prosumers with renewable energy resources. One of the consumers represents a public library while the others are regular households. Six of the households have battery energy storage systems, which start at full capacity and are connected to real batteries. The batteries will help them reduce energy costs by using their energy to reduce the energy bought from the market. In addition to the 50 consumers, there is a community manager who is responsible for calculating the energy consumption, generation, and costs of the whole community and publishing the results. The

scenario was simulated for one day and it was divided by periods of 15 minutes. In the beginning, the community members will get their data from an Excel file and at each period they will send the energy consumption and generation, and the battery usage data to the community manager. The community manager will aggregate the data and calculate the total energy usage of the community for that period by subtracting the generation from the consumption. If there is more consumption than generation the community manager will buy energy. Otherwise, it will sell it.

Figure 26 shows the purchase and sale prices, as well as the energy profile of the energy community for the simulated day. The sale prices used are from the MIBEL market and correspond to the day of 10th March 2022. The purchase prices are the result of the sale prices multiplied by 1.5 to account for the distribution company's profit. The prices for this day are higher at the wake-up time and at dinner time.

Figure 26 - Energy profile of the community and the market prices (Ribeiro, Pereira, Gomes, et al., 2023).

Figure 27 shows the difference the battery energy storage system has on the energy costs of a consumer. The consumer shown had less energy costs when using batteries. The battery was used from the beginning of the simulation until hour six, when it reached the minimum state of charge, 25%. This shows that even without optimization of the battery, it can be profitable to use one.

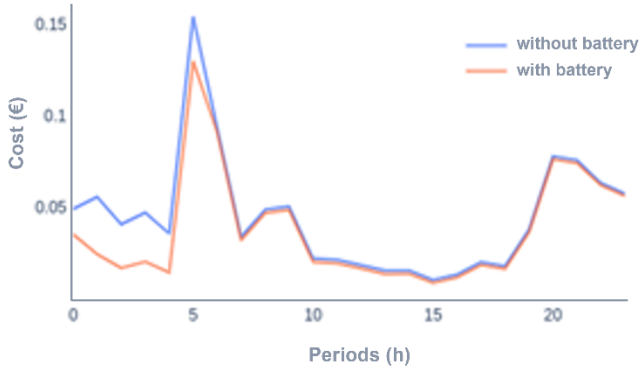


Figure 27 –One consumer's costs with and without battery (Ribeiro, Pereira, Gomes, et al., 2023).

In the second scenario, PEAK was used to simulate an energy community with access to photovoltaic generation and batteries. In this scenario, a simple demand and response approach was modelled. Demand and response events are, in simple terms, events that aim to balance the energy demand in power grids by encouraging the consumers to shift their energy consumption to the time when the energy production is higher, so there is less waste of energy. The encouragement is normally made monetarily. The optimization algorithm will allow the consumers to shift their energy consumption to receive more remuneration and have fewer energy costs.

In this scenario's community, there are 19 prosumers with access to battery energy storage systems and photovoltaic energy generation. They can suppress their loads or sell the excess to the grid using those resources. Five of those prosumers had real batteries connected to the simulation. It is assumed in the simulation that the batteries have all 2 kWh capacity. To map the simulated batteries to the real ones, given that the capacities are not equal, it was used the state of charge of the simulated batteries and the maximum and minimum charge and discharge limits of the real batteries. The tariffs used for the scenario are tri hourly as shown in Figure 28. The values applied in the tariffs are considered by the main Portugal DSO. Only prosumers with contracted power below 20.7 kW were considered for this scenario. Each agent is responsible for the optimization of its resources.

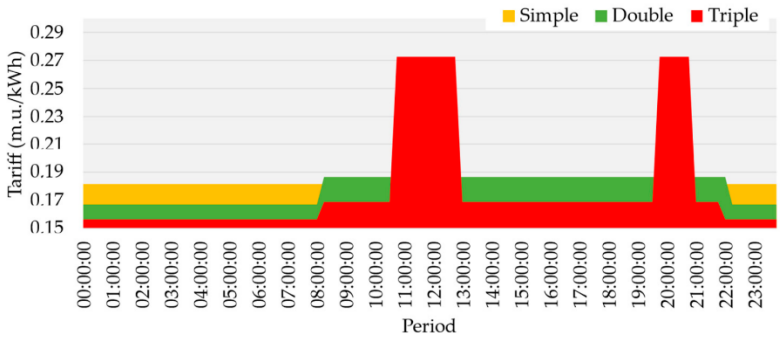
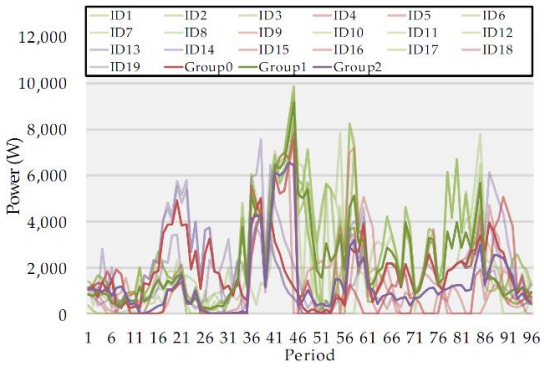


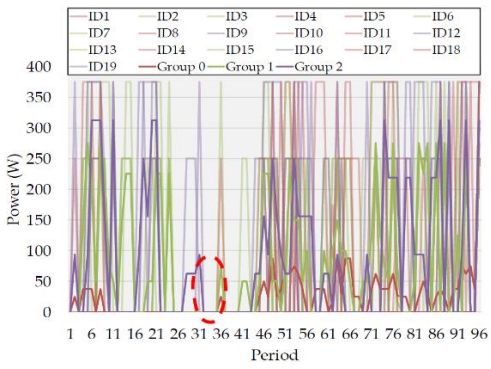
Figure 28 - Schedule for the tariff used (C. Silva et al., 2022).

For the MAS there were 21 agents, 19 represent the members of the energy community, 1 is the community manager and the other is the synchronizer. In the beginning, the agents would join the same group so the simulation could start, and the synchronizer could synchronize them all. The community members first read their datasets and for each period (15 minutes) the agent ran the model and sent the results to the community manager. The community manager would then aggregate the results and publish them to the community. The values regarding the battery would be used directly in the battery to charge and discharge. The data from the battery would be used in the optimization model so it could calculate every period what to do with the battery for the next period. Five of the consumers were connected to real batteries to see the impact of the algorithm when applied to real devices. The simulated batteries were considered to have a 2 kWh capacity while the real batteries have 2.4 kWh and 3.6 kWh. The mapping of the simulated batteries to the real ones was done by using the state of charge in percentage and the maximum and minimum charge and discharge rate of the real batteries.

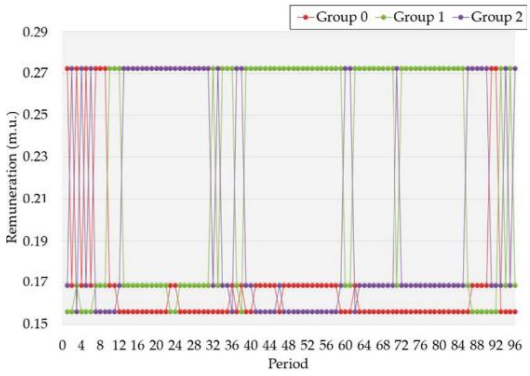
In Figure 29 it can be seen the flexibility of the consumers and the discharge of the battery scheduled throughout the day and the respective remunerations. To facilitate the analysis, it was used k-means clustering method to aggregate consumers with similar load profiles. For the prosumer flexibility, it is possible to see that in the three groups they have more flexibility during the lunch hours which is exactly when the energy price is higher. For battery usage, it is possible to see several requests for discharging the batteries throughout the day. The load flexibility and the BES discharge were attributed to monetary remuneration which can be seen in Figure 29.c and Figure 29.d. The higher the flexibility or the discharge the higher the remuneration value as well.



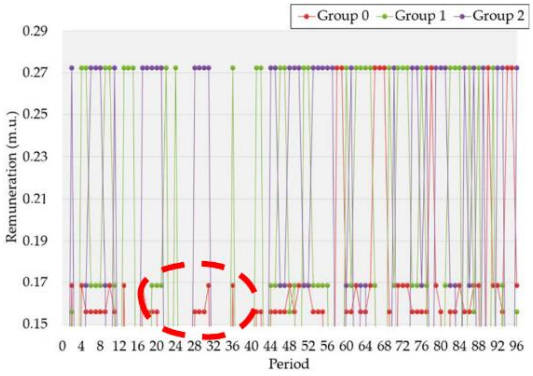
(a) Prosumers' flexibility



(b) BES discharge



(c) Prosumers' flexibility remuneration



(d) BES discharge remuneration

Figure 29 – Prosumers' flexibility and BES discharge and their remuneration (C. Silva et al., 2022).

Figure 30 compares the real and simulated battery charges and discharges during the execution of the scenario. Despite the values not being exactly equal they followed the same trend throughout the execution. They stared at full capacity, and they were promptly discharged until they reached their minimum capacity, 25%. In all batteries, between the periods 42 and 62, there were requests to charge the battery that were followed by the real battery, except for battery 4. One reason for that difference is that the real battery does not always have a fixed maximum charge and discharge rates because they have a security mechanism that does not allow excessive charge or discharge depending on the state of charge of the battery and on the voltage being read from the grid.

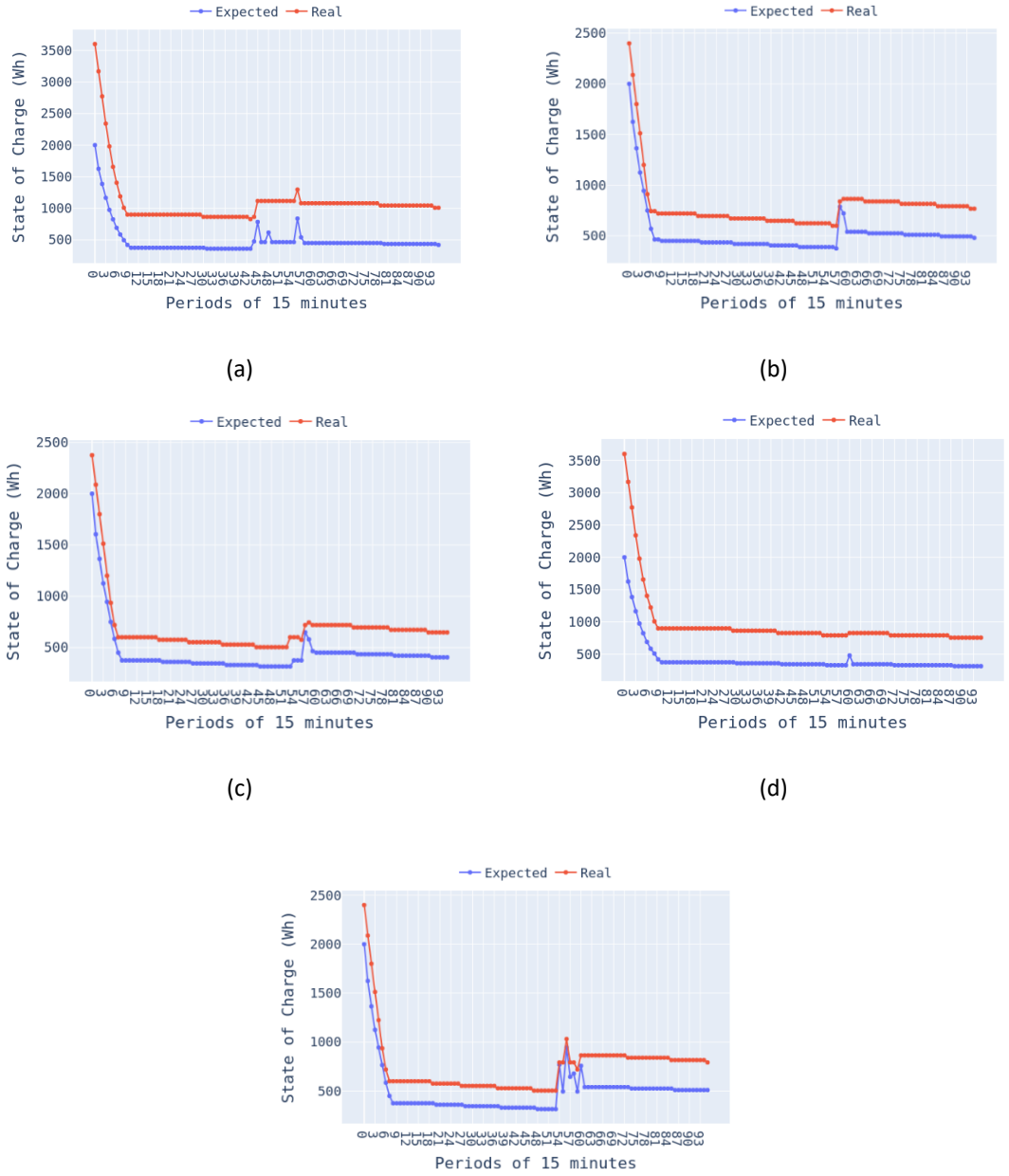


Figure 30 - Comparison between simulated and real battery usage.

In the third part of this case study, PEAK was used to develop a smart home system based on MAS that controlled and managed a real building. The building has devices that control and monitor the battery energy storage system, the total consumption and production of energy and the light intensity of a room. The MAS was also integrated with an energy optimization model, to allow the user to see how to best schedule the use of the devices so that the energy cost can be minimized. To interact with the smart home system remotely, the user must talk with the AI chatbot via text. The smart devices of the building were represented by single agents, as shown in Figure 31. One agent for the lights, another for the battery and the other for the

consumption and production of energy for the building. Two other agents were used, one that was responsible for the energy optimization model and the other for the human-machine interface, the assistant, that had the natural language model integrated into it. The data generated by the MAS was all published in the PEAK Dashboard in real time.

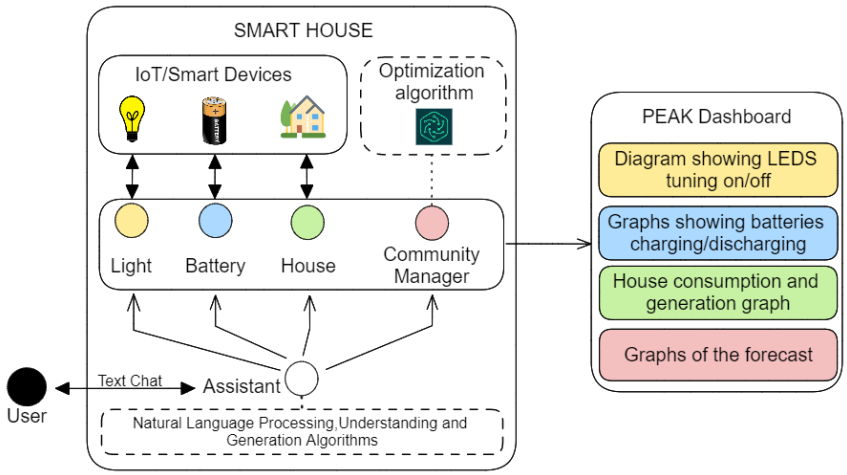


Figure 31 - Architecture of the smart home app (Ribeiro, Faia, Gomes, et al., 2023).

The user could ask for the bot to perform certain tasks regarding the monitorization and management of the building. Regarding the lights, the user can turn on and off the lights and adjust the intensity of the lights. Regarding the battery, the user can charge and discharge the battery according to his needs. The user could also ask the bot to provide the real-time energy consumption and production of the home in watts. If the user wanted to run the optimization algorithm or update the data related to it the user just needs to ask the bot to run the optimization algorithm. This was all possible by communicating via text with the assistant chatbot using an XMPP client called Prosody.

In the system, there was a natural language model and an energy optimization model. The natural language model was implemented with Rasa, which provided a pre-trained natural language model. A dataset was created, with the help of Chat-GPT, with examples of instructions to fine-tune the model. The energy optimization model is a mixed integer linear programming algorithm implemented using the MIP Python library using the free solver coin-or branch and cut (CBC). The algorithm will receive as input the forecasted consumption including the lights and other loads of the building, photovoltaic generation, the maximum of charge, discharge and total battery power, market prices to buy and sell energy and the limits of the import and export energy to the building. The model output was calculated for 15-minute periods, and it encompassed the exported and imported energy from the grid, the charge and discharge values of the battery and the costs of the operations (i.e., buy, sell and the costs at the end of the day).

Figure 32 shows the PEAK Dashboard in the Ecosystem view where it is possible to see the structure of the MAS deployed. There is a group called smart house which the agents are part of. Each of the agents, the assistant, the battery agent, the lights agent, the energy consumption,

and production agent (house agent) and the energy optimization algorithm agent (community manager) all joined the group.

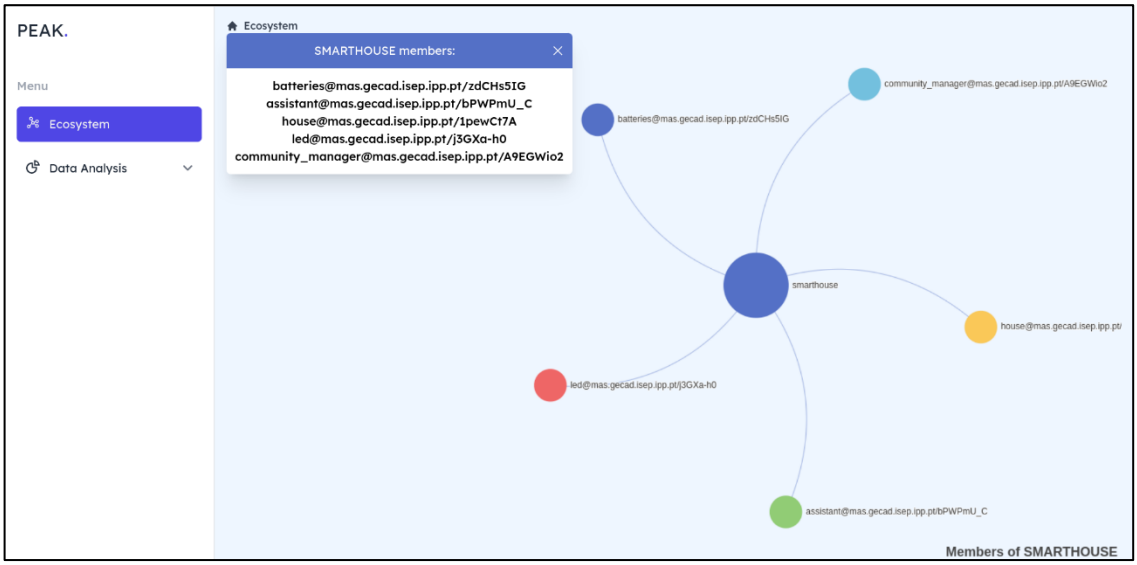


Figure 32 – Print of PEAK Dashboard showing the smart home agents (Ribeiro, Faia, Gomes, et al., 2023).

Figure 33 and Figure 34 are prints of the PEAK Dashboard showing the data being monitored in real-time. There is the lights graph showing if the light is off or on and its intensity, the graphs showing the consumption and production of the home in real-time, the battery state-of-charge and the graphs showing the recommended operations for an optimized use of the smart devices present in the smart home according to the current state of each device.

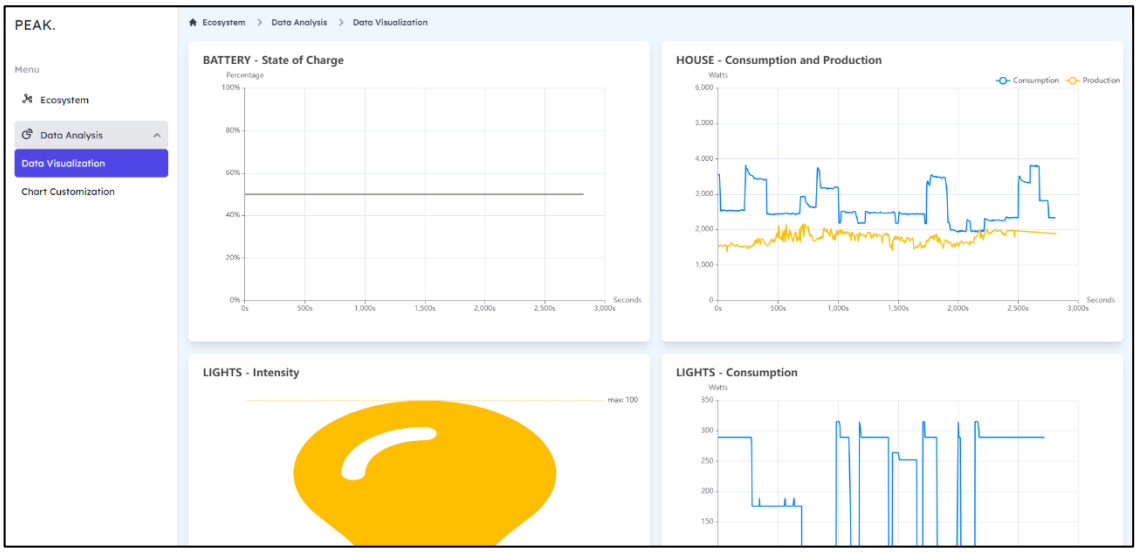


Figure 33 - Print of PEAK Dashboard showing the agent's data (Ribeiro, Faia, Gomes, et al., 2023).

This case study showed and validated the capabilities of PEAK to integrate and implement different scenarios and components to develop complex systems. It allowed the simulation of

energy communities where the agents represented different prosumers, and the integration of complex mathematical models to optimize the energy costs of the community. It also showed the feasibility of PEAK to model energy communities using real devices and map the simulated devices from the simulation to the real devices. This helps test a variety of systems while assuring consistency. PEAK also allowed the deployment of a real MAS into a smart building to help the remote management of the smart devices present in the home. In addition, an energy optimization model and a natural language model were integrated into the system. This case study demonstrated that PEAK is a reliable framework that handles multitasking and can be integrated with real smart environments.

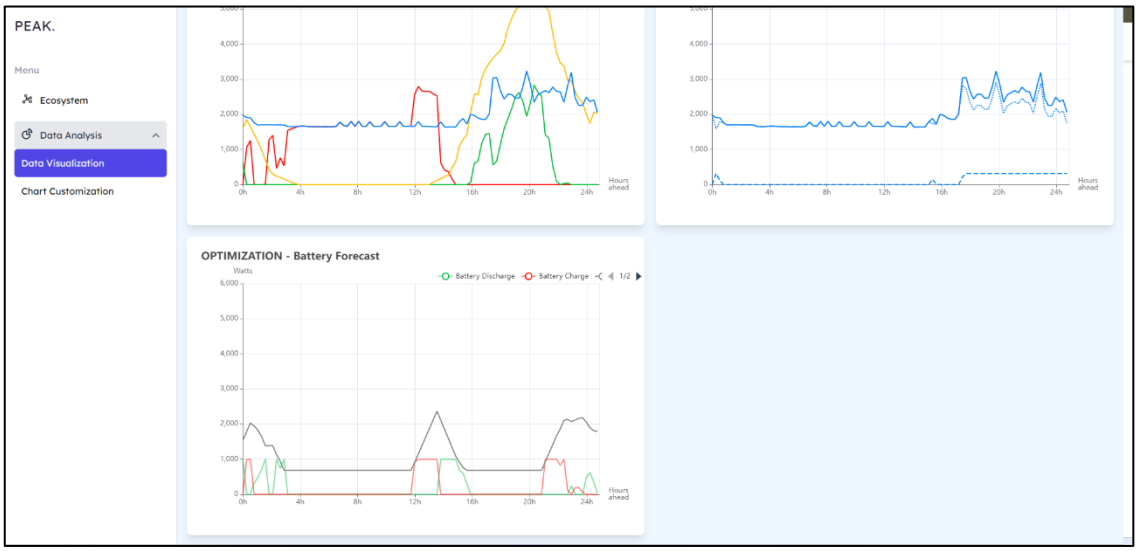


Figure 34 - Print of PEAK Dashboard showing the optimization results (Ribeiro, Faia, Gomes, et al., 2023).

5.4 Federated Learning based on Agent Communications: an Energy Forecast Case Study

The main objective of this case study is to validate the proposed solution, PEAK-FL, in different applications and with different models. The models used to test PEAK-FL are the FGP, a GP algorithm for image classification, and a DNN made by Daniel Ramos, for energy consumption forecast. The PEAK-FL is used here mainly as a communication layer of the FL system. This case study is divided into two different scenarios. In the first scenario, PEAK-FL is used to implement a collaborative learning environment to train a GP algorithm. In the second scenario, it is used to simulate an energy community training collaboratively a deep neural network using FL for energy forecasting of the next day. This case study contributed to the publication of the following paper:

- [Conference] **Bruno Ribeiro**, Luis Gomes, Rafael Barbarroxa, Zita Vale (2023) “A Novel Framework for Multiagent Knowledge-based Federated Learning Systems”, presented

and published in The PAAMS Collection: Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. (PAAMS 2023), DOI: 10.1007/978-3-031-37616-0_25

In the first scenario, PEAK-FL was used PEAK-FL to test and validate its performance in an image classification problem. The dataset used to train the model was the Rectangles dataset (see Section 3.1.5) and it was used only a fraction of it. In the simulation there were in total 11 agents, 10 were clients and one was the server. Every client has a dataset with 200 images chosen randomly from the original dataset. The FGP model was used (see Section 3.1.4) with its original parameters defined, only the population size and the number of generations were adapted to 100 and 10, respectively. The agents started the federation by joining the same community, as shown in Figure 35, and then they trained the model for 30 rounds.

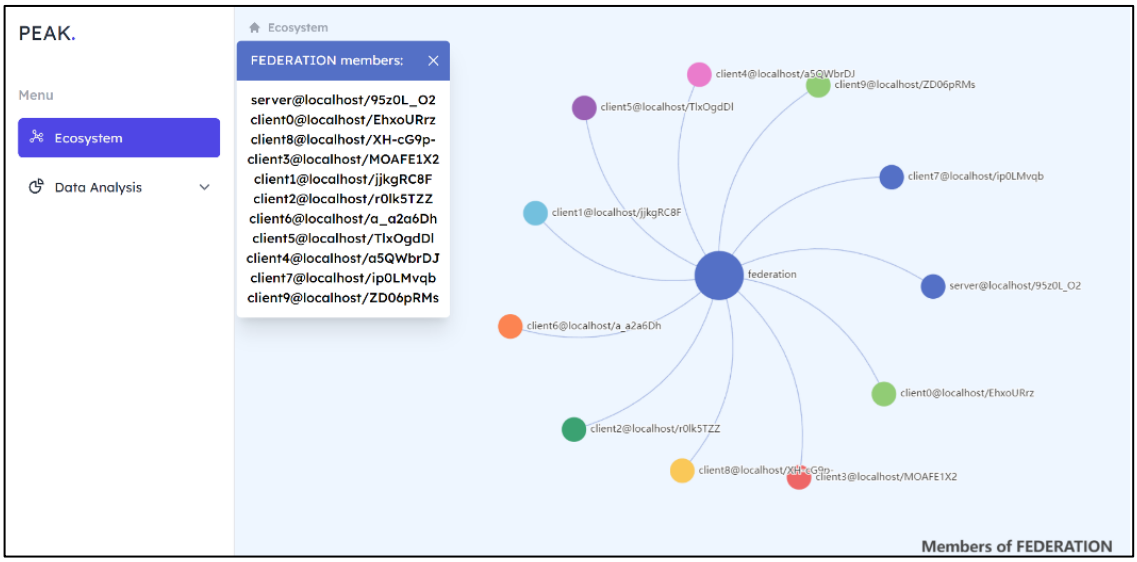


Figure 35 - Print of PEAK Dashboard showing the federation (Ribeiro, Gomes, Barbarroxa, et al., 2023).

For the analysis of the performance, it was tracked the classification error rate (%) of each client throughout the whole training, as shown in Figure 36. The x-axis represents the individuals in the halls of fame ordered from the first to the last one for each round. The y-axis is the classification error rate (%) of one individual at a given time. The graph resembles a Pareto distribution that stabilized near the beginning of the execution. Every client ended up with an error of 0%, except for clients three and seven who ended up with 0.4%. This shows the FGP's efficacy in the classification task using the Rectangles dataset.

In the second scenario, PEAK-FL was tested and validated against an energy forecasting problem. In this scenario, it was simulated an energy community with 10 members who trained collaboratively in a load forecasting model that predicts the energy consumption of the day ahead. To do that a DNN proposed by Ramos et al. (2021) was used (see Section 3.1.4). Normally, in FL scenarios, the whole dataset is used every round, but to make the federation more realistic to an energy community scenario, the clients trained the model every week with the data

gathered from that same week. The dataset used was the Energy Dataset 1 (see Section 3.1.5) and each client had access to two years of data making a total of 104 weeks/rounds.

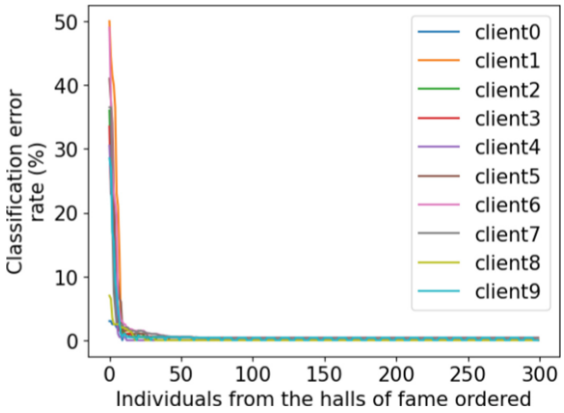


Figure 36 - Evolution of the halls of fame ordered by best to worst (Ribeiro, Gomes, Barbarroxa, et al., 2023).

The results of the case study can be seen in Figure 37. After the model training, the results of the clients were aggregated by weighted averaging. The green line corresponds to the root mean square error (RMSE) after the weights averaging of the whole federation and is measured in Watts. The blue line is the temperature during the two years 2019 and 2020 in Porto. The x-axis is expressed in weeks. It is possible to see that the overall error is very small since 200/300 W are very small values when it comes to energy consumption. It is possible to note that there was some fluctuation in the performance of the model in the transition from the first year to the second. As shown by the temperature variation it is possible that the fluctuation was originated from the abrupt change in the temperature since it perfectly matches the temperature drop. This is due mainly because of the use of air conditioning devices that are used in the winter and the energy profiles of the consumers. Given that the model is not prepared for that energy profile change, the error increases, coming down again after the temperature comes back up.

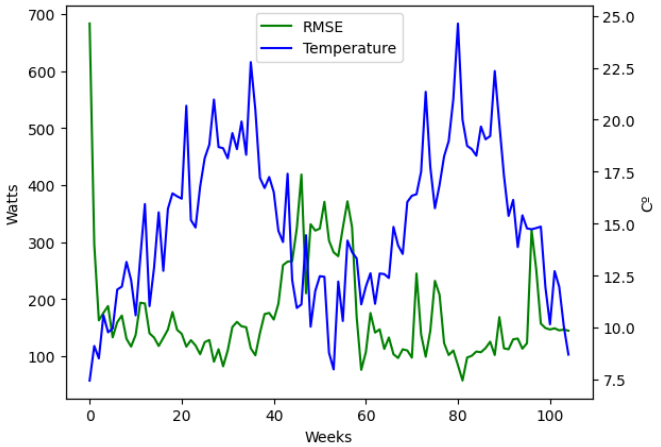


Figure 37 – Aggregated RMSE of the federation.

Overall, in this case study, PEAK-FL was tested and validated against two different scenarios. One where a group of clients collaboratively trained a model to solve an image classification problem, using a GP model, and the second one was an energy community that trained a model to predict energy load for the day ahead, using a DL model. PEAK-FL has shown to be reliable and efficient in implementing and executing these dynamic federated learning systems based on agents. Both federations effectively trained the models and ended up with good results, showing that PEAK-FL can indeed integrate different kinds of models in the FL environments. In addition to that, PEAK-FL has access to the PEAK Dashboard which allows to observe the model training in real-time, as well as the community structures that the agents created.

5.5 Federated Learning within Multi-Agent Systems: a Dynamic Mobility Case Study

The main objective of this case study is to assess the impact of dynamic clients, powered by agents, in an FL environment. Here the proposed solution, PEAK-FL, was used to develop the autonomous clients and allow them to change federations during execution time. Each federation model was based on the same DNN, previously proposed by Ramos et al. (2021), and described in Section 3.1.4, but each one was to be trained with different data. The idea was for an energy community to be able to forecast energy for the next day depending on the type of day they were trying to forecast. For this case study, it was first studied which type of inputs are better suitable for the energy consumption forecast, using the DNN model, to optimize its performance. Then, it was tested five different types of federated clients, each one with a different training procedure, to optimize the results of each client and the federations. Lastly, two scenarios were tested and compared where in one of them the clients are fixed in the same federation throughout the FL training process, and in the other, the clients can move freely between federations.

Firstly, the forecasting model was tested using six different types of input to see which input configuration better optimizes the performance of the DNN model used. Each type of input corresponds to different dataset transformations, which originated from the real consumption of one of the buildings managed by the research group where this dissertation was developed. It used data from the year 2013 divided into periods of 15 minutes (see Section 3.1.5). The performed transformations enabled the creation of six datasets where units and dimensions were in Watts (W), kilo Watts (kW), Watt-hour (Wh), kilo Watt-hour (kWh), Wh with cumulative sum for each day (Wh-cumsum), and kWh with the cumulative sum for each day (kWh-cumsum).

Regarding the DNN model, it was used the root mean square error (RMSE) as its loss function. For the evaluation process, it was used the following metrics, the relative absolute error (RAE), the mean absolute percentage error (MAPE), three versions of the normalized root mean square error (NRMSE), and the mean version of the normalized mean absolute error (NMAE). The NRMSE was used considering the following three calculations: the mean of the actual

values, the range between the minimum and maximum values of the actual values, and the standard deviation of the actual values. These metrics were chosen as they are better suited to compare models with different output units and dimensions.

Table 12 shows the results of the model using the six different datasets. Each row corresponds to a different dataset, and each column corresponds to a different metric evaluation. The evaluation is done using the last week of the dataset, which was separated from training specifically for testing. The results indicate, for all the metrics, that the best configuration is the use of Wh-cumsum transformation.

Table 12 – Comparison of the five different input configurations.

Dataset	Max	RAE	MAPE (%)	NMAE	NRMSE MaxMin	NRMSE Mean	NRMSE STD
W	49,256.000 W	3.924	0.078	0.081	1.066	0.098	3.834
kW	49.256 kW	4.293	0.084	0.087	1.170	0.104	4.162
Wh	12,314.000 Wh	3.333	0.069	0.072	0.934	0.090	3.386
kWh	12.314 kWh	13.214	0.194	0.193	3.155	0.199	11.367
Wh-cumsum	1,088,107.500 Wh	**0.344**	**0.047**	**0.048**	**0.094**	**0.052**	**0.316**
kWh-cumsum	1,088.108 kWh	0.642	0.104	0.107	0.173	0.117	0.583

Next, it was tested five different FL client training procedures which are: the traditional FL procedure (Full Data), the daily training procedure (Day), the weekly training procedure (Week), the daily cumulative training procedure (Cumulative Day), and the weekly cumulative training procedure (Cumulative Week). The Full Data is based on the traditional FL approach seen in the literature where the researchers assume that the clients have all the necessary data available at the time of training. The Day is a daily training procedure where one FL round is run every day. The Week is a weekly training procedure where one FL round is run every week. The Cumulative Day is a daily cumulative training procedure where instead of using only one day for training, it uses also all the data used to train the days before. The Cumulative Week is a weekly cumulative training procedure that performs one FL round every week but instead of using only one week of data to train the model, it also uses the data used to train the weeks before.

The dataset used was the Wh-cumsum transformation from the previous study. This dataset has a total of 51 clients but only 50 clients were used since they all represent households except for client 0. Each client has data from one leap year of consumption, containing a total of 366 days of data. After parsing the dataset to the model input format, 7 days as the input variable and 1 day as the expected variable, making a total of 8 days in the dataset, it reduces the number of days to 359. This is because if the window has a size of 8 and it starts at the beginning of the dataset it is only possible to shift it 359 times until you reach the other end of the dataset. From these 359, the last day was removed for testing purposes, reducing the training dataset to 358 days. Regarding the daily training procedures, the starting window, if combining the training and validation window, has a size of 2 days, meaning the maximum number of rounds it can perform is a total of 357. For the weekly training procedure, the starting windows, if also combining the training and validation windows, have the size of 8 days, but since it moves 7

days per round, it makes the total of 51 rounds possible. The Full Data since it trains with the whole dataset was defined as 30 rounds.

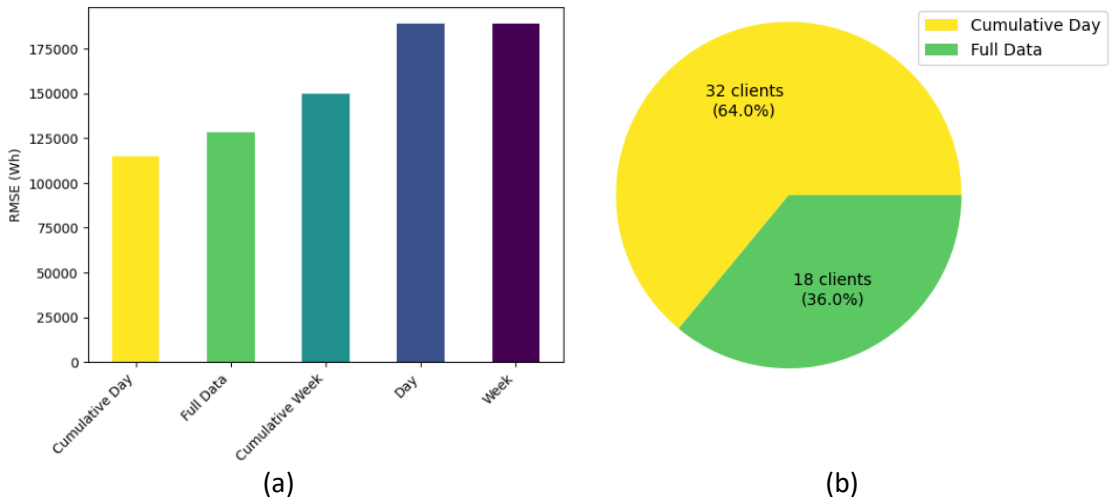
Table 13 shows the differences between each training procedure regarding the data being used in the training, validation, and testing phases. The training set is used in the training phase of FL, the validation set is used in the evaluation phase of FL, and the testing set is used after the whole FL process to measure the performance of the model. In the table, the *n* represents the last day of the dataset of the client, and the *i* is the number of the current round, or in other words, the day on which the FL round is being run. So, for training procedures Day, Cumulative Day, Week and Cumulative Week the training and validation sets are constantly updating every round while in the Full Data procedure, the training set uses the entirety of the training set every round, leaving the last day for validation. The validation set is not from the current day, the day *i*, because realistically the day when the round is running is not completed yet, so the clients would not have access to the complete data from that day making it unreasonable to use it as validation data.

The model was adapted to use seven days as input and the batch size was also according to the different training procedures. For the Full Data, the batch was the size of one month, making a total of 12 batches per training. In the Day it was used a batch size of one hour. The Week approach it was used a batch size of a day. The Cumulative Day had a dynamic batch size throughout the training process, where the training dataset was divided by 24, the number of hours per day. Since the cumulative approach increases the training dataset per round it also increases the batch size, by the size of one day. The same goes for the Cumulative Week, but instead of dividing the training dataset by 24, it is divided by seven. The rest of the model parameters stayed the same. This batch configuration was done to optimize the computation time needed to train the models while keeping the performance optimized as well.

Table 13 - Description of the federated learning training procedures tested.

Federated Learning Training Procedures	Training window	Validation window	Training set	Testing set	Number of Rounds
Full Data	[0; *n*-2]	*n*-1	[0; *n*-1]	*n*	30
Day	*i*-2	*i*-1			357
Cumulative Day	[0; *i*-2]	*i*-1			357
Week	[*i*-8; *i*-2]	*i*-1			51
Cumulative Week	[0; *i*-2]	*i*-1			51

The results of the study can be seen in Figure 38. Figure 38.a shows the accumulation of RMSE, calculated using the last day from the dataset kept aside for testing, from the 50 clients using each training procedure. The procedure that led to the least errors in total was the Cumulative Day. Figure 38.b shows the best performant procedure for each client individually. In this case, 32 clients had the best results using the Cumulative Day, and 18 clients had the best results using the Full Data. For these reasons, the Cumulative Day approach was used to validate the use of dynamic clients based on agents in federated learning environments.



(a) (b)

Figure 38 – Comparison between the different federated learning training procedures.

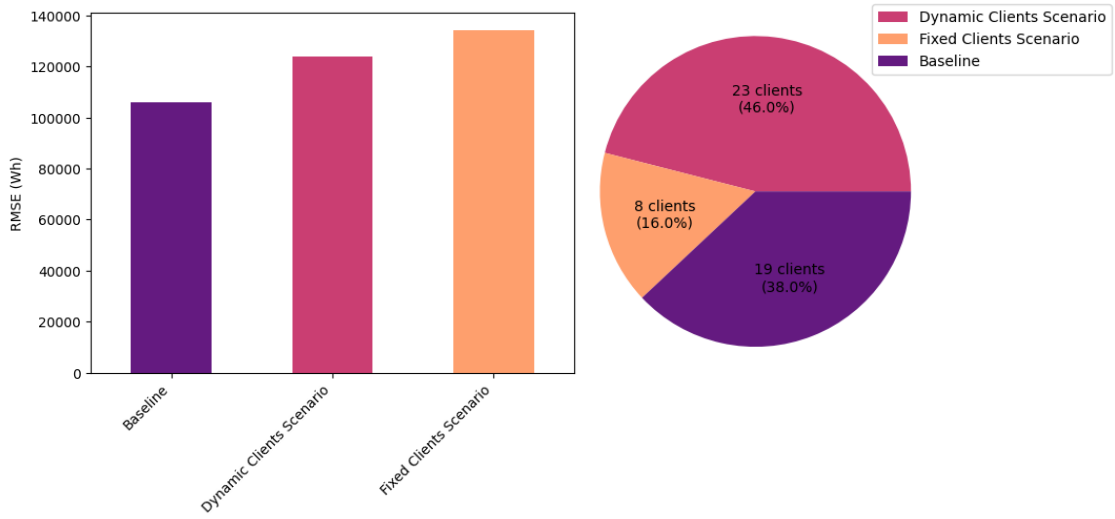
After knowing which dataset transformation is better for the DNN model and which FL training procedure is better, for fixed scenarios, it was studied the performance gain that dynamic clients based on agents can have when compared to fixed clients. To test this it was elaborated three scenarios: (i) the baseline scenario, where the clients train their model individually using traditional machine learning training, (ii) the fix clients scenario, where the clients will train the model collaboratively in federations by sticking with their initial federation until the end, and (iii) the dynamic clients scenario, where clients will train the model collaboratively, as well, but they will be able to change federations every round. It used the Wh-cumsum dataset and the last week of data of each client was preserved from training for testing purposes.

In energy forecasting models is better to train models for a specific type of day or energy profile. For this reason, it was decided to calculate the clusters for this energy community using a cluster technique. The technique used was the K-means algorithm in conjunction with Silhouette. All the data from the 50 clients were used. The clustering was based on three dimensions, namely, (i) the cumulative energy consumption of the given day, (ii) the average consumption, and (iii) the standard deviation of the consumption. The K-means algorithm was run with K from 2 to 10, and it was observed that the best configuration was K=3. From these results, it was made two distinct cluster approaches, one for the fixed client scenario and the other for the dynamic client scenario.

In the dynamic clients scenario, the clients will be able to choose every round which federation to go. With that possibility, it is not reasonable for clients to use any of the cumulative training procedures since they would use all the previous training data on the federation that they are on, worsening its performance. Since the clustering was based on single-day characteristics, the clients, in the dynamic client scenario, can only train with one day of data. For this reason, the Day training procedure was used in this scenario. Regarding the clustering, it was used the results directly from the K-means algorithm. The client sees which cluster the day it is trying to predict belongs to and goes to the corresponding federation, that is how the decision-making

will take place. For the fixed client scenario, on the other hand, since they will train on the same federation for the whole time, it was calculated the number of days for each client that belonged to each cluster. The cluster with the most days was then attributed to the client.

The results of these three scenarios can be seen in Figure 39 and Figure 40. Figure 39.a shows the amount of RMSE accumulated from the federation for each scenario, using the test dataset. The baseline shows the smallest error, next is the dynamic client scenario and the worst is the fixed client scenario. Figure 39.b shows the model preference of each client, and it indicates that the dynamic client scenario is preferred among the clients, with 46% approval, when considering all three scenarios. If comparing only the dynamic client scenario with the baseline it shows a higher approval with 58% of the clients preferring the dynamic client approach. Despite the dynamic client scenario having more total RMSE than the baseline, more clients have better results with it. In Figure 40 is possible to see the results of the (a) worst and (b) best clients. The worst client has the highest RMSE accumulation from the whole federation and it is possible to observe that the dynamic client is better throughout the first part of the day, until period 45, and then baseline takes the lead until the end of the day. The best client, on the other hand, is the client with the least RMSE accumulation among the whole federation and it is possible to observe that the dynamic client scenario is better throughout all day except for 3 or 4 periods.

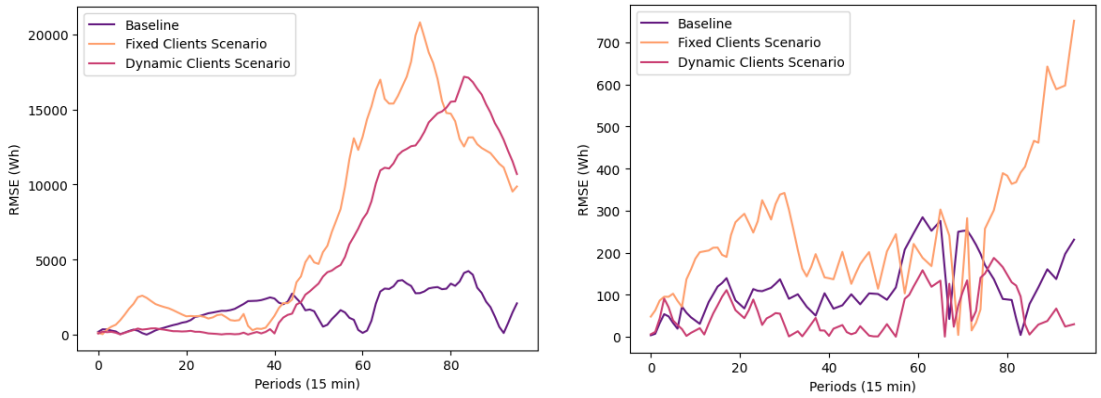


(a) (b)

Figure 39 – Comparing the results of the baseline, fixed client, and dynamic client scenarios.

If comparing the fixed client scenario against the dynamic client scenario it is obvious that the dynamic approach is better. It has better overall RMSE and there are way more clients that prefer the dynamic approach. When comparing the dynamic client scenario with the baseline it becomes more difficult to decide which one is better. On one hand, more clients prefer the dynamic client scenario than the baseline. However, that is not sufficient for the error of the clients to be small in the dynamic client scenario when considering the errors of the whole federation. This shows that the dynamic client may not be the perfect solution but is yet another

good option when trying to design, simulate, test, and validate a FL environment. When applied it can benefit the clients in the federation. In the case of this scenario, to maximize the performance for all clients, a solution would be to predict or calculate which approaches are better for which clients beforehand, and then they could all benefit from their best approaches. Other clustering techniques and heuristics could also be investigated to make the clusters even more precise, and therefore increase the precision of the FL model.



(a) (b)

Figure 40 - Comparing each scenario using (a) the worst and (b) the best clients.

6 Conclusions

This chapter describes the conclusions of the dissertation, how the research questions were answered, how the objectives were accomplished and the limitations and future directions of the proposed solution.

6.1 Summary

Since 2018, after the data protection regulations started to appear, companies have become more aware of how they are using the user's data to build new products. To overcome the limitation in the accessibility to the user's data a new technique was developed called federated learning, which permits the development of machine learning models without the necessity of users to share their data. But the way federated learning has been explored it assumes static and realistic environments, making it useless in real environments where changes in the system are unpredictable. This is where the concept of dynamic federated learning based on agents comes in, a combination of multi-agent systems with federated learning. The dissertation explores the benefits and drawbacks of using such systems. It proposes a new framework for building dynamic federated learning systems based on agents to support and help researchers test the combination of these two approaches and increase the applications and case studies for federated learning. The case studies tested the proposed solution against several scenarios, and almost every one of them contributed to the publication of scientific articles. The results showed that the framework works as intended and the combination of multi-agent systems and federated learning can benefit each other by allowing more dynamic environments, automating the clients, and increasing their model performance. This proposed solution can help the scientific community and users who want to deploy this kind of system, to go beyond and explore these concepts together to create more advanced, reliable, and efficient solutions in the federated learning field.

6.2 Research Questions and Objectives Achieved

During the development of the dissertation, research questions were used to guide the research and validation process. The research questions were answered throughout the dissertation as will be described next. The questions and answers are the following:

RQ1 - What benefits and drawbacks can multi-agent systems, a dynamic self-improvement distributed approach, have in automating and dynamizing federated learning, a private and collaborative learning environment approach?

The dissertation describes the gap in the state of the art regarding the merging of intelligent agents with federated learning systems. The benefits of this combination are the increase in realism in testing scenarios and case studies, enabling dynamic mobility in clients and the possibility of participating in more than one federation. When used right, it can even increase the performance of the model at personal and global levels. Despite not being tested, there is evidence that agents can provide a solution for the single point of failure in the federated learning system, by allowing agents to rapidly substitute other agents when they crash (Section 2.1). Agents can also make the federated learning system automatic, reducing the clients' burden to constantly monitor the federations they are participating.

The drawbacks are that developing agents can take more time and effort, and for some applications, there is no need to develop such a complex system, a simple federated learning system can do the job. The users also need to be careful since the bad use of agents can reduce the model performance and communication efficiency. If not implemented carefully the agents can become a problem rather than a solution. The proposed solution, PEAK-FL, helps the users to explore the impacts of continuous changes in the clients in federated learning environments allowing for more robust federated learning architectures in real-life situations.

This research question was answered by the fulfilment of objectives O1 ("Explore the state of the art of multi-agent systems, federated learning, and machine learning"), O2 ("Explore the state of the art about previous integrations of federated learning with multi-agent systems in what regards dynamic distributed learning environments.") and O7 ("Conceptualize, implement, and validate a federated learning development framework based on intelligent, autonomous, and goal-oriented agents."). These objectives helped identify the gap in the state of the art regarding the merging of intelligent agents with federated learning systems and its benefits and drawbacks. The results can be found throughout the whole dissertation from the state-of-the-art to the implementation and validation of the proposed solution.

RQ1.1 - What open-source state-of-the-art federated learning frameworks are flexible enough to allow the integration of different machine learning models (i.e. genetic programming) and other frameworks?

Before developing the proposed solution, a survey was done to see if there was any framework that already allowed the development of dynamic federated learning systems based on agents.

Few papers mention the use of agents in their solutions, but there is no indication of how they are implemented. Since no such frameworks were found, the next step was to find any federated learning framework that was flexible enough to allow the manipulation of different components and the integration with multi-agent systems. After some research and testing, only the Flower framework stood out as it was the only one that allowed the manipulation of the different components inside the framework and the integration of uncommon machine learning algorithms like genetic programming. The testing of Flower originated from a case study that also contributed to a scientific publication and as expected the results showed that the framework was highly customizable and flexible.

This research question was answered by the fulfilment of objective O3 ("Explore the open-source federated learning frameworks available and their flexibility in integrating different models and other frameworks."), O4 ("Explore the use of machine learning alternatives, like genetic programming, in federated learning settings.") and O5 ("Conceptualize, implement, and validate a solution to integrate genetic programming as the machine learning model to train in federated learning."). These objectives helped identify the list of federated learning frameworks available and which ones were more adequate to be used in the development of the proposed solution. The results can be found in Chapter 2, Section 2.2.4, through the survey conducted about the federate learning frameworks and in Chapter 3, Section 3.1.2, through the testing and selection of the framework.

RQ1.2 – How can federated learning be integrated with multi-agent systems to provide a dynamic distributed learning environment?

In multi-agent systems, as seen in Section 2.1.1, agents can have several different characteristics ranging from adaptation capabilities to physical mobility. Given their versatility to solve tasks and represent different entities, agents can be modelled specifically to participate and make use of federated learning environments. In federated learning, two different types of participants can be mapped to two different roles in a multi-agent system, the client, and the server. Each role has very specific actions and responsibilities that can easily be implemented in agents. This approach makes the integration open for the agent to acquire more roles, not limiting it only to federated learning systems. That was the driving thought when the proposed solution, PEAK-FL, was being designed. This does not only make the agents more open but the federated learning systems are more open as well, given that agents from different multi-agent systems are allowed to participate in them as well, if the owner allows it, and dynamically participate and join in the federations publicly available.

This research question was answered by the fulfilment of objectives O2 ("Explore the state of the art about previous integrations of federated learning with multi-agent systems in what regards dynamic distributed learning environments."), O6 ("Propose new agent architecture models for client and server participation in federated learning environments.") and O7 ("Conceptualize, implement, and validate a federated learning development framework based on intelligent, autonomous, and goal-oriented agents."). It was concluded that federated learning and multi-agent systems can indeed be combined, and it is possible to do that by

creating specific roles for the participation of the agents in federated learning systems. The result can be found in Chapter 2, in Section 2.2, through the exploration of how federated learning works theoretically and practically and in Chapter 4 through the actual implementation of the proposed solution.

RQ1.3 - How can federated learning help the agents achieve their goals more effectively and securely?

One important agent characteristic is the ability to learn (Section 2.1.3) which relates to the ability to gather new insights from the information the agent is receiving. This trait can be enhanced with the help of federated learning. In multi-agent system scenarios, normally the agents do not have sufficient computational resources, they have only partial views of the environment and together with other agents they can achieve better results by sharing some of their help and knowledge (Section 2.1). Traditionally, a multi-agent system would solve these problems with machine learning, by training models in centralized entities. In these circumstances, a distributed approach like federated learning is better applicable since it can distribute the computation power among several nodes while protecting the agent's data from being read by malicious agents. In addition to that, federated learning can even help with the sharing of knowledge in connection with constricted scenarios where the communication channels are not resilient as shown in Section 2.2.3. The results of using federated learning in agents and its benefits can be seen in case studies 5.4 and 5.5.

This research question was answered by the fulfilment of objectives O1 ("Explore the state of the art of multi-agent systems, federated learning, and machine learning.") and O7 ("Conceptualize, implement, and validate a federated learning development framework based on intelligent, autonomous, and goal-oriented agents.") and it was concluded that federated learning can help the agents share knowledge securely with other agents that aim to achieve similar objectives and it can help distribute computational power in machine learning tasks. The results can be found in Chapter 2, Sections 2.1 and 2.2, through the exploration of current multi-agent systems and federated learning works and Chapters 4 and 5, through the implementation and validation of the proposed solution.

RQ1.4 -What characteristics inherent in agent-based systems make federated learning scenarios more realistic?

In federated learning, most of the research assumes that the environment does not change over time, either regarding the data distribution of clients, the type of clients that join the federation, or even the model structure that is used in the federation (Section 2.2.3). In realistic scenarios, federations may run for several weeks and even months, and like any other system or long-time scenarios, the individual objectives and data distribution of each entity can change over time, making the need to alter some aspects of the federation or even switch to other federations to effectively pursue their updated objectives. For that reason, the dynamism, flexibility, proactivity, and intelligence that these entities have can help the federated learning scenarios become more realistic as they can simulate more accurately the client's behaviours. In addition

to that, PEAK-FL, specifically, allows the users to change any part of the federated learning process even during its execution time, making it flexible enough for any type of scenario that there is.

This research question was answered by the fulfilment of objective O1 ("Explore the state of the art of multi-agent systems, federated learning, and machine learning."), O2 ("Explore the state of the art about previous integrations of federated learning with multi-agent systems in what regards dynamic distributed learning environments") and O7 ("Conceptualize, implement, and validate a federated learning development framework based on intelligent, autonomous, and goal-oriented agents."). It was concluded that agent-based systems can indeed make federated learning more realistic by mimicking more accurate client profiles and allowing a more dynamic change of several aspects during the system runtime. The results of this research questions can be found in Chapter 2, through the exploration of federated learning and multi-agent system works and in Chapter 5, through the validation of the proposed solution.

RQ1.5 – To what extent can the dynamism of intelligent goal-oriented agents improve model accuracy in federated learning contexts?

Dynamism can be related to several characteristics in federated learning environments. The dynamism that is studied in this dissertation is the possibility of the clients joining or leaving federations during execution time, like in case study 5.5. This tries to replicate more realistically the scenarios where clients have the free will to choose from which federation they want to participate. Either good or bad outcomes can come out when free will is applied (Section 2.1.2). The idea is that agents have the right decision-making mechanism to allow them to choose the best federation to achieve their objectives. For the federations, the idea is to minimize the bad impact that these dynamic changes in the environment can bring to the model, normally the reduction of the model performance. As shown in Section 5.5, the right decision-making mechanism can bring the model accuracy up while allowing the clients to make the most out of the models available to them through federated learning.

This research question was answered by the fulfilment of objective O7 ("Conceptualize, implement, and validate a federated learning development framework based on intelligent, autonomous, and goal-oriented agents."). It was concluded that a good application of the multi-agent system in a federated learning environment can indeed increase the performance of the model, however, thoughtful considerations must be made when implementing such systems. The results of these research questions can be found mainly in Chapter 5 through the last case study.

6.3 Future Work

The proposed solution of this dissertation, PEAK-FL, was developed as a proof of concept. Some details were not considered during its development which does not make it suitable right away to be open to the public. However, PEAK-FL is planned to be open to the public soon after the

necessary fixes and updates are implemented. One fix that should be done is to make the framework more user-friendly since the main idea of the framework is to provide the necessary tools to the user so it can construct his system without being unnecessarily complex. Another objective is to make the framework more optimized, reducing the computational resources used by the agents since the execution time is very important in this type of system. One feature that should be added to the framework is creating a window in the PEAK Dashboard specifically for federated learning, for the management and monitorization of these systems inside the PEAK ecosystems. Next, more strategies can be added to the framework to help the users transition from the Flower framework to PEAK-FL.

Regarding the federated learning research field, there are also several different ways to further investigate the area. One property that can be investigated is the mitigation of the single point of failure using multi-agent systems. Another is to capacitate the agents to autonomously rearrange themselves in the federations to increase the accuracy of the models, and with that explore the impact of different decision-making mechanisms inside federated learning environments.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *ArXiv*.

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep Learning with Differential Privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318. https://doi.org/10.1145/2976749.2978318

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938

Abiodun, O. I., Kiru, M. U., Jantan, A., Omolara, A. E., Dada, K. V., Umar, A. M., Linus, O. U., Arshad, H., Kazaure, A. A., & Gana, U. (2019). Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition. *IEEE Access*, *7*, 158820–158846. https://doi.org/10.1109/ACCESS.2019.2945545

Agapitos, A., Loughran, R., Nicolau, M., Lucas, S., O'Neill, M., & Brabazon, A. (2019). A Survey of Statistical Machine Learning Elements in Genetic Programming. *IEEE Transactions on Evolutionary Computation*, *23*(6), 1029–1048. https://doi.org/10.1109/TEVC.2019.2900916

Aggarwal, A., Mittal, M., & Battineni, G. (2021). Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, *1*(1), 100004. https://doi.org/10.1016/j.jjimei.2020.100004

Ahmed Abbas, H. (2015). Organization of Multi-Agent Systems: An Overview. *International Journal of Intelligent Information Systems*, *4*(3), 46. https://doi.org/10.11648/j.ijiis.20150403.11

Ahvanooey, M., Li, Q., Wu, M., & Wang, S. (2019). A Survey of Genetic Programming and Its Applications. *KSII Transactions on Internet and Information Systems*, *13*(4). https://doi.org/10.3837/tiis.2019.04.002

Asaad, R. R., Ashqi Saeed, V., & Masud Abdulhakim, R. (2021). Smart Agent and it's effect on Artificial Intelligence : A Review Study. *ICONTECH INTERNATIONAL JOURNAL*, *5*(4), 1–9. https://doi.org/10.46291/ICONTECHvol5iss4pp1-9

Bae, H., Jang, J., Jung, D., Jang, H., Ha, H., Lee, H., & Yoon, S. (2018). *Security and Privacy Issues in Deep Learning*.

Baeta, F., Correia, J., Martins, T., & Machado, P. (2021). TensorGP – Genetic Programming Engine in TensorFlow. In *Applications of Evolutionary Computation - 24th International Conference, EvoApplications 2021* (pp. 763–778). Springer. https://doi.org/10.1007/978-3-030-72699-7_48

Balaji, P. G., & Srinivasan, D. (2010). An introduction to multi-agent systems. *Studies in Computational Intelligence*, *310*. https://doi.org/10.1007/978-3-642-14435-6_1

Barrientos, A. H., & Luevano, A. N. (2022). A State-of-the-Art Survey on Various Domains of Multi-Agent Systems and Machine Learning. In I. Sheremet (Ed.), *Multi-Agent Technologies and Machine Learning*. IntechOpen. https://doi.org/10.5772/intechopen.107109

Behera, M. R., Upadhyay, S., Shetty, S., & Otter, R. (2021). *Federated Learning using Peer-to-peer Network for Decentralized Orchestration of Model Weights*. https://doi.org/10.36227/TECHRXIV.14267468.V1

Belani, H., Vukovic, M., & Car, Z. (2019). Requirements Engineering Challenges in Building AI-Based Complex Systems. *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, 252–255. https://doi.org/10.1109/REW.2019.00051

Bellifemine, F., Caire, G., & Greenwood, D. (2007). Developing Multi-Agent Systems with JADE. In *Developing Multi-Agent Systems with JADE*. https://doi.org/10.1002/9780470058411

Beltrán, E. T. M., Pérez, M. Q., Sánchez, P. M. S., Bernal, S. L., Bovet, G., Pérez, M. G., Pérez, G. M., & Celdrán, A. H. (2022). *Decentralized Federated Learning: Fundamentals, State-of-the-art, Frameworks, Trends, and Challenges*.

Berna-Koes, M., Nourbakhsh, I., & Sycara, K. (2004). Communication efficiency in multi-agent systems. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 2129-2134 Vol.3. https://doi.org/10.1109/ROBOT.2004.1307377

Bertino, E., Kantarcioglu, M., Akcora, C. G., Samtani, S., Mittal, S., & Gupta, M. (2021). AI for Security and Security for AI. *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 333–334. https://doi.org/10.1145/3422337.3450357

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P. B., & Lane, N. D. (2020). Flower: A Friendly Federated Learning Research Framework. *ArXiv*.

Bi, Y., Xue, B., & Zhang, M. (2021a). *Genetic Programming for Image Classification* (Vol. 24). Springer International Publishing. https://doi.org/10.1007/978-3-030-65927-1

Bi, Y., Xue, B., & Zhang, M. (2021b). Genetic Programming With Image-Related Operators and a Flexible Program Structure for Feature Learning in Image Classification. *IEEE Transactions on Evolutionary Computation*, *25*(1), 87–101. https://doi.org/10.1109/TEVC.2020.3002229

Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., & van der Torre, L. (2001). The BOID architecture. *Proceedings of the Fifth International Conference on Autonomous Agents*, 9–16. https://doi.org/10.1145/375735.375766

Bulling, N. (2014). A Survey of Multi-Agent Decision Making. *KI - Künstliche Intelligenz*, *28*(3), 147–158. https://doi.org/10.1007/s13218-014-0314-3

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., & Talwalkar, A. (2018). LEAF: A Benchmark for Federated Settings. *ArXiv*.

Charbonnier, F., Morstyn, T., & McCulloch, M. D. (2022). Scalable multi-agent reinforcement learning for distributed control of residential energy flexibility. *Applied Energy*, *314*, 118825. https://doi.org/10.1016/j.apenergy.2022.118825

Chen, H., Asif, S. A., Park, J., Shen, C.-C., & Bennis, M. (2021). *Robust Blockchained Federated Learning with Model Validation and Proof-of-Stake Inspired Consensus*.

Chen, H., & Babar, M. A. (2022). *Security for Machine Learning-based Software Systems: a survey of threats, practices and challenges*.

Cho, Y. J., Wang, J., & Joshi, G. (2020). *Client Selection in Federated Learning: Convergence Analysis and Power-of-Choice Selection Strategies*. https://doi.org/10.48550/arxiv.2010.01243

Chou, L., Liu, Z., Wang, Z., & Shrivastava, A. (2021). *Efficient and Less Centralized Federated Learning*.

Choudhury, A., & Shamszare, H. (2023). Investigating the Impact of User Trust on the Adoption and Use of ChatGPT: Survey Analysis. *Journal of Medical Internet Research*, *25*, e47184. https://doi.org/10.2196/47184

Chudy, J., Popov, N., & Surynek, P. (2020). Emulating Centralized Control in Multi-Agent Pathfinding Using Decentralized Swarm of Reflex-Based Robots. *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 3998–4005. https://doi.org/10.1109/SMC42975.2020.9283368

Cointe, N., Bonnet, G., & Boissier, O. (2018). Ethics-based Cooperation in Multi-Agent Systems. *14th Social Simulation Conference (SSC18)*.

Concepcion II, R. S., Dadios, E. P., & Cuello, J. (2021). Non-destructive in Situ Measurement of Aquaponic Lettuce Leaf Photosynthetic Pigments and Nutrient Concentration Using Hybrid Genetic Programming. *AGRIVITA Journal of Agricultural Science*, *43*(3). https://doi.org/10.17503/agrivita.v43i3.2961

Cortacero, K., McKenzie, B., Müller, S., Khazen, R., Lafouresse, F., Corsaut, G., Acker, N. Van, Frenois, F.-X., Lamant, L., Meyer, N., Vergier, B., Wilson, D. G., Luga, H., Staufer, O.,

Dustin, M. L., Valitutti, S., & Cussat-Blanc, S. (2023). Kartezio: Evolutionary Design of Explainable Pipelines for Biomedical Image Analysis. In *arXiv*.

De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP. *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, 85–92. https://doi.org/10.1145/2330784.2330799

DeAngelis, D. L., & Diaz, S. G. (2019). Decision-Making in Agent-Based Modeling: A Current Review and Future Prospectus. *Frontiers in Ecology and Evolution*, *6*. https://doi.org/10.3389/fevo.2018.00237

Desai, H. B., Ozdayi, M. S., & Kantarcioglu, M. (2020). *BlockFLA: Accountable Federated Learning via Hybrid Blockchain Architecture*.

Dilmaghani, S., Brust, M. R., Danoy, G., Cassagnes, N., Pecero, J., & Bouvry, P. (2019). Privacy and Security of Big Data in AI Systems: A Research and Standards Perspective. *2019 IEEE International Conference on Big Data (Big Data)*, 5737–5743. https://doi.org/10.1109/BigData47090.2019.9006283

Dinelli, C., Racette, J., Escarcega, M., Lotero, S., Gordon, J., Montoya, J., Dunaway, C., Androulakis, V., Khaniani, H., Shao, S., Roghanchi, P., & Hassanalian, M. (2023). Configurations and Applications of Multi-Agent Hybrid Drone/Unmanned Ground Vehicle for Underground Environments: A Review. *Drones*, *7*(2), 136. https://doi.org/10.3390/drones7020136

Dorri, A., Kanhere, S. S., & Jurdak, R. (2018). Multi-Agent Systems: A Survey. *IEEE Access*, *6*, 28573–28593. https://doi.org/10.1109/ACCESS.2018.2831228

Drew, D. S. (2021). Multi-Agent Systems for Search and Rescue Applications. *Current Robotics Reports*, *2*(2), 189–200. https://doi.org/10.1007/s43154-021-00048-3

Duan, Y., Cui, B. X., & Xu, X. H. (2012). A multi-agent reinforcement learning approach to robot soccer. *Artificial Intelligence Review*, *38*(3), 193–211. https://doi.org/10.1007/s10462-011-9244-8

El Mrabet, M. A., El Makkaoui, K., & Faize, A. (2021). Supervised Machine Learning: A Survey. *Proceedings - 4th International Conference on Advanced Communication Technologies and Networking, CommNet 2021*. https://doi.org/10.1109/COMMNET52204.2021.9641998

Esterle, L. (2022). Deep learning in multiagent systems. In *Deep Learning for Robot Perception and Cognition* (pp. 435–460). Elsevier. https://doi.org/10.1016/B978-0-32-385787-1.00022-1

European Comission. (2018). *Ethics Guidelines for Trustworthy AI*. https://ec.europa.eu/futurium/en/ai-alliance-consultation.1.html

European Union. (2018). *General Data Protection Regulation*. https://gdpr.eu/

Fabiano, N. (2019). Robotics, intelligent systems, ethics and data protection. *Proceedings of the 2nd International Conference on Applications of Intelligent Systems*, 1–5. https://doi.org/10.1145/3309772.3309787

Faia, R., Ribeiro, B., Goncalves, C., Gomes, L., & Vale, Z. (2023). Multi-agent based energy community cost optimization considering high electric vehicles penetration. *Sustainable Energy Technologies and Assessments*, *59*, 103402. https://doi.org/10.1016/j.seta.2023.103402

Falco, M., & Robiolo, G. (2019). *Tendencies in Multi-Agent Systems: A Systematic Literature Review* (Vol. 23, Issue 1).

Fan, Q., Bi, Y., Xue, B., & Zhang, M. (2022). Genetic programming for feature extraction and construction in image classification. *Applied Soft Computing*, *118*, 108509. https://doi.org/10.1016/j.asoc.2022.108509

FATE. (2019, January 24). Federated AI. https://fate.fedai.org/

Ferber, J., Gutknecht, O., & Michel, F. (2004). *From Agents to Organizations: An Organizational View of Multi-agent Systems* (pp. 214–230). https://doi.org/10.1007/978-3-540-24620-6_15

Fernandez de Vega, F., Olague, G., Lanza, D., Chavez de la O, F., Banzhaf, W., Goodman, E., Menendez-Clavijo, J., & Martinez, A. (2020). Time and Individual Duration in Genetic Programming. *IEEE Access*, *8*, 38692–38713. https://doi.org/10.1109/ACCESS.2020.2975753

Foley, P., Sheller, M. J., Edwards, B., Pati, S., Riviera, W., Sharma, M., Narayana Moorthy, P., Wang, S., Martin, J., Mirhaji, P., Shah, P., & Bakas, S. (2022). OpenFL: the open federated learning library. *Physics in Medicine & Biology*, *67*(21), 214001. https://doi.org/10.1088/1361-6560/ac97d9

Fourez, T., Verstaevel, N., Migeon, F., Schettini, F., & Amblard, F. (2022). How to Solve a Classification Problem Using a Cooperative Tiling Multi-agent System? In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection* (pp. 166–178). https://doi.org/10.1007/978-3-031-18192-4_14

Gal, K., & Grosz, B. J. (2022). Multi-Agent Systems: Technical & Ethical Challenges of Functioning in a Mixed Group. *Daedalus*, *151*(2), 114–126. https://doi.org/10.1162/daed_a_01904

Gal, Y., Grosz, B., Kraus, S., Pfeffer, A., & Shieber, S. (2010). Agent decision-making in open mixed networks. *Artificial Intelligence*, *174*(18), 1460–1480. https://doi.org/10.1016/j.artint.2010.09.002

Gao, D., Chen, D., Li, Z., Xie, Y., Pan, X., Li, Y., Ding, B., & Zhou, J. (2023). FS-Real: A Real-World Cross-Device Federated Learning Platform. *Proceedings of the VLDB Endowment*, *16*(12), 4046–4049. https://doi.org/10.14778/3611540.3611617

Gao, L., Fu, H., Li, L., Chen, Y., Xu, M., & Xu, C.-Z. (2022). FedDC: Federated Learning with Non-IID Data via Local Drift Decoupling and Correction. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10102–10111. https://doi.org/10.1109/CVPR52688.2022.00987

Gavigan, P., & Esfandiari, B. (2021). Agent in a Box: A Framework for Autonomous Mobile Robots with Beliefs, Desires, and Intentions. *Electronics*, *10*(17), 2136. https://doi.org/10.3390/electronics10172136

Georgeff, M., Pell, B., Pollack, M., Tambe, M., & Wooldridge, M. (1999). *The Belief-Desire-Intention Model of Agency* (pp. 1–10). https://doi.org/10.1007/3-540-49057-4_1

Gherairi, S. (2023). Design and implementation of an intelligent energy management system for smart home utilizing a multi-agent system. *Ain Shams Engineering Journal*, *14*(3), 101897. https://doi.org/10.1016/j.asej.2022.101897

Gholami, A., Torkzaban, N., & Baras, J. S. (2022). Trusted Decentralized Federated Learning. *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, 1–6. https://doi.org/10.1109/CCNC49033.2022.9700624

Gogineni, V. C., Werner, S., Huang, Y.-F., & Kuh, A. (2022). Communication-Efficient Online Federated Learning Strategies for Kernel Regression. *IEEE Internet of Things Journal*, 1–1. https://doi.org/10.1109/JIOT.2022.3218484

Gomes, L., Ribeiro, B., Lezama, F., & Vale, Z. (2023). A Multi-Agent System Empowered by Federated Learning and Genetic Programming. *2023 31st Signal Processing and Communications Applications Conference (SIU)*, 1–4. https://doi.org/10.1109/SIU59756.2023.10223778

Goncalves, C., Barreto, R., Faria, P., Gomes, L., & Vale, Z. (2022). Dataset of an energy community's generation and consumption with appliance allocation. *Data in Brief*, *45*, 108590. https://doi.org/10.1016/j.dib.2022.108590

Gouws, G. (2021). *Holonic Superposition Collaborative Multi-Agent Systems Architecture*.

Gronauer, S., & Diepold, K. (2022). Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, *55*(2), 895–943. https://doi.org/10.1007/s10462-021-09996-w

Hagendorff, T. (2020). The Ethics of AI Ethics: An Evaluation of Guidelines. *Minds and Machines*, *30*(1), 99–120. https://doi.org/10.1007/s11023-020-09517-8

Han, C., & Yang, T. (2021). Privacy Protection Technology of Maritime Multi-agent Communication Based on Part-Federated Learning. *2021 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, 266–271. https://doi.org/10.1109/ICCCWorkshops52231.2021.9538897

Hanson Robotics. (2016). *Sophia*. https://www.hansonrobotics.com/sophia/

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

He, L., Bian, A., & Jaggi, M. (2018). COLA: Decentralized Linear Learning. *ArXiv*.

Hedin, Y., & Moradian, E. (2015). Security in Multi-Agent Systems. *Procedia Computer Science*, *60*, 1604–1612. https://doi.org/10.1016/j.procs.2015.08.270

Hernandez-Leal, P., Kaisers, M., Baarslag, T., & de Cote, E. M. (2019). A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. In *arXiv*.

Hoen, P. J. 't, Tuyls, K., Panait, L., Luke, S., & La Poutré, J. A. (2006). *An Overview of Cooperative and Competitive Multiagent Learning* (pp. 1–46). https://doi.org/10.1007/11691839_1

Horling, B., & Lesser, V. (2004). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, *19*(4), 281–316. https://doi.org/10.1017/S0269888905000317

Huang, C., Huang, J., & Liu, X. (2022). *Cross-Silo Federated Learning: Challenges and Opportunities*.

Jabeur, R., Boujoudar, Y., Azeroual, M., Aljarbouh, A., & Ouaaline, N. (2022). Microgrid energy management system for smart home using multi-Agent system. *International Journal of Electrical and Computer Engineering*, *12*(2). https://doi.org/10.11591/ijece.v12i2.pp1153-1160

Jain, A., Patel, H., Nagalapatti, L., Gupta, N., Mehta, S., Guttula, S., Mujumdar, S., Afzal, S., Sharma Mittal, R., & Munigala, V. (2020). Overview and Importance of Data Quality for Machine Learning Tasks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3561–3562. https://doi.org/10.1145/3394486.3406477

Javadpour, A., Pinto, P., Ja'fari, F., & Zhang, W. (2023). DMAIDPS: a distributed multi-agent intrusion detection and prevention system for cloud IoT environments. *Cluster Computing*, *26*(1), 367–384. https://doi.org/10.1007/s10586-022-03621-3

Jeon, J., Kim, J., Kim, J., Kim, K., Mohaisen, A., & Kim, J.-K. (2019). Privacy-Preserving Deep Learning Computation for Geo-Distributed Medical Big-Data Platforms. *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S)*, 3–4. https://doi.org/10.1109/DSN-S.2019.00007

Jin, Y., Zhu, H., Xu, J., & Chen, Y. (2023). *Federated Learning*. Springer Nature Singapore. https://doi.org/10.1007/978-981-19-7083-2

Jubair, M. A., Mostafa, S. A., Mustapha, A., & Hafit, H. (2018). A Survey of Multi-agent Systems and Case-Based Reasoning Integration. *2018 International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR)*, 1–6. https://doi.org/10.1109/ISAMSR.2018.8540549

Jung, A. (2022). *Machine Learning*. Springer Nature Singapore. https://doi.org/10.1007/978-981-16-8193-6

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., El Rouayheb, S., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., … Zhao, S. (2021). Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, *14*(1–2), 1–210. https://doi.org/10.1561/2200000083

Kastampolidou, K., Nikiforos, M. N., & Andronikos, T. (2020). *A Brief Survey of the Prisoners' Dilemma Game and Its Potential Use in Biology* (pp. 315–322). https://doi.org/10.1007/978-3-030-32622-7_29

Kavalionak, H., Carlini, E., Dazzi, P., Ferrucci, L., Mordacchini, M., & Coppola, M. (2021). Impact of Network Topology on the Convergence of Decentralized Federated Learning Systems. *2021 IEEE Symposium on Computers and Communications (ISCC)*, 1–6. https://doi.org/10.1109/ISCC53001.2021.9631460

Khanal, S. S., Prasad, P. W. C., Alsadoon, A., & Maag, A. (2020). A systematic review: machine learning based recommendation systems for e-learning. *Education and Information Technologies*, *25*(4), 2635–2664. https://doi.org/10.1007/s10639-019-10063-9

King, M. R. (2023). The Future of AI in Medicine: A Perspective from a Chatbot. *Annals of Biomedical Engineering*, *51*(2), 291–295. https://doi.org/10.1007/s10439-022-03121-w

Kreps, D. M. (1989). Nash Equilibrium. In *Game Theory* (pp. 167–177). Palgrave Macmillan UK. https://doi.org/10.1007/978-1-349-20181-5_19

Lalitha, A., Kilinc, O. C., Javidi, T., & Koushanfar, F. (2019). *Peer-to-peer Federated Learning on Graphs*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *Proceedings of the*

24th International Conference on Machine Learning, 473–480. https://doi.org/10.1145/1273496.1273556

Lavanchy, M. (2018). *Amazon's sexist hiring algorithm could still be better than a human*. https://www.imd.org/research-knowledge/articles/amazons-sexist-hiring-algorithm-could-still-be-better-than-a-human/

Lavinas, Y., Cortacero, K., & Cussat-Blanc, S. (2023). Evolving Graphs with Cartesian Genetic Programming with Lexicase Selection. *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 1920–1924. https://doi.org/10.1145/3583133.3596402

Lee, S., & Choi, D.-H. (2022). Federated Reinforcement Learning for Energy Management of Multiple Smart Homes With Distributed Energy Resources. *IEEE Transactions on Industrial Informatics*, *18*(1), 488–497. https://doi.org/10.1109/TII.2020.3035451

Lewicki, R. J., Barry, B., & Saunders, D. M. (2020). *Negotiation*. McGraw-Hill.

Li Deng. (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, *29*(6), 141–142. https://doi.org/10.1109/MSP.2012.2211477

Li, J., Shao, Y., Wei, K., Ding, M., Ma, C., Shi, L., Han, Z., & Poor, H. V. (2021). *Blockchain Assisted Decentralized Federated Learning (BLADE-FL): Performance Analysis and Resource Allocation*.

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., & He, B. (2023). A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. *IEEE Transactions on Knowledge and Data Engineering*, *35*(4), 3347–3366. https://doi.org/10.1109/TKDE.2021.3124599

Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, *37*(3), 50–60. https://doi.org/10.1109/MSP.2020.2975749

Li, X., Lu, L., Ni, W., Jamalipour, A., Zhang, D., & Du, H. (2022). Federated Multi-Agent Deep Reinforcement Learning for Resource Allocation of Vehicle-to-Vehicle Communications. *IEEE Transactions on Vehicular Technology*, *71*(8), 8810–8824. https://doi.org/10.1109/TVT.2022.3173057

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999–7019. https://doi.org/10.1109/TNNLS.2021.3084827

Liao, Y., Xu, Y., Xu, H., Wang, L., & Qian, C. (2022). Adaptive Configuration for Heterogeneous Participants in Decentralized Federated Learning. *ArXiv*.

Lin, J., Ma, J., & Zhu, J. (2022). Privacy-Preserving Household Characteristic Identification With Federated Learning Method. *IEEE Transactions on Smart Grid*, *13*(2), 1088–1099. https://doi.org/10.1109/TSG.2021.3125677

Liu, B., Lv, N., Guo, Y., & Li, Y. (2023). Recent Advances on Federated Learning: A Systematic Survey. *ArXiv*.

Liu, C., Sun, S., Tao, C., Shou, Y., & Xu, B. (2021). Sliding mode control of multi-agent system with application to UAV air combat. *Computers & Electrical Engineering*, *96*, 107491. https://doi.org/10.1016/j.compeleceng.2021.107491

Liu, J., Huang, J., Zhou, Y., Li, X., Ji, S., Xiong, H., & Dou, D. (2022). From distributed machine learning to federated learning: a survey. *Knowledge and Information Systems*, *64*(4), 885–917. https://doi.org/10.1007/s10115-022-01664-x

Liu, P., Xu, X., & Wang, W. (2022a). Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecurity*, *5*(1). https://doi.org/10.1186/s42400-021-00105-6

Liu, P., Xu, X., & Wang, W. (2022b). Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecurity*, *5*(1), 4. https://doi.org/10.1186/s42400-021-00105-6

Liu, W., Chen, L., & Zhang, W. (2021). Decentralized Federated Learning: Balancing Communication and Computing Costs. *ArXiv*.

Liu, X., Xie, L., Wang, Y., Zou, J., Xiong, J., Ying, Z., & Vasilakos, A. V. (2021). Privacy and Security Issues in Deep Learning: A Survey. *IEEE Access*, *9*, 4566–4593. https://doi.org/10.1109/ACCESS.2020.3045078

Liu, Z., Chen, S., Ye, J., Fan, J., Li, H., & Li, X. (2023). DHSA: efficient doubly homomorphic secure aggregation for cross-silo federated learning. *The Journal of Supercomputing*, *79*(3), 2819–2849. https://doi.org/10.1007/s11227-022-04745-4

Lyu, L., Yu, H., & Yang, Q. (2020). *Threats to Federated Learning: A Survey*.

Ma, Z., Schultz, M. J., Christensen, K., Værbak, M., Demazeau, Y., & Jørgensen, B. N. (2019). The Application of Ontologies in Multi-Agent Systems in the Energy Sector: A Scoping Review. *Energies*, *12*(16), 3200. https://doi.org/10.3390/en12163200

Mäenpää, D. (2021). *Towards Peer-to-Peer Federated Learning: Algorithms and Comparisons to Centralized Federated Learning*. http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-176778

Mahajan, H. B., Uke, N., Pise, P., Shahade, M., Dixit, V. G., Bhavsar, S., & Deshpande, S. D. (2023). Automatic robot Manoeuvres detection using computer vision and deep learning

techniques: a perspective of internet of robotics things (IoRT). *Multimedia Tools and Applications*, *82*(15), 23251–23276. https://doi.org/10.1007/s11042-022-14253-5

Mahela, O., Khosravy, M., Gupta, N., Khan, B., Alhelou, H., Mahla, R., Patel, N., & Siano, P. (2020). Comprehensive overview of multi-agent systems for controlling smart grids. *CSEE Journal of Power and Energy Systems*. https://doi.org/10.17775/CSEEJPES.2020.03390

Mai, L., Li, G., Wagenländer, M., Fertakis, K., Brabete, A. O., & Pietzuch, P. (2020). Kung Fu: Making training in distributed machine learning adaptive. *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020*.

Malan, E., Peluso, V., Calimera, A., Macii, E., & Montuschi, P. (2023). Automatic Layer Freezing for Communication Efficiency in Cross-Device Federated Learning. *IEEE Internet of Things Journal*, 1–1. https://doi.org/10.1109/JIOT.2023.3309691

Martin, D. L., Cheyer, A. J., & Moran, D. B. (1999). The open agent architecture: a framework for building distributed software systems. *Applied Artificial Intelligence*, *13*(1–2). https://doi.org/10.1080/088395199117504

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2016). *Communication-Efficient Learning of Deep Networks from Decentralized Data*.

McMaster, C., Chan, J., Liew, D. F. L., Su, E., Frauman, A. G., Chapman, W. W., & Pires, D. E. V. (2023). Developing a deep learning natural language processing algorithm for automated reporting of adverse drug reactions. *Journal of Biomedical Informatics*, *137*, 104265. https://doi.org/10.1016/j.jbi.2022.104265

Meurisch, C., & Mühlhäuser, M. (2022). Data Protection in AI Services. *ACM Computing Surveys*, *54*(2), 1–38. https://doi.org/10.1145/3440754

Michalski, R. S. (1993). Learning = Inferencing + Memorizing. In *Foundations of Knowledge Acquisition* (pp. 1–41). Springer US. https://doi.org/10.1007/978-0-585-27366-2_1

Moradi, R., Alikhani, A., & Jegarkandi, M. F. (2017). Multi-objective optimization in graceful performance degradation and its application in spacecraft attitude fault-tolerant control. *Aerospace Science and Technology*, *69*, 465–473. https://doi.org/10.1016/j.ast.2017.07.009

Mualla, Y., Najjar, A., Daoud, A., Galland, S., Nicolle, C., Yasar, A.-U.-H., & Shakshuki, E. (2019). Agent-based simulation of unmanned aerial vehicles in civilian applications: A systematic literature review and research directions. *Future Generation Computer Systems*, *100*, 344–364. https://doi.org/10.1016/j.future.2019.04.051

Müller, S. D., Kristensen, M., Lauridsen, K. G., Zwanenburg, M., & Løfgren, B. (2021). Decentralized federated learning: An introduction and the road ahead. In *Proceedings of the Annual Hawaii International Conference on System Sciences* (Vols. 2020-January). IEEE Computer Society. https://doi.org/10.24251/hicss.2021.441

Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2020). Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Transactions on Cybernetics*, *50*(9), 3826–3839. https://doi.org/10.1109/TCYB.2020.2977374

Nuriyev, E., Manumachu, R. R., Aseeri, S., Verma, M. K., & Lastovetsky, A. L. (2024). SUARA: A scalable universal allreduce communication algorithm for acceleration of parallel deep learning applications. *Journal of Parallel and Distributed Computing*, *183*, 104767. https://doi.org/10.1016/j.jpdc.2023.104767

NVIDIA, Vingelmann, P., & Fitzek, F. H. P. (2020). *CUDA, release: 10.2.89*. https://developer.nvidia.com/cuda-toolkit

Olthof, A. W., Shouche, P., Fennema, E. M., IJpma, F. F. A., Koolstra, R. H. C., Stirler, V. M. A., van Ooijen, P. M. A., & Cornelissen, L. J. (2021). Machine learning based natural language processing of radiology reports in orthopaedic trauma. *Computer Methods and Programs in Biomedicine*, *208*, 106304. https://doi.org/10.1016/j.cmpb.2021.106304

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., & Walsh, J. (2020). Deep Learning vs. Traditional Computer Vision. In *CVC 2019: Advances in Computer Vision* (pp. 128–144). https://doi.org/10.1007/978-3-030-17795-9_10

OpenAI. (2022). *ChatGPT: Optimizing Language Models for Dialogue*. https://openai.com/blog/chatgpt/

Oroojlooy, A., & Hajinezhad, D. (2023). A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, *53*(11), 13677–13722. https://doi.org/10.1007/s10489-022-04105-y

Owoputi, R., & Ray, S. (2022). Security of Multi-Agent Cyber-Physical Systems: A Survey. *IEEE Access*, *10*, 121465–121479. https://doi.org/10.1109/ACCESS.2022.3223362

Pal, C.-V., Leon, F., Paprzycki, M., & Ganzha, M. (2020). A Review of Platforms for the Development of Agent Systems. *ArXiv*.

Palanca, J., Terrasa, A., Julian, V., & Carrascosa, C. (2020). SPADE 3: Supporting the New Generation of Multi-Agent Systems. *IEEE Access*, *8*, 182537–182549. https://doi.org/10.1109/ACCESS.2020.3027357

Palconit, M. G., Pareja, M., Bandala, A., Espanola, J., Vicerra, R. R., Concepcion, R., Sybingco, E., & Dadios, E. (2021). FishEye: A Centroid-Based Stereo Vision Fish Tracking Using Multigene Genetic Programming. *2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC)*, 1–5. https://doi.org/10.1109/R10-HTC53172.2021.9641654

Pandey, M., & Rautaray, S. S. (Eds.). (2021). *Machine Learning: Theoretical Foundations and Practical Applications* (Vol. 87). Springer Singapore. https://doi.org/10.1007/978-981-33-6518-6

Patil, M. S., & Chickerur, S. (2023). Study of Data and Model parallelism in Distributed Deep learning for Diabetic retinopathy classification. *Procedia Computer Science*, *218*, 2253–2263. https://doi.org/10.1016/j.procs.2023.01.201

Pereira, H., Ribeiro, B., Gomes, L., & Vale, Z. (2022). Smart Grid Ecosystem Modeling Using a Novel Framework for Heterogenous Agent Communities. *Sustainability*, *14*(23), 15983. https://doi.org/10.3390/su142315983

Petroc Taylor. (2023, November 16). *Data growth worldwide 2010-2025*. Statista. https://www.statista.com/statistics/871513/worldwide-data-created/

Presidente, G., & Frey, C. (2022, March 10). *The GDPR effect: How data privacy regulation shaped firm performance globally*. VoxEU. https://cepr.org/voxeu/columns/gdpr-effect-how-data-privacy-regulation-shaped-firm-performance-globally

Price, S. R., Anderson, D. T., & Price, S. R. (2019). GOOFeD: Extracting Advanced Features for Image Classification via Improved Genetic Programming. *2019 IEEE Congress on Evolutionary Computation (CEC)*, 1596–1603. https://doi.org/10.1109/CEC.2019.8790347

Priestley, M., O'donnell, F., & Simperl, E. (2023). A Survey of Data Quality Requirements That Matter in ML Development Pipelines. *Journal of Data and Information Quality*, *15*(2), 1–39. https://doi.org/10.1145/3592616

Puica, M.-A., & Florea, A.-M. (2013). Emotional belief-desire-intention agent model: Previous work and proposed architecture. *International Journal of Advanced Research in Artificial Intelligence*, *2*(2), 1–8.

PySyft. (2017, July 18). OpenMined. https://github.com/OpenMined/PySyft

Qammar, A., Wang, H., Ding, J., Naouri, A., Daneshmand, M., & Ning, H. (2023). Chatbots to ChatGPT in a Cybersecurity Space: Evolution, Vulnerabilities, Attacks, Challenges, and Future Recommendations. *ArXiv*.

Qiu, X., Pan, H., Zhao, W., Ma, C., de Gusmão, P. P. B., & Lane, N. D. (2023). Efficient Vertical Federated Learning with Secure Aggregation. *ArXiv*.

Ramos, D., Khorram, M., Faria, P., & Vale, Z. (2021). Load Forecasting in an Office Building with Different Data Structure and Learning Parameters. *Forecasting*, *3*(1), 242–254. https://doi.org/10.3390/forecast3010015

Rasa Technologies Inc. (2023). *Rasa*. https://rasa.com/

Rehman, K. ur, Li, J., Pei, Y., Yasin, A., Ali, S., & Mahmood, T. (2021). Computer Vision-Based Microcalcification Detection in Digital Mammograms Using Fully Connected Depthwise Separable Convolutional Neural Network. *Sensors*, *21*(14), 4854. https://doi.org/10.3390/s21144854

Ribeiro, B., Faia, R., Gomes, L., & Vale, Z. (2023). Energy Community Integration of a Smart Home Based on an Open Source Multiagent System. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection* (pp. 415–421). https://doi.org/10.1007/978-3-031-37616-0_35

Ribeiro, B., Gomes, L., Barbarroxa, R., & Vale, Z. (2023). A Novel Framework for Multiagent Knowledge-Based Federated Learning Systems. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection (PAAMS 2023)* (pp. 296–306). https://doi.org/10.1007/978-3-031-37616-0_25

Ribeiro, B., Gomes, L., Faia, R., & Vale, Z. (2023). Federated Genetic Programming: A Study About the Effects of Non-IID and Federation Size. In *Distributed Computing and Artificial Intelligence, 20th International Conference (DCAI 2023)* (pp. 193–202). https://doi.org/10.1007/978-3-031-38333-5_20

Ribeiro, B., Gomes, L., & Vale, Z. (2023). A Novel Federated Learning Approach to Enable Distributed and Collaborative Genetic Programming. In *Progress in Artificial Intelligence (EPIA 2023)* (pp. 195–206). https://doi.org/10.1007/978-3-031-49011-8_16

Ribeiro, B., Pereira, H., Gomes, L., & Vale, Z. (2023). Python-Based Ecosystem for Agent Communities Simulation. In *17th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2022)* (pp. 62–71). https://doi.org/10.1007/978-3-031-18050-7_7

Rincon, J., Julian, V., & Carrascosa, C. (2022). FLaMAS: Federated Learning Based on a SPADE MAS. *Applied Sciences*, *12*(7), 3701. https://doi.org/10.3390/app12073701

Rizk, E., Vlaski, S., & Sayed, A. H. (2020). Dynamic Federated Learning. *ArXiv*.

Rizk, Y., Awad, M., & Tunstel, E. W. (2020). Cooperative Heterogeneous Multi-Robot Systems. *ACM Computing Surveys*, *52*(2), 1–31. https://doi.org/10.1145/3303848

Rodrigues, N., Leitão, P., & Oliveira, E. (2014). Dynamic Composition of Service Oriented Multi-agent System in Self-organized Environments. *Proceedings of the 2014 Workshop on Intelligent Agents and Technologies for Socially Interconnected Systems*, 1–6. https://doi.org/10.1145/2655985.2655990

Rousset, A., Herrmann, B., Lang, C., & Philippe, L. (2016). A survey on parallel and distributed multi-agent systems for high performance computing simulations. *Computer Science Review*, *22*, 27–46. https://doi.org/10.1016/j.cosrev.2016.08.001

Ruiz de Gauna, D. E., Villalonga, C., & Sanchez, L. E. (2020). Multi-agent systems in the field of urbane-mobility: A Systematic Review. *IEEE Latin America Transactions*, *18*(12), 2186–2195. https://doi.org/10.1109/TLA.2020.9400447

Russell, S., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall Press.

Russo, M. (2020). A novel technique to self-adapt parameters in parallel/distributed genetic programming. *Soft Computing*, *24*(22), 16885–16894. https://doi.org/10.1007/s00500-020-04982-w

Sachindra, D. A., & Kanae, S. (2019). Machine learning for downscaling: the use of parallel multiple populations in genetic programming. *Stochastic Environmental Research and Risk Assessment*, *33*(8–9), 1497–1533. https://doi.org/10.1007/s00477-019-01721-y

Safarov, F., Akhmedov, F., Abdusalomov, A. B., Nasimov, R., & Cho, Y. I. (2023). Real-Time Deep Learning-Based Drowsiness Detection: Leveraging Computer-Vision and Eye-Blink Analyses for Enhanced Road Safety. *Sensors*, *23*(14), 6459. https://doi.org/10.3390/s23146459

Saint-Andre, P. (2011). RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. *Internet Engineering Task Force*.

Sen, P. C., Hajra, M., & Ghosh, M. (2020). Supervised Classification Algorithms in Machine Learning: A Survey and Review. *Advances in Intelligent Systems and Computing*, *937*, 99–111. https://doi.org/10.1007/978-981-13-7403-6_11/COVER

Shahid, O., Pouriyeh, S., Parizi, R. M., Sheng, Q. Z., Srivastava, G., & Zhao, L. (2021). Communication Efficiency in Federated Learning: Achievements and Challenges. *ArXiv*.

Shao, Z., Zhao, R., Yuan, S., Ding, M., & Wang, Y. (2022). Tracing the evolution of AI in the past decade and forecasting the emerging trends. *Expert Systems with Applications*, *209*, 118221. https://doi.org/10.1016/j.eswa.2022.118221

Sharma, A., Bahl, S., Bagha, A., Javaid, M., Shukla, D., & Haleem, A. (2020). Multi-agent system applications to fight COVID-19 pandemic. *Apollo Medicine*, *0*(0), 0. https://doi.org/10.4103/am.am_54_20

Shen, S., Zhu, T., Wu, D., Wang, W., & Zhou, W. (2022). From distributed machine learning to federated learning: In the view of data privacy and security. *Concurrency and Computation: Practice and Experience*, *34*(16). https://doi.org/10.1002/cpe.6002

Shen, W., Norrie, D. H., & Barthès, J.-P. A. (2019). *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*. CRC Press. https://doi.org/10.1201/9780429182112

Siau, K., & Wang, W. (2020). Artificial Intelligence (AI) Ethics. *Journal of Database Management*, *31*(2), 74–87. https://doi.org/10.4018/JDM.2020040105

Silva, B. M. P., Bernardino, H. S., & Barbosa, H. J. C. (2021). Human Activity Recognition Using Parallel Cartesian Genetic Programming. *2021 IEEE Congress on Evolutionary Computation (CEC)*, 474–481. https://doi.org/10.1109/CEC45853.2021.9504793

Silva, C., Faria, P., Ribeiro, B., Gomes, L., & Vale, Z. (2022). Demand Response Contextual Remuneration of Prosumers with Distributed Storage. *Sensors*, *22*(22). https://doi.org/10.3390/s22228877

Soomro, P. N., Abduljabbar, M., Castrillon, J., & Pericàs, M. (2023). Shisha: Online Scheduling Of CNN Pipelines On Heterogeneous Architectures. *Parallel Processing and Applied Mathematics: 14th International Conference, PPAM 2022, Gdansk, Poland, September 11–14, 2022, Revised Selected Papers, Part I*, 249–262. https://doi.org/10.1007/978-3-031-30442-2_19

Sousa, M. J., & Rocha, Á. (2021). Decision-Making and Negotiation in Innovation & Research in Information Science. *Group Decision and Negotiation*, *30*(2), 267–275. https://doi.org/10.1007/s10726-020-09712-z

Srirama, S. N., & Vemuri, D. (2023). CANTO: An actor model-based distributed fog framework supporting neural networks training in IoT applications. *Computer Communications*, *199*, 1–9. https://doi.org/10.1016/j.comcom.2022.12.007

Steinhardt, J., Koh, P. W., & Liang, P. (2017). *Certified Defenses for Data Poisoning Attacks*.

Substra. (2022, September 1). Owkin. https://www.owkin.com/substra

Tan, J., Khalili, R., Karl, H., & Hecker, A. (2022). *Multi-Agent Distributed Reinforcement Learning for Making Decentralized Offloading Decisions*. https://doi.org/10.1109/INFOCOM48880.2022.9796717

The TensorFlow Federated Authors. (2018, December 12). *TensorFlow Federated*. Google. https://www.tensorflow.org/federated?hl=pt-br

Tian, J., Smith, J. S., & Kira, Z. (2022). *FedFOR: Stateless Heterogeneous Federated Learning with First-Order Regularization*.

Vemuri, V. K. (2020). *The Hundred-Page Machine Learning Book* (Vol. 22). https://doi.org/10.1080/15228053.2020.1766224

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., & Rellermeyer, J. S. (2021). A Survey on Distributed Machine Learning. *ACM Computing Surveys*, *53*(2), 1–33. https://doi.org/10.1145/3377454

Vergaray, A. D., Peralta Robles, W. F., & Salazar Jiménez, J. A. (2023). The Impact of Chatbots on Customer Satisfaction: A Systematic Literature Review. *TEM Journal*, 1407–1417. https://doi.org/10.18421/TEM123-21

Vincent, J. (2016). *Twitter taught Microsoft's AI chatbot to be a racist asshole in less than a day*. https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist

Wang, B., Xie, J., & Atanasov, N. (2022). *DARL1N: Distributed multi-Agent Reinforcement Learning with One-hop Neighbors*.

Wang, H., Kaplan, Z., Niu, D., & Li, B. (2020a). Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 1698–1707. https://doi.org/10.1109/INFOCOM41043.2020.9155494

Wang, H., Kaplan, Z., Niu, D., & Li, B. (2020b). Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 1698–1707. https://doi.org/10.1109/INFOCOM41043.2020.9155494

Wang, J., Hong, Y., Wang, J., Xu, J., Tang, Y., Han, Q.-L., & Kurths, J. (2022). Cooperative and Competitive Multi-Agent Systems: From Optimization to Games. *IEEE/CAA Journal of Automatica Sinica*, *9*(5), 763–783. https://doi.org/10.1109/JAS.2022.105506

Wang, T., Du, Y., Gong, Y., Choo, K.-K. R., & Guo, Y. (2023). Applications of Federated Learning in Mobile Health: Scoping Review. *Journal of Medical Internet Research*, *25*, e43006. https://doi.org/10.2196/43006

Wei, X., Li, Q., Liu, Y., Yu, H., Chen, T., & Yang, Q. (2019). Multi-Agent Visualization for Explaining Federated Learning. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 6572–6574. https://doi.org/10.24963/ijcai.2019/960

Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J., & Zhang, W. (2023). A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, *14*(2), 513–535. https://doi.org/10.1007/s13042-022-01647-y

Witt, L., Heyer, M., Toyoda, K., Samek, W., & Li, D. (2022). *Decentral and Incentivized Federated Learning Frameworks: A Systematic Literature Review*.

Wong, A., Bäck, T., Kononova, A. V., & Plaat, A. (2023). Deep multiagent reinforcement learning: challenges and directions. *Artificial Intelligence Review*, *56*(6), 5023–5056. https://doi.org/10.1007/s10462-022-10299-x

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2nd ed.). Wiley Publishing.

Wrona, Z., Buchwald, W., Ganzha, M., Paprzycki, M., Leon, F., Noor, N., & Pal, C.-V. (2023). Overview of Software Agent Platforms Available in 2023. *Information*, *14*(6), 348. https://doi.org/10.3390/info14060348

Wu, H., Zhang, Z., Guan, C., Wolter, K., & Xu, M. (2020). Collaborate Edge and Cloud Computing With Distributed Deep Learning for Smart City Internet of Things. *IEEE Internet of Things Journal*, *7*(9), 8099–8110. https://doi.org/10.1109/JIOT.2020.2996784

Xie, J., & Liu, C.-C. (2017). Multi-agent systems and their applications. *Journal of International Council on Electrical Engineering*, *7*(1), 188–197. https://doi.org/10.1080/22348972.2017.1348890

Yang, Q., Fan, L., & Yu, H. (Eds.). (2020). *Federated Learning*. *12500*. https://doi.org/10.1007/978-3-030-63076-8

Yang, Y., & Wang, J. (2020). An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. *ArXiv*.

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, *31*(7), 1235–1270. https://doi.org/10.1162/neco_a_01199

Zhang, B., Zhu, J., & Su, H. (2023). Toward the third generation artificial intelligence. *Science China Information Sciences*, *66*(2), 121101. https://doi.org/10.1007/s11432-021-3449-x

Zhang, K., Song, X., Zhang, C., & Yu, S. (2022). Challenges and future directions of secure federated learning: a survey. *Frontiers of Computer Science*, *16*(5), 165817. https://doi.org/10.1007/s11704-021-0598-z

Zhang, S. Q., Lin, J., & Zhang, Q. (2022). A Multi-Agent Reinforcement Learning Approach for Efficient Client Selection in Federated Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(8), 9091–9099. https://doi.org/10.1609/aaai.v36i8.20894

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018a). *Federated Learning with Non-IID Data*. https://doi.org/10.48550/arXiv.1806.00582

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018b). *Federated Learning with Non-IID Data*. https://doi.org/10.48550/arXiv.1806.00582

Zhu, H., Xu, J., Liu, S., & Jin, Y. (2021). *Federated Learning on Non-IID Data: A Survey*.

Znaidi, M. R., Gupta, G., & Bogdan, P. (2022). *Secure Distributed/Federated Learning: Prediction-Privacy Trade-Off for Multi-Agent System*.