



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING
Ciclo 37

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI
Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

ENHANCING FEDERATED LEARNING THROUGH
DISTRIBUTED LEDGER TECHNOLOGY INTEGRATION

PRESENTATA DA: NICOLÒ ROMANDINI

COORDINATORE DOTTORATO:
PROF.SSA ILARIA BARTOLINI

SUPERVISORE:
PROF.SSA REBECCA MONTANARI
Co-SUPERVISORE:
PROF. PAOLO BELLAVISTA

ESAME FINALE ANNO 2025

ABSTRACT

In today's data-driven world, vast amounts of information power Machine Learning (ML) models for a wide range of applications. However, this data flow raises significant privacy concerns, as individuals are often reluctant to share personal information, especially given increasing regulations on data protection. Federated Learning (FL) offers a solution by training ML models directly on users' devices and sending only model updates to a central server. This distributed approach enables collaboration without sharing personal data, but challenges remain. Centralization may lead to server bottlenecks, reduced resilience, and fairness concerns if updates from certain devices are prioritized. Additionally, the lack of transparency and accountability can erode trust, while security risks, such as data poisoning and model inversion attacks, further complicate FL. Deployment can be costly and time-consuming, and participants may also lack incentives. Regulatory compliance, such as ensuring the *right to be forgotten*, adds complexity, as removing data from FL models without full retraining is challenging.

This dissertation proposes integrating Distributed Ledger Technologies (DLTs) with FL to address these challenges. DLT decentralizes the aggregation process, enhancing security, transparency, and fairness through immutable record-keeping and traceability. Two DLT-based architectures are presented: one blockchain-based and the other using a Directed Acyclic Graph (DAG) for scalability. These approaches utilize Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) to track contributions and verify participants. Furthermore, a DLT-based FL as a Service (FLaaS) is introduced to simplify deployment, incorporating model validation to mitigate poisoning attacks and token-based incentives to encourage participation. Additionally, this dissertation outlines design guidelines for Federated Unlearning (FU), covering key evaluation metrics, existing techniques, and future research. Finally, a new unlearning algorithm is proposed to address adversarial settings and protect model integrity. These contributions pave the way for more secure, transparent, and resilient FL systems that can meet the needs of next-generation data-driven applications.

CONTENTS

ACRONYMS	XI
1 INTRODUCTION	1
2 BACKGROUND	5
2.1 Federated Learning	5
2.1.1 Mathematical Formulation of FL	7
2.1.2 Types of FL	8
2.1.3 Promising Research Paths	9
2.2 Distributed Ledger Technologies	9
2.2.1 Blockchain	10
2.2.2 Directed Acyclic Graph Ledger	11
2.2.3 Decentralized Identifiers	13
3 CHALLENGES IN FEDERATED LEARNING	15
3.1 Centralization Risks	15
3.2 Security Threats	16
3.3 Deployment and Participation	17
3.4 Regulatory Compliance	18
3.5 Summary of Key Contributions and Proposed Solutions	19
4 DLT AS AN ENABLING BACKBONE FOR TRUSTWORTHY FL	21
4.1 Motivation	21
4.2 Related Work	23
4.2.1 Blockchain-based FL	23
4.2.2 DAG-based FL	23
4.3 FlowChain: a Blockchain-based Architecture	24
4.3.1 Architecture and Workflow	24
4.3.2 Implementation	27
4.3.3 Experimental Evaluation	30
4.3.4 Conclusion	35
4.4 FRAMH: a FlowChain Deployment for Risk-Based Authorization in Healthcare	36
4.4.1 Architecture	37
4.4.2 Experimental Evaluation	39
4.4.3 Conclusion	44
4.5 FETA: a DAG-based Architecture	45
4.5.1 Architecture	45
4.5.2 Workflow	48

Contents

4.5.3	Experimental Evaluation	51
4.5.4	Conclusion	56
5	TRUSTWORTHY FL AS A SERVICE	59
5.1	Motivation	59
5.1.1	Trustworthiness	60
5.1.2	Privacy-preserving Techniques	61
5.1.3	Authentication and Authorization	61
5.1.4	Incentive and Penalization	62
5.2	Related Work	62
5.2.1	Accountability and Fairness	63
5.2.2	Validation Mechanisms	63
5.2.3	Reputation and Weighted Contributions	64
5.3	TruFLaaS: a Framework for Trustworthy FLaaS	64
5.3.1	Architecture	65
5.3.2	Workflow	68
5.3.3	Experimental Evaluation	73
5.3.4	Conclusion	80
6	FEDERATED UNLEARNING: TAXONOMY, APPROACHES, AND SOLUTIONS	81
6.1	Motivation	81
6.2	Background	82
6.2.1	Machine Unlearning	82
6.2.2	Catastrophic Forgetting and Differential Privacy	83
6.3	Federated Unlearning	85
6.3.1	Objectives	85
6.3.2	Challenges of Adapting MU in FL Settings	86
6.3.3	Entity Performing Unlearning	88
6.4	Design and Implementation Guidelines for Efficient FU Algorithms	89
6.4.1	Requirements of Federated Unlearning Algorithms	89
6.4.2	Metrics to Assess Unlearning	90
6.5	Literature Review of Federated Unlearning Methods	95
6.5.1	Client Unlearning	96
6.5.2	Class Unlearning	101
6.5.3	Sample Unlearning	101
6.6	Distilled Lessons Learned	104
6.7	FedUP: Efficient Federated Unlearning Through Pruning	106
6.7.1	Architecture and Workflow	106
6.7.2	Technical Insights	110
6.7.3	Experimental Evaluation	112
6.7.4	Conclusion	116
7	CONCLUSION AND FUTURE WORK	117
	BIBLIOGRAPHY	119

LIST OF FIGURES

2.1	Difference between ML and FL approaches.	6
2.2	The blockchain data structure.	10
2.3	The Tangle data structure.	12
3.1	Main challenges in FL.	15
4.1	FlowChain architecture.	25
4.2	Power consumption for the first experiment.	33
4.3	Power consumption for the second experiment.	33
4.4	Power consumption statistics of a blockchain node.	34
4.5	Power consumption statistics of a client connector.	34
4.6	Considered risk levels.	36
4.7	FRAMH architecture	38
4.8	Performance comparison between centralized and federated approaches.	42
4.9	ROC comparisons between the centralized and federated approaches.	43
4.10	FETA architecture.	46
4.11	Publishing local models.	47
4.12	Authorization workflow.	48
4.13	Aggregating local models.	49
4.14	Accuracy comparison in the first experiment.	52
4.15	Latency comparison in the first and second experiments.	53
4.16	Power consumption comparison in the first and second experiments.	55
5.1	TruFLaaS architecture.	65
5.2	Authorization workflow.	69
5.3	Validation and aggregation workflow.	72
5.4	Predictive maintenance with heterogeneous data distribution.	77
5.5	Botnet attack detection under heterogeneous data distribution.	77
5.6	Predictive maintenance with rare case heterogeneous data distribution.	78
5.7	Botnet attack detection with rare case heterogeneous data distribution	78
5.8	Predictive maintenance under model forging attack.	79
5.9	Botnet attack detection under model forging attack.	79
6.1	FU overview.	84
6.2	Visualization of FU objectives.	86
6.3	Unlearning procedure after detecting malicious clients.	107
6.4	Normalized squared difference between averaged benign and malicious weights.	108

LIST OF TABLES

4.1	Memory (MB) and CPU (%) usage.	34
4.2	Comparison of ML approaches that use the MIMIC-III Dataset.	43
4.3	Latency for authorization procedures and FL training.	54
4.4	Comparison of each component’s power consumption (W).	55
4.5	Comparison of memory (MB) and CPU (%) usage for each component.	56
4.6	Power consumption (W), memory (MB), and CPU (%) usage of the AS.	56
5.1	Comparison of related work based on their features.	62
5.2	Configuration parameters.	68
5.3	Botnet attack detection results.	75
6.1	Summary of the main metrics in FU literature.	90
6.2	Summary of FU referenced works.	95
6.3	Unlearning performance with CIFAR-10.	114
6.4	Unlearning performance with FashionMNIST.	114
6.5	Comparison with FedEraser.	115

ACRONYMS

AS	Authorization Service
AUROC	Area Under the Receiver Operating Characteristics
CA	Certificate Authority
CCPA	California Consumer Privacy Act
CID	Content IDentifier
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
dApp	Decentralized Application
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
DON	Decentralized Oracle Network
DoS	Denial-of-Service
DP	Differential Privacy
EU	European Union
FedAVG	Federated Averaging
FedSGD	Federated SGD
FL	Federated Learning
FLaaS	Federated Learning as a Service
FTL	Federated Transfer Learning
FU	Federated Unlearning
GDPR	General Data Protection Regulation
HFL	Horizontal Federated Learning
IID	Independent and Identically Distributed
IIoT	Industrial Internet of Things
IoMT	Internet of Medical Things
IoT	Internet of Things
IPFS	InterPlanetary File System
KD	Knowledge Distillation
KL	Kullback-Leibler
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MIA	Membership Inference Attack
MIB	Membership Inference Backdoor
ML	Machine Learning
MLaaS	Machine Learning as a Service

Acronyms

MLP	Multilayer Perceptron
MU	Machine Unlearning
P2P	Peer-to-Peer
PKI	Public Key Infrastructure
PoS	Proof of Stake
PoW	Proof of Work
ReLU	Rectified Linear Unit
SAPE	Symmetric Absolute Percentage Error
SGA	Stochastic Gradient Ascent
SGD	Stochastic Gradient Descent
SOTA	State-of-the-Art
SVM	Support Vector Machine
VC	Verifiable Credential
VFL	Vertical Federated Learning
VP	Verifiable Presentation

1 INTRODUCTION

The rapid expansion of the Internet of Things (IoT) contributes to the design and development of next-generation services [1]. By 2025, the number of IoT devices will be estimated to exceed 20 billion [2], generating an immense data flow. This information explosion creates new opportunities to develop intelligent Machine Learning (ML) systems capable of driving innovations in various sectors, from healthcare to smart cities and industrial automation. However, conventional ML approaches, which require data centralization for training, face significant challenges in this context. The widespread distribution of data across different devices complicates its management, raising privacy concerns and introducing regulatory limitations, such as the European Union (EU) General Data Protection Regulation (GDPR) [3] and the California Consumer Privacy Act (CCPA) [4]. Furthermore, continuous raw data transfer from edge devices to central servers can overload network infrastructure and increase latency, making real-time analysis more difficult. In this scenario, Federated Learning (FL) [5] has emerged as a promising solution to overcome these challenges. FL is a distributed ML paradigm that enables multiple clients to collaboratively train a global model without sharing their raw data. In this framework, each client computes local model updates based on its private dataset and transmits them to a central server, aggregating them to refine the global model. This process preserves data privacy and significantly reduces the communication overhead associated with traditional centralized ML approaches. Healthcare, finance, and manufacturing sectors can benefit from this collaborative learning without compromising sensitive data, enabling intelligent applications in various environments.

However, despite the advantages of FL, several challenges limit its full potential and adoption [6]. Centralization in a distributed framework is a significant issue. As the number of participating devices grows, the central server may struggle to handle the increased communication load, leading to bottlenecks that degrade overall system performance. Moreover, centralization diminishes the system's resilience, as the failure of the central server could halt the entire learning process [7]. Another critical aspect of centralization is its impact on fairness [8]. The central server is necessary for aggregating updates from various devices; however, it often functions as a black box, lacking transparency in its decision-making processes. If the server does not treat all participants equitably, biases can be introduced into the global model. For instance, the server may prioritize

Introduction

updates from devices with superior computational capabilities or more stable connections, potentially sidelining contributions from weaker or resource-constrained devices. Moreover, there is a risk that certain participants could influence the server's decisions to favor their updates, leading to the exclusion of other valuable contributions. This unequal treatment can distort the model, favoring data from select participants and undermining fairness while diminishing the dataset's overall diversity and affecting the global model's performance. In addition to centralization and fairness issues, trust is another critical challenge in FL [9], where multiple stakeholders interact. The inherent lack of transparency surrounding the server's decision-making processes can lead to suspicions among participants about how their contributions are utilized. Without clear insights into the aggregation of updates and the criteria used for prioritization, stakeholders may question the model's integrity and the collaboration's fairness. Moreover, the absence of accountability mechanisms exacerbates these trust issues. When participants do not have assurances that their data will be handled responsibly or that any biases in the model will be addressed, their willingness to collaborate diminishes. This lack of accountability can create an environment where contributors feel powerless, leading to concerns about potential misuse of their data or the marginalization of their contributions. Security and privacy are also major concerns in FL [10]. Although the approach keeps raw data on individual devices, it remains vulnerable to data or model poisoning and backdoor attacks [11], where malicious clients can compromise the integrity of the global model. Moreover, model inversion attacks [12] pose risks by potentially inferring sensitive information from shared model updates. Moreover, setting up a FL process could be costly and time-consuming, hindering its successful widespread adoption. One of the primary challenges lies in coordinating and managing diverse participant devices, each with varying computational capabilities, data quality, and connectivity [13]. Additionally, ensuring that all nodes can effectively contribute to the FL process in a secure and trustworthy manner can complicate deploying the FL framework. Participants may also lack sufficient incentives to engage in FL. This is particularly problematic in scenarios where data is sensitive or proprietary. Organizations may be reluctant to share their data, fearing that their competitive advantage could be compromised without adequate recognition or compensation. Finally, ensuring regulatory compliance [3], especially concerning the *right to be forgotten*, poses a significant challenge in the context FL. This legal principle requires that individuals have the right to request the deletion of their data from systems and records, which becomes particularly complicated within the context of ML models. In traditional centralized systems, removing data contributions could be more manageable, as data resides in a single location. However, in FL, where data remains on individual devices, and only model updates are aggregated by a central server, deleting specific data contributions from a trained model is computationally expensive and technically intricate. The complexities arise be-

cause each participant's contribution influences the global model, making it challenging to isolate and remove the effects of any single data point without retraining the entire model.

CONTRIBUTIONS BEYOND THE STATE OF ART

This dissertation proposes leveraging Distributed Ledger Technologies (DLTs) to enhance FL by addressing its inherent challenges. By decentralizing the aggregation process, DLT improves security and reliability, reducing the risks associated with a central point of failure. Additionally, it fosters transparency and fairness through immutable record-keeping, allowing participants to verify and audit model updates, which builds trust and mitigates bias. Furthermore, DLT enhances traceability and accountability, enabling tracking updates and identifying malicious activities.

- Two DLT-based architectures are presented, one based on an enterprise-ready blockchain solution and the other on a Directed Acyclic Graph (DAG)-based efficient and scalable ledger. Both architectures utilize advanced Decentralized Identifiers (DIDs) and associated technologies to identify participants and track their contributions within the FL process. This approach enhances participant verification and ensures accountability by clearly delineating who contributed what, enabling a transparent and trustworthy collaborative environment.
- To simplify the deployment of FL processes and address the associated challenges, this dissertation proposes the first robust framework for DLT-based Federated Learning as a Service (FLaaS). This framework facilitates the implementation of FL while ensuring a trustworthy, secure, and reliable environment, promoting effective participant collaboration. It incorporates a model validation mechanism to prevent the poisoning of the global model. This mechanism can identify and mitigate potential threats before they affect the FL process by evaluating the quality and integrity of the model updates received from individual participants. This proactive approach helps aggregate only reliable and accurate contributions. Additionally, the framework incorporates token-based incentives to motivate active participation, rewarding clients for their contributions while deterring malicious behaviors. Promoting a collaborative environment can enhance data diversity, improving model performance and accuracy.
- Moreover, this dissertation proposes comprehensive design guidelines for Federated Unlearning (FU) [14], a novel technique to ensure regulatory compliance and enhance security. FU enables the removal of specific data contributions from trained FL model, whether due to privacy requests under the *right to be forgotten* or to eliminate malicious influence.

This is achieved while maintaining the model's integrity, eliminating the need for complete retraining. The dissertation categorizes existing FU techniques based on the entity responsible for unlearning and their methodological approach. It also examines key evaluation metrics, such as unlearning effectiveness, computational overhead, and model degradation. The study identifies open challenges and outlines future research directions to improve FU strategies.

- Finally, recognizing a gap in managing adversarial settings, this dissertation introduces a novel unlearning algorithm to address challenges posed by malicious participants. This approach is based on pruning and is specifically designed to protect the model's integrity and security in the presence of adversarial contributions, providing a resilient solution to enhance security and compliance in FL environments.

THESIS STRUCTURE AND ORGANIZATION

The remainder of this work is structured as follows. The remainder of this work is organized as follows: Chapter 2 provides a comprehensive overview of FL, outlining its foundational principles and including essential background on DLTs to address the challenges involved and establish a basis for further developments. Chapter 3 examines specific challenges within FL, such as risks of centralization, security vulnerabilities, deployment and participation issues, regulatory compliance, and limitations related to its distributed nature. Chapter 4 explores how DLTs can serve as a foundational infrastructure to strengthen trust in FL systems. This chapter introduces various architectures and their implementations to integrate FL with DLTs, focusing on enhancing trust, scalability, and data integrity. Chapter 5 presents the concept of trustworthy FLaaS, stressing security and transparency in providing FL as a service. It also introduces TruFLaaS, the first framework incorporating blockchain, a validation mechanism, and a reputation and reward system to securely and reliably deliver FLaaS. Chapter 6 explores the concept of FU, providing a taxonomy of current approaches and proposing a new unlearning algorithm specifically designed to remove malicious contributions from the FL model. Finally, Chapter 7 synthesizes the dissertation's key findings, contributions, and implications, suggesting directions for future research.

2 BACKGROUND

This chapter provides an overview of FL, highlighting its fundamental principles and the mechanisms that underpin this innovative approach to distributed ML. Furthermore, it discusses the technologies and tools employed in this dissertation to address these issues and explore new paths for development. It is important to note that this is not meant to be exhaustive; instead, it aims to introduce the fundamental concepts necessary for a comprehensive understanding of the subsequent sections of this dissertation.

2.1 FEDERATED LEARNING

FL is an emerging distributed ML framework that enables multiple clients, such as mobile devices, sensors, or institutions, to collaboratively train a shared ML model without directly exchanging their data [15]. This paradigm ensures that the raw data remains local on each client, and only model updates or gradients are transmitted to a central server for aggregation. This approach addresses privacy concerns, reduces communication costs, and enables collaborative learning across geographically distributed datasets [10]. FL has garnered significant attention in recent years due to its potential to revolutionize industries. FL was first introduced by Google in 2016 [5] as a means to improve mobile applications by training predictive models across millions of devices without collecting raw data. However, its potential applications extend far beyond mobile devices, encompassing sectors such as healthcare, finance, and the IoT, where data privacy is critical [16].

As illustrated in Figure 2.1, FL (b) distinguishes itself from traditional centralized ML (a) in several ways. In the latter approach, data is collected from various sources and aggregated at a central server for model training, raising significant concerns regarding data privacy and security, particularly in sensitive sectors governed by strict regulations, such as the GDPR [3].

In contrast, FL operates on the principle of distributed data handling, enabling model training directly on local devices or data sources. The process begins with the aggregator initializing a global model with specified learning parameters, which each client downloads. Clients then compute model updates using their local datasets and send these updates back to the aggregator. The

Background

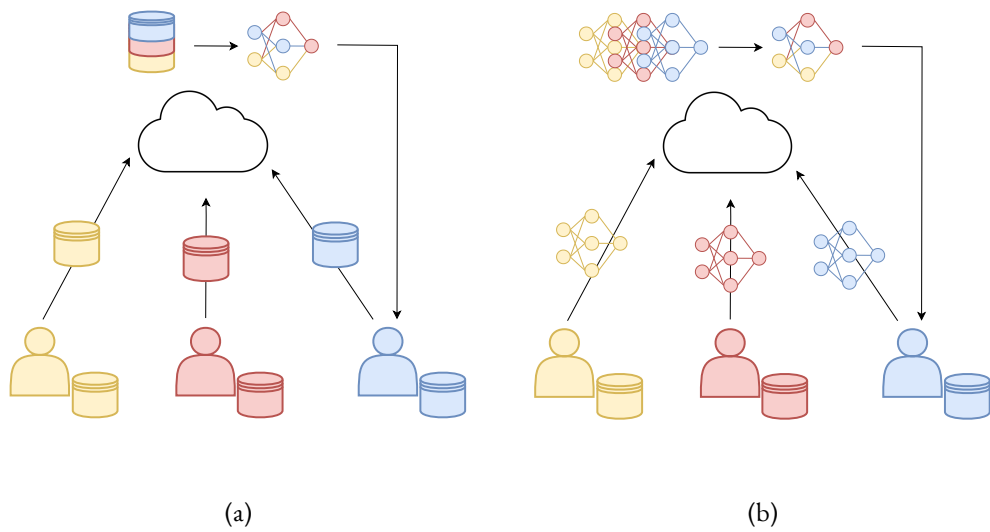


Figure 2.1: Difference between ML (a) and FL (b) approaches.

aggregator combines these local updates to refine and enhance the global model, employing various aggregation algorithms based on the type of updates received. In the 2016 Google proposal, McMahan et al. [5] introduced Federated SGD (FedSGD), an algorithm that leverages Stochastic Gradient Descent (SGD) for federated optimization. In each training round, clients perform a single SGD step on the current model and transmit the computed gradients back to the server. They also proposed Federated Averaging (FedAVG), a variation where clients send model weights instead of gradients, allowing multiple local updates before transmission and enhancing efficiency. This iterative process continues as clients download the improved global model, compute further updates, and repeat until global training is complete. Moreover, new aggregation algorithms have been developed to enhance model training across diverse client datasets. One notable example is FedProx [9], which introduces a proximal term during the aggregation process to address the challenges posed by data heterogeneity. By penalizing deviations from the global model, FedProx encourages clients to remain close to the global state, improving convergence and robustness. The SCAFFOLD algorithm [17] is a further advancement in this domain, which employs control variates to stabilize and enhance the aggregation of updates from clients with non-Independent and Identically Distributed (IID) data distributions. By using local control variates, SCAFFOLD adjusts the updates contributed by each client, reducing the variance and promoting more consistent contributions to the global model. This method not only accelerates convergence but also improves the robustness of the training process.

This distributed architecture of FL significantly mitigates the risk of exposing sensitive information, as only model updates are communicated back to the central server, thereby preserving privacy and ensuring compliance with regulatory requirements. While most FL strategies rely on synchronous aggregation, i.e., updating the global model only after contributions from all participating clients are received, recent advancements in asynchronous approaches [18] enable continuous updates to the global model, allowing it to be sent back to clients for additional training rounds even if some have yet to contribute.

2.1.1 MATHEMATICAL FORMULATION OF FL

The core objective of FL is to optimize a global ML model based on the data distributed across multiple clients. A typical FL setting has K clients, each with a local dataset D_k where $k = 1; 2; \dots; K$. The goal is to minimize a global objective function $F(\mathbf{w})$ that combines the local loss functions of each client. This global objective is represented as:

$$\min_{\mathbf{w}} F(\mathbf{w}) = \sum_{k=1}^K \frac{n_k}{n} F_k(\mathbf{w}) \tag{2.1}$$

where:

- \mathbf{w} denotes the global model parameters,
- $F_k(\mathbf{w})$ is the local loss function of the k -th client,
- n_k represents the number of data points held by client k ,
- $n = \sum_{k=1}^K n_k$ is the total number of data points across all clients.

Each client k solves its local optimization problem by minimizing $F_k(\mathbf{w})$ based on its local dataset D_k :

$$F_k(\mathbf{w}) = \frac{1}{n_k} \sum_{(x_i, y_i) \in D_k} \ell(\mathbf{w}; x_i, y_i) \tag{2.2}$$

where $\ell(\mathbf{w}; x_i, y_i)$ is the loss of the model \mathbf{w} on the data sample (x_i, y_i) from client k .

Once the local optimization is complete, the client transmits its model updates (typically in the form of gradients or model weights) to a central server. The server aggregates these updates to compute the new global model, often using the FedAVG [5] algorithm, which calculates the updated global model $\mathbf{w}^{(t+1)}$ as a weighted average of the client models:

$$\mathbf{w}^{(t+1)} = \sum_{k=1}^K \frac{n_k}{n} \mathbf{w}_k^{(t)} \quad (2.3)$$

where $\mathbf{w}_k^{(t)}$ is the local model of client k after round t of training. The process continues iteratively until the global model converges to an optimal solution.

2.1.2 TYPES OF FL

FL can be classified into three main types based on the distribution of data across clients and the types of learning scenarios [19]:

- *Horizontal Federated Learning (HFL)*: This type of FL is applicable when the datasets on different clients share the same feature space but contain different data samples (i.e., different rows but similar columns). In this scenario, clients contribute to training the same model on data with similar characteristics, making it particularly useful in cases where multiple organizations handle similar types of data but do not want to pool that data together [8]. A prime example of HFL is multiple hospitals collaborating to train a diagnostic model using patient records from different regions [20].
- *Vertical Federated Learning (VFL)*: In VFL, different clients hold datasets that contain different features but refer to the same set of data points (i.e., different columns but the same rows). This scenario arises in cases where organizations possess complementary information about the same users or entities [21]. For example, a financial institution and an e-commerce company may want to collaborate on building a credit scoring model, with each party contributing different attributes (e.g., transaction history and purchasing behavior) [22]. The challenge here lies in aligning the data and ensuring that only necessary information is shared.
- *Federated Transfer Learning (FTL)*: FTL applies when both the feature spaces and the data samples differ across clients. This type of FL leverages transfer learning techniques to enable collaboration between entities that do not share similar data distributions or data points. For instance, an online retailer might collaborate with a telecommunications company to enhance product recommendations by learning from data derived from different domains [23]. FTL is particularly useful in scenarios where data heterogeneity is extreme, but collaborative learning is still desired.

2.1.3 PROMISING RESEARCH PATHS

The field of FL is evolving rapidly, with several promising research directions shaping its future [15, 24]. One noteworthy area of focus is communication efficiency, which aims to optimize the interaction between clients and the central server to mitigate bandwidth limitations and latency issues. Data heterogeneity presents another challenge, as models must perform effectively despite the non-IID data typically found across client devices. Ensuring privacy is also required, leading to the development of advanced techniques like Differential Privacy (DP) [25] and secure multi-party computation to protect sensitive information during model training. Researchers are also exploring personalization methods that tailor models to individual clients while benefiting from the global model. As the number of clients increases, scalability becomes crucial, necessitating algorithms that sustain high performance in large networks. Other notable research areas include improving robustness to attacks to ensure model integrity and addressing issues of fairness and bias mitigation to guarantee impartial outcomes. Lastly, regulatory compliance is also increasingly important, requiring adherence to data protection laws and regulations to ensure trustworthy data handling. These research directions collectively emphasize the potential of FL and the challenges that must be addressed to fully exploit its benefits in various applications. As will be discussed in Section 3, this dissertation focuses on just some of these research directions, examining their implications and potential solutions.

2.2 DISTRIBUTED LEDGER TECHNOLOGIES

In recent years, DLTs have obtained attention for their ability to enhance security across untrusted parties. One of the key innovations of DLTs is their decentralized and distributed nature, as information is shared across a network of nodes rather than being controlled by a single central authority. This decentralized structure enables each node to maintain an up-to-date copy of the data, providing transparency and reducing dependency on intermediaries. The decentralized design of DLTs, enabled by a Peer-to-Peer (P2P) network, also offers a significant advantage in terms of security. Traditional databases, which rely on centralized control, allow for potential modifications to stored data. In contrast, DLTs utilize an append-only model, which ensures that data, once recorded, becomes immutable and resistant to unauthorized changes. The absence of a central authority also reduces the risk of single points of failure, a common vulnerability in centralized systems. To ensure data integrity and consensus across a decentralized network, DLTs employ cryptographic protocols. These mechanisms allow participants to collectively validate and agree on transactions, ensuring that only legitimate updates are recorded. By decentralizing the valida-

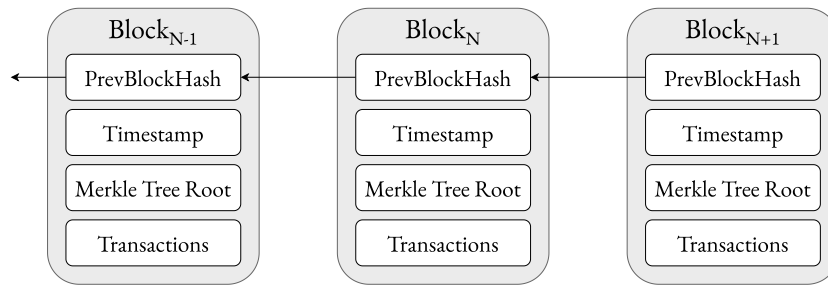


Figure 2.2: The blockchain data structure.

tion process, DLTs foster trust among participants without the need for a central governing entity, further strengthening the reliability and security of the system.

2.2.1 BLOCKCHAIN

Blockchain technology is a foundational framework for decentralized digital interactions, characterized by its unique structure and consensus mechanisms. At its core, a blockchain consists of a series of linked blocks, each containing a collection of transactions or data secured through cryptographic hashing. Figure 2.2 shows the blockchain data structure. This design ensures that once data is recorded, it cannot be altered without modifying all subsequent blocks, thereby maintaining integrity and trust throughout the network. The blockchain operates as a distributed ledger, with each participant or node maintaining a copy, enhancing transparency and reducing data manipulation risk. Satoshi Nakamoto introduced the concept of blockchain in a whitepaper [26] published in 2008. Nakamoto presented a digital currency operating without a central authority, utilizing cryptographic techniques to ensure security and integrity. The first block of the Bitcoin blockchain, known as the *genesis block*, was mined by Nakamoto in January 2009, marking the beginning of a revolutionary movement in digital finance. As a cornerstone of blockchain, there are consensus mechanisms, such as Proof of Work (PoW) and Proof of Stake (PoS). They play a crucial role in validating transactions and maintaining the blockchain's state. These protocols enable distributed network participants to agree on the legitimacy of transactions without a central authority, fostering a secure and collaborative environment for data management. In PoW, for example, the nodes known as *miners* compete to solve complex mathematical puzzles, validating transactions and adding new blocks to the chain. This operation, known as *mining*, necessitates considerable computational resources and energy, providing a high level of security while also being highly energy-intensive. Although PoW has established itself as a reliable mechanism for ensuring security and decentralization, it has raised significant environmental concerns due to its substantial energy demands, leading to a considerable impact on overall energy consumption and

sustainability [27]. In contrast, in PoS, validators are chosen based on the amount of cryptocurrency they hold and are willing to stake, incentivizing honest participation [28]. Blockchains can be broadly categorized into permissionless and permissioned systems, each serving distinct purposes and use cases. Permissionless blockchains, often called public blockchains, allow anyone to participate in the network without restrictions. This openness enables users to contribute to transaction validation and participate in governance, fostering decentralization and inclusivity. However, this model can face challenges such as scalability and increased vulnerability to malicious attacks, as anyone can join the network. In contrast, permissioned blockchains restrict access to a select group of participants who are granted specific rights and responsibilities. These private networks often prioritize efficiency and control, making them suitable for enterprises and organizations that require greater privacy and regulatory compliance. Limiting participation allows permissioned blockchains to achieve faster transaction speeds and enhanced security, though at the cost of decentralization. Each type offers unique benefits and drawbacks, making them suitable for different applications depending on the desired transparency, control, and scalability level. Additionally, integrating smart contracts into blockchain technology has further expanded its capabilities. These programs enable Decentralized Applications (dApps) to perform computations without the need for intermediaries. Smart contracts operate directly on the blockchain, leveraging its security and transparency to ensure transaction trust and reliability. However, smart contracts could only rely on information already stored on the blockchain. This is where Decentralized Oracle Networks (DONs) are essential. These networks connect the blockchain to the outside world, supplying smart contracts with reliable and tamper-resistant data inputs, such as market prices, weather conditions, or real-world events. By utilizing decentralized oracle networks, smart contracts can make informed decisions based on real-time information while preserving the trustless nature inherent to blockchain technology. Combining blockchain's structural integrity with the automation provided by smart contracts creates a new paradigm for decentralized applications. This innovation paves the way for advancements in finance, supply chain management, and various other fields.

2.2.2 DIRECTED ACYCLIC GRAPH LEDGER

DAGs represent an innovative approach to data structures that have gained considerable interest in recent years due to their ability to enhance scalability and efficiency. Unlike traditional blockchain systems that rely on a linear arrangement of blocks, DAGs utilize a more flexible, branching structure, allowing transactions to connect to multiple predecessors. This interconnected network facilitates parallel processing of transactions, significantly improving throughput and reducing latency. The asynchronous validation process enables various nodes to con-

Background

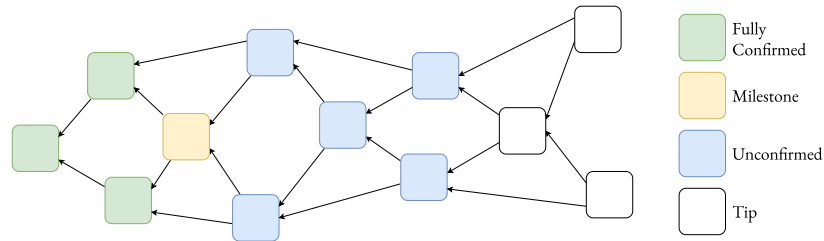


Figure 2.3: The Tangle data structure.

firm transactions simultaneously, eliminating the need for every node to validate each transaction. Additionally, DAGs do not depend on energy-intensive mining processes, making them a more energy-efficient alternative among rising environmental concerns associated with conventional blockchain technologies.

IOTA [29] is a leading example of a DAG tailored for the IoT, employing a unique structure known as the Tangle, shown in Figure 2.3. Like traditional blockchain systems, IOTA networks can be implemented in private and public configurations. IOTA differentiates between two key roles: clients and nodes. Clients submit transactions to the network, while nodes verify their accuracy and integrate them into the Tangle. In addition to these basic roles, IOTA introduces specialized node types: the *Coordinator* and *Permanode*. The Coordinator is the sole entity responsible for generating *milestones*, trusted signed transactions that nodes use to validate other transactions. The integrity of these milestones is ensured through cryptographic signatures, making it impossible to forge them. A transaction only receives confirmation when directly or indirectly linked to a validated milestone. However, it's important to highlight that the Coordinator's role is intended to be temporary, as future updates aim to eliminate this centralization point. Permanodes serve another vital function by maintaining a comprehensive history of all transactions. This is particularly important in scenarios where nodes might be constrained devices with limited storage capacity. Permanodes prevent data loss by periodically archiving transaction histories and managing the pruning of older transactions from the Tangle. Moreover, this framework supports zero-value transactions that can be attached to the Tangle without requiring validation, thus speeding up information sharing.

However, despite these advantages, DAGs face limitations. For instance, the reliance on a Coordinator in IOTA for transaction confirmation introduces a centralization point, which can compromise the system's trustless nature. Additionally, the complexity of the DAG structure can make it challenging to implement robust consensus mechanisms compared to the more established methods used in blockchain systems. Consequently, while DAGs offer compelling bene-

fits, blockchain technology may still be preferable in scenarios where decentralization, security, and proven consensus protocols are crucial.

2.2.3 DECENTRALIZED IDENTIFIERS

DIDs [30] are a new kind of identifier proposed by the W3 Consortium to enable verifiable, decentralized digital identities. A DID can represent any subject (such as a person, organization, or device) as its controller specifies. Unlike federated identifiers, DIDs are independent of centralized authorities, such as identity providers or certificate authorities, making them well-suited for distributed environments. DIDs are typically stored in distributed ledgers like blockchains and, therefore, are not controlled by a single entity. Technically, DIDs are Uniform Resource Identifiers (URIs) that associate a DID subject to a DID document, facilitating trusted interactions. These URIs comprise three parts: the DID URI scheme identifier, a DID method identifier, and a method-specific identifier. When resolved, the URI links to a specific DID document containing cryptographic materials, verification methods, and services that enable a DID controller to demonstrate control over the DID. They can be linked to various digital credentials, including Verifiable Credentials (VCs) and Verifiable Presentations (VPs) [31], which can be easily shared and verified without relying on a third party. VCs are digital certificates that assert specific attributes about an individual or organization, while VPs provide a way to present these credentials securely, allowing users to disclose information based on the context of their interactions selectively. This decentralized approach empowers users to maintain ownership of their personal information and fosters trust in digital interactions by ensuring that identities can be verified without central intermediaries. Furthermore, DIDs enable interoperability among various platforms and services, fostering a more open and user-centric internet.

3 CHALLENGES IN FEDERATED LEARNING

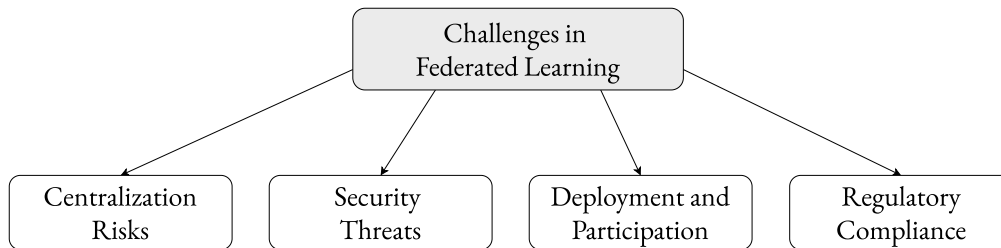


Figure 3.1: Main challenges in FL.

While FL offers significant advantages, it also introduces a range of challenges [6] that must be addressed for successful implementation. Section 2.1.3 already introduced several promising research directions. This chapter focuses on the specific challenges chosen for further exploration in this dissertation, including centralization risks, security threats, deployment and participation issues, regulatory compliance, and the inherent risks associated with the system’s distributed nature. Through a detailed examination of these concerns, the chapter aims to shed light on the current limitations of FL.

3.1 CENTRALIZATION RISKS

Despite FL being a distributed learning paradigm, the aggregation process creates a single point of control. This centralization can lead to a bottleneck, particularly as the number of participating clients increases. The server may struggle to efficiently handle the volume of data being processed, resulting in delays and increased latency during the model training phase [7]. This inefficiency can hinder the system’s scalability and limit its ability to adapt to real-time data changes. Additionally, the central server represents a critical point of failure in the FL architecture. The entire collaborative learning process can be halted if the server experiences technical issues, goes offline, or suffers network disruptions. This lack of resilience can be particularly problematic in applications requiring continuous learning and quick adaptation, as the system’s performance can be severely impacted by even minor disruptions [10]. Moreover, centralized aggregation may lead to bias in

the global model. The server has the discretion to prioritize specific updates over others, which can inadvertently skew the model toward the data characteristics of the preferred clients. This selective aggregation can result in a lack of diversity in the training data, undermining the model's ability to generalize well across different populations and scenarios. Consequently, the model may perform inadequately for underrepresented groups, raising ethical concerns regarding fairness in decision-making processes [8]. Transparency is another challenge with a centralized server, as it can lead to a potential lack of accountability in the aggregation process. Clients may have limited visibility into how their updates are being processed and integrated, which can create trust issues among participants. Suppose clients cannot verify the integrity of the model updates or the fairness of the aggregation process. In that case, they may be reluctant to participate, limiting the overall effectiveness of the FL system and ensuring its credibility [9]. Additionally, a malicious server can conduct eavesdropping on client updates. Even though clients do not share raw data, model updates can still leak information about the underlying data. A malicious server can analyze these updates over time, potentially uncovering sensitive information about clients, particularly in applications involving sensitive data, such as financial or medical records [12]. Collusion with malicious clients presents another severe vulnerability in FL. In this scenario, the server collaborates with compromised clients to manipulate the global model. These malicious clients may send corrupted updates, which the server selectively aggregates. This can introduce biases or backdoors into the global model, all while degrading overall model performance [32].

3.2 SECURITY THREATS

Security is among the most pressing concerns in FL. While the decentralized design of FL improves privacy by keeping data on local devices, it also introduces new and unique security threats [10]. One of the most significant challenges arises from untrusted or compromised clients, which can lead to poisoning and backdoor attacks. One of the most likely attacks in FL is data poisoning [11, 33], where adversaries manipulate their local datasets or models to introduce misleading or harmful information. This creates a high risk of malicious contributions being incorporated into the global model. Poisoning can occur during the training phase, impacting either the local dataset or the local model itself, degrading the global model's accuracy and performance. In FL, model updates are aggregated from many clients, increasing the chances of a poisoning attack from one or more compromised clients. Poisoning attacks in FL can target various parts of the process. These attacks can be broadly classified into the following categories [34]:

- *Data Poisoning*: This type of attack was first introduced by authors in [35], where attackers manipulated the training data to maximize classification errors in Support Vector Ma-

chine (SVM). Since then, several approaches have been proposed to mitigate such attacks across centralized and distributed ML systems. In FL, malicious clients can contribute tampered training samples, resulting in corrupted local model updates that, when aggregated, compromise the global model. A specific form of this attack is data injection, where malicious data is directly introduced into the local model’s training process. This can allow an attacker to influence multiple clients and ultimately control the global model.

- *Model Poisoning*: Unlike data poisoning, which focuses on manipulating training data, model poisoning targets the global model directly. Malicious participants can modify their local model updates before submitting them to the central server for aggregation, which can severely compromise the global model’s accuracy. Model poisoning is particularly effective in large-scale FL systems with many clients, as it becomes harder to detect and mitigate these attacks [32, 36].

Another type of potential attack that can occur is a backdoor attack. Unlike poisoning attacks, this type of attack involves inserting a malicious task into the model while maintaining the accuracy of the primary task. These attacks are difficult to detect because the overall model performance remains unaffected in the short term. Backdoor attacks exploit specific triggers in the data to cause misclassification only when certain conditions are met [11, 37].

Moreover, in FL systems, certain clients may assume a passive role, benefiting from the global model without actively participating in the training process. These *free-riders* may submit dummy updates instead of training their local models on real data [38]. The impact of free-riding attacks is more pronounced in smaller FL settings, where a lack of active client participation can significantly hinder the global model’s training. Although the likelihood of free-riding is relatively low, its impact is moderate due to the potential degradation of model quality.

3.3 DEPLOYMENT AND PARTICIPATION

The deployment of FL faces several significant challenges, particularly scalability and trust. As the number of clients grows, coordinating and aggregating model updates become increasingly complex, leading to potential communication bottlenecks and latency issues [10]. Moreover, creating a trustworthy environment is crucial, as participants often do not trust each other or the central server. Trust is essential in motivating all parties to participate in FL, which in turn facilitates its widespread adoption [39]. In fact, when stakeholders, including clients, service providers, and researchers, have confidence in the integrity and security of the FL process, they are more likely to engage and contribute their data and resources. Traditional FL scheme alone is insuf-

efficient; new features must be introduced to enhance the system's security and trustworthiness [9]. Edge deployment further complicates FL, as real-time data processing across a distributed network requires efficient handling of heterogeneous devices and dynamic environments [13]. Furthermore, most devices at the edge, like in IoT environments, are resource-constrained, which introduces additional burdens in deploying FL. In fact, these devices often have limited computational power, storage, and energy capacity, making it challenging to train complex models or handle large datasets efficiently [5]. Setting up such FL processes in these scenarios and with these requirements can be extremely expensive and time-consuming, especially in some sectors, such as industry or healthcare, where the necessary infrastructure and expertise are often lacking. Additionally, maintaining the system's functionality in dynamic environments where devices may be offline or unable to participate at all times further complicates the coordination of model updates, making it difficult to ensure timely and accurate results [9].

Another critical challenge in FL is ensuring broad participation from diverse clients. Since FL requires computation and communication resources to participate, not all clients may be equally motivated to contribute their local updates [9]. This imbalance can lead to biased models, as a small subset of clients could dominate the training process [40]. Many clients may also seek compensation for their contributions, especially in commercial contexts. For example, a client with a large dataset may not wish to participate in a FL process with clients having smaller datasets. This could be due to concerns about the unequal contribution of data or the potential impact on the overall performance of the FL model. This raises questions about how to fairly incentivize participation without compromising the integrity of the FL process [41]. Additionally, fairness in FL is not only about incentivizing participation but also ensuring that clients with weaker computational capacities or limited data resources are not overlooked in the aggregation process [42].

3.4 REGULATORY COMPLIANCE

With the increasing focus on data privacy and the emergence of stringent regulations such as the GDPR [3] and the CCPA [4], FL must navigate complex regulatory environments. One of the main challenges is addressing the *right to be forgotten*, a fundamental principle encapsulated in many privacy regulations that allow individuals to request the deletion of their data from an organization's systems. Specifically, for example, under the GDPR, individuals have the right to have their personal data erased when it is no longer necessary for the purposes for which it was collected, if they withdraw consent, or if their data has been processed unlawfully. While deleting records from a database may not seem too complicated, it becomes significantly more complex when dealing with ML models. Although FL avoids directly sharing raw data, the global model may still

retain information derived from individual datasets, complicating compliance with these regulations [43]. This retention of information raises ethical and legal questions about how the *right to be forgotten* can be effectively implemented within a distributed framework. For instance, if an individual requests deletion, it becomes challenging to determine whether their contributions to the model can be entirely removed without compromising the integrity of the global model.

Additionally, ML, and for extension FL, is vulnerable to membership inference attacks [44], where an adversary can infer whether an individual's data was used in the training process based on the model's output. These attacks exploit the tendencies of overfitting in models trained on limited datasets and can reveal sensitive information about participants, further complicating compliance with privacy regulations. Addressing these concerns requires the development of robust mechanisms that ensure compliance with regulatory frameworks and ethical management of participant data within the FL paradigm.

Moreover, ensuring compliance with such regulations requires FL systems to be auditable and transparent, particularly in healthcare sectors where patient data is highly sensitive [20]. Accountability frameworks are needed to trace the flow of model updates, verify the legitimacy of participants, and ensure that data contributions comply with legal standards [45].

3.5 SUMMARY OF KEY CONTRIBUTIONS AND PROPOSED SOLUTIONS

To address these challenges, this dissertation has focused on identifying potential solution directions, and this section presents a concise overview of its key contributions. In particular, one major challenge is centralization risks, where the aggregation process can create bottlenecks and bias the global model, undermining participant trust due to limited transparency. This dissertation presents multiple DLT-based architectures to enhance transparency and accountability, addressing centralization risks (see Chapter 4). Security threats, particularly from data poisoning, backdoor attacks by compromised clients, and the risk of free-rider clients that can degrade overall model performance, are also critical concerns. To mitigate these, a robust framework for executing FLaaS is proposed, which enhances security and encourages participation (see Chapter 5). Additionally, this dissertation introduces the concept of FU through a comprehensive literature review and proposes an unlearning algorithm to address regulatory compliance challenges, particularly the *right to be forgotten* and security concerns. This algorithm, in fact, also facilitates the removal of malicious contributions without compromising the integrity of the global model (see Chapter 6). By tackling these challenges, the proposed solutions aim to enhance the effectiveness and adoption of FL in various applications.

4 DLT AS AN ENABLING BACKBONE FOR TRUSTWORTHY FL

This chapter explores how DLT can be a foundational infrastructure to support trustworthy FL. As the landscape of FL evolves, ensuring transparency, security, and trustworthiness becomes essential, especially when working with sensitive or proprietary data distributed across various participants. DLT, with its decentralized and immutable characteristics, offers an effective solution to enhance these aspects [46, 47, 48]. This chapter introduces the proposed architectures and implementations for integrating FL with DLT, beginning with an enterprise-ready blockchain solution, followed by an efficient, scalable architecture based on a DAG-based ledger. Each approach is designed to address specific challenges in FL, such as trust, scalability, and data integrity while ensuring that the system is adaptable to the needs of both large-scale enterprise environments and resource-constrained distributed networks. Extensive experiments and real-world deployments demonstrate how these architectures can be implemented in practical case scenarios.

4.1 MOTIVATION

Chapter 3 already highlighted the key challenges inherent in FL. In the present context, DLT emerges as a promising solution for addressing all the concerns within the domain of FL.

There are several key motivations for integrating DLT with FL:

- *Increased Security and Reliability through Decentralization:* In FL, the aggregation process can create a central point of failure, posing scalability and system resilience risks. Integrating DLT removes the need for a central authority by distributing control across all participants. This decentralized approach ensures that the system remains operational even if individual nodes fail, enhancing the overall robustness of the FL process [49]. Moreover, removing the central entity in FL significantly enhances the security of the entire process. By decentralizing control, the system becomes less vulnerable to attacks, as no single point adversaries can target to disrupt or manipulate the global model. Instead, security is distributed across the network, making any attempt to compromise the system far more challenging. The de-

centralized nature of DLT ensures that tampering with data would require simultaneous manipulation of multiple nodes, a challenging task for attackers [50]. Aggregation, traditionally performed by a central server, can now be decentralized using DLT. Clients can collaborate to aggregate model updates in a distributed manner, ensuring that no single entity controls the process. Alternatively, the aggregation process could be automated through smart contracts, which are self-executing pieces of code on the blockchain. These smart contracts offer a transparent, verifiable, and immutable method for aggregating model updates, ensuring fairness and security while reducing the risk of tampering or bias.

- *Enhanced Process Transparency and Fairness:* Transparency is critical to ensuring trust and fairness in FL. Using DLT, every model update and aggregation is recorded on an immutable ledger, making the entire process auditable and verifiable by all participants. This transparency helps detect and trace tampering or malicious activity, building trust and fostering a more secure collaborative environment [46]. The use of DLT can ensure fairness in the FL process by providing an equal opportunity for all clients' contributions to be acknowledged and validated. Through its decentralized consensus mechanism, DLT prevents bias in the aggregation process, ensuring that all updates, regardless of their source, are fairly integrated into the global model. This reduces the risk of privileging certain clients, promoting equity in the model-building process.
- *Improved Traceability and Accountability:* With DLT, every action within the FL process is logged on an immutable ledger, providing full traceability of updates. This capability allows participants to track model updates back to their origin, facilitating the identification of malicious actors or detecting anomalies in the model training process. This traceability is key to maintaining accountability, security, and integrity within FL.
- *Incentivizing Participation:* DLT can introduce token-based incentives to encourage clients to participate in FL actively. By rewarding participants for submitting updates or penalizing malicious behavior, DLT can promote active and honest participation in the federated learning process [51]. Furthermore, this incentive mechanism can increase the diversity of data contributions, thereby improving the performance and accuracy of models.

In summary, integrating DLT into FL addresses several key challenges, from improving trust and transparency to enhancing security, ensuring system reliability, and preserving fairness.

4.2 RELATED WORK

Many researchers have recently explored the potential of integrating FL with DLTs. This section reviews some notable works that leverage blockchain and DAGs to enable FL at the edge.

4.2.1 BLOCKCHAIN-BASED FL

Blockchain offers a promising solution to address the centralized challenges in FL [52]. Many existing approaches, such as [50], introduce blockchain-based platforms for FL, replacing the centralized server with a P2P network. These peers aggregate partial models through consensus protocols and generate a global model. This decentralized structure ensures that the shared model is unbiased, fostering trustworthiness among participants who may not know each other [53]. For instance, the system architecture proposed in [54] combines a permissioned blockchain with an FL component, storing IoT transactions and enabling P2P data sharing, thereby facilitating data flow auditing. In another study, [55] introduced BlockFL, a blockchain-enhanced FL architecture where participants exchange local model updates and are rewarded for their contributions. This approach decentralizes the system and provides incentives for blockchain network members. Similarly, BD-FL, proposed by Liu et al. [56], integrates blockchain with edge computing to encourage active participation in training, enhancing model accuracy. BD-FL also uses a stable matching algorithm to pair local devices with appropriate edge servers, optimizing resource use and reducing transmission delays. Liu et al. [57] further extended these ideas with FedAC. This framework incorporates a staleness coefficient and uses blockchain for automated aggregation, designed to withstand security threats like poisoning attacks while remaining efficient in edge computing scenarios. Lu et al. [58] focused on combining FL and blockchain for IoT users, facilitating neural network collaboration and training parameter sharing with edge servers. While these studies propose novel algorithms, most experiments were conducted in simulated environments, leaving gaps in practical implementation. Furthermore, evaluation metrics such as accuracy and latency are often used, but they overlook critical factors like energy consumption, a major concern for edge environments with heterogeneous devices.

4.2.2 DAG-BASED FL

Given the constrained capabilities of IoT and edge devices, blockchain-based architectures for FL face limitations. Consequently, new approaches using DAGs have emerged. Schmid et al. [59] were among the first to propose leveraging the Tangle for decentralized learning, where transactions represent the complete set of shared ML model parameters. Consensus is achieved by validating parent transactions through their contributions to the training process, and the model is

published only if it performs better than a reference model. However, the authors do not address the challenge of embedding large ML models within transactions, given the Tangle’s 64 KB size limit. Similarly, Cao et al. [60] introduced DAG-FL. This framework facilitates asynchronous FL using DAGs, along with two algorithms to regulate consensus and schedule ML tasks via smart contracts. While promising, the paper lacks sufficient implementation details regarding the DAG and smart contracts, which are crucial components. Hybrid approaches, like [61], have also been explored, combining the Tangle with a consortium blockchain to enable cooperative FL. In particular, they use the Tangle to secure the sharing of local and global model updates. Lee et al. [62] proposed a hierarchical blockchain system that integrates a public blockchain (e.g., Ethereum) for global aggregation with local shards based on DAGs. Each shard is autonomously established by participants according to their data distribution, and DAGs are employed to compute local models for nodes with low entropy. These models are then aggregated on the public blockchain. However, this approach introduces significant computational overhead, making it unsuitable for constrained environments. While evaluated in terms of accuracy, most of these solutions fail to address other relevant metrics like latency and energy consumption, which are essential for assessing feasibility in edge computing contexts. Moreover, a lack of details on implementing these proposals limits the understanding of how they can be effectively applied in practice.

4.3 FLOWCHAIN: A BLOCKCHAIN-BASED ARCHITECTURE

FlowChain [63] is a distributed, decentralized, secure framework for executing FL tasks. These features are achieved through the exploitation of blockchain technology and decentralized identifiers. Figure 4.1 shows significant differences between a traditional FL framework and FlowChain. The latter exploits blockchain technology to support local models’ storage and aggregation in a fully automated way by relying on smart contract technology. This makes the framework fully decentralized and distributed. Furthermore, as a key feature, each FlowChain client has a DID [30] that certifies its identity. This technology allows to establish who sent the specific local model and to check whether it is allowed to participate in the FL process, thus achieving a priori security.

4.3.1 ARCHITECTURE AND WORKFLOW

The FlowChain architecture consists of two main interacting elements: the blockchain at the framework’s core and the participants in the FL process. FlowChain clients are the elements in charge of performing the logic of the training process. More specifically, clients use their own local data to train a local model. The modalities of this training are based both on local choices, such as the algorithm for calculating gradients and optimization and on architectural choices, such as the

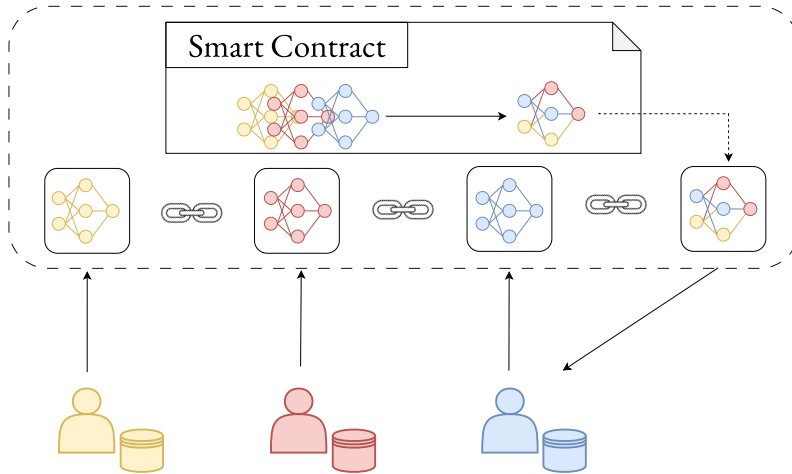


Figure 4.1: FlowChain architecture.

model aggregation algorithm. The latter, in fact, discriminates which data should be published on the blockchain, gradients or simple weights, and the frequency of publication. The clients in the traditional framework and the clients in FlowChain are similar. The main difference lies in how they interact with the entity responsible for calculating the global model. FlowChain achieves this through the smart contract component by executing appropriate transactions on the blockchain. At the core of the architecture, the blockchain replaces all the functionalities of the central entity present in traditional Federated Learning schemes, namely, storing all local models and calculating the global one. In particular, a smart contract is deployed on the blockchain, which exposes the methods invoked to execute the FL process, namely the method to publish local models and a method to read the global model. As a primary role, the FlowChain smart contract validates the local models it receives by checking that the signature on the transaction comes from a known and enabled identity. It then keeps track of these models in its internal state for their subsequent aggregation and the generation of the global model. It is worth noticing that, given the immutability property of the blockchain, local models can be retrieved from the smart contract state and the blockchain by analyzing the sequence of all executed transactions. Another distinctive smart contract functionality is aggregating local models to generate the global model automatically. Toward this goal, the FlowChain smart contract integrates several algorithms, such as FedSGD or FedAVG. An explicit transaction must be performed and submitted to the smart contract to activate local model aggregation. This depends on the fact that the blockchain does not allow the self-execution of smart contracts. In FlowChain, different aggregation models can be envisioned. On the one hand, in a synchronous aggregation context, where all participants are expected to

submit their models, one client could exploit the publication of its last model to perform the aggregation. On the other hand, in an asynchronous context, a specific third entity can be delegated to submit the transaction, enabling the smart contract to perform the aggregation. The other important feature that FlowChain exhibits is the support of identity management via DID standard. Each device participating in the FL process must have an associated DID published on the blockchain so that an a priori secure execution environment can be created. In fact, by certifying the identities of the devices, it is possible to control who can and cannot participate in the training process and to track who has published a particular local model. Apart from access control, DIDs can also be used to keep track of information about devices involved in industrial processes, such as the status of the device, its location, and other data that may be useful. In addition, it is also possible to assign a DID to the personnel working on these devices to record any interaction with them, such as installation or maintenance operations. To be clearer and more precise, the main workflow among the FlowChain architecture components is summarized below. To simplify the explanation, FedAVG is used as the aggregation algorithm, and the synchronous aggregation context is considered, which means all models from the participants are waited before aggregating them. Furthermore, all participants possess certified identities and are, therefore, authorized to perform operations on the blockchain or at least invoke the smart contract's methods.

1. The first step is for all clients to agree on the architecture of the neural model to be used. In other words, each participant must have the same starting configuration (weights and biases) of the model used to perform the training process. This can be achieved either by configuring the model statically, namely setting up all clients with the same hard-coded model, or dynamically, where one client publishes the initial configuration of its model, and all the others adopt it.
2. Once all participants have the model, they can start the training phase using their local data. The training of a neural network is based on repeating the update of the network weights several times, according to the gradient of the error function. Traditionally, training ends when a particular model accuracy threshold is reached. In our case, the number of updates depends on the configuration parameters of the entire system, namely how often we want to send the updated local model. Once the partial training of the model is complete, a transaction is performed on the blockchain, using the proper method of the deployed smart contract, to publish the model's weights.
3. Once all participants have published their local models, namely the networks' weights, the smart contract can perform local model aggregation. In this case, it is simply averaging all the weights. Once this is done, the smart contract updates the state of the blockchain, pub-

lishing the global model. This is done by saving the newly generated global model in its internal state. Different global model notification schemas can be realized on FlowChain, following a push or a pull approach. In a push approach, it is possible to notify the participants of the aggregation, possibly even sending them the updated global model. In a pull approach, participants are responsible for retrieving the new updated global model through the specific smart contract method.

4. Once participants receive the updated global template, the training process can continue. In short, the previous points in the list are repeated. Depending on the needs, the process may or may not end. For example, it may stop when the accuracy of the global model is satisfactory. Conversely, if, for example, the training data changes over time, it is necessary to continue updating the global model to keep track of these changes.

4.3.2 IMPLEMENTATION

BLOCKCHAIN

To implement the FlowChain blockchain component, we opted for Hyperledger Fabric [64], a framework part of the large family of Hyperledger projects supported by the Linux Foundation. It is a decentralized operating system for permissioned blockchains that can execute dApps written in general-purpose programming languages. The Fabric's modular architecture and versatile design fit into various industrial distributed ledger use cases. Fabric-based blockchain applications are enterprise-grade and offer high security, scalability, confidentiality, and performance. More specifically, recent work [65] has shown that in an industrial environment, Fabric can support about 200 transactions per second, with an average latency of about 0.16 seconds and up to 100,000 participants. These values represent a balanced approach between the benefits of blockchain and the potential overheads of its implementation. We deployed a Fabric network as a first step in building our architecture. A Fabric network consists of nodes that can be clients that propose transactions to execute and broadcast them for ordering, peers that maintain the ledger, and the state of the latter, ordering service nodes that establish the order of all the transactions. Moreover, each node must have an identity given by a membership service provider. Indeed, Hyperledger Fabric uses a Public Key Infrastructure (PKI) to verify the actions of all network participants. Every node, network administrator, and user submitting transactions must have a public certificate and private key to verify their identity. These identities need to have a valid root of trust, for example, a Certificate Authority (CA), establishing that the certificates were issued by an organization member of the network. In our implementation we used the test network that the Fabric developers made available. This network includes two peer nodes, two CAs (one per peer), and

an ordering node. After configuring and launching the Fabric network, we built and deployed the smart contract to run the FL process. Hyperledger Fabric allows various programming languages; we adopted the Node.js SDK to implement the smart contract and the methods for publishing a local model and reading the global model. We implemented FedAVG as the adopted aggregation algorithm. This implies that the local models are simple weights and biases of the neural networks. Furthermore, we have implemented synchronous aggregation, allowing us to automatically generate the global model following the publication of the last local model. Hyperledger Fabric also enables the implementation of a push notification approach through event management. Generating a new global model causes an event to be emitted to the registered clients. This event also contains the new model configuration. So, having N participants for each training cycle exactly means N transactions are executed on the blockchain, one for each local model. The N -th also includes the generation of the new global model. As stated in [65], Fabric can handle a very large N without performance or scalability issues. As mentioned above, Hyperledger Fabric is a permissioned blockchain that manages identities via PKI. Unfortunately, to date, it does not support the DID standard. However, this is not a problem since, being permissioned, anyone performing a transaction must have a known identity. In this way, control over who can execute smart contract methods can be performed directly via the verification of the signature on the transaction. If using Hyperledger Fabric, the use of DIDs seem useless. However, DIDs can also be used to keep track of all devices in the environment, together with relevant information about them. For this reason, we decided to implement support for DIDs in Hyperledger Fabric anyway, pointing out that this technology is not used for access control in the specific case of the Hyperledger Fabric-based implementation. To achieve this support, we relied on a project called TrustID [66], incubated in Hyperledger Labs, the development space for new Hyperledger projects. Using a smart contract, TrustID implements the management of the entire lifecycle of a DID, from its creation to its update and revocation. Moreover, it allows to assign a DID also to smart contracts already deployed on the blockchain so that they can be seamlessly discovered and accessed. Doing so makes it possible to use the TrustID smart contract to execute other smart contracts by exploiting the support of DIDs as identity checks.

OPTIMIZATION TECHNIQUES

One of the most significant limitations that hinder the use of blockchain for ML is the size of transactions. Indeed, although Fabric is one of the few blockchains that allows large transaction sizes (around 100MB), ML models can easily go beyond this threshold. For example, some models available on Keras [67] reach up to 500MB in size. To address such concerns, we adopted the following model size optimization techniques:

- **Model Compression:** this technique can be used with any type of data. It consists of compressing the global and local models before saving them on the blockchain. We used *pako* [68], a porting of *zlib* for NodeJS. *zlib* is a free, open-source software library for lossless data compression and decompression. It is based on the DEFLATE algorithm [69], which uses a combination of the LZ77 lossless data compression algorithm and the Huffman coding;
- **Model Quantization:** this method is specific to neural network weights. Although compression is generally useful in decreasing data size, it turns out to be underperforming in the case of ML models. The *oat* weights of a neural network are usually unsuitable for such compression due to the noise-like variation in the parameter values, which contains few repeating patterns [70]. For this reason, we decided to quantize the neural network weights after performing local training. Quantization refers to mapping the 32-bit float values of the original network weights to a more compact representation, such as 8-bit integers. In our case, we mapped the weights ($w \in [a; b]$) to 8-bit signed integer values ($w_q \in [q; -q]$), with $q = 128$ and $q = 127$. The quantization process is defined for each weight as follows :

$$w_q = \text{round}\left(\frac{1}{S}w + Z\right) \quad (4.1)$$

and the de-quantization process is defined as:

$$w = S(w_q - Z) \quad (4.2)$$

where S is the scale and Z is the zero point [71]. The scale is an arbitrary positive real number, while the zero point is the quantized value corresponding to the value 0. It is possible to derive S and Z as follows:

$$\begin{aligned} \exists \\ < S = \frac{1}{q - a} \\ \vdots Z = \text{round}\left(\frac{a - q}{q - a}\right) \end{aligned} \quad (4.3)$$

Since some neural networks have weights that differ by orders of magnitude between layers, S and Z are calculated by considering one layer at a time. This approach improves the accuracy of the quantization process and makes it less subject to variance in values.

CLIENTS

For the devices participating in the training process, we decided to use a new framework for FL called Flower [72]. Flower provides FL infrastructure to ensure low engineering effort, enabling the programmer to concentrate on his own ML use case. It is compatible with many existing and future ML frameworks, such as TensorFlow and PyTorch. Moreover, the use of Flower allows

FlowChain to be ready to deploy in any real-world context since Flower has been built to meet various real-world configurations with many clients and to be interoperable with different operating systems and hardware platforms to work well in heterogeneous environments. In addition, Flower includes means to simulate real-world system conditions, such as limited computational resources that are common for typical FL workloads. However, the integration between Flower and Hyperledger Fabric was not straightforward. In fact, Flower is implemented, as is the norm in the ML field, using Python, and, unfortunately, Hyperledger Fabric does not yet have an official SDK for Python. In addition, the SDK for interacting with the TrustID smart contract is implemented in Node.js. For these reasons, we decided to implement the clients as consisting of two communicating parts: one in Python to perform the ML model manipulation procedures and one in Node.js for communication with the blockchain. These two parts are integrated via gRPC, a modern open-source high-performance Remote Procedure Call (RPC) framework. By doing this, we remain consistent with the Flower framework, the TrustID project, and the official Hyperledger Fabric SDKs.

4.3.3 EXPERIMENTAL EVALUATION

Despite the full-blown benefits of using blockchain, its widespread adoption is often inhibited by energy consumption, which is one of the significant concerns also due to the current discussions on climate change and sustainability [27]. In blockchain environments, determining the amount of energy consumption is a hard task that depends on several factors, such as the number of participants and the consensus mechanism adopted. For example, PoW consensus requires a large extent of electrical energy, as the entities involved in the validation process (i.e., miners) demand a huge amount of computational resources to validate blocks [73]. Miners compete to confirm a block, and who wins the race advances the blockchain status. All the other block candidates are discarded, resulting in a massive waste of electricity put into their calculation. The difficulty in accurately estimating the general energy consumption of blockchain technology is further confirmed by the limited results available in the literature [74]. Some studies [75, 76] have provided general estimates for the lower and upper bounds of the energy consumed, but a more comprehensive analysis is needed. Energy is one of the major concerns also for FL environments [77] because participants may have heterogeneous capabilities. In FL, clients do not offload heavy computations to cloud-based resources. Thus, their computing resources are directly involved in the training phase. For this reason, Multi-access Edge Computing servers are often leveraged to allow users to offload a portion of their dataset for model training [78]. Despite the increasing interest in integrating blockchain and FL while keeping energy consumption under control, en-

ergy and power analysis are often neglected and hardly ever presented in novel proposals. In the literature, no works estimate blockchain's energy and power consumption for FL purposes.

To fill this gap, this section aims to estimate the power consumption of FlowChain. Our evaluation mainly focuses on the power consumption of the blockchain for those operations needed to enable FL (e.g., aggregating local models). Experimental results demonstrate that the overall power consumption heavily depends on the considered ML model.

POWER CONSUMPTION MODEL

To effectively enable the use of blockchain in FL environments, the power consumption of a blockchain node while performing the operations connected to an FL process needs an adequate investigation. It is clear that all the parties involved in FL consume energy, and saving energy is fundamental to reducing costs and avoiding environmental damages associated with carbon emission [79].

We model the power consumption of the overall FL process P_f as a function of the different modules involved in the training phase. Consider the power consumption of an FL training during the i -th round. The client that sends local models consumes P_c^i . The power consumption of a blockchain node that collects local model P_b^i is a combination of power needed to run the node P_n^i and that P_s^i to execute the smart contract to aggregate local models and generate the $i + 1$ -th global model. We denote with C the number of FL clients and with B the number of blockchain nodes. Therefore, the power consumption of the i -th FL round can be defined as follows:

$$P_f^i = C P_c^i + B P_b^i \quad (4.4)$$

where P_b^i is defined as:

$$P_b^i = P_n^i + P_s^i \quad (4.5)$$

Therefore, given N rounds of FL, the overall power consumption of an FL process is given by:

$$P_f = \sum_{i=1}^N P_f^i = C \sum_{i=1}^N P_c^i + B \sum_{i=1}^N P_b^i \quad (4.6)$$

The formulas above show that the total power consumption of an FL training depends on C and B . However, determining the optimal number of FL clients and blockchain nodes that constitute the network goes beyond the scope of this work.

EXPERIMENTS

This section evaluates the feasibility of employing blockchain in an FL environment. We deploy the blockchain component of FlowChain on 2 nodes, each equipped with an Intel(R) Core(TM) i5-3470 CPU running at 3.20GHz and 12 GB of RAM.

The main focus of our analysis is to evaluate the power consumption of the blockchain while performing FL operations. Therefore, we did not consider the amount of power clients consume while training local models. We also measure the consumption of the connector that allows Flower Clients to interact with the blockchain. To do this, we use the well-known PowerTop [80] tool available for Linux. Concerning the blockchain, since in FlowChain each component is run inside a Docker container, we obtain power metrics through docker-activity [81], a tool developed to monitor the statistics of containers and their power consumption. Additionally, since resource consumption directly affects feasibility, we also measure memory and CPU usage to gain deeper insights into the system's efficiency and evaluate its practicality for real-world deployment.

To provide a comprehensive estimation of power and resource consumption, we conducted two experiments varying the complexity of the employed ML models and datasets. To obtain a more accurate estimate, each experiment was repeated 10 times, and the results were aggregated. For the first case, we used the Fashion-MNIST [82] dataset, a collection of Zalando item images consisting of a training set of 60.000 examples and a test set of 10.000 examples. Each example is a 28x28 grayscale image associated with a label of 10 classes. The used neural network has only three layers: a Flatten input layer, a Dense layer consisting of 128 neurons, and an output layer consisting of 10 neurons, one per class. In total, the network has 101.770 parameters.

Then, we increased the complexity for the second experiment by employing CIFAR-10 [83] and the neural network MobileNetV2 proposed in [84]. CIFAR-10 is a well-known dataset of 60.000 32x32 color images in 10 classes, with 6.000 images per class. There are 50.000 training images and 10.000 test images. MobileNetV2 has 2.270.794 parameters, about 20 times more than the network used for the first experiment. In both cases, the neural network is trained locally for 5 epochs, 10 FL rounds are performed, and datasets are fairly split among all the FL clients.

RESULTS

Figures 4.2 and 4.3 give an idea of the overall power consumption of FL processes, while Table 4.1 reports the memory and CPU usage. In particular, for each experiment, we plot the results obtained during one of the 10 executions. The graphs sharply highlight the difference in terms of seconds needed to perform the FL processes. Indeed, the simpler case was about 850 seconds less than the experiments with a more complex setting. In the second experiment, the blockchain's

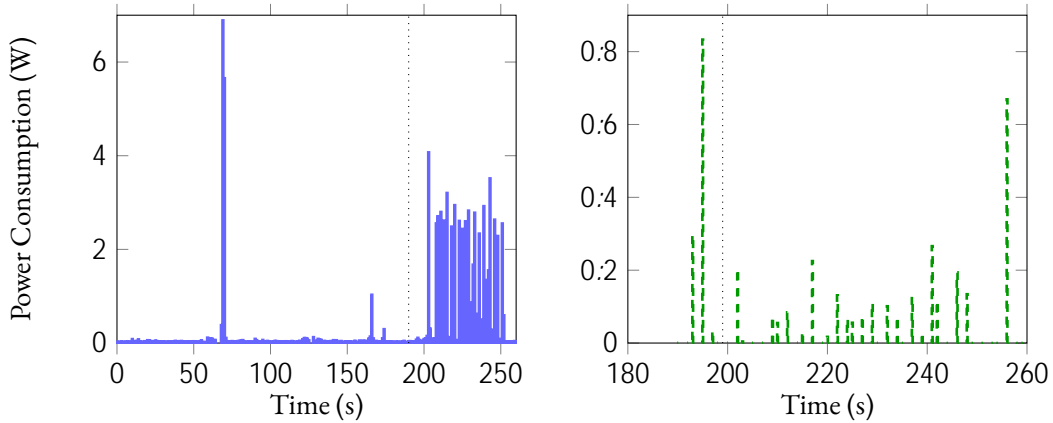


Figure 4.2: Power consumption of a blockchain node while performing FL operations (blue) and client connector while interacting with the blockchain (green) for the first experiment.

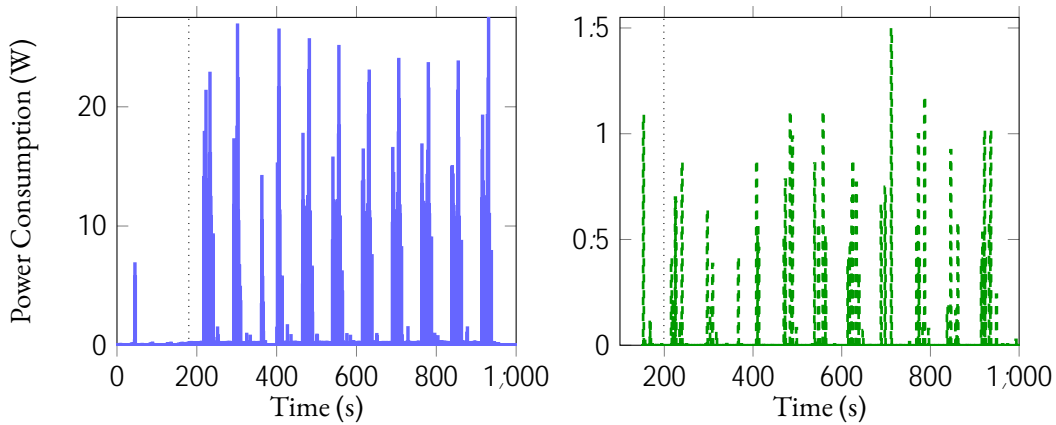


Figure 4.3: Power consumption of a blockchain node while performing FL operations (blue) and client connector while interacting with the blockchain (green) for the second experiment.

power consumption is about an order of magnitude higher than in the first. In contrast, Figures 4.4 and 4.5 report the aggregate results of the 10 executions for both experiments, calculating the maximum and mean value of power consumption in different time slots. Table 4.1 proves that, in both experiments, the blockchain node has a greater memory and CPU consumption than the client connector. Thus, it deems more powerful resources to be executed. In the following, we analyze the power consumption of the blockchain node and the client while performing FL-related operations.

Blockchain. Blockchain power consumption can be divided into two main phases: the instantiation of the smart contract and the execution of the transactions to get local models and combine them. In the blockchain part of Figures 4.2 and 4.3, the vertical dotted line divides the instan-

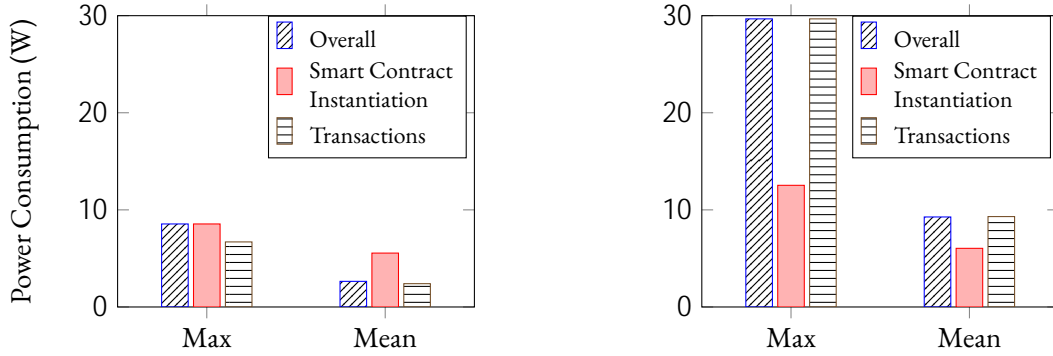


Figure 4.4: Power consumption statistics of a blockchain node, respectively, for the first and second experiments.

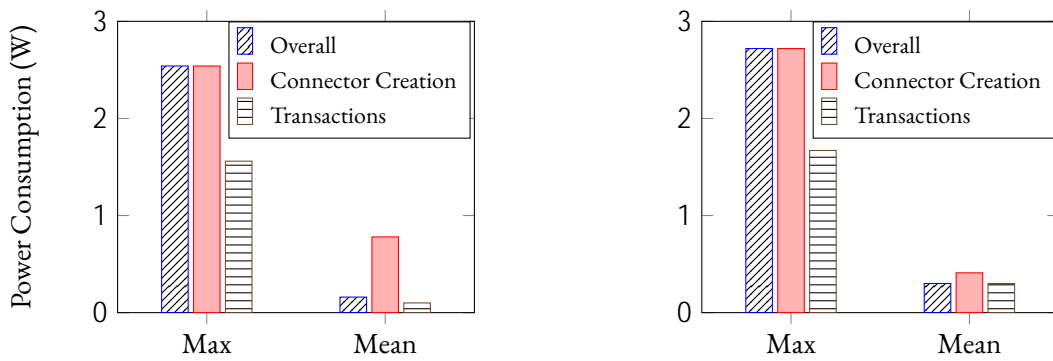


Figure 4.5: Power consumption statistics of a client connector, respectively, for the first and second experiments.

Table 4.1: Memory (MB) and CPU (%) usage.

Experiment	Component	Memory (MB)		CPU (%)	
		Mean	Max	Mean	Max
#1	Blockchain Node	161	315	1.53	32.09
	Client Connector	106	124	1.57	9.77
#2	Blockchain Node	1357	2688	4.16	49.78
	Client Connector	329	385	4.17	18.81

tiation of the smart contract and the execution of the FL process. Concerning the power consumption during the instantiation, there are no remarkable differences since instantiating a smart contract does not depend on the complexity of the ML model. The second peak, which can be seen in Figure 4.2, represents the automatic invocation of the smart contract initialization function. This is clearly shown in Figure 4.4, where the mean power consumption during the smart

contract instantiation phase is almost equal between the two experiments. The FL process involves three main transactions: publishing one model, publishing a second model with subsequent aggregation to create the global model, and reading the latter. Figures 4.2 and 4.3 highlight that, in both experiments, the aggregation is more power-intensive. Furthermore, there is a sharp difference in power consumption between the two experiments: the second one is deemed 5 times the consumption of the former. Figure 4.4 shows where the peak of maximum power consumption occurs throughout the execution of the experiments. In the first one, it is interesting to note that the consumption is higher during the smart contract instantiation than during the transaction execution. However, consumption is significantly higher in the second experiment during transaction execution. Figure 4.4 shows how a more complex ML model markedly affects power consumption during the FL process.

Client. On the client side, also the execution can be divided into two phases: the creation of the connector and the sending of the various transactions to the blockchain. In the client connector part of Figures 4.2 and 4.3, the vertical dotted line represents the division between the creation of the connector and the execution of the FL process. By observing the figures, we can state that the creation of the connector, barring small variations, has comparable power consumption between the first and second experiments. In both cases, as Figure 4.5 shows, the peak power consumption occurs precisely during the creation of the connector. The consumption during sending and receiving of the various FL models is markedly different between the first and second experiments. As Figure 4.5 shows, the mean power consumption during the transaction phase increases in the second experiment by having to send a much larger amount of data. As with the first experiment, Figure 4.5 shows how a more complex ML model noticeably affects power consumption during the FL process. However, since the overall power consumption is lower than 3W, it can be deployed in real-world scenarios. For example, a client connector can be deployed on a Jatson Nano [85] since it provides satisfying compute performance with 5-10W of power consumption.

4.3.4 CONCLUSION

This work introduces FlowChain, a distributed, secure framework for executing FL tasks within the Industrial Internet of Things (IIoT). By leveraging blockchain, FlowChain addresses the limitations of traditional frameworks where centralization can create single points of failure and bottlenecks. FlowChain utilizes blockchain for storage, as seen in other solutions, and for the automated aggregation of local models through smart contract technology. With DIDs, FlowChain creates a secure, trustworthy environment while integrating the Flower framework enhances it to an enterprise-grade level. We validated the framework's feasibility through an initial implementation, demonstrating complete functionality. Our experiments highlighted that power consump-

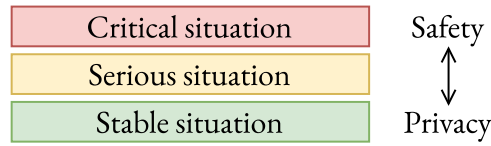


Figure 4.6: Considered risk levels.

tion is closely tied to the complexity of the ML model in use. Based on these results, we can assert that FlowChain is prepared for real-world deployment.

4.4 FRAMH: A FLOWCHAIN DEPLOYMENT FOR RISK-BASED AUTHORIZATION IN HEALTHCARE

To demonstrate the benefits of FlowChain, the framework is applied in a healthcare setting where data privacy and security are critical. Healthcare environments typically involve sensitive patient data distributed across multiple institutions, each tasked with adhering to stringent privacy regulations. This section introduces the Federated Learning Risk-Based Authorization Middleware for Healthcare (FRAMH) [86], which enables users to access patient data and medical equipment based on contextual information, such as the health status of the patient. It leverages FL to train a ML model that assesses the level of risk associated with a patient’s current condition. Local models are securely stored on the blockchain, ensuring integrity and trustworthiness. To our knowledge, this is the first approach to apply FL for risk inference related to patient health status and, more broadly, to estimate health risks in risk-based access control models for healthcare scenarios [87]. We implement a prototype of the middleware and, through extensive experimental testing, demonstrate the effectiveness of FL in assessing patient health when individual medical institutions have limited datasets.

Risk Classification. FRAMH classifies the severity of a patient’s health status using the same risk categories proposed in [88], where a patient’s condition is categorized as either *critical*, *serious*, or *stable*. While adopting a more granular risk classification is possible, this would introduce additional complexity in managing policies, particularly in handling overlapping policies across different risk levels. Furthermore, we have not identified practical scenarios that would necessitate a finer-grained risk classification. It is important to note that FRAMH is flexible and can support different risk classifications with minimal adjustments. Figure 4.6 illustrates the adopted risk levels and their relationship to safety and privacy. As the risk increases, safety precedes privacy, whereas, in stable situations, privacy is prioritized.

- *Critical*: patient life is significantly in danger. In this case, privacy becomes secondary and requests to access data or medical equipment can be granted. Medical personnel must obtain information as soon as possible to do their best and save patient life.
- *Serious*: patient conditions are considered urgent. In such a situation, doctors can access patient data from other departments. We grant access to riskier data to expedite patient treatments.
- *Stable*: users can only access authorized data. There is no reason to grant access to additional information since the patient is not facing a life-threatening situation.

The proposed risk classification is adopted to label the outputs predicted by our ML model. Further details about this phase are discussed in Section 4.4.2.

4.4.1 ARCHITECTURE

FRAMH architecture consists of three layers: *Authorization*, *Patient*, and *Learning*, which collaborate to manage effective risk-based access control. The Authorization Layer provides the components to verify access requests and enforce access control decisions. The Patient Layer offers services for the collection of patient context data. Finally, based on the FlowChain framework, the Learning Layer supports the prediction of the level of risk associated with the current patient condition. Figure 4.7 shows the FRAMH architecture.

AUTHORIZATION LAYER

The Authorization Layer consists of various components from the XACML architecture [89], as depicted in Figure 4.7.

Policy Enforcement Point (PEP). The PEP is responsible for managing all incoming access requests. It extracts relevant information from the requests and constructs a query that can be processed by the PDP. If the request lacks necessary information, the PEP gathers additional data from external sources to complete the query before forwarding it for evaluation.

Policy Decision Point (PDP). The PDP evaluates access requests and decides whether to grant or deny access. It relies on predefined policies and data to make these decisions. The PDP must have access to all the necessary information and rules to execute this function effectively.

Policy Information Point (PIP). The PIP is deployed on each node and provides additional data the PDP requires for access control decisions. Specifically, it supplies patient information based on the current health status predicted by the global model.

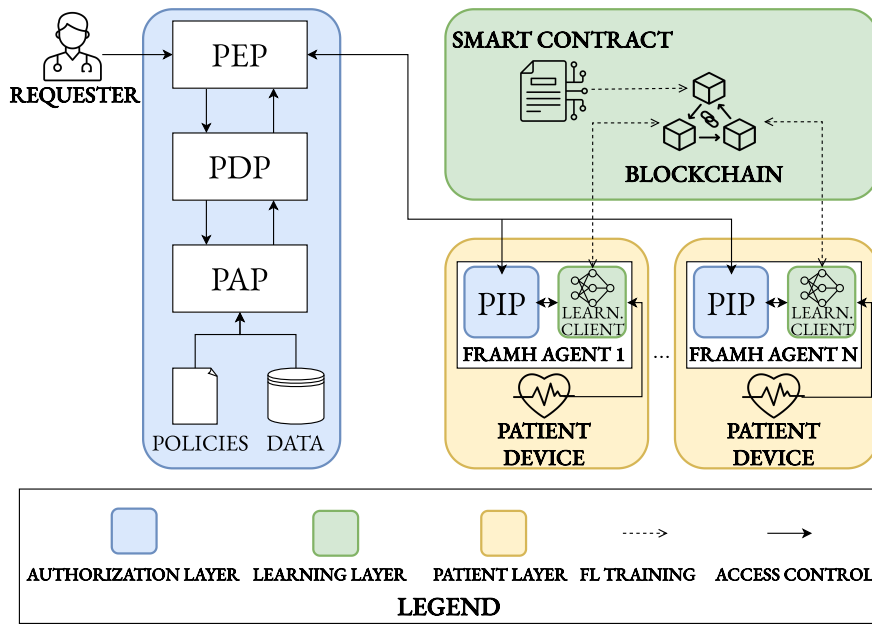


Figure 4.7: The FRAMH architecture with the Authorization, Patient, and Learning layers highlighted with different colors.

Policy Administration Point (PAP). The PAP is the service that enables administrators to manage policies and upload the data needed during policy evaluations. It ensures that the system’s rules and information are updated.

FRAMH Agent. Each Patient Device has a FRAMH Agent associated with it. FRAMH Agent is a local component deployed within smart homes, hospitals, and clinics that comprises all services for calculating the risk level of the patient’s health status and for providing patient data along with the current health status to other FRAMH architecture modules. The FRAMH Agent also comprises the Learning Client module, presented below. Every FRAMH Agent receives vital physiological data from the Patient Device. Such information is periodically provided as an input to the global model embedded in the Learning Client, which outputs the patient’s health status. The FRAMH Agent grants access to these data if the access request satisfies the corresponding access control policy. When deemed by the PDP, it sends context information about the patient needed to make access control decisions.

LEARNING LAYER

Let us examine the key components involved in the learning process. These are the same components as those in FlowChain presented in Section 4.3, which we revisit below for clarity.

Blockchain. We use a blockchain because it guarantees transparency, fairness, and impartiality even among untrusted parties that collaborate with the FL process (e.g., [47]). In addition, by relying on smart contract technology, we eliminate the need for a centralized entity typically responsible for aggregating all local models in the traditional FL approach. The adoption of smart contracts allows clients to publish local models on the blockchain. It is the smart contract that aggregates them in an automated way. The blockchain also stores all the versions of the global model. One global model version can be retrieved by looking at the transaction history. One advantage is, for example, that if performance drops during the training phase due to overfitting, the best version of the model can be restored. Thanks to blockchain features of immutability and non-repudiation, all changes to the partial and global models can be easily tracked. Furthermore, when keys used to sign transactions are linked to physical entities, it is possible to provide accountability during the FL process. These properties are relevant in healthcare contexts, where an error in the model generation may have legal consequences. Using a blockchain also prevents a malicious user from generating the global model without being detected. An attacker may, for instance, provide a local model with some backdoor to corrupt the performance of the global model on specific sub-tasks [90] (e.g., classifying critical patient conditions as stable). Although the legal nature of blockchain is still debatable, some countries have begun to admit data saved on blockchain as a piece of legal evidence. For example, on June 27, 2018, ruling on *Hangzhou Huatai Yimei Culture Media Co., Ltd. ("Huatai") v. Shenzhen Daotong Technology Development Co., Ltd. ("Daotong")*[91], the Hangzhou Internet Court was the first to accept electronic data stored on a blockchain as legal evidence in a process.

Learning Client. Learning Clients (also called clients) are the entities involved in the training process. They use vital physiological data from patients to train an ML model locally. The training takes advantage of both local decisions (optimization algorithm) and global decisions (algorithm for aggregating the various local models). The latter discriminates which data should be published on the blockchain as well as the frequency of publication. Once the local training has ended, clients rely on smart contract support to publish their results. The blockchain server exploits information the clients provide to update the global model, which is then sent back to clients and used as a starting point for the next round of local training. Once the training has ended, Learning Clients are provisioned with a global model that enables them to infer the health status of patients.

4.4.2 EXPERIMENTAL EVALUATION

We evaluate the feasibility of applying the proposed FL to estimate the level of risk corresponding to the current patient conditions. We first describe the employed dataset. Then, we introduce the adopted ML model. Finally, we discuss experimental results and draw some considerations.

DATASET

We used the public dataset from the PhysioNet/Computing in Cardiology Challenge 2012 *Predicting Mortality of ICU Patients* [92]. This dataset includes the medical records of 12,000 Intensive Care Unit (ICU) patients who survived or died. All patients were adults and were hospitalized for several reasons. For each patient, data were collected for 48 hours after their admission to the ICU. Observations covered 42 variables, including information from laboratory tests or non-invasive examinations. However, not all variables were collected every hour of hospitalization. Before performing the training, we pre-processed the dataset. First, we followed the data cleaning and feature extraction process described in [93]. Then, we checked each patient's measurements to correct any errors using domain knowledge. Physiologically implausible values were replaced with valid measures or *NaN*. Finally, we transformed the time series of the individual variables into scalar features. We extracted the following features for each temporal variable: minimum, maximum, median, first, and last values. Furthermore, to simulate a scenario where patients are monitored through Internet of Medical Things (IoMT) devices, we used features that can be monitored through non or minimally invasive readings. Non-invasive means no need to cut the skin or enter any body space to measure a vital parameter. For this reason, we only take into account the following features: age (Age), glucose in blood (Glucose), heart rate (HR), non-invasive diastolic arterial blood pressure (NIDiasABP), non-invasive mean arterial blood pressure (NIMAP), non-invasive systolic arterial blood pressure (NISysABP), respiration rate (RespRate), O_2 saturation in hemoglobin (SaO₂), and temperature (Temp). Each feature X was normalized so that each value x was mapped to the range $[0; 1]$. Not all measurements were available for all the patients, producing missing data in the dataset. For this reason, we first eliminated those patients with at least two time variables that had never been collected. In such a way, the dataset was reduced from 12,000 to nearly 8,000 patients. In addition, we replaced the *NaN* values with the median for each feature. As a remarkable issue, we note that the dataset is highly unbalanced since it contains many more survived than dead patients. After various transformations, the ratio between deceased and survivors is approximately 1:6. Many techniques handle unbalanced data sets (e.g., [94]). Since we decided only to use real data and avoid synthetic, we performed a random under-sampling of the dominant class to obtain a balanced dataset. This step further reduced the size of the dataset to 2254 patients.

MODEL

As our ML model, we use the Multilayer Perceptron (MLP), a fully connected class of feedforward neural networks. The choice of hyper-parameters, such as the number of layers, neurons, and

initial learning rate, is crucial to obtaining a performing model. There is no unique strategy for discovering the optimal configuration. However, some recommendations can be followed [95]. In this work, we adopt a search technique based on a manual trial and error of different hyperparameter configurations. The best configuration comprises four dense layers, with 256, 512, 128, and 1 neuron, respectively. The first three layers use Rectified Linear Unit (ReLU) as the activation function [96], while the last layer (the output layer) adopts the Sigmoid to obtain a value between 0 and 1, representing the probability of the patient death. In addition, layers using the ReLU function are initialized using the He Normal initializer, while the last layer is initialized using the Glorot Normal initializer. All biases are initialized to zero. As the optimization algorithm, we employed Adam, a stochastic gradient descent method based on adaptive estimation of the first-order and second-order moments [97], with a learning rate of 0.01 and a decay of 0.005. As demonstrated in [98], decreasing the learning rate during training contributes to achieving better performance. Finally, since we aim to address a binary classification problem, we used binary cross-entropy as the loss function. The output value o of the model is mapped into the risk levels shown in Figure 4.6 as follows:

$$\begin{aligned} & \infty \\ & \rightsquigarrow o \in [0; 0.33[\quad \textit{Stable} \\ & \rightsquigarrow o \in [0.33; 0.66[\quad \textit{Serious} \\ & \rightsquigarrow o \in [0.66; 1] \quad \textit{Critical} \end{aligned} \tag{4.7}$$

As anticipated in Section 4.4.1, vital physiological data collected through IoMT devices are provided to the global model that outputs one of such risk levels that is then used by the PDP to make access control decisions.

EXPERIMENTS

We applied our model in centralized and federated configurations to show the effectiveness of FL in healthcare contexts. The dataset was split into 90% training data and 10% testing data. We first defined a performance benchmark by performing centralized training using the entire training set. In the centralized approach, a single client, which could be a hospital, performs the training process for 300 epochs. However, since the hospital may have a smaller training set, we ran a second experiment using 75% of the training data. To evaluate the federated approach, we considered three clients that can represent three hospitals of the same region that join the federated learning process. Each hospital uses 33% of the training data, with no overlap, to train a local model. On each client, the model is trained for 3 epochs before being sent for aggregation, while the global training involves 100 rounds of communication. Thus, we are in the same condition as the centralized approach since each model will be trained for 300 epochs.

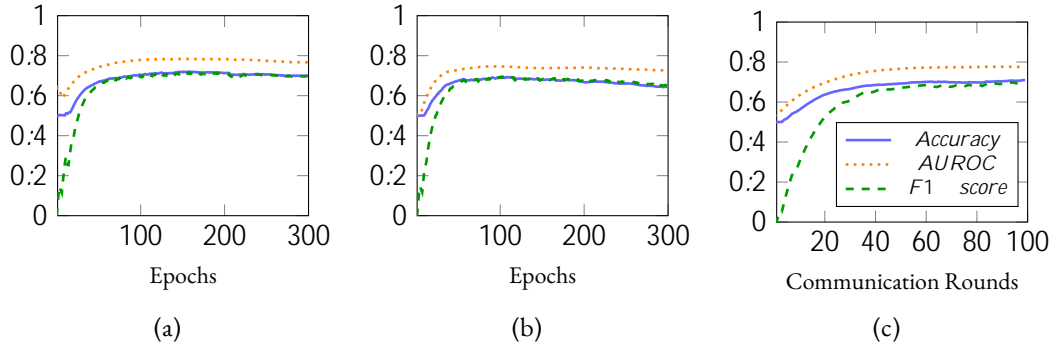


Figure 4.8: Performance comparison of the centralized approach with 100% of the training set (a), centralized approach with 75% of the training set (b), and federated approach with 3 clients each using 33% of the training set (c).

RESULTS

In Figure 4.8, we report the main results of the experiments. The three graphs show that the model obtained through FL achieves comparable performance with the centralized model trained with the entire training set. Interestingly, the centralized model trained with 75% of the training set performs worse than the federated one. Fewer data will result in a worse ML model, and a single clinic cannot have data comparable to a federation of medical institutions collaborating to achieve a common goal. This realistic observation justifies our choice to adopt FL to infer the health status of patients in healthcare scenarios where sharing sensitive data is one of the major concerns. Based on the collected results, we can state that medical institutions with a small dataset can benefit from joining the FL process to infer the risk related to the current patient condition. Although each client trains a model using a restricted dataset, this knowledge is shared through the aggregated model with other peers. Thanks to the proposed process, medical institutions can achieve comparable results to a centralized ML approach as they employ the whole amount of data produced by all involved organizations.

Furthermore, in Table 4.2, we compare the obtained results with those of the State-of-the-Art (SOTA) research proposals that use the MIMIC-III dataset or derivatives and only consider vital signs. Our comparisons are based on three metrics widely adopted in literature: Accuracy, Area Under the Receiver Operating Characteristics (AUROC), and F1-score. AUROC is a performance measurement for classification problems at various threshold settings. It tells how much the model is capable of distinguishing between classes. Figure 4.9 reports the ROC of our experiments. The graphs highlight that the AUROC of the centralized approach using 100% of the training set and that of the federated one are very similar. This may be because the task is relatively simple, allowing the federated approach to achieve performance close to the centralized

Table 4.2: Comparison of ML approaches that use the MIMIC-III Dataset.

Project	Dataset	Model	Performance
Centralized 100%	Subset of MIMIC-III	MLP	Accuracy = 0.70 AUROC = 0.76 F1-score = 0.70
Centralized 75%	Subset of MIMIC-III	MLP	Accuracy = 0.64 AUROC = 0.72 F1-score = 0.65
Federated	Subset of MIMIC-III	MLP	Accuracy = 0.70 AUROC = 0.77 F1-score = 0.70
Baker et al. [99]	Complete MIMIC-III	CNN-BiLSTM	Accuracy = 0.76 AUROC = 0.85
Sadeghi et al. [100]	Complete MIMIC-III	Decision Tree	AUROC = 0.93 F1-score = 0.91
Brand et al. [101]	Complete MIMIC-III	CNN	AUROC = 0.87 F1-score = 0.77

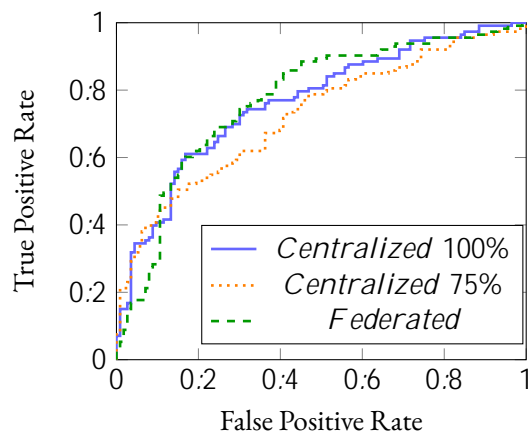


Figure 4.9: ROC comparisons of the centralized approach with 100% of the training set, centralized approach with 75% of the training set, and federated approach with 3 clients each using 33% of the training set.

one. In contrast, the centralized configuration using only 75% of the training set results in lower performance, indicating that it makes more wrong predictions than the other two configurations. The F1-score can be interpreted as a harmonic mean of precision and recall. The precision is the number of true positives divided by the number of all positives. At the same time, the recall is the number of true positives divided by the number of all samples that should have been identified as positive. In healthcare scenarios, predicting false negatives and effectively distinguishing

between patients with and without disease are two serious concerns. For this reason, the F1-score and the AUROC are the two key metrics to consider while evaluating the ML model. However, some of the papers considered do not show all the metrics. AUROC is the only one present in each. The model presented in [99] was trained to predict three different cases of risks of mortality: within 3 days, 7 days, and 14 days, respectively. Table 4.2 shows the performance of risk mortality within 14 days. The model comprises a hybrid neural network comprising a Convolutional Neural Network (CNN) and a Long Short Term Memory (LSTM). CNNs are widely used to identify patterns, while LSTM networks are known for their ability to remember which information in a sequence is the most important. The data used for training are the readings of some vital signs recorded over a 24-hour window. In [100], the authors trained eight different classifiers: decision tree, linear discriminant, logistic regression, SVM, random forest, boosted trees, gaussian SVM, and k-nearest neighbors (k-NN) algorithm. In Table 4.2, we reported the performance of the decision tree since it achieves the best performance. Quantitative features were extracted from the heart rate signals of ICU patients with cardiovascular disease to predict risk. Each signal is described in terms of 12 statistical and signal-based features. Finally, the authors of [101] used a CNN architecture to predict the patient risk of death. They employed the time series of readings of some vital signs (Heart Rate, Respiratory Rate, Systolic Blood Pressure, and Diastolic Blood Pressure) as training data. They compared their model with other architectures (i.e., recurrent neural network and logistic regression), showing how it outperformed them. The papers show how models not based on deep neural networks manage, in some cases, to match or even outperform them. The most recent works generally prefer models based on 1-dimensional CNNs, whose usage has recently emerged in processing time series.

4.4.3 CONCLUSION

FRAMH is a risk-based authorization framework that combines FL and blockchain to assess patient health risk levels, enhancing access control decisions while balancing security and patient safety. As healthcare data from IoMT devices continues to grow, FL helps medical institutions comply with strict legal and regulatory restrictions that prohibit sharing sensitive patient data. FRAMH allows hospitals to collaborate on shared goals while preserving data privacy, using FL to infer patient health status without centralizing data. The prototype shows that FL can match and sometimes outperform centralized approaches in performance, especially when training data is limited, highlighting FL's efficiency with smaller, distributed datasets. This approach has potential applications in other healthcare settings, like home healthcare and online services needing dynamic, risk-based access control.

4.5 FETA: A DAG-BASED ARCHITECTURE

Traditional blockchains could improve the security of FL approaches. However, their long waiting time for transaction confirmation and high energy consumption are among the most relevant factors limiting their adoption in edge deployment environments, which remain largely unexplored. We believe that DAG-oriented DLTs have the potential to guarantee the needed security features as the blockchain while providing improved performance in terms of latency and energy consumption, thus well meeting the requirements of IoT and edge computing. Therefore, this section proposes a novel architecture called FL at the Edge through the IOTA Tangle (FETA) [102], which leverages the IOTA Tangle [29], discussed in Section 2.2.2, and the InterPlanetary File System (IPFS), which is commonly used in the literature [103] to overcome scalability issues related to the size of data shared in DLT environments. The Tangle, which is the underlying structure of the network, enables fast and feeless transactions on a large scale. Portability is ensured by allowing each component to execute within a container. This enables the deployment and management of FL models on edge devices independently from the underlying hardware and software configurations by providing flexibility and ease of use. Furthermore, the Tangle guarantees the same features as the blockchain, such as decentralized control and immutability. However, local models are not directly shared on it due to their large size, which makes it hard to embed them within transactions. To tackle this problem, they are shared through the decentralized storage of IPFS, while only their Content Identifier (CID) are stored on the Tangle.

In addition, authentication and authorization are implemented through the IOTA Identity framework [104], which offers DID and VC functionalities. Asynchronous communication, which is fundamental in edge scenarios, is enabled by the consensus protocol: this allows parallel transaction processing in FETA and speeds up the sharing of local models. Local models are shared through zero-value transactions that do not require any transfer of value from one entity to another, thus not requiring the provision of cryptocurrencies or tokens to FL participants. Although resource allocation algorithms and incentive mechanisms are out of the scope of our FETA proposal, at least at this stage of evolution, they can be easily integrated. Finally, as discussed above, the proposed architecture can be effectively deployed on real-edge devices.

4.5.1 ARCHITECTURE

As shown in Figure 4.10, to enable FL at the edge, our proposed solution consists of multiple edge nodes that receive data from IoT devices and train local models. The edge nodes host the following components:

- FL client: it trains a local model using data received from IoT devices.

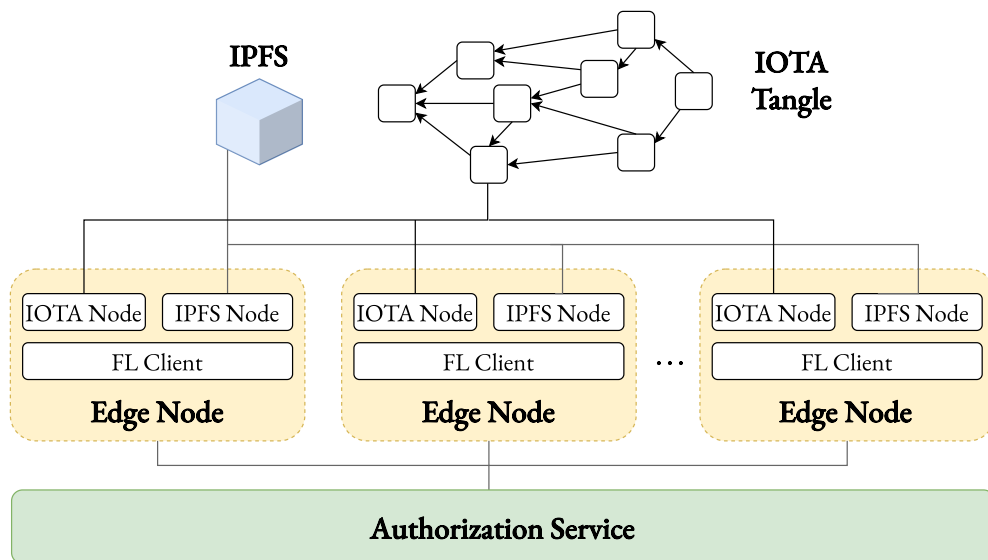


Figure 4.10: FETA architecture.

- IPFS node: it stores and distributes the updated models across the network.
- IOTA node: it belongs to the IOTA Network and cooperates with other nodes to maintain the unified view of the Tangle. It shares references of local models among participants by enabling their retrieval via IPFS nodes.

In addition to these components, our architecture includes an Authorization Service (AS) that generates VCs for clients willing to participate in the FL process. Only local models with valid VC will be included in the aggregation process. The AS is managed by the FETA participants that can establish whether participants are allowed to join.

AUTHORIZATION SERVICE

The AS is a critical component regulating participation in an FL process. Its primary function is to issue VCs that validate the eligibility of FL clients to contribute to FL training. To ensure the authenticity and prevent the tampering of the VCs, they are signed by the AS using the private key associated with its DID. As DIDs are public, FL clients can automatically retrieve the corresponding public key of the AS and verify whether a local model has been published by an authorized participant or not. Therefore, this approach allows AS to be involved only in the initial issuance of VCs. The participating clients can perform subsequent verification operations, promoting decentralization and improving scalability.

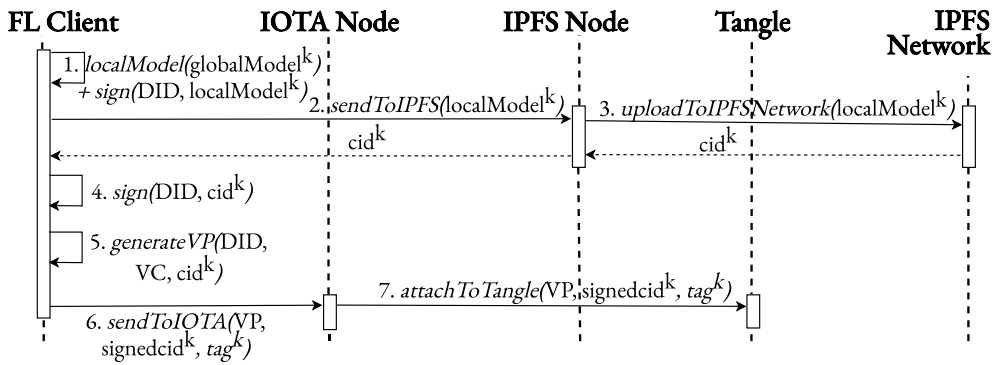


Figure 4.11: Publishing local models.

FL CLIENT

The FL clients receive data from IoT devices to locally train their models by using any desired ML algorithm. To contribute to FL training, each client must first obtain a valid VC from the AS that allows participating in FL training. The client signs the VC using the private key associated with its DID, obtaining a VP that proves its eligibility. Before including a local model in the aggregation, each client also verifies that the model has been published by an authorized participant.

IOTA NODE

An IOTA network consists of multiple IOTA nodes, which store a copy of the Tangle: these nodes are vital to ensuring the integrity and reliability of the IOTA network by participating in the consensus process and validating transactions before adding them to the Tangle. Besides, IOTA nodes can perform other functions, such as acting as gateways for communication between users and the IOTA network, participating in transaction routing, and providing access to the distributed ledger for applications and other network participants.

In our architecture, each edge node includes an IOTA node that attaches the local models provided by the FL client and retrieves other participants' contributions, which are used to generate the global model. This approach ensures that the FL training process is transparent and secure, as all participants' contributions are recorded on the Tangle, which is immutable and tamper-proof. Additionally, the IOTA node capabilities contribute to the overall FL scalability and reliability by allowing a large number of participants to contribute to model training simultaneously.

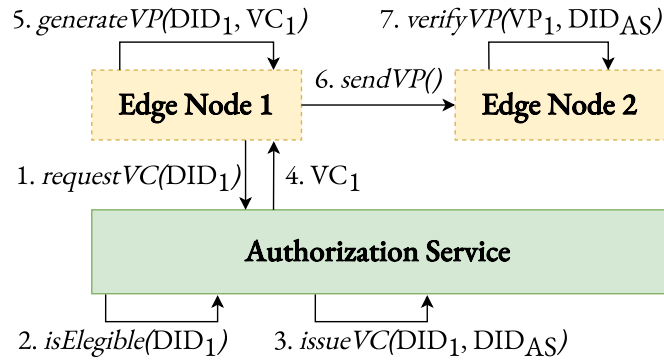


Figure 4.12: Authorization workflow.

IPFS NODE

IPFS participants, or simply IPFS nodes, are programs that can be executed on a local computer to store files and establish connections with the IPFS network. In IPFS, files are stored and shared using a P2P network. This means that rather than depending on centralized servers to store and retrieve files, IPFS nodes communicate with each other directly to transfer files. In FETA, each edge node includes an IPFS node responsible for storing the local models. FETA exploits efficient integration with IPFS because transactions on IOTA have space limitations, i.e., not allowing the direct publication of ML models on it. Models published on IPFS are signed to ensure the origin of the data can be verified.

4.5.2 WORKFLOW

This subsection describes the authorization and FL training stages, which are the main phases that compose the FETA protocol.

AUTHORIZATION

The authorization phase ensures that only authorized FL clients can contribute to the FL training process. In the following, we detail the authorization workflow depicted in Figure 4.12:

1. Each FL client requests a valid VC to the AS, providing its DID.
2. The AS verifies the eligibility of the FL client associated with the provided DID.

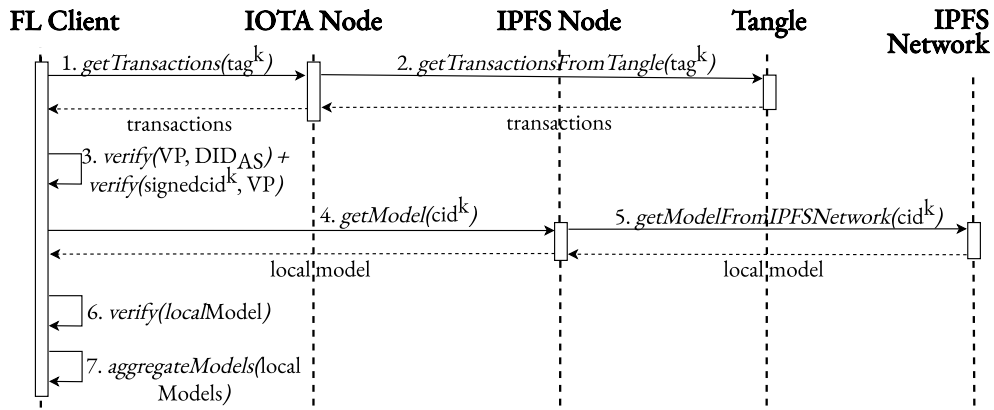


Figure 4.13: Aggregating local models.

3. Upon successful verification, the AS issues a valid VC associated with the client’s DID and signed through its private key.
4. The FL client receives the issued VC from the AS.
5. The FL client generates a VP by signing the collected VC with its own private key, thus proving its eligibility.
6. The FL client attaches the VP to the transaction that contains the necessary information to retrieve its local model.
7. Before downloading local models submitted by other participants, each FL client verifies the validity of the VP using the public keys associated with the AS and other FL clients, which are linked to their respective DIDs.

FL TRAINING

The FL training phase consists of two sub-phases: *publishing local models* and *aggregating local models*. Each sub-phase contributes to the collaborative training process in FL.

Publishing Local Models. After issuing VCs, FL clients can train their models using local data and share the trained local models with other participants. The protocol for publishing local models, as depicted in Figure 4.11, is described below:

1. Each FL client trains a local model using its on-premises data. Once the training is complete, the hash of the local model is signed with the private key associated with its DID.

2. The signed hash, along with the model, is sent to the IPFS node, which forwards the data to the IPFS network, which returns a CID to the FL client.
3. The FL client signs the received CID, ensuring the integrity of the model.
4. Using the VC and its associated DID, the FL client generates a VP that serves as proof of eligibility for FL training.
5. Finally, the VP and the signed CID are bundled in a message and sent to an IOTA node. The IOTA node attaches this data to the Tangle and indexes it using a tag corresponding to the FL round for easy retrieval.

Aggregating Local Models. An FL client that has completed its training listens for transactions associated with the tag for that specific FL round to retrieve the local models of other participants. Specifically, the steps needed to perform the aggregation of local models, whose workflow is sketched in Figure 4.13, are reported below:

1. Using the tag of the FL round, the FL client sends a request to the IOTA node to retrieve all the transactions associated with that tag.
2. The IOTA node collects the transactions indexed with the tag and returns them to the FL client.
3. For each transaction, the FL client first verifies the validity of included VP using the public key of the AS. Upon successful verification, it checks that the CID has been signed by the FL client linked with that VP.
4. Given the CID, the FL client requests the corresponding local model to the IPFS node.
5. The IPFS interacts with the IPFS network and responds to the FL client with the requested local model and its signed hash associated with that CID.
6. The FL client verifies the hash has been signed by the FL client associated with that VP and checks that the hash of the retrieved local model matches the signed one.
7. Finally, local models are aggregated to generate the global model, which will be used for the following FL round.

Algorithm 1 presents the algorithm implemented by the FL client, which comprises both the publication and aggregation of local models.

Algorithm 1 FL client - k FL round

```

Input:  $did_{AS}; did; VC; globalModel^{k-1}; tag^k; stopCond$ 
Output:  $globalModel^k$ 
 $localModel^k \leftarrow trainModel(globalModel^{k-1})$ 
 $modelHash^k \leftarrow hash(localModel^k)$ 
 $signedHash^k \leftarrow sign(did; modelHash^k)$ 
 $cid^k \leftarrow sendToIPFS(signedHash^k; localModel^k)$ 
 $signedCid^k \leftarrow sign(did; cid^k)$ 
 $VP \leftarrow generateVP(did; VC)$ 
 $sendToOTA(VP; signedCid^k; tag^k)$ 
 $partialModels^k \leftarrow []$ 
while not stopCond do
   $VP_i; signedCid_i^k \leftarrow getFromOTA(tag^k)$ 
  if  $verify(VP_i; did_{AS})$  and  $verify(signedCid_i^k; VP_i)$  then
     $signedHash_i^k; model_i^k \leftarrow getFromIPFS(cid_i^k)$ 
    if  $verify(signedHash_i^k; VP_i)$  and  $check(signedHash_i^k; model_i^k)$  then
       $partialModels^k.push(model_i^k)$ 
    end if
  end if
end while
 $globalModel^k \leftarrow aggregateModels(partialModels^k)$ 

```

4.5.3 EXPERIMENTAL EVALUATION

This section aims to demonstrate the practical feasibility of deploying our FETA solution in real-world scenarios. Our proposed architecture is designed to be highly adaptable and flexible, enabling the integration of a wide range of FL algorithms and strategies. Our experiments are focused on showcasing the practical applicability of FETA, with particular emphasis on the respect of challenging requirements in terms of latency and power consumption, which are central to the most original aspects of our proposal for FL at the edge.

We deployed FETA on a real cluster by scaling the number of edge nodes involved in the training phase. Each edge node is equipped with an Intel(R) Core(TM) i5-3470 CPU running at 3.20GHz and 12 GB of RAM, while each component executed at the edge is run within a Docker container.

EXPERIMENTS

Guaranteeing efficient communication and assessing energy consumption are two critical concerns to successfully enable FL in edge computing scenarios. Therefore, our analysis primarily aims to evaluate FETA's latency and power consumption during an FL process. Since each component is executed within a Docker container, we obtained resource usage using docker-activity

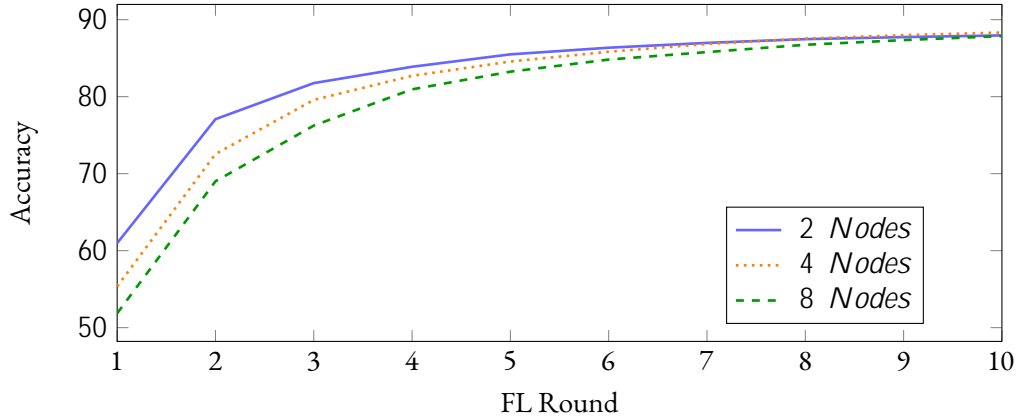


Figure 4.14: Accuracy comparison with 2, 4, and 8 edge nodes in the first experiment.

[81], a tool that provides accurate power consumption, memory, and CPU usage. To offer a comprehensive estimation, we conducted a series of experiments by scaling the number of FL clients from 2 to 8 while also varying the complexity of ML models and datasets. Each experiment was repeated 10 times to ensure accuracy, and the results were aggregated.

For the first experiment, we utilized the widely used MNIST dataset [105], which includes 60,000 training and 10,000 test samples. Each sample consists of a 28x28 grayscale image of a handwritten digit (0 to 9) and its corresponding label. Our ML model was a neural network with three layers: a Flatten input layer that transformed the 28x28 image into a vector of 224 elements, a Dense layer with 128 neurons using the ReLU activation function, and an output layer with 10 neurons, one for each digit class. We employed the Adam optimization algorithm with a learning rate of 0.001 and used the sparse categorical cross-entropy loss function. We aimed to increase the complexity for the second experiment by employing the CIFAR-10 dataset [83] and a Deep CNN. The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. It contains 50,000 training images and 10,000 test images. On the other hand, the model is a more complex neural network with 2.626.634 parameters, about 20 times more than the network used for the first experiment. The network comprises multiple convolutional and fully connected layers, interspersed with activation functions, batch normalization, max pooling, and dropout layers to enhance performance and prevent overfitting. This setup allowed us to assess the framework’s performance on a more intricate dataset and a larger neural network. In both cases, the neural network was trained locally for 5 epochs, and 10 FL rounds were performed. To ensure fairness and comparability, the datasets were fairly split among all the FL clients.

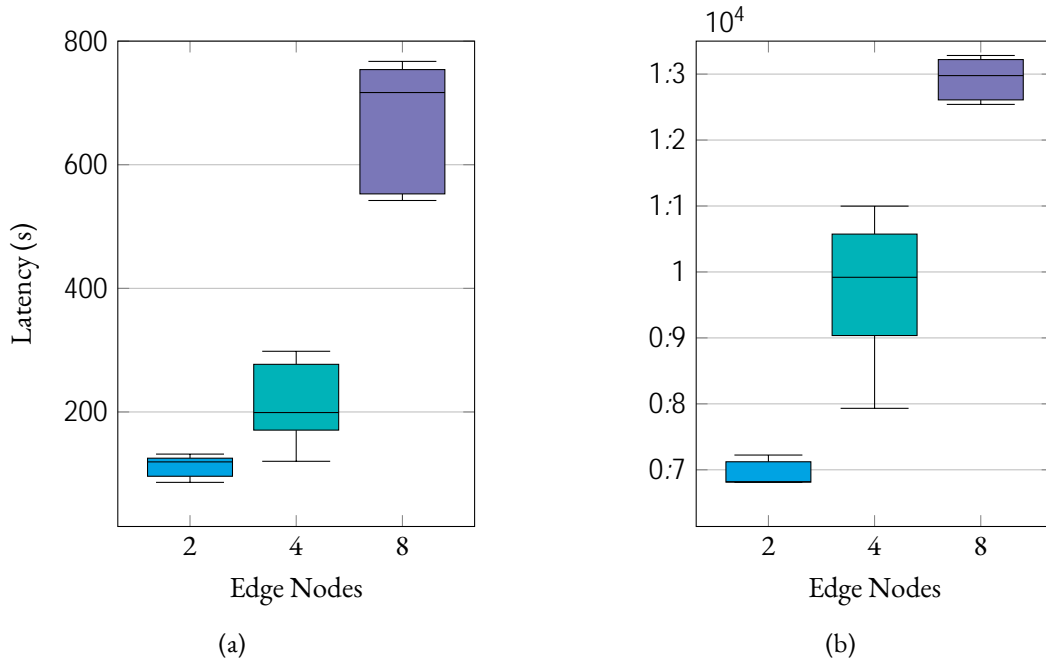


Figure 4.15: Latency comparison with 2, 4, and 8 edge nodes in the first (a) and second (b) experiments.

PERFORMANCE RESULTS

In this section, we present the results of our experiments, which evaluated the accuracy, latency, power, and resource consumption of our proposal.

Accuracy. Our experiments prioritize the evaluation of latency and energy consumption. However, achieving high accuracy is still crucial to any FL platform. An FL platform that sacrifices accuracy for low latency and energy consumption may not be widely adopted. Thus, while ML metrics are not the primary focus of our proposal, we understand their importance and have taken steps to ensure that our platform can deliver satisfactory results in terms of accuracy. Figure 4.14 reports the training results obtained using the MNIST dataset, showcasing the high accuracy achieved by FETA. For brevity, we only depicted the results from the first experiment. However, it is worth noting that the training performed with the CIFAR-10 dataset also demonstrates notable accuracy, reaching approximately 80%. Notably, these results are independent of the FETA architecture and are solely determined by the adopted aggregation strategy. Specifically, we employed FedAVG in these experiments.

Latency. In Figure 4.15, we present the results of our experiments on the mean latency of the FL training as we scale up the number of edge nodes involved. The reported latency includes both the FL training and the authorization procedure. The higher latency of the second experiment is due to the complexity of the CIFAR-10 dataset and the employed ML model that demands a longer

Table 4.3: Latency for authorization procedures and FL training.

Operation	No. of Nodes	Mean Exp 1	Mean Exp 2
DID Creation	2	90.1s	94.87s
	4	96.94s	72.51s
	8	94.35s	104.48s
VC Creation	2	5.41ms	5.49ms
	4	5.38ms	5.32ms
	8	5.82ms	6.88ms
VP Creation	2	1.76ms	1.87ms
	4	1.72ms	1.87ms
	8	1.92ms	2.00ms
FL Rounds	2	112.80s	6951.02s
	4	212.19s	9258.29s
	8	675.52s	12918.49s

training phase. Indeed, the overhead related to FETA operations is minimal compared to the time needed to perform the training. Please refer to Table 4.3 for a detailed breakdown. The table highlights a sublinear relationship between the number of edge nodes and the training latency. Our observations indicate that a smaller number of participants results in lower latency. This can be attributed to the fact that FL clients have to wait for fewer contributions before aggregating them. The creation of DIDs, VCs, and VPs is an occasional process that is not affected by the number of nodes, as well as the ML model and dataset. On the other hand, the latency of their verification depends on the number of participants, which is included in the whole FL process.

Power Consumption. Figure 4.16 depicts, for both experiments, the results of the mean power consumption of each component deployed on an edge node, while Table 4.4 presents their respective mean and maximum power consumption. The power consumption of the IPFS node can not be directly observed in Figure 4.16 since its contribution is lower than 0.02 W and can be neglected. The IOTA node does not show remarkable differences when scaling the number of edge nodes or increasing the complexity of the ML model.

Interestingly, a smaller number of participants leads to higher medium power consumption. This can be attributed to the fact that with fewer FL clients, the latency related to waiting for all participants is reduced. However, this also means that the waiting time is very low, causing FL clients to continuously perform operations, which increases the mean power consumption per unit of time. It is worth outlining that the whole energy consumption of the FL training with

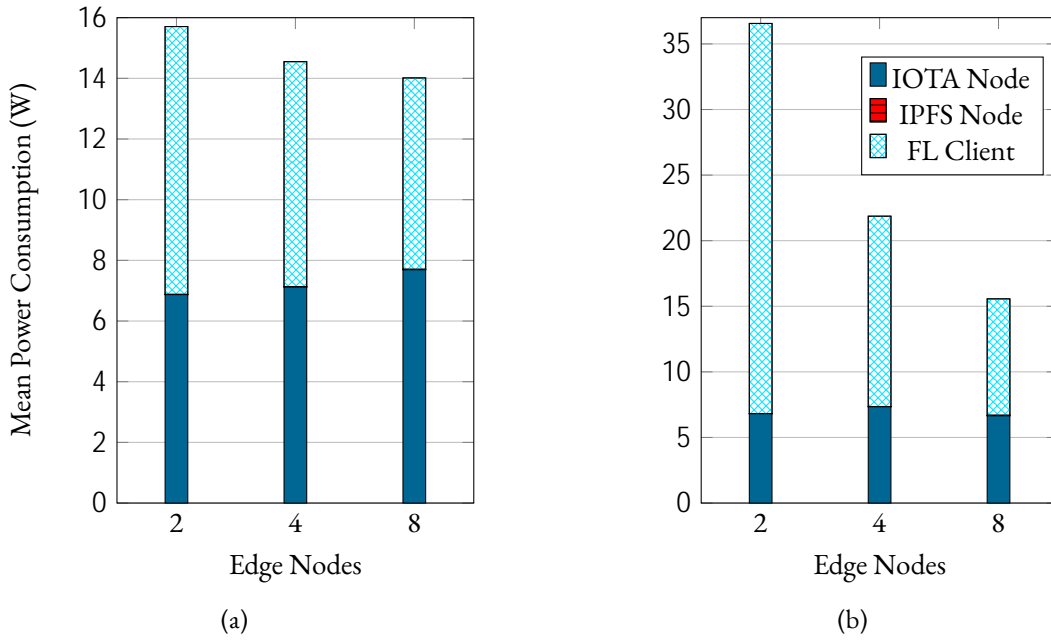


Figure 4.16: Power consumption comparison with 2, 4, and 8 edge nodes in the first (a) and second (b) experiments.

Table 4.4: Comparison of each component’s power consumption (W).

Component	No. of Nodes	Exp 1		Exp 2	
		Mean	Max	Mean	Max
IOTA Node	2	6.87	44.43	6.81	33.65
	4	7.12	43.61	7.34	44.33
	8	7.69	58.25	6.95	38.17
IPFS Node	2	0.007	0.32	0.009	1.87
	4	0.01	0.49	0.01	3.64
	8	0.013	1.00	0.03	6.44
FL Client	2	8.83	64.32	29.74	74.29
	4	7.42	63.33	14.51	69.72
	8	6.31	54.16	8.88	86.70

4 and 8 nodes will be higher since it demands more time. The collected metrics for the second experiment show a slight increase in power consumption due to the longer training time.

Resource Consumption. To provide a comprehensive analysis of our proposal, we included Table 4.5 and 4.6, which detail the memory, CPU, and power usage for each component. We found that increasing the number of participants results in higher memory consumption for the

Table 4.5: Comparison of memory (MB) and CPU (%) usage for each component.

Component	No. of Nodes	Exp 1				Exp 2			
		Memory (MB)		CPU (%)		Memory (MB)		CPU (%)	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
IOTA Node	2	76.88	112.39	15.06	75.88	93.24	133.95	14.95	67.75
	4	74.57	115.68	15.46	76.38	87.95	134.96	15.64	75.53
	8	76.51	117.96	16.18	97.97	91.65	141.65	15.13	73.54
IPFS Node	2	18.89	94.23	0.13	3.81	219.78	1487.89	0.11	7.70
	4	25.89	156.55	0.16	4.80	392.86	2780.80	0.15	8.43
	8	47.79	282.28	0.19	6.65	528.13	3265.12	0.24	15.34
FL Client	2	134.37	728.02	16.46	100.0	1239.20	2264.21	43.33	96.43
	4	151.62	622.24	12.83	99.0	1117.36	2280.95	21.57	95.29
	8	184.45	458.30	10.24	90.0	1107.95	3010.91	12.28	96.9

Table 4.6: Power consumption (W), memory (MB), and CPU (%) usage of the AS.

No. of Nodes	Power Consumption (W)		Memory (MB)		CPU (%)	
	Mean	Max	Mean	Max	Mean	Max
2	0.0004	0.05	5.42	6.73	0.008	1.03
4	0.0003	0.05	6.32	8.35	0.007	1.03
8	0.0003	0.09	9.35	11.34	0.005	1.80

FL client, AS, and IPFS Node, as they have to store more data. The IOTA node is unaffected since it does not directly manage the local models. It is worth noting that the AS exclusively handles information pertaining to the authorization phase, making its resource consumption independent of the complexity of ML models. Regarding CPU usage, we observed that only the FL clients show remarkable differences. Specifically, we found that CPU usage decreases when scaling the number of nodes. This trend arises because low latency requires clients to use more resources by continuously engaging in operations like training local models. As expected, the resource consumption of the FL clients is notably impacted by the complexity of the dataset and the ML model. Employing a more complex ML model and larger dataset results in longer training times, consequently increasing the overall usage of the CPU and memory.

4.5.4 CONCLUSION

Deploying FL in edge computing environments presents various challenges that need to be thoughtfully tackled. To address scalability and enhance platform robustness, many researchers have ex-

plored integrating FL with DLT. However, traditional blockchain technologies introduce challenges, including long transaction confirmation times and high energy consumption. In response, this work proposes FETA, a novel architecture utilizing the IOTA Tangle to mitigate these concerns for edge-based FL deployments. Extensive experiments demonstrated that FETA effectively enables FL at the edge, supporting IoT applications in real-world deployment environments.

5 TRUSTWORTHY FL AS A SERVICE

With security as one of its primary objectives, this chapter presents the concept of trustworthy FLaaS [106], a paradigm designed to offer FL as a service while ensuring high levels of security, transparency, and trust throughout the process. The first primary focus is how to establish and deliver trustworthy FLaaS, where participants can securely contribute their data for distributed learning without concerns over misuse, data breaches, or malicious behavior. Moreover, this chapter introduces TruFLaaS, a framework specifically designed to provide a trustworthy FLaaS solution. Building on the foundation of FlowChain, presented Section 4.3, TruFLaaS delivers blockchain technology, smart contracts, and DIDs as a service, creating a transparent, secure, and verifiable FL process for enhanced trust and accountability. These technologies provide the infrastructure to ensure that all model updates and data interactions are tamper-resistant and auditable, thereby increasing trust among participants, even in untrusted or distributed environments. Moreover, offering FLaaS is essential to facilitate the deployment of the FL process, making it more accessible and encouraging broader adoption of FL across various domains.

5.1 MOTIVATION

Despite the benefits of FL, deploying and managing FL systems can be costly and time-consuming, particularly in industries such as healthcare or manufacturing, where the necessary infrastructure and expertise are often lacking. FLaaS addresses this issue by providing clients with a simplified way to leverage FL without needing significant technical knowledge. This solution removes the burden of developing and fine-tuning algorithms and tools while offering flexibility to accommodate participants' unique requirements during the FL training process.

Consider a company that sells industrial machinery and provides predictive maintenance services. Customers using these machines would benefit from participating in FLaaS, as predicting the failure of machine components delivers notable advantages, such as reducing maintenance costs and increasing production efficiency [107]. Since all machines of a specific model share common characteristics, they also follow similar degradation patterns over time. As a result, insights gathered from one machine's environmental conditions could help others avoid similar compo-

ment failures. In this scenario, when a smart manufacturing company purchases a machine, it also receives the infrastructure necessary to participate in FL training. The local training results from each client are then sent to the vendor, who aggregates these into a global model capable of predicting machine behavior across various operational contexts. Although its benefits, there has been limited research on offering FLaaS to third parties. Cloud providers have long provided Machine Learning as a Service (MLaaS) services [108], which include computing resources, APIs, and tools for data analytics. However, these services generally lack support for collaborative training. The goal of FLaaS is to minimize technical overhead while promoting the adoption of FL by offering easy access to collaborative training environments. An effective FLaaS must: (i) relieve developers from data collection tasks, allowing them to focus on algorithms, (ii) preserve data privacy by preventing data transfers, and (iii) ensure trust among participants, which is a core focus of this work.

To achieve this, several critical components must be integrated into a trustworthy FL system, ensuring the learning process's security, fairness, and integrity. As explored in the following section, no existing solutions currently incorporate all of these elements in a comprehensive manner.

5.1.1 TRUSTWORTHINESS

Trust is crucial in FL, especially when participants or service providers may not fully trust one another. For instance, a bias in model aggregation could occur if the entity responsible for aggregating models prefers certain updates over others. Additionally, models submitted by clients must be validated to detect potential malicious contributions. If a compromised model is aggregated, it could degrade the global model's performance or introduce vulnerabilities like backdoors [11]. To mitigate this, service providers must guarantee that (i) all local models are treated equitably, with no biases favoring particular updates, and (ii) validation mechanisms are in place to detect and remove malicious contributions.

Returning to the smart manufacturing example, both the service provider (vendor) and customers (smart manufacturing companies) are potential targets of attacks. A competitor might join the FL process to degrade the global model or disrupt the service. Likewise, a rival to a third-party customer could intentionally degrade the global model, causing machine failures for their competitor.

This highlights why trustworthiness is critical for offering an effective FLaaS. While trustworthy FL has been explored in various studies focusing on challenges like privacy and client selection [109, 110, 111, 112], its application as a service remains underexplored. To address this gap, this work proposes incorporating blockchain and smart contracts to enhance the trustworthiness of local model validation and aggregation processes. While blockchain has been suggested in many

studies to improve reliability and track participant contributions [113, 114, 115, 116], the validation and aggregation processes typically occur off-chain. Performing these operations off-chain diminishes the potential benefits of blockchain’s transparency and reliability. Moreover, off-chain methods lack public verification, meaning clients might question the results.

Sometimes, a participant may not possess sufficient validation data to verify model accuracy. For example, in the predictive maintenance scenario, a manufacturer may not have data on specific faults and thus cannot validate the model’s ability to predict those failures. On-chain validation, while offering better security, faces its challenges. It would require publishing validation data on the blockchain, which could be exploited by malicious actors. To provide a reliable FLaaS, service providers must develop mechanisms that allow smart contracts to validate local models without making validation data publicly accessible. Additionally, blockchain’s transparency should allow participants to verify how validation was performed.

5.1.2 PRIVACY-PRESERVING TECHNIQUES

Although FL helps protect privacy by keeping data on the client’s side, sensitive information can still be inferred from the local models submitted [117]. To further enhance privacy, various privacy-preserving techniques are commonly employed [118, 119]. DP is one such approach, where artificial noise is added to local models before aggregation. However, while stronger noise improves privacy, it can also degrade model performance and slow down training. Finding the right balance between privacy and performance is necessary. Homomorphic encryption [120] is another method that allows operations on encrypted data without decryption. Clients can encrypt their models, send them for aggregation, and receive an encrypted global model that clients can use once decrypted.

5.1.3 AUTHENTICATION AND AUTHORIZATION

For secure interactions in FLaaS, participants must be authenticated and authorized transparently. Centralized identity management is often unsuitable for distributed environments like FL, as it can expose client data to leakage risks and unauthorized sharing. Decentralized authentication mechanisms, such as DIDs and VCs [121], offer a solution by allowing clients to maintain control over their identity and participation. Specifically, to use FLaaS, clients need VCs with the necessary permissions to join the desired FL processes. Moreover, since participants may have different requirements, the service provider has to guarantee that the issued VCs allow clients only to join the FL training that satisfies its demands.

Table 5.1: Comparison of related work based on their features.

Work	Blockchain	Smart Contracts	Validation Mechanism	Validation Dataset	Statistical Analysis	Weight Contributions	Reputations	Incentive
[123]	-	-	X	X	-	-	-	-
[113]	X	X	X	-	X	-	X	X
[114]	X	-	X	X	X	-	X	X
[115]	X	-	-	-	-	-	-	-
[116]	X	-	-	-	-	-	-	X
[124]	X	-	-	-	X	-	X	X
[125]	-	-	-	-	X	X	-	-
[126]	-	-	X	-	X	-	-	-
[127]	-	-	X	-	-	-	-	-
Ours	X	X	X	X	X	X	X	X

5.1.4 INCENTIVE AND PENALIZATION

Since clients may hesitate to share their data, incentives are essential to encourage participation in FL training, ensuring high-quality contributions and a robust global model. At the same time, contributions must be evaluated accurately to reward participants appropriately, avoiding low engagement or financial loss. However, incentives alone are insufficient. Malicious actors could exploit the system for rewards or simply to obtain the global model, as in the case of free-rider attackers. For this reason, penalization mechanisms are equally important to prevent spam or incorrect computations that could harm the global model’s performance [122].

5.2 RELATED WORK

Reviewing the existing literature, it has been found that no FLaaS solution has been proposed that fully addresses all the critical features outlined in the previous section. While various works have introduced individual features such as accountability [115], validation mechanisms [113], or reputation systems [124], none offer a comprehensive solution that integrates all of these elements into a single, trustworthy FL framework. Consequently, a gap remains in developing a fully secure, transparent, and incentive-driven FLaaS solution. This section will discuss related work incorporating some of these key features, focusing on existing efforts that address specific aspects of trustworthiness in FL. However, none provide the complete set of requirements. Table 5.1 provides a comparative summary of key features from existing literature while also previewing our proposal, TruFLaaS, which will be elaborated on in Section 5.3. This comparison highlights the novelty and distinct advantages of TruFLaaS over current approaches.

5.2.1 Accountability and Fairness

Blockchain technology can be a reliable data source, providing participants with a transparent and consistent view of stored data. Lo et al. [115] leverage blockchain to enhance accountability and promote fairness in FL systems by ensuring data-model provenance through blockchain-stored hashed values for data and local and global model versions. To address fairness, they introduce an algorithm that dynamically samples underrepresented classes during training based on the inverse of the weight distribution in the test dataset. While this approach helps build fairer models, assuming mutual trust among participants, it does not safeguard against including malicious contributors in the model aggregation process. This is a significant challenge in FLaaS environments. Basset et al. [116] propose Fed-Trust, a blockchain-enabled edge intelligence framework aimed at detecting cyberattacks in IIoT, but their contribution lacks innovation in local model validation. In their design, fog nodes collect blocks containing local models to compute the global model without addressing verification improvements. The trust component in their work is seen as a target of cyberattacks, with protection provided through a distributed temporal convolution generative network.

5.2.2 Validation Mechanisms

The absence of effective validation mechanisms can lead to the aggregation of potentially harmful models, opening doors for malicious backdoors. Several papers propose new validation techniques to address this issue. TrustFed [113], a blockchain-based framework for decentralized FL systems, removes malicious participants from the training pool using statistical outlier detection methods and relies on blockchain and smart contracts to maintain participant reputations. However, this approach risks excluding contributions that outperform others, especially if those clients benefit from larger local datasets. Chen et al. [114] introduce a blockchain-based decentralized FL framework with a validation process where selected devices act as validators during each training round. Validators cast votes on the legitimacy of local models, allowing for the identification and removal of compromised devices and increasing robustness even in the presence of faulty validators.

Li et al. [123] propose a dynamic verification strategy that uses a server-side dataset to validate client contributions. Only models meeting accuracy thresholds are included in the aggregation process, but their dependence on a centralized server introduces vulnerability. Other recent approaches have moved away from blockchain, but these systems remain susceptible to central points of failure. Wang et al. [126] present PTDFL, a decentralized FL scheme that uses a local proof mechanism to verify the authenticity of submitted models. However, it still permits the

5 Trustworthy FL as a Service

aggregation of malicious models if their proofs are valid. Gao et al. [127] propose SVeriFL, which uses BLS and multi-party security protocols to verify the integrity of local models and aggregation correctness. However, like previous methods, it does not prioritize the quality of local models and depends on a trusted authority, reintroducing centralization risks.

5.2.3 Reputation and Weighted Contributions

To build trust among participants, reputations can guide client selection and model contributions can be weighted based on trust scores. Kang et al. [124] propose using reputation stored on a consortium blockchain to select participants in FL training, with reputation measured by task completion history and past behaviors. While effective for mobile devices, this approach may not be suitable for settings with limited clients, as it risks excluding too many participants, thus limiting the effectiveness of FL. Each contribution may be essential to improve the global model in scenarios with fewer clients.

Cao et al. [125] introduce FLTrust, a Byzantine-robust FL approach that protects against malicious attacks by using a small, clean training dataset on the server to create a baseline model. Trust scores are assigned to local model updates based on their similarity to the server model, which are then used to weigh contributions. However, this method depends heavily on the server's dataset and risks assigning low trust scores to well-performing, honest clients.

5.3 TruFLaaS: a Framework for Trustworthy FLaaS

This section presents TruFLaaS, a blockchain-based, trustworthy FLaaS solution designed to ensure trustworthiness among third-party contributors in the FL process. Building upon the foundation of FlowChain, TruFLaaS leverages blockchain, smart contracts, and a DON to create a secure and reliable validation framework. It introduces a novel validation strategy for aggregating local models, improving the quality of the global model. Clients interact with the service offering FLaaS, and the blockchain, smart contracts, validation set, and DON work together to establish trust throughout the process. The DON is a middleware layer that facilitates the secure and reliable delivery of off-chain validation data to the blockchain. The smart contract validates local models based on a sample of the validation dataset provided by the service provider via the DON. It evaluates them against defined quality metrics (e.g., accuracy). This process enables the generation of high-quality global models. TruFLaaS associates a trust level with each client to ensure fairness, updated during each round of the FL process. This trust level is used to weigh client contributions appropriately. Honest participants are incentivized to contribute their models by rewarding them with tokens or other benefits, while malicious actors are discouraged through

5.3 TruFLaaS: a Framework for Trustworthy FLaaS

Figure 5.1: TruFLaaS architecture.

penalties. By creating these incentives, TruFLaaS fosters a system where trust is earned and maintained through positive contributions.

To the best of our knowledge, TruFLaaS is the first designed and implemented framework that provides trustworthiness in the FLaaS paradigm and performs validation of local models in FL by leveraging smart contracts and a DON. TruFLaaS main components are highlighted in Figure 5.1.

5.3.1 Architecture

As depicted in Figure 5.1, the main components are the Service Provider, the Clients, and the Blockchain, which are described below.

Service Provider

The service provider is the entity that delivers FLaaS to its clients for tasks beneficial to all involved parties. For example, in a scenario where several smart manufacturing companies use the same machinery, preventing equipment breakdown is a shared objective. However, despite this common goal, clients may have varying requirements in terms of metrics, the number of nodes involved, and aggregation strategies. One client, for instance, may want the global model as soon as possible so that they might set a threshold of participants required for model aggregation. Conversely, another client might prefer to wait longer to collect additional contributions for enhanced accuracy.

5 Trustworthy FL as a Service

Therefore, the service provider must implement a flexible service that accommodates these diverse requirements.

Clients share these specifications with the service provider, which then uses them to implement a smart contract that facilitates an FL process tailored to these requirements. Additionally, the service provider is responsible for registering clients and assigning them valid identifiers for blockchain interaction, ensuring that only authorized clients participate in the FL training process.

Since the service provider directly manages tasks under its control (e.g., predictive maintenance of its machines), it is assumed to have a sufficiently large validation dataset for validating local models [123]. After each FL training round, the service provider publishes a sample of this validation set on the blockchain, which is used to validate the local models before aggregation. To prevent model-forging attacks, each validation sample must be unique across rounds, as repeating the same sample could enable malicious participants to craft models that pass validation checks while embedding backdoors. For secure and accurate data publication on the blockchain, the service provider relies on a DON that facilitates the transfer of off-chain data to the blockchain.

Decentralized Oracle Network. Oracles are trusted intermediaries between blockchains and external systems, allowing smart contracts to access real-world inputs and outputs. An oracle verifies, authenticates, and relays data from external sources to the blockchain. In TruFLaaS, oracles supply the validation dataset managed by the service provider to the smart contract, supporting the security and reliability of the validation process. However, a single oracle can create a central point of failure, undermining the decentralized nature of blockchain and introducing security risks, a known issue termed **blockchain oracle problem** [128]. To overcome this, DONs are employed [129]. A DON uses multiple independent oracle nodes and data sources to deliver decentralized, secure off-chain information access. By leveraging information from multiple sources, a DON ensures accuracy and reliability in blockchain data inputs while preserving decentralization. Without a DON, validation data would need to be published on the blockchain, which could allow participants to exploit this information to craft models that, even if malicious, pass validation. Validation data are only provided after aggregation to maintain integrity and security, preventing participants from manipulating the global model.

Authorization Service. Access to FLaaS is regulated using DIDs and VCs. Each client possesses a single digital identity (DID issued by the service provider) with multiple claims (VCs) that prevent misuse and Sybil attacks [130]. This identity information is stored in the client's wallet rather than a centralized location, improving data control and security for external entities (e.g., apps or service providers) [131].

Our DID-based access control system involves:

5.3 TruFLaaS: a Framework for Trustworthy FLaaS

- ^ Claim Holder Clients require VCs issued by the claim issuer to access FLaaS. A VC acts as proof of membership for a specific FL process.
- ^ Claim Issuer The service provider generates a VC, signs it with its DID, and returns it to the client, allowing participation in the FL process.
- ^ Claim Verifier: The smart contract verifies claims. A client signs a VP (containing a VC) with its DID and submits it to the verifier, which checks for a valid VC authorizing participation in the FL process.

Client

Clients collect data from IoT and IIoT devices to independently train local models using their preferred ML algorithm. Upon completion, the local model is enhanced with DP to protect individual data, where carefully calibrated noise is added to model updates, ensuring that the influence of any single data point is indistinguishable, thus protecting sensitive information. The local model is then submitted to a blockchain-deployed smart contract. Each client has a module for FL-related tasks and may either run a local blockchain node, as depicted in Figure 5.1, or connect to one of the provider's nodes. Clients indicate their intent to join an FL process. If existing processes don't meet their requirements, they can communicate new conditions to the service provider, which then initiates a new FL process tailored to their specifications.

Blockchain

As outlined, blockchain and smart contracts establish trustworthiness among the FL participants. Therefore, the service provider must deploy blockchain nodes, while clients may choose whether or not to participate in maintaining the blockchain and contributing like a node in the network. Running a blockchain node allows clients to monitor FL processes directly but can consume significant resources, which could be challenging for clients with limited computing power or storage. This flexible setup allows clients to participate by simply training and submitting local models without managing a blockchain node.

Validation and Aggregation. The smart contract handles validation and aggregation of the global model in alignment with client requirements. First, it verifies client authorization to participate in the specified FL process. Then, it waits until training requirements, such as the desired aggregation strategy, are met (e.g., all participants have submitted their models). The validation dataset, provided by a DON, is used to test local models. Models that meet the performance threshold are included in the aggregation.

5 Trustworthy FL as a Service

Table 5.2: Configuration parameters.

Parameter	Value
Minimum rounds	Positive integer
Minimum participants	Positive integer
Budget	Positive integer
Aggregation algorithm	JSON
Structure of the model	JSON
Privacy techniques	JSON
Metrics	JSON

Reputation System and Incentives Mechanism. If a model is accepted, it results in an increase in the client's trust level. Conversely, if a model fails to meet this threshold, it is discarded, leading to a reduction in the client's trust level. At the end of the FL process, rewards are distributed in proportion to the client's trust levels, promoting a merit-based environment. This is managed automatically through the smart contract, ensuring transparency and fairness in the allocation of incentives, while motivating clients to consistently deliver high-quality models.

5.3.2 Workflow

This section discusses the main phases enabling trustworthiness in a FLaaS environment. Let us consider a service provider that offers FL training to its clients. To train collaboratively, according to given requirements, a global model on a given task. To join the FLaaS offered by a client, a client must be already registered with the service. Once a client is registered with the service, it owns a DID issued by enabling it to join the FLaaS.

Starting and Joining FL Training

A client can either join an existing FL training or start a new one if the requirements implemented by existing processes do not satisfy its demands. TruFLaaS employs a robust authorization workflow, illustrated in Figure 5.2, that leverages DIDs and VCs to regulate all interactions. In the following, we detail the steps involved in initiating a new FL training or becoming a member of an existing one:

1. A client presents its DID and provides the requirements that it has to address. Table 5.2 summarizes the parameters that can be customized.
2. In case there are no pre-existing smart contracts that meet the client's needs, a novel smart contract is created that verifies and aggregates local models according to the specified criteria. However, if such a smart contract does exist, please refer to step 3).

Figure 5.2: Authorization workflow.

3. s returns to c_i a verifiable credential $vc_{i,l}$ signed with its DID that enables c_i to interact with the deployed s_l .
4. once the local training is completed, c_i signs with its DID the previously obtained $vc_{i,l}$, generating a verifiable presentation $vp_{i,l}$. Then, it provides such $vp_{i,l}$ and the local model $mp_{i,l}$ to s_l .
5. s_l verifies the validity of $vp_{i,l}$ through the DID of s , which has released $vc_{i,l}$, and subsequently grants or denies the participation to c_i .

It is worth noting that starting a new s_l is an expensive operation that should be avoided if existing training already satisfies the client's demands.

Trust level

Each client $c_i \in C$ is assigned a trust level $t_i \in [0; 1]$. As pointed out in [132], most trust models in P2P networks distinguish trust towards a peer into direct and indirect. Direct trust is based on previous interactions with that peer, while indirect trust is based on that peer's global reputation. We denote with $TAR_{i,l}$ the Transaction Acceptance Rate, which is defined as:

$$TAR_{i,l} = \frac{TA_{i,l}}{T_{i,l}} \quad (5.1)$$

5 Trustworthy FL as a Service

where $T A_{i,j}$ is the number of accepted transactions T and the total number of transactions made, both referred to the process. The Global Trust Value which is denoted with GT_i , is defined as:

$$GT_i = \frac{\sum_{j=1}^N t_{i,j}}{N} \quad (5.2)$$

where the trust levels are obtained by the clients in past or current FL processes. Thus, leveraging the direct and indirect trust, we calculate the trust level as follows:

$$t_{i,l} = \frac{GT_i + TAR_{i,l}}{2} \quad (5.3)$$

These values are updated at each round through the specific smart contract (Figure 5.3, step 11). At this point, we also consider whether to revoke the client c_i in case its $TAR_{i,l}$ does not meet minimum requirements (i.e., pass a threshold). The average TAR among all participants, and the corresponding standard deviation are calculated at each round. Then, we estimate whether or not a client can, considering the number of remaining rounds, exceed the threshold value set to TAR_{min} . If it fails, the corresponding c_i is revoked, and the client is excluded from η .

Validation and Aggregation

After starting a novel or joining an existing one, a client is given the necessary contribute to that training. The details of the validation and aggregation workflow are depicted in Figure 5.3. These steps are repeated for each round

1. each c_i collects local data from the deployed IoT/IIoT devices.
2. data are used to locally train a local model, enhanced with DP.
3. each c_i provides $mp_{i,l}^k$ and $vp_{i,l}$, which is obtained by signing $mp_{i,l}^k$ through its DID, to sq that validate and aggregate all local models. Algorithm 2 shows the algorithm implemented by the smart contract to validate and aggregate local models.
4. sq forwards $vp_{i,l}$ to a smart contract responsible for authorizing participants.
5. this smart contract will grant or deny access. Specifically, it jointly verifies the validity of $vp_{i,l}$ and ensures that the embedded $mp_{i,l}^k$ has not been revoked.
6. before aggregating all local models, sq waits until the aggregation requirements are met and validates $MP_{i,l}^k$ against a validation set $V_{i,l}$ provided by a DON.

5.3 TruFLaaS: a Framework for Trustworthy FLaaS

7. d requests V_i^k to s .
8. s provides V_i^k to d . Given two rounds t, z , where $t < z$, V_i^z must be $\leq V_i^t$, otherwise $c_{i,j}$ could craft mp_i^z to achieve satisfying performance on a known V_i^k .
9. d returns V_i^k to sq .
10. sq validates MP_i^k against V_i^k . To be accepted, a local model must achieve performance equal to or better than a specific threshold. We employ the Interquartile Range (IQR) method for detecting outliers. This method does not use the median, mean, and standard deviation, being more robust to extremely large or small values. The IQR is calculated as $Q3 - Q1$, where $Q1$ is the first quartile of the data, and $Q3$ is the third quartile. We calculate the threshold as $1.5 \times IQR$ to detect outliers. Any data that falls below this value is considered an outlier and consequently discarded.
11. sq sends the updated reputations to a smart contract employed to trace the proof of each $c_{i,j}$.
12. the smart contracts calculate all the trust levels of each $c_{i,j}$ and returns them to sq .
13. sq aggregate all validated MP_i^k weighting them according to the corresponding $t_i^k \in T^k$.
14. the global model g_i^k is provided to all $c_{i,j}$.

It is worth outlining that transparent collaboration among smart contracts plays a key role in achieving a trustworthy FLaaS. In particular, such a design choice is justified by the following considerations:

- ^ all MP_i^k are validated and aggregated without any biases, guaranteeing the correctness of g_i^k .
- ^ only $c_{i,j}$ having satisfying the process can join
- ^ reputations of $c_{i,j}$ is calculated by a smart contract using the output as input. Thus, we ensure the correctness for each participant.

Figure 5.3: Validation and aggregation work ow.

Incentives and Penalization

To start a novel f_l or join an existing one, clients use tokens that are, by design, the natural incentive mechanism for blockchain-based platforms. Tokens are purchased and earned by clients through positive participation. Moreover, to participate in an already started f_l , tokens are also required to discourage malicious behavior. Participants who provide incorrect contributions are penalized by having a portion of their tokens withdrawn in proportion to their contribution quality score (Q_i). This ensures that all participants are interested in contributing high-quality work and helps maintain the integrity of the f_l . More in detail, at the time of the creation of a new f_l , the budget b_l (i.e., tokens) is locked up into the corresponding $c_{i,l}$ that initialized that f_l . This budget represents an incentive to promote participation to freshly started f_l . At the end of f_l , it will be distributed among the participants according to the corresponding $TAR_{i,l}$. The reward $r_{i,l}$ assigned to each c_i is calculated as follows:

$$r_{i,l} = b_l P \frac{TAR_{i,l}}{\sum_{j=1}^N TAR_{j,l}} \quad (5.4)$$

Such an incentive scheme fairly distributes b_l according to the contributions of all the $2 C_l$. For each $c_{i,l}$, the contribution corresponds to $TAR_{i,l}$. It is clear that, given $c_{i,l}; c_{j,l} \in 2 C_l$, and $TAR_{i,l} > TAR_{j,l}$, it follows that $r_{i,l} > r_{j,l}$.

Furthermore, to deter malicious behavior, before joining each $c_{i,l}$ has to deposit an amount of tokens $d_{i,l}$ bounded by $d_{i,l} \leq \frac{1}{Q_i}$. Thus, the higher a client's reputation, the less it will have

Algorithm 2 Smart Contract - Validation & Aggregation

```

Input: didsk; localModelsk; valSetk; metric
Output: globalModelk
acceptedLocalModelsk []
acceptedDidsk []
performancesk []
for i = 1; localModelsk:size() do
    mp = localModels(i)k
    performance = evaluate(mp; valSetk; metric)
    performancesk:push(performance)
end for
threshold = generatedThreshold(performances)
for i = 1; didsk:size() do
    did = dids(i)k
    mp = localModels(i)k
    accepted = false
    if performancesk(i) > threshold then
        acceptedLocalModelsk:push(mp)
        acceptedDidsk:push(did)
        accepted = true
    end if
    updateReputation(did; accepted)
end for
tk = getTrustLevels(acceptedDidsk)
globalModelk =  $\frac{\sum_{i=1}^M t(i)^k \text{acceptedLocalModels}(i)^k}{\sum_{i=1}^M t(i)^k}$ 

```

to deposit, and vice versa. This amount will be fully returned to the participant at the end of f_i only if the $TAR_{i,j}$ is greater than a threshold. Otherwise, the amount returned will equal $d_{i,j} \cdot TAR_{i,j}$. Such a mechanism is a strong deterrent to voluntarily submitting malicious or inaccurate models, as it would result in an economic loss of tokens.

5.3.3 Experimental Evaluation

To validate and compare our proposal with the existing literature, we consider predictive maintenance and botnet attack detection use cases, which are highly interesting for industrial deployment environments and call for data collection from multiple distributed sources. We first describe the implementation setup for our experiments and the employed datasets, then present the details of the performed experiments, and, finally, discuss the performance indicators we have experimentally measured by drawing some related considerations.

Implementation Setup

TruFLaaS can be integrated into any blockchain infrastructure that supports smart contracts and DONs. For the blockchain component, we leverage FlowChain, which provides a robust founda-

5 Trustworthy FL as a Service

tion for decentralized FL. We add the extra features discussed in the previous sections, such as DP, validation and reputation mechanism, and incentives. FlowChain ensures that all model updates are securely recorded on an immutable ledger, enabling transparent and tamper-resistant tracking of each participant's contributions. We have implemented our smart contracts in NodeJS using TensorFlow libraries to implement the proposed validation protocol. This choice is motivated by the need to recreate an ML model directly in the smart contract: TensorFlow is one of the few frameworks that implements ML also in JavaScript [133]. Concerning the DON, we have used Provable [134], whose only requirement is to deploy a specific smart contract that connects the blockchain and the outside world. Our experiments were run on a Python-simulated FL framework.

Dataset

For the predictive maintenance use case, we selected the NASA Turbofan Jet Engine dataset [135], which is a widely accepted and well-known baseline dataset from NASA for engine degradation modeling. It enables estimating the RUL of the considered engine; the dataset was generated through the simulation of the commercial modular aero-propulsion system. Specifically, it comprises 4 sub-datasets, with temporal signals from 21 sensors (e.g., temperature and fuel flow ratio); each sub-dataset considers different combinations of operational conditions and fault modes. To employ the dataset effectively, first, we performed a data pre-processing step to remove features with non-consistent values. In addition, since the training set does not present RUL values but only the number of time cycles of engine usage, we were forced to calculate them manually. For the purpose of the following evaluation, We assume that RUL decreases linearly over time so that it would have a value of 0 at the last time cycle of the engine: for each engine, RUL is calculated as $\frac{\text{max_time_cycle} - \text{time_cycle}}{\text{max_time_cycle}}$; moreover, as usual for regression problems, we have normalized the input values. Finally, we have split the dataset into training and testing subsets for 100 engines to replicate the behavior of an FL network during the training phase.

Concerning the botnet attack detection use case, we employed the N-BaloT dataset [136], which contains real traffic data gathered from 9 commercial IoT devices authentically infected by Mirai and BASHLITE. Malicious traffic is divided into multiple attacks (e.g., network scanning and rmware), thus enabling us to use it for multi-class classification: 10 classes of attacks, plus 1 class benign. To prepare the data for training, we first used a Label Encoder to convert the target value for each sample into a numerical value. The target value indicates the type of network traffic, either benign or belonging to one of the ten possible attacks. Next, we applied one-hot encoding to these values, resulting in a vector for each sample. Additionally, we normalize each feature using a MinMaxScaler, which scales the data in the range [0, 1]. We implemented a feature

Table 5.3: Botnet attack detection results.

Exp.	Nodes (%)	Cross Entropy	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
#1	0	0.549	77.51	79.31	75.77	73.92
	10	0.461	75.83	74.15	76.42	70.51
	25	0.419	77.5	75.86	79.11	72.20
#2	0	0.390	77.55	81.13	79.47	73.38
	10	0.409	75.83	78.12	76.81	70.7
	25	0.387	74.16	71.43	77.53	69.39
#3	0	0.41	77.39	70.31	77.39	71.95
	10	0.41	77.38	70.29	77.38	71.94
	25	0.42	76.43	82.04	76.43	70.74

selection mechanism based on an ExtraTreesClassifier to minimize the number of features. Tree estimators are utilized to compute feature importance through impurity calculations, which can be combined with the SelectFromModel meta-transformer to eliminate irrelevant features.

Experiments

To show the effectiveness of our solution, we conducted several experiments considering the application domains of predictive maintenance and botnet attack detection. In each of them, we considered both honest clients with a limited dataset and malicious nodes, which aim to either disrupt the training process or introduce backdoors within the global model. We compare TruFLaaS against both the conventional baseline (i.e., no validation mechanisms) and TrustFed [113], i.e., a framework for fair and trustworthy FL. For fairness, we use the same FL model in the predictive maintenance use case, configured as follows. The input layer takes 24×24 input neurons for each window. Two middle dense layers represent 24×24 neurons, while the output dense layer is mapped on 24×1 neurons. The ReLU activation function is used for all the layers, and weights are adjusted through SGD optimizer. Mean Absolute Percentage Error (MAPE) is used to evaluate the accuracy of each model at the end of the global aggregation. In contrast, we calculated Mean Absolute Error (MAE) to validate local models and identify the most beneficial ones for training. However, TrustFed was not thoroughly validated against multiple datasets and models. Therefore, we developed a novel FL model consisting of four layers for the botnet attack detection scenario. The first two layers are Dense layers with ReLU activation functions, comprising 64 and 32 neurons, respectively, followed by a Dropout layer with a rate of 0.2 to minimize overfitting. The final layer is a Dense layer with 11 neurons, one for each class to be predicted, utilizing a softmax activation function. Due to the problem being a multiclass classification, we

5 Trustworthy FL as a Service

used categorical cross-entropy as the loss function. Moreover, since this use case does not aim to solve a regression problem, in Table 5.3, we report the final evaluation metrics for the experiments conducted.

Heterogeneous Data Distribution First, we consider a scenario where data distribution among honest clients is heterogeneous. Hence, some nodes have more or fewer data samples than others. Having clients with heterogeneous data distribution is one of the major reasons for adopting FL. This situation is quite common in industrial environments since the size of enterprises directly affects the amount of data generated. We run this type of experiment by varying the number of participants and using different percentages of nodes with augmented data.

Heterogeneous Data Distribution on Rare Cases This experiment is a special case of the previous set. In FLaaS, clients may be interested only in a specific subtask (e.g., RUL under a given threshold or a specific botnet attack). We focus on a deployment environment where some nodes have no data on a particular class of events, which we denote as rare cases. For example, there may be smart manufacturing enterprises that have not yet experienced the breakdown of specific machinery or have never been affected by a certain attack. In this experiment, we discriminate the data records according to their RUL values for the predictive maintenance use case. In particular, we tag all the samples inside a low percentile of a pseudo-normal distribution as rare records by separating the low RUL values from the others. Through statistical analysis, we identify a subset of the data containing the data records with low RUL. We calculate the i th percentile values on the training and the validation sets. Since NASA data do not follow a normal distribution, we operate a standardization process to use the score table to calculate exact areas for any given normally distributed populations. Mathematically, the standardization operation is described by the formula below:

$$z = \frac{x - \mu}{\sigma} \quad (5.5)$$

where z is the z -score value, x is the observation value, μ is the mean of the distribution and σ is the standard deviation of the distribution. A multiclass classification approach is used for the botnet attack detection use case. Thus, we consider the two attack classes with the lowest occurrence to be rare cases. Specifically, such classes are represented by junk and scanning attacks. Our experiments involve varying the number of nodes without rare cases and the strategy for discarding nodes. We test four strategies using two validation sets: one with only rare cases (Rares) and another with the same data distribution as the global test set (Overall). The first strategy entails discarding nodes that perform poorly on the validation set that contains only rare cases. The second strategy involves discarding nodes that perform poorly on the second validation set. The third strategy requires discarding nodes that perform poorly on both the first and second

5.3 TruFLaaS: a Framework for Trustworthy FLaaS

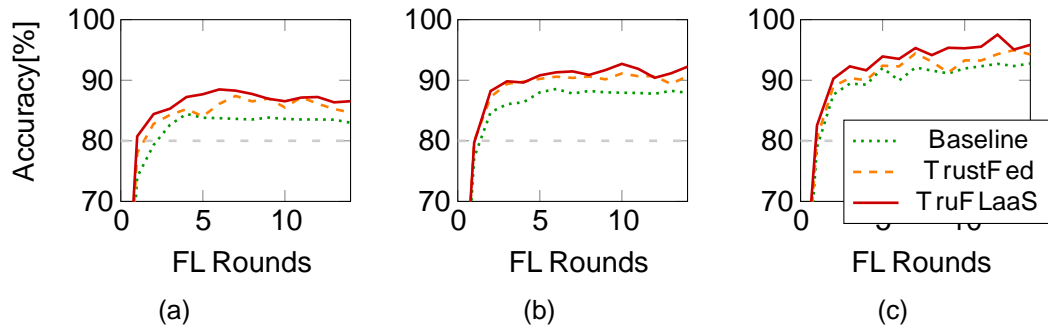


Figure 5.4: Predictive maintenance with heterogeneous data distribution showing accuracy comparison of different node selection strategies with 0 nodes (a), 10 nodes (b), and 25 nodes (c) containing augmented data.

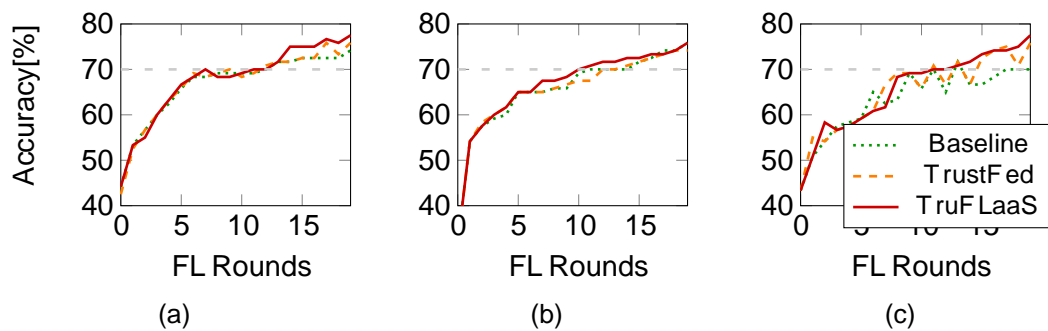


Figure 5.5: Botnet attack detection under heterogeneous data distribution showing accuracy comparison of various node selection strategies with 0 nodes (a), 10 nodes (b), and 25 nodes (c) containing augmented data.

validation sets. Lastly, the fourth strategy involves discarding nodes that perform poorly on either the first or second validation set.

Model Forging Attack. Ensuring the security and integrity of FL platforms is a major concern due to their widespread adoption in various domains. In model forging attacks, a malicious participant could craft a local model to introduce backdoors into the global model or simply disrupt the training process. To assess the resilience of TruFLaaS against this class of attacks, we conducted several experiments with different percentages of malicious nodes. In these experiments, as done in TrustFed, we simulated malicious nodes' behavior by performing training on data containing random noise to corrupt the model.

Results and Associated Considerations

The experimental results reported in this section show that our solution outperforms conventional baselines and TrustFed under all the considered circumstances.

5 Trustworthy FL as a Service

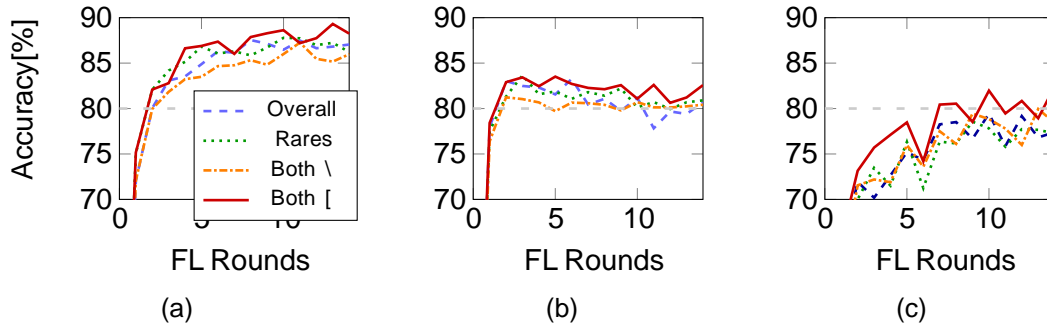


Figure 5.6: Predictive maintenance with rare case heterogeneous data distribution showing accuracy comparison for different node selection strategies with 0 nodes (a), 10 nodes (b), and 25 nodes (c) lacking rare data.

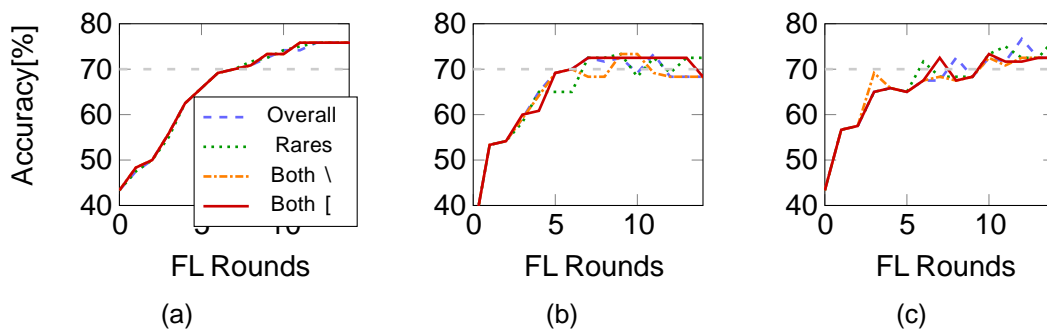


Figure 5.7: Botnet attack detection with rare case heterogeneous data distribution showing accuracy comparison across node selection strategies with 0 nodes (a), 10 nodes (b), and 25 nodes (c) lacking rare data.

Heterogeneous Data Distribution TrustFed only aggregates local models whose accuracy falls in the interval identified by the neighborhood of the medium and standard deviation. However, the TrustFed approach neglects clients with a heterogeneous data distribution that results in high-quality local models (which can achieve performance results that overcome the bound of the interval). Also, TruFLaaS discards local models whose performance is below its threshold; on the opposite, in the aggregation phase, TruFLaaS involves all the local models whose accuracy is more significant than the lower neighborhood of the medium and the standard deviation. Figure 5.4 and 5.5 show how TruFLaaS outperforms TrustFed by better identifying clients' contributions with significantly larger datasets. TruFLaaS reaches the target accuracy faster than the others and gains a more significant advantage with the increased number of nodes with augmented data.

Heterogeneous Data Distribution on Rare Cases In this type of experiment, we have not compared TruFLaaS with TrustFed because the latter does not distinguish rare cases, and our approach would undoubtedly perform better. Figure 5.6 and 5.7 highlight the experiment results by

5.3 TruFLaaS: a Framework for Trustworthy FLaaS

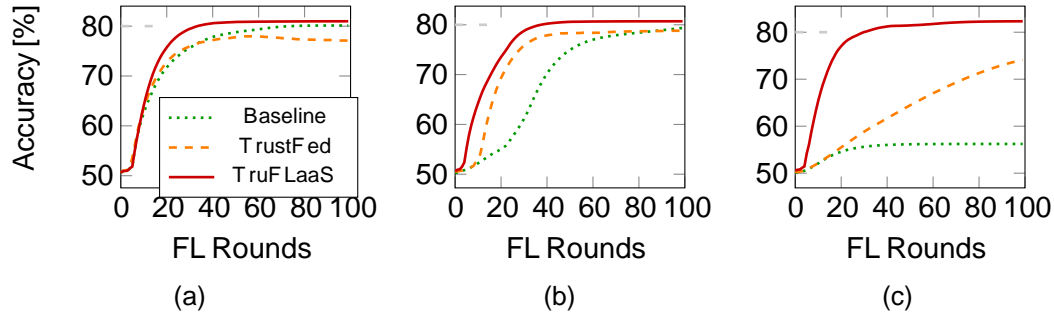


Figure 5.8: Predictive maintenance under model forging attack showing accuracy comparison of approaches with 0 malicious nodes (a), 10 malicious nodes (b), and 25 malicious nodes (c).

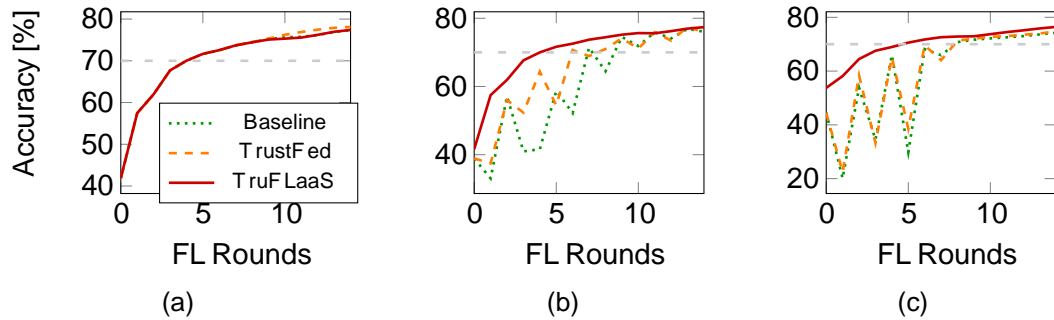


Figure 5.9: Botnet attack detection under model forging attack showing accuracy comparison of approaches with 0 malicious nodes (a), 10 malicious nodes (b), and 25 malicious nodes (c).

varying the number of nodes with no rare data and the strategy used to discard nodes with poor performance. We can observe that in the predictive maintenance scenario, the first two strategies, i.e., discarding nodes that perform poorly on the validation set comprising only rare cases and discarding nodes that perform poorly on the validation set with the same distribution as the test set, performing better than the other aggregation strategies considered. In the botnet attack scenario, we can see the resilience of our solution to an increasing amount of nodes without rare data. The accuracy is not affected negatively by the higher amount of heterogeneous nodes.

Model Forging Attack. In the predictive maintenance use case, since the FL configuration employed by TrustFed was not leading to acceptable results compared to our solution, we increased the number of FL rounds to 100. Figure 5.8 and 5.9 sharply outline how TruFLaaS is more robust than TrustFed against model forging attacks by varying the number of malicious nodes. This is mainly motivated by the fact that TrustFed, by using the mean and standard deviation to detect outliers, is less accurate in the presence of excessively large or small values. For example, an outlier with poor performance might significantly shift the total mean. In this case, TrustFed accepts outliers with performance that should not be accepted under nominal conditions. More-

5 Trustworthy FL as a Service

over, not weighing the local models and aggregating an outlier will completely ruin the training performed up to that point. It is also interesting to observe that, in the predictive maintenance experiment, TrustFed performs worse than the baseline with 0 malicious nodes. This could be caused by TrustFed also removing nodes that reach performances much greater than the average.

On the contrary, TruFLaaS employs a more robust outlier detection algorithm, achieving, in all cases, very similar performance. Moreover, due to the use of weights based on the number of accepted transactions (i.e., level of trust), sporadic errors during the validation process would result in very little influence on the global model without disrupting the whole training process. These results provide valuable insights into the robustness of our proposal and help ensure that it can effectively protect from model forging attacks.

5.3.4 Conclusion

Service providers can facilitate and promote the adoption of FL by offering it as a service. FLaaS reduces the overhead and technical expertise required to develop and re-tune collaborative algorithms and tools for training a global model on shared tasks. However, while FLaaS offers distinct advantages, designing an effective FLaaS solution presents various technical challenges that must be addressed. A significant obstacle is the lack of trust among untrusted participants, limiting FL adoption in real-world scenarios. To address this, the work proposes a blockchain-based architecture that transparently validates local models using blockchain, smart contracts, and a DON. Specifically, before aggregation, local models are validated against a round-specific validation sample provided by the service provider via a DON. TruFLaaS employs smart contracts to track each participant's trust level, which in turn influences the weight of each model in the aggregation phase. Extensive experiments demonstrate that TruFLaaS outperforms traditional baselines and SOTA approaches in detecting malicious nodes across various use case scenarios.

6 Federated Unlearning: Taxonomy, Approaches, and Solutions

To ensure regulatory compliance and enhance security and privacy in FL systems, this chapter explores the concept of FU, a crucial aspect of ensuring privacy and regulatory compliance in FL systems. We provide a comprehensive taxonomy of FU, exploring and categorizing the proposed approaches and solutions to implement unlearning in federated settings. Additionally, this chapter presents a novel unlearning algorithm specifically designed to operate in malicious settings, ensuring that even in adversarial environments, participants' privacy is safeguarded. By examining current solutions and introducing a novel approach, this chapter aims to provide a clear understanding of the state of FU and its role in enhancing trust, privacy, and compliance in FL.

6.1 Motivation

The growing reliance on distributed user data has stimulated the demand for privacy-preserving measures. To address this, recent years have seen the development of privacy protection regulations reflecting increased awareness around third-party handling of personal information. The EU and California, which introduced the GDPR [3] and the CCPA [4], respectively, are among the pioneers in this area. These regulations include directives regarding data collection, storage, and processing and allow individuals to request deletion of previously shared personal data. For example, the right of erasure, also known as the right to be forgotten, is one of the cardinal articles in the GDPR [3] and states that the data subject shall have the right to obtain from the controller the erasure of personal data concerning him or her without undue delay [...] where the data subject withdraws consent on which the processing is based. Similarly, CCPA includes the right to delete, which empowers business users to request that businesses delete the personal information they collected from them and tell their service providers to do the same. In ML, fulfilling requests for data removal can be achieved through Machine Unlearning (MU) [137]. Data owners may wish to retract their contributions from a trained model due to privacy or security concerns [138]. This unlearning process involves removing the influence of specific training samples from

6 Federated Unlearning: Taxonomy, Approaches, and Solutions

an already-trained model. A straightforward approach might involve retraining the model from scratch with the remaining data; however, this solution is computationally intensive and impractical, particularly when multiple unlearning requests arise. In FL, the implementation of the right to be forgotten remains complex, as the jointly trained model includes insights from individual data that could leak sensitive information or be vulnerable to Membership Inference Attacks (MIAs) [139] [44], i.e., the ability to infer whether a specific data sample was part of the training set. This raises the question: how should FL respond when clients exercise this right during runtime? In MU, the most basic approach to removing a data sample's impact would be to omit it from the training data entirely. However, retraining a federated model with a sanitized dataset is often unfeasible, as clients may be unavailable for additional rounds, or the computational costs may be prohibitive. Thus, effective FU techniques are essential to modify models directly without discarding good knowledge acquired before the unlearning request.

6.2 Background

To provide context for the following section, it is essential to introduce some background information first.

6.2.1 Machine Unlearning

Considering a set of model weights trained on a dataset, MU aims to remove the influence of a specific subset of D . Let D_u be the subset of D to be forgotten, and D_r be its complement, i.e. $D_u \cup D_r = D$. We will refer to D_u as the forgetting dataset and to D_r as the retain dataset. Let w^r be a set of model weights trained only on D_r , usually referred to as the retrained model. The goal of an MU algorithm is to efficiently obtain, a sanitized version of w , which is indistinguishable or approximately indistinguishable from w^r [140, 137]. Two sets of model weights are indistinguishable if an attacker with access to D_u cannot design a function $g(\cdot)$ to extract information specific to D_u . Considering, for example, a Kullback-Leibler (KL) divergence as a measure for similarity among distributions (distributions of weights in this case), the unlearned model weights and the retrained model weights are formally indistinguishable if their KL divergence is zero [140]:

$$KL(g(w^u); g(w^r)) = 0 \tag{6.1}$$

with $w^u = U(w)$ and $U(\cdot)$ representing an unlearning algorithm. Similarly, the unlearned model weights w^u and the retrained model weights w^r are approximately (or indistinguishable) if their KL divergence is lower than a small threshold ϵ value

$$KL(g(w^u); g(w^r)) < \epsilon \quad (6.2)$$

As anticipated, the naive solution to unlearn the contribution of D_r would consist of retraining the model from scratch only on D_r . However, this method can be costly in terms of time, computation, and energy consumption, especially considering large models. Hence, MU mechanisms should be more efficient than the retraining strategy. Furthermore, in some scenarios, the entire corpus of D_r may not be directly accessible or may not be available anymore after the training of the original model, i.e., as we will see in the following sections, for FL settings.

It is worth noting that MU mechanisms are not only needed to meet regulations about individuals' privacy, such as GDPR and CCPA, but also represent a security tool for the provider of the ML service. Training data may include natural or malicious outliers, which can harm the model performances or introduce potential security threats (e.g., model backdoor). Under this perspective, MU algorithms represent a mechanism to selectively remove the influence of such a subset of threatening data without discarding the useful knowledge already acquired.

6.2.2 Catastrophic Forgetting and Differential Privacy

In this subsection, we introduce catastrophic forgetting and DP, two concepts correlated with FU. In particular, we clarify why these concepts are orthogonal to unlearning.

Catastrophic Forgetting. A significant challenge in Continual Learning (CL) [141] is catastrophic forgetting, which arises when a neural network, while adapting to new tasks, disrupts the representations learned for previous tasks [142]. Recent research shows that catastrophic forgetting also occurs in FL due to shifts in data distribution across rounds. The participant pool can change each round, sometimes including clients with data distributions that are skewed relative to the global data, which may cause the global model to over-specialize in the specific per-round data distributions of active clients while struggling to generalize overall [143, 144, 145].

Although catastrophic forgetting might appear as a natural method for unlearning where the global model forgets specific data contributions once a client is excluded from FL rounds studies indicate [146] that contributions from a participant's data can still persist in the model after they leave training. Thus, catastrophic forgetting is unreliable as an unlearning approach. Additionally,

6 Federated Unlearning: Taxonomy, Approaches, and Solutions

Figure 6.1: FU overview. When the client requests to be forgotten, the FL server erases its contributions from the global model, generating a new version.

unlearning methods applied to an ML model risk inducing catastrophic forgetting as a side effect, potentially removing essential knowledge that should be retained.

Differential Privacy. In ML, it is essential to ensure publicly shared model parameters do not disclose excessively sensitive information about the training data, even when an adversary has auxiliary information. DP [25, 147] addresses this issue by ensuring that an individual's data's inclusion or exclusion in a private database does not significantly impact the analysis outcomes. Ideally, an individual should be largely indifferent to disclosing their data. In essence, DP is a precise mathematical framework to quantify privacy, where thresholds are essentially defined by a privacy budget, represented as

In ML, DP uses ϵ -calibrated noise such as Gaussian noise to perturb model updates or gradients, balancing model performance and privacy, as ϵ values introduce stronger noise and greater privacy [148, 149, 150]. Since DP must be integrated at the start of FL training, it cannot be applied retrospectively. Although DP helps mask individual participation, limiting vulnerability to membership attacks, it does not remove malicious contributions from adversarial clients, which makes it a complementary tool rather than a standalone solution for unlearning.

6.3 Federated Unlearning

FU is a technique to selectively remove certain data contributions from a federated model. This approach enables the model to effectively forget specific data without requiring a full retraining cycle, maintaining model performance and integrity. Suppose that, as part of FL, a given client u submits an unlearning request for a subset of its private data D_u and an unlearning algorithm U is executed to fulfill the request. An unlearning algorithm is applied to the global model w_t , which has been trained including client u in the pool of clients, generating an unlearned global model w_t^u , i.e., $w_t^u = U(w_t)$. $U(\cdot)$ can be defined as a function that ensures that exhibits performances indistinguishable or approximately indistinguishable from a global model w_t trained without the contribution of D_u . Figure 6.1 offers an illustrative overview of FU.

FU can have different objectives (sample unlearning, class unlearning, and client unlearning), as we detail in Subsection 6.3.1. As we describe in Subsection 6.3.2, the architectural design of FL introduces a more challenging setting than regular MU due to the decentralization and inscrutability of data and the iterative and stochastic nature of the FL process. In Subsection 6.4.1, we present the requirements that FU algorithms should meet. Finally, Subsection 6.4.2 outlines and describes the metrics used in the related literature to assess whether an FU mechanism fulfills the previously introduced requirements.

6.3.1 Objectives

As depicted in Figure 6.2, FU aims to achieve one or more of the following objectives [151]:

Sample Unlearning. Sample unlearning aims to remove the contribution of specific data samples from the trained model. In this case, the client that requests the unlearning would like to erase the contribution of a subset of its data, i.e., D_u .

Class Unlearning. Class unlearning aims to remove the contribution of all the data samples that belong to a certain class across clients, i.e., to remove the contribution of S_k with $S_k = \{x_i; c_i\}_{i=1}^{n_k^c}$, with n_k^c the number of local samples at client k labeled with class k . Ideally, if provided with samples from a removed class, the unlearned model should produce an outcome randomly between the remaining classes.

Client Unlearning. Client unlearning is peculiar to federated settings, and it relates to the to be forgotten clients. In this case, the client that requests the unlearning would like to erase the

¹If including or excluding data had no impact on outcomes, the data would be essentially useless.

Figure 6.2: Visualization of FU objectives. We consider four clients in the federation, i.e., clients A, B, C, and D, and four total classes for the data held by clients, i.e., L1, L2, L3, and L4. The figure reports the label that corresponds to a specific sample in the local dataset of clients (e.g., data sample 1 in client A's dataset belongs to class L1).

contribution of its entire local dataset, $S_u = D_u$. Client Unlearning can be seen as applying Sample Unlearning to all the samples a specific client holds during local training. Similarly, Client Unlearning can also be a special case of class unlearning when a single client holds all the samples for class c across the federation, i.e., $S_u = \{x_i; c\}_i^{n_u}$ and $S_u = D_u$.

6.3.2 Challenges of Adapting MU in FL Settings

Concerning centralized MU, FL settings introduce even more challenging constraints. The iterative nature of collaborative learning, the stochastic selection of per-round participating clients, and the decentralization and inscrutability of local data represent additional degrees of complexity that have to be accounted for in FU algorithms.

Non-deterministic Training Process. Most MU mechanisms suppose that retained data are available when unlearning is performed [137]. Contrarily, the data available in an FL round depends on the client selection, which is subject to various dynamics. For example, clients eligible for a certain round may not be available in the future due to connection or battery issues or unwillingness to participate. This peculiarity of FL makes the possibility of exactly reproducing the

rounds after t_u unrealistic. At the same time, discarding all the knowledge acquired between the aggregation at round t_u and the aggregation at round t_i (i.e., the most recent round), that possibly may be distant in time with $t_i \ll t_u$, would waste significant learning efforts of the federation. In this sense, methods that can embed that knowledge are most appealing.

Inscrutable Model Updates and Finite Memory. In principle, the FL aggregator or server should not or could not be able to inspect the model updates of single clients (e.g., because the updates could be aggregated via Multi-Party Computation [152, 153]). This would make unfeasible the naive solution of rolling back to the global aggregation of round t_u and removing the individual contribution of client i because it requires access to single updates of clients. Moreover, even if it could be possible to have direct access to the past model updates, the central FL aggregator would be required to store the entire history of aggregated models and collected updates of each round, translating into unfeasible storage capacity.

Inscrutable Local Datasets. In centralized ML, all the data used for training are supposed to be directly accessible and potentially be used for MU. On the contrary, datasets remain private by design in FL contexts. This prevents the application of solutions from plain MU that require direct access to the data to forget and/or retain. For example, the recent work in [154], targeting centralized unlearning, selectively induces noisy knowledge on the data to forget while reinforcing good knowledge to retain sets respectively employing an incompetent or competent teacher in a Knowledge-Distillation framework. This kind of mechanism cannot be directly adapted to FL due to the inscrutability of local datasets of clients, which cannot be exported or accessed by other parties.

Iterative Learning Process. An additional factor that differentiates FU from MU is the iterative nature of FL. FL algorithms proceed in cyclical rounds, and the global model aggregated at round t is the starting state for round $t + 1$. Suppose that at round t_i a client i , which has been included in a round t_u ($t_u < t_i$), requires its contribution deletion (e.g., client unlearning). The naive solution to client unlearning would be to recover the global model at round t_u to exclude client i 's updates from the aggregation of round t_i . However, the iterative nature of FL algorithms implies that all the global aggregations after t_u should be invalidated.

Data Heterogeneity. In centralized ML, the data is supposed to be IID since data are centralized in the same location and can be shuffled and arbitrarily partitioned among machines in case of distributed training. On the contrary, one of the defining characteristics of the FL setting is rep-

6 Federated Unlearning: Taxonomy, Approaches, and Solutions

resented by data heterogeneity among clients, i.e., clients hold data that are likely to be non-IID. While this may hamper the convergence of the global model [155, 156], this trait of FL settings could be convenient for an FU algorithm since it may be easier to erase the learned representations peculiar to a skewed subset of data. However, since data are not directly accessible, it may not be straightforward to assess the level of data heterogeneity in advance.

6.3.3 Entity Performing Unlearning

The unlearning phase can be performed by one or more parties in the federation, with relative benefits and drawbacks [157]:

- ^ **Server.** The server has more computation power and storage capacity than clients. It could store historical global models and keep track of individual clients' historical updates. In this way, the server could remove the contribution of a specific client when requested. However, to perform such a removal, the server should be able to associate one or more specific historical updates with the identity of a specific client, which poses additional privacy concerns.
- ^ **Target Client.** The target client, i.e., client u , has direct access to the data to be forgotten. It can locally perform and assess the unlearning success. However, this poses further security concerns when considering possibly malicious clients, who may request a local unlearning to inject a backdoor in the model or to hamper the training on purpose, taking advantage of their role in the unlearning phase. Furthermore, regular MU methods often require both retaining and forgetting data, e.g., to alternate forgetting and retaining phases so that the model can selectively unlearn. The technical issue in only-local FU is to retain good knowledge without having access to other clients' data.
- ^ **Remaining Clients.** The remaining clients, i.e., the federation without client u , should retain data and are typically used to recover the performance of an unlearned model.

Given the above benefits and drawbacks, mixed solutions may also be possible, e.g., target client u and the server may cooperate during the unlearning phase.

6.4 Design and Implementation Guidelines for Efficient FU Algorithms

Here, we present the primary requirements that an FU algorithm should meet to be efficiently adopted and integrated into practical scenarios. In addition, we thoroughly examine the metrics that have been proposed to assess the extent to which data has been effectively unlearned from the global model.

6.4.1 Requirements of Federated Unlearning Algorithms

1. **Enhanced Efficiency Compared to Retraining.** An effective unlearning algorithm must be more efficient than the naive approach of retraining from scratch, which may be infeasible. Ideally, it should leverage and retain useful knowledge from the original global model instead of discarding all progress through a full rollback.
2. **Performance Retention and Recovery.** The unlearned model should ideally retain or quickly recover the performance of the original model, either immediately after unlearning or after a few additional FL rounds:
 - ^ **Comparable Test Performance to Retrained Models.** The unlearned model's performance (e.g., accuracy, loss) on test data should be comparable to that of a model trained solely on sanitized data. As the unlearned model progresses through additional rounds, it should approach or match the original model's performance.
 - ^ **Performance Relative to the Original Model.** The unlearned model's test performance may differ from that of the original model, particularly when data from the removed client, u , had a significant impact. For example, in cases where u 's data were unique or dominant for a specific class, the unlearned model may struggle with generalization on that class. Therefore, directly comparing performance between the unlearned and original models may be complex.
3. **Effective Forgetting.**
 - ^ **Comparable Performance on Forgetting Data to Retrained Models.** Ideally, the unlearned model should perform similarly on forgetting data as a model retrained on the remaining dataset only. After unlearning, the model should no longer benefit from client u 's data.
 - ^ **Significantly Different Performance on Forgetting Data from the Original Model.** Successful unlearning should cause the unlearned and original models to yield no-

6 Federated Unlearning: Taxonomy, Approaches, and Solutions

Table 6.1: Summary of the main metrics in FU literature.

Category	ID	Metric	Range	Exp. Value	Sample	Class	Client
Improved Efficiency	IE.1	Speed-up ratio	(0; +1)	" ; w^f	X	X	X
	IE.2	Communication cost reduction ratio	(0; +1)	" ; w^f	X	X	X
Recovered Performance	RP.1	SAPE	[0, 1]	#; w^f	X	-	X
	RP.2	Performance on test dataset		; w^f ; w^i	X	-	X
		Accuracy	[0, 1]				
	RP.3	Loss	[0; +1)				
		Performance on retained training dataset		; w^f ; w^i	X	-	X
	RP.4	Accuracy	[0, 1]				
		Loss	[0; +1)				
	RP.5	Performance on retained or unlearned classes (test dataset)		; w^f ; w^i	-	X	-
		Accuracy	[0, 1]				
	RP.5	Loss	[0; +1)				
ECE [158]		[0, 1]	; w^f ; w^i	-	-	X	
Forgetting Verification	FV.1	Performance on clients data		(; w^f); (# = " ; w^i)	X	-	X
		Accuracy	[0, 1]				
		Loss	[0; +1)				
	FV.2	Accuracy on removed class(es) (test dataset)	[0, 1]	#	-	X	-
	FV.3	KL divergence (model's output vs. uniform distrib.)	[0; +1)	#	X	-	X
	FV.4	KL divergence (per-class accuracy distributions)	[0; +1)	#	-	X	-
	FV.5	KL divergence between retrained and unlearned weights	[0; +1)	0	-	-	X
	FV.6	ℓ_2 -norm between retrained and unlearned weights	[0; +1)	#	X	-	X
	FV.7	FR (Liu et al. [159])	[0, 1]	"	X	-	X
	FV.8	IF [160] on forgetting data	(1 ; +1)	0	-	-	X
	FV.9	RE [161]	[0; +1)	#	X	-	-
	FV.10	AD [161]	[0; +1)	(#; w^f), (" ; w^i)	X	-	X
	FV.11	BASR ([37, 11, 36])	[0, 1]	(; w^f), (#; w^i)	X	-	X
	FV.12	MIA (Shokri et al. [44])	[0, 1]	(; w^f), (#; w^i)	X	-	X
	FV.13	MIA (Golatkar et al. [162])	[0, 1]	(; w^f), (#; w^i)	X	-	X
FV.14	MIA (Yeom et al. [163])	[0, 1]	(; w^f), (#; w^i)	X	-	X	
FV.15	MIB (Hu et al. [164])	[0, 1]	(; w^f), (#; w^i)	X	-	X	

ticeably different results on forgetting data. For example, a reduced success rate for backdoor attacks in the unlearned model indicates effective unlearning. This difference between the original and unlearned models on forgetting data should be maximized according to relevant performance metrics.

4. Privacy Preservation Consistent with Standard FL Any unlearning algorithm should uphold the privacy-centric design of FL, ensuring that data remains distributed and controlled by individual data owners throughout the unlearning process.

6.4.2 Metrics to Assess Unlearning

The requirements described in Subsection 6.4.1 call for metrics that can assess the efficacy of unlearning algorithms. In this subsection, we gather and introduce the metrics used in literature to evaluate FU methods, and we classify them according to the scope of their assessment. Table 6.1 summarizes the main characteristics of the considered metrics. All the reported metrics are supposed to be applied to the unlearned model. The first column indicates the scope of the

6.4 Design and Implementation Guidelines for Efficient FU Algorithms

metric. The ID column associates an identifier to the metrics that will be useful in the next sections. Range column reports the limit values of the metric. Expected Value column indicates if higher or smaller values are better. Expected Value column contains tuples (s,) in the form of (f" ; #; g ; w), it indicates if higher values, smaller values or values similar to a baseline w (e.g., the retrained model's performances) are better. w^r represents the global model before unlearning and w^r represents the retrained model. Accuracy and success rate are expressed as probability and can be equivalently expressed in percentage (i.e., in a range [0%, 100%]). The three rightmost columns report the unlearning objectives for which the metric is most appropriate.

Improved Efficiency Metrics and Indicators. This class of metrics relates to the Requirement (1) of the previous section and usually compares, as a primary dimension, the time needed by the proposed unlearning algorithm to recover performance similar to a baseline, e.g., the original model or the retrained model. For example, the ratio $\frac{T_{w^r}}{T_{w^u}}$, with T_w being the time needed by a model trained with a specific baseline w^r and with T_{w^u} being the time needed by the unlearned model to attain a certain test performance, can be used as an indicator of speed-up. However, recovery time is not the only measure used in the FU literature. In [165], Halimi et al. consider the required cumulative communication cost in client-to-server model updates to achieve similar performance (e.g., in test accuracy) to assess the efficiency of the unlearned method compared to the retrained model, with the lower, the better. This comparison could also be expressed as a ratio $\frac{\text{Bytes}_w}{\text{Bytes}_{w^u}}$ (higher is better), with Bytes representing the size (e.g., in MB) of the cumulative client-to-server updates. In [166], Su et al. also focus on minimizing the number of clients participating in the recovery process.

Retained or Recovered Performance Metrics This class of metrics relates to the Requirement (2) of the previous section. The straightforward metrics that can be employed to assess whether the unlearned model achieves performances similar to a baseline model are the test performances (e.g., loss, accuracy). Another potential metric is the Expected Calibration Error (ECE) [158]. It is used to evaluate the calibration of probabilistic models, focusing on the alignment between predicted probabilities and actual observed frequencies. It involves dividing predictions into probability bins, comparing average predicted probabilities with observed accuracy in each bin, and calculating the average absolute differences. ECE measures how well a model's predicted probabilities reflect the true likelihood of events, indicating the model's reliability and confidence calibration. In [167], Liu et al. propose to use a compact indicator for comparing the test perfor-

manances of the unlearned model to a reference model. Such a metric is the Symmetric Absolute Percentage Error (SAPE), computed as

$$\frac{|\text{Acc}(w^u; D_{\text{test}}) - \text{Acc}(w; D_{\text{test}})|}{|\text{Acc}(w^u; D_{\text{test}})| + |\text{Acc}(w; D_{\text{test}})|}, \quad (6.3)$$

with the baseline model indicated with (e.g., the retrained model or the original model) and $\text{Acc}(w; D_{\text{test}})$ being the accuracy on test data. SAPE spans in a range and the lower the value, the closer the unlearned model is to match the baseline model's test performance. To assess Class Unlearning, the test performance should be measured and compared only on the subset of test data belonging to the retained classes [168].

Forgetting Verification Metrics and Indicators. This class of metrics relates to the Requirement (3) of the previous section. Comparing the unlearned model's performance (e.g., loss and accuracy) to a baseline model (e.g., the retrained model) on ~~client~~ data can represent a preliminary indicator to assess selective forgetting. For example, the retrained and unlearned models should perform similarly on ~~client~~ train data. However, from a real-world, practical perspective, clients can only evaluate this locally since their data are private.

Another metric used more often to evaluate FU algorithms is the success rate of backdoor attacks (BASR) [37, 11, 36]. In a nutshell, the attacker client crafts malicious inputs to instill a backdoor into the model, meaning that when the model encounters a similar input, it will misclassify it. For example, to inject a backdoor pattern in a visual classification task, the malicious client modifies some pixels in the original images and associates wrong labels to those inputs. The attack's success rate is the probability that the model predicts induced wrong class for any malicious inputs. From the metric perspective, backdoor attacks should induce the global model to mispredict specific inputs that embed the backdoor pattern. In contrast, the global model's performance should remain unaffected by regular inputs. The unlearning algorithm removes the contributions of harmful backdoored training data in the global model. Hence, a successfully unlearned global model should reduce the success rate of backdoor attacks when triggered by backdoored samples but should perform well on the test dataset (i.e., similarly to the retrained or original model).

Inspired by [37], Hu et al. [164] proposed a novel membership inference technique, namely Membership Inference Backdoor (MIB). In MIB, a data owner marks their samples before releasing them. If the marked samples have been used by an unauthorized ML model, a backdoor will be injected into the model. To prove that their data was used for training, data owners can demonstrate the presence of the backdoor in the model. In the context of FU, MIB can be leveraged to evaluate unlearning methods by introducing backdoor triggers to the erased data samples and val-

6.4 Design and Implementation Guidelines for Efficient FU Algorithms

identifying the unlearned model's susceptibility to backdoor attacks. As backdoored samples are hard to remove, the efficacy of unlearning these data will be even higher when removing regular data.

Another alternative attack for assessing the effectiveness of unlearning methods is the MIA [44, 163, 162]. MIA tries to guess whether a specific sample was included in the training set. Then, the success of MIA can be measured via standard accuracy, precision, and recall metrics, with the unlearned and retrained models exhibiting similar sensitivity to MIA. The rationale when employing MIA in assessing the quality of unlearning is that a model has forgotten only if an attacker cannot guess the membership of the forgetting data with reliable confidence. The pioneering MIA proposed by Shokri et al. [44] firstly trains the so-called attack model that, once trained, will take as input the predictions of a model on client train data and will output a membership probability, building on the assumption that if a model is very confident in his predictions, then the input example is likely to be used for training. It is worth noting that membership attacks like the one proposed by Shokri et al. [44] require that the attacker knows the type and architecture of the ML model and has access to some data from the same underlying distribution. In [162], Golatkar et al. propose a black-box MIA that trains the attacker models on the entropy of the model's output probabilities. During the training phase, the attacker model learns to predict membership by seeing samples from the training data as members and samples from the test set as non-members. In [163], Yoem et al. propose an MIA that requires knowing the average training loss of the model as well as having access to its output logits. Then, a data sample is inferred as a member if the training loss of the model on the input data point is smaller than the known average training loss.

To assess client unlearning within a visual classification task, Halimi et al. [165] introduce a subset of clients that hold flipped images in the federation. A successful unlearning algorithm should reduce the accuracy on the flipped samples while retaining good accuracy on regular samples

In [159], Liu et al. introduce the Forgetting Rate (FR), a measure to evaluate the performance of unlearning. To calculate FR, it is necessary to use a membership oracle that can discern whether a given sample is categorized as a member or not. Given a trained ML model and an arbitrary sample x , an ideal membership oracle outputs true if x belongs to the training set; otherwise, outputs false. FR is an indicator to measure how many samples are changed from the memorized set to the unknown set after unlearning, and it is calculated as the transformation rate between unknown states, i.e.:

$$FR = \frac{BT}{BT + BF} - \frac{AT}{AT + AF}; \quad (6.4)$$

²When the flip-based evaluation method presented in [165] is used, no data augmentation is applied during training.

denoting as BT (the number of eliminated training samples that are predicted to be true by the oracle before unlearning), BF (the number of eliminated training samples that are predicted to be false by the oracle before unlearning), AF (the number of eliminated training samples that are predicted to be false by the oracle after unlearning), AT (the number of eliminated training samples that are predicted to be true by the oracle after unlearning).

Reconstruction Error (RE) and Activation Distance (AD) are two metrics utilized in [161]. RE metric evaluates the model's ability to reconstruct data subjected to unlearning, with a lower score indicating enhanced performance. Simultaneously, AD metric evaluates the indistinguishability between model outputs by measuring the average distance between the model's predictions before and after unlearning, employing the distance metric on forgetting data, the higher the better. A similar metric is used in [169] and referred to as prediction difference

In [146], Gao et al. propose to employ metrics on specific subsets of forgetting data, called markers, on which calculating more reliable indicators about unlearning performance. Most notably, Gao et al. consider markers for forgettable samples, samples with the highest loss variance across several rounds, and loss outliers, the high loss samples. To assess unlearning, Gao et al. mainly track four metrics: loss, accuracy, and Influence Function (IF) [160], which estimates the impact of a training sample on model prediction and a metric based on KL divergence. The latter is computed as follows:

$$KL(f_w(S_u); p) \tag{6.5}$$

The formulation above measures the distributional distance between a model's output on forgetting data $f_w(S_u)$ and a uniformly distributed output probability with $p = (\frac{1}{L}; \dots; \frac{1}{L})$, L the number of total classes, and the KL divergence. If the KL divergence is low or even 0, the model cannot produce any meaningful predictions on forgetting data, i.e., the model has forgotten such data. The KL divergence can also be used to measure the distance between weight distributions, i.e., to assess the disparity between the weight distribution obtained through the unlearning process and that obtained through retraining [170], with lower values expected. While the metrics detailed above are mainly employed to assess the forgetting in client or sample unlearning methods, other metrics tailored to evaluate class unlearning exist. For example, suppose a test set is divided into two subsets, i.e., the set of samples labeled with the unlearned class and the set of all the other samples. The unlearned model's accuracy on the former set should ideally be 0, whereas the accuracy on the latter set should be similar to the retrained model [171, 168]. However, it should be ensured that the unlearned model does not just misclassify the removed class with another. A possible metric to assess that the unlearned model spans the removed class's

6.5 Literature Review of Federated Unlearning Methods

Table 6.2: Summary of FU referenced works.

Objective	Technique	Ref.	Where	Metrics	Proxy Data	Open Source
Client	Re-calibration of Historical Updates	[169]	Remaining Clients & Server	IE.1, RP.2, FV.1, FV.12	-	X
		[172]	Remaining Clients & Server	RP.2, FV.11	-	-
		[173]	Remaining Clients & Server	IE.2, RP.2, FV.11	-	-
		[146]	Target Client & Server	IE.1, IE.2, FV.1, FV.3, FV.8	-	-
		[174]	Server	IE.1, IE.2, RP.2, FV.1	Yes	-
	Knowledge Distillation	[175]	Server	FV.1, FV.11	Yes	-
		[13]	Target Client & Neighborhood	RP.2	Yes	-
		[171]	All Clients & Server	IE.1, RP.3, FV.1, FV.11	-	-
	Gradient Modification	[165]	All Clients & Server	IE.1, IE.2, RP.3, FV.11, FV.12, FV.14	-	-
		[176]	All Clients & Server	RP.2, FV.11	-	-
		[177]	Target Client	RP.2, FV.11	-	X
	Clustering	[166]	Clients in the Cluster & Server	IE.1, RP.2	-	X
	Bayesian FL	[170]	Target Client	FV.5	-	-
		[178]	Target Client	RP.3, FV.1	-	-
		[179]	Target Client	RP.2, RP.3, RP.5, FV.1	-	-
		[180]	Target Client	IE.1, RP.2, RP.3, FV.1, FV.5, FV.6, FV.15	-	X
		Differential Privacy	[181]	Server	IE.1, RP.2, FV.12	-
	Other Approaches	[182]	Server	IE.1, RP.2, RP.3	-	-
		[183]	Target Client & Server	IE.1, RP.2	-	X
		[184]	All Clients & Server	IE.1, RP.2, FV.12	-	X
Class	Pruning	[168]	Target Client	IE.1, RP.3, FV.1, FV.2, FV.4, FV.12	-	-
	Knowledge Distillation	[171]	All Clients & Server	IE.1, RP.3, FV.1, FV.11	-	-
Sample	Gradient Modification	[167]	All Clients	IE.1, RP.1, RP.2	-	X
		[185]	Target Clients & Remaining Clients	IE.1, IE.2, RP.3, RP.4, FV.1, FV.13	-	X
		[186]	Target Client & Server	IE.1	Yes	-
		[187]	Target Client	RP.2, RP.3, FV.1, FV.13	Yes	-
		[159]	Target Client	RP.2, RP.3, FV.7	-	-
		[188]	Target Client	RP.2, FV.12	-	-
	Quantization	[189]	Target Client & Server	IE.1, RP.1, FV.12	-	X
		[190]	All Clients & Server	IE.1, RP.2, RP.3, FV.1	-	-
	Reinforcement Learning	[161]	Target Client	RP.2, FV.9, FV.10	-	-
	Other Approaches	[191]	All Clients & Server	IE.1, IE.2, RP.2, FV.1	-	X
[183]		Target Client & Server	IE.1, RP.2	-	X	
[184]		All Clients & Server	IE.1, RP.2, FV.12	-	X	

prediction across the remaining classes is presented in [168]. Wang et al. [168] measure the KL divergence among the distribution of the unlearned model's per-class accuracy and the distribution of retrained model's per-class accuracy. The closer to 0 the distance, the more similar the distributions are.

6.5 Literature Review of Federated Unlearning Methods

This section offers a comprehensive survey of the existing literature on FU methods. We categorize the referenced works based on the unlearning objectives, techniques, and metrics employed for evaluation. The key features of the reviewed papers are summarized in Table 6.2. Each referenced paper is linked with the primary unlearning objective specified by the author. The column indicates the entities involved in the unlearning process, while the column reports the identifiers of the metrics (refer to Table 6.1) employed for evaluating the effectiveness of the unlearning scheme. Additionally, the Proxy Data column denotes whether supplementary

data, typically required by the server for recovery, are necessary. Finally, we highlight whether the authors have made their solution accessible to the community (or if they plan to do so).

6.5.1 Client Unlearning

Most methods proposed for client unlearning can be seen as fast retraining strategies, which design specific ways to retain as much knowledge as possible from the ongoing training while selectively overwriting the influence of forgetting data.

Re-calibration of Historical Updates. FedEraser [169] is one of the first methods proposed to perform client unlearning, which is achieved through a retraining that speeds up the recovering phase by leveraging historical parameter updates stored on the server side. Suppose that the current round is s_i (when unlearning is requested) and that client u_i participated in round t_i . FedEraser conducts a calibration phase only including the retained clients to iteratively sanitize the updates that have been produced after client u_i participated instead of discarding them. FedEraser performs $s_i - t_i$ calibration rounds, and each round includes local training of calibrated updates and server-side aggregation of such updates to produce the sanitized global model to be broadcast at the next re-calibration round, until all the past updates have been recovered. In [172], Yuan et al. introduce FRU, an unlearning algorithm designed for federated recommendation systems based on FedEraser [169]. To minimize storage space required, FRU suggests a user-item mixed semi-hard negative sampling component to decrease the number of item embedding updates and an importance-based update selection component to retain only crucial model updates.

Cao et al. [173] propose FedRecover, a method designed to recover a poisoned model using historical information. FedRecover utilizes retained updates from clients to estimate the update produced during retrain-from scratch, employing the Cauchy mean theorem and the L-BFGS algorithm [192] to approximate the integrated Hessian matrix efficiently. It implements various strategies for fine-tuning the recovered model to enhance its performance. These strategies include warm-up, periodic correction, abnormality fixing, and final tuning. In the warm-up phase, clients calculate exact updates and send them to the server to create buffers used by the L-BFGS algorithm in constructing Hessian matrices. These buffers are periodically updated during the periodic correction phase, where the server requests clients to send exact updates. These updates are also used to correct the recovered global model. When an estimated model update for a client is substantial, the server asks the client to send the exact update to mitigate the impact of potentially inaccurately estimated large model updates, constituting the abnormality fixing phase. Finally, in the last phase of final tuning, clients send exact updates to fine-tune the recovered global model, a step proven to enhance the model's performance.

6.5 Literature Review of Federated Unlearning Methods

Gao et al. [146] propose a unified framework for unlearning and verification named VeriFi. It is the first work that grants users **Right to Verify (RTV)**, in which participants could actively check the unlearning effect. VeriFi consists of three parts: (1) an unlearning module, (2) a verification module, and (3) an unlearning-verification mechanism that chains (1) and (2) into an integrated pipeline. Although the unlearning module can be any unlearning method, the authors also propose a more efficient one-step unlearning approach **Scale-to-Unlearn (S2U)**. The unlearning process is triggered only when a participant leaves during the later stages of the FL process when the model has already stabilized. S2U scales up the contribution of the clients staying in the federation and scales down the contribution of the leaving client. This adjustment is designed to align the global model more closely with the local models of those who remain while distancing it from the leaving one. The verification process is performed locally by the leaving client. It consists of two steps: **marking and checking**. The marking step involves utilizing marked data, known as markers, to re-tune the local model. Subsequently, the checking step verifies how much the global model has unlearned the markers.

FU serves as a viable tool for eliminating contributions from malicious clients. Guo et al. [174] introduce FAST, a protocol designed to remove the influence of byzantine participants on the global model. The server retains all the clients' updates, using this information to adjust the model parameters and derive the unlearning model directly. Specifically, the unlearning model is obtained by subtracting the contributions of malicious clients in each training round from the original global model. For every iteration, the server compares the accuracy of the current unlearning model with that of the previous one. If the current unlearning model consistently outperforms its predecessor, it indicates that the historical influence of the malicious client persists, prompting the continuation of the step to delete contributions in the next round. Conversely, if the accuracy consistently surpasses the previous unlearning model, the unlearning operation proceeds until the maximum number of attempts to eliminate malicious clients' contributions is reached. However, their method may eventually lead to a loss in the predictive power for other data that are not required to be forgotten. To overcome this concern, the server maintains an additional small benchmark dataset for supplementary training of the unlearning model, thereby enhancing the overall accuracy.

Knowledge Distillation. In [175], Wu et al. propose a mechanism for FU based on Knowledge Distillation (KD), following a recent trend that sees many KD-based mechanisms designed to improve the FL process [193]. The solution in [175] leverages KD to quickly recover the global model's performance after removing the unlearning client's historical model updates. The unlearning algorithm proceeds in two server-side steps: (1) The historical updates of the unlearned client are removed from the current global model to produce a sanitized model; (2) The previous

global model works as the teacher, and the sanitized model works as the student. The sanitized global model mimics the outputs of the previous global model on proxy unlabeled data to recover the drop in performance after step (1). In [175], clients do not directly have to participate in the unlearning phase. However, as a drawback, the solution proposed by Wu et al. requires (unlabeled) data on the server side, which may not be easily available for specific tasks. Furthermore, the server should maintain a full history of the updates collected from clients, and it should be able to identify clients' updates to then respond to unlearning requests.

Zaho et al. [171] present a knowledge erasure strategy called Momentum Degradation (MoDe). The proposed method decouples the FU process from training, making it applicable to any model architecture that has completed FL training without altering the training procedure. Unlearning consists of two phases: knowledge erasure and memory guidance. In the first phase, the pre-trained model parameters are adjusted to reduce discriminability for target data, aligning them with retrained model parameters. The server initializes a degradation model, sends it to clients for FL training, and uses its weights as the guiding direction for updating the pre-trained model. Following momentum degradation, the pre-trained model experiences reduced discriminability for all data points. The memory guidance phase restores performance by sending the pre-trained model to all clients and the degradation model to the target client. The target client uses the degradation model's output on its local dataset as pseudo labels, guiding the local training of the pre-trained model. This ensures the unlearned model's output closely resembles the retrained model's output. All clients contribute to aggregating the pre-trained model. The knowledge erasure and memory guidance steps are iterated multiple times.

Ye et al. [13] present a decentralized unlearning framework called HDUS, employing distilled (smaller) models to build erasable ensembles for each client. HDUS operates in a fully decentralized learning scenario, where clients exchange information without needing a central server. In addition to its local main dataset and model, each client owns a distilled model and reference dataset shared among all clients. The client aims to align its distilled model as closely as possible with the main model, optimizing the former using a specific loss function and the reference dataset. Distilled models are exchanged with neighboring clients. During the inference phase, the client generates an ensemble model output by weighting the output of its main model and the outputs of all neighbors' distilled models. If a client requests unlearning of its data, its neighbors simply exclude that client's distilled model from the ensemble.

Gradient Modification. In [165], Halimi et al. introduce a method to perform unlearning directly on the target client. The method employs projected gradient ascent to maximize the loss from a reference model, defined as the average of the models of the remaining clients. A sphere with radius around the reference model constrains the unbounded loss. To improve the

6.5 Literature Review of Federated Unlearning Methods

performance of the unlearned model, the remaining clients run additional rounds of FL, achieving good performance in a few rounds. Similarly, Li et al. [176] propose a subspace-based FU method (SFU) that leverages gradient ascent, constrained by the orthogonal space of input gradient spaces formed by other clients, to eliminate the target client's contribution. SFU does not require storing intermediate updates on the clients or the server. The process includes three steps: (1) each client, except the target one, creates a representation matrix using a part of its local data and sends it to the server. A representation matrix for each layer in a neural network is formed by combining outputs from the forward pass of random samples through the network, with each column representing the output for a different sample. The privacy of the representation matrix is protected through a DP algorithm. Random factors are added to each client's layer representation to prevent individual client data information leakage. This addition does not affect the subspace search process due to the orthogonal nature of the matrix. (2) The target client executes some round of gradient ascent and sends the updated gradient to the server. (3) The server then performs Singular Value Decomposition (SVD) [194] on the set of representation matrices to get the set of input gradient subspace. It gets the orthogonal projection of the target gradient for this subspace and removes the contribution of the target client by updating the global model.

In [177] Alam et al. shift the perspective and explore FU to eliminate backdoors for the benefit of adversaries. Once attackers have accomplished their intended objectives or suspect the possibility of detection, they may wish to eliminate the backdoors they previously injected and erase any trace of their presence. The proposed method leverages gradient ascent on the target client to forget its contribution. However, to face challenges like catastrophic forgetting, the generation of deviating models during the unlearning process, and the varying importance of individual parameters, the authors propose two strategies: memory preservation and dynamic penalization. The first one implies utilizing the benign dataset of the target client to prevent catastrophic forgetting. The second strategy implements a dynamic penalty mechanism to penalize diversion from the global model. Moreover, weights are also incorporated into the penalty term, assigning a unique weight to each parameter based on its significance. Introducing non-uniform penalties for all parameters may yield superior results. The loss function is modified following the strategies depicted before.

Clustering. KNOT [166] performs client clustering to speed-up retraining in asynchronous FL. Indeed, when unlearning is requested, retraining is conducted exclusively within a client cluster while the remaining clients are unaffected.

Bayesian FL Bayesian FL combines Bayesian methods with FL, introducing a distribution of models to capture uncertainty instead of a fixed model. Gong et al. [170] introduce the first FU method in a decentralized network within a Bayesian framework. It is based on exponential

family parametrization and leverages local gossip-driven communication. When the unlearning client is scheduled, it removes its local variational parameter from the current global one, and the result is forwarded to the next client. Gong et al. [178] also propose a non-parametric federated Bayesian unlearning method, referred to as Forget-Stein Variational Gradient Descent (Forget-SVGD). Forget-SVGD is an extension of SVGD [195], a particle-based approximate Bayesian inference method utilizing gradient-based deterministic updates, and its federated variant called Distributed SVGD (DSVGD) [196]. After the conclusion of the FL process, one or more clients may request to forget their data. Unlearning a dataset from the variational posterior is formulated as the minimization of the unlearning free energy. This involves identifying a distribution that closely aligns with the current variational posterior while maximizing the average training loss for the datasets designated for forgetting. An optimized version for rate-constrained channels is presented in [179]. It applies quantization and sparsification across multiple particles.

BFU [180] is another Bayesian FU method that incorporates parameter self-sharing. The proposed scheme introduces an unlearning rate that balances the trade-off between forgetting the erased data and remembering the original global model. Data erasure and maintaining learning accuracy are two different objectives to limit potential performance degradation.

Differential Privacy. In Sec. 6.2.2, we presented DP as an orthogonal tool for unlearning methods. In [181], Zhang et al. propose FedRecovery, an algorithm that embeds DP in their unlearning mechanism. Indeed, when a client requests unlearning, firstly, the server removes its influence by eliminating all its gradient residuals, which are generated from clients' historical submissions. Then, the server injects Gaussian noise into the resulting model to make such an unlearned model statistically indistinguishable from the retrained one. The noise is calibrated according to the upper bound of the distance between the two models, estimated through the smoothness of clients' loss functions. By leveraging DP, an observer cannot assess whether a model has been trained by $K - 1$ clients.

Other Approaches. Tao et al. [184] propose a framework for exact FU that leverages the concept of total variation (TV) stability, which measures the discrepancy between two distributions. A learning algorithm is TV-stable if the model produced using the entire dataset and the model produced using a subset differ by at most ϵ . The authors introduce FATS, a novel algorithm for fast retraining based on FedAvg and proved to be TV-stable.

RevFRF [182] is a framework designed for federated random forests (RFs) that supports secure participant revocation. The authors delve into the revocation process from two distinctive perspectives: (1) ensuring the removal of targeted data from the RF and (2) preempting the potential collusion of removed clients with the server to continue utilizing the outdated RF illicitly. To achieve the first level of revocation, the RevFRF traverses the Random Decision Trees (RDTs)

6.5 Literature Review of Federated Unlearning Methods

within the trained RF and prunes all splits contributed by the participant, seeking to be forgotten. Subsequently, the RF undergoes a reconstruction, wherein only the remaining RDTs are considered. Due to their isolation from each other, the rebuilding of one RDT does not affect the others. However, it is crucial to note that this approach is not easily extendable to other ML models, and the worst-case scenario coincides with the reconstruction of the model from scratch. The second level of revocation involves executing additional computations to refresh the revoked splits with random values, rendering them inaccessible to any potentially colluding server.

In [183], Pan et al. introduce a novel FU algorithm to perform federated K-means++. In a nutshell, clients maintain centroid vectors computed on their local data. Those vectors are then transmitted to the server, which uses them to obtain a centroid set. If a client wishes its data to be forgotten, it resets its vector of centroids to zero, prompting the server to re-execute clustering on the remaining clients' vectors.

6.5.2 Class Unlearning

In CNN models, visualizing feature maps activated by different channels reveals their varying contributions to distinct image categories in image classification. Wang et al. [168] leverage this feature to introduce FU by selectively pruning channels with high discrimination on the target category to forget. Class discrimination is determined using the term frequency-inverse document frequency (TF-IDF), a statistical metric originally designed to determine the relevance of a word to a document within a collection of documents. In this context, each channel's output can be seen as a word, the feature maps representing different categories as documents, and TF-IDF evaluates the relevance between channels and categories. Once the discriminative channels have been pruned, a fine-tuning operation is applied to restore the model's performance.

The MoDe method proposed by Zaho et al. [171] is also presented as a solution for category removal without modifying the algorithm introduced in the previous subsection.

6.5.3 Sample Unlearning

Since sample unlearning can be viewed as a more specialized instance of client unlearning, these methods inherently adopt strategies inspired by the same principles. We identify similar categories of methods.

Gradient Modification. Liu et al. [167] propose a rapid retraining strategy that leverages a low-cost Hessian approximation method and applies it to the local training of clients after forgetting data deletion. Specifically, at the core of the mechanism, a diagonal empirical Fisher information matrix (FIM) is used, with the update rule of the unlearned model employing first and

second-order moments for Hessian momentum, similar to optimizer like Adam [197]. However, this method may significantly reduce performance when dealing with complex models.

Dhasade et al. [185] introduce QuickDrop, an FU method that leverages dataset distillation (DD) to perform unlearning and recovery. DD condenses a large training dataset into a compact and smaller synthetic dataset [198]. However, generating good-quality synthetic data requires many local update steps. To reduce the computational overhead, the clients reuse the gradient updates computed during FL training. Clients perform Stochastic Gradient Ascent (SGA) on their local distilled dataset when an unlearning request comes. While in the recovery phase, the network executes recovery rounds, during which they also use the distilled data not extracted from the samples to forget. It is worth noting that distilled data slightly decreases the performance. To avoid decreasing performance in the recovery phase, the authors include a few original samples in the distilled datasets that nullify potential performance drops.

FedFilter [186] is an edge caching scheme that uses FU to remove invalid and outdated data from the global model. The unlearning is initiated by the server that selects the content to be forgotten and generates a reverse gradient. Moreover, to maximize the unlearning while minimizing its impact on the model accuracy, the parameters are iterated through SGD. The reverse gradient is applied to the local model to erase the selected content locally. Finally, for the basic layer parameters, FU is achieved by aggregating and broadcasting model parameters.

Jin et al. [187] present the forgettable federated linear learning (2F2L) framework. 2F2L introduces a federated linear learning strategy that leverages a linear approximation of a deep neural network (DNN) given by the first-order Taylor expansion. The first initial value for the weights to perform linear approximation is built using the server's dataset. With a reasonable amount of data on the server, this value is believed to be close to the optimal model weight based on the entire dataset. Having a linear approximation, it is possible to perform data removal performing gradient ascent using the quadratic forgetting loss function [162]. However, to achieve removal, it is necessary to compute Hessian matrix inversion, which is a complex task. For this reason, 2F2L leverages a numerical approximation based on the server's data.

Forsaken [159] is a memorization elimination algorithm that employs dummy gradients to stimulate the neurons of the ML model and erase the knowledge of specific data points. Client possesses a trainable generator that produces gradients. During each unlearning epoch, the generator is optimized based on an objective function aiming to reduce the distance between the current confidence vector and the confidence vector perfectly forgotten sample rules. Then, the client produces a new dummy gradient and sends it to the server.

In [188], Xia et al. propose FedME², a FU framework for digital twins for mobile networks (DTMN). FedME² comprises two modules: MEval and MEraser. The first one aims to build

6.5 Literature Review of Federated Unlearning Methods

a memory evaluation model to detect whether the data to be forgotten is remembered and the degree of memory. This information is used by the MEraser module to guide the privacy erasure process. On the other hand, the MEraser module optimizes the local model leveraging the model loss, the MEval loss, and a penalty term, which constrains similarity between the global model before and after data erasure. The server then re-aggregates the local model from the client and completes the construction of the global model without privacy information.

Quantization. Xiong et al. [189] introduce Exact-Fun, a quantized FU algorithm designed to eliminate the influence of a subset of data from a target client within a global model. When a client requests the removal of a specific portion of their data, the algorithm calculates a local model using the client's trained model up to that iteration. If the resulting quantized model matches the previously global model, the deletion of that data subset has no impact on the overall model. However, if quantization leads to a mismatch, retraining becomes necessary to effectively eliminate the influence of the unlearned data.

In [190], Che et al. extend their prior work on prompt certified machine unlearning (PCMU) [199] to operate effectively in a federated setting. PCMU employs randomized gradient smoothing and quantization to execute training and unlearning operations, enhancing overall efficiency concurrently. Random smoothing on gradients implies that the MU model, directly trained on the entire dataset, shares identical gradients (and parameters) with the model retrained solely on the remaining data. This occurs within a specific certified radius concerning the gradient change before and after data removals and a certified budget for data removals. To implement PCMU in a federated setting, the authors propose a method that involves creating MU models on clients using the PCMU algorithm. These models are then reformulated as output functions of a Nemyskii operator, inducing a Frechet differentiable smooth manifold. This manifold possesses a global Lipschitz constant that bounds the difference between two local MU models. A global gradient is computed at the server by averaging gradients from all clients, creating a global MU model. The global Lipschitz property ensures that this model closely aligns with each local MU model on the clients within a distance determined by the Lipschitz constant. Consequently, the global MU model can maintain the certified radius and budget of data removals from the local MU models to a certain degree.

Reinforcement Learning. In [161], Shaik et al. present FRAMU, an attention-based MU framework that leverages federated reinforcement learning. FRAMU can unlearn outdated, irrelevant, and private data in single-modality and multi-modality scenarios. It adopts a federated reinforcement learning architecture, employing local agents for real-time data collection and model updates, a central server for aggregation and global model updates using the FedAvg algorithm, and an attention layer to dynamically weigh the relevance of each data point in learning and un-

learning. This layer acts as a specialized approximator, enhancing individual agents' learning capabilities by assigning attention scores to data points, indicating their relevance in local learning, and continually refining the model through interactions with the environment and feedback. If these scores fall below specific predetermined thresholds, the data is considered outdated or irrelevant and is consequently removed from the model.

Other Approaches.The FATS algorithm, presented in [184], can also be applied for sample unlearning by starting the retraining from the iteration preceding the inclusion of that specific sample.

FL can also be employed with alternative data representation such as a knowledge graph (KG), a structured knowledge base of real-world entities and their relations in terms of triplets. KGs have also been adopted to advance representation learning techniques, i.e., KG embeddings combine entities and their relations into a unified semantic space [200]. FL can be integrated with KG embeddings, giving birth to a novel paradigm, namely federated KG embedding learning [201].

In this direction, Zhu et al. [191] propose FedLU, an FL framework for heterogeneous KG embedding. FedLU uses an unlearning method to erase specific knowledge from local embeddings. On the one hand, local knowledge from clients is erased through an interference phase based on hard and soft confusions, which reflect the distance between scores of the triplet in the forgetting set and its negative samples, i.e., the set of triplets whose intersection with the local KG is empty. On the other hand, to limit the decrease in model performance caused by the interference, it is recovered via KD by providing the local client KG without the forgetting set.

The framework introduced in [183] is also applicable when a client wishes to remove some of their data. No action is needed if the data to be removed is not a centroid. However, if it is a centroid, the client needs to eliminate that centroid and select a new one by sampling from its local data using a specific distribution. The updated centroid vector is then sent to the server, which re-executes clustering on the new vector set, producing a new set of server centroids.

6.6 Distilled Lessons Learned

This section summarizes the main lessons learned from the reported in-depth review of SOTA FU research.

1. **Motivation.** FU schemes are proposed as valuable mechanisms to preserve privacy and/or enhance security. From the privacy perspective, FU can guarantee that the forgotten data cannot be forgotten. On the security side, it is a technique for mitigating the impact of malicious clients by eliminating their contributions, i.e., poisoned data and backdoors.

2. **Empirical Evidences** The contribution of a specific client has a tangible influence on the global model, which outputs significantly more accurate predictions when presented with client's samples if client has been included in the pool of clients. Also, the participation of client translates to evident spikes of enhanced accuracy (and reduced loss) on client's train data for that round. Natural forgetting, i.e., just excluding client from the federation in the hope of catastrophic forgetting to remove its contribution rapidly, is not a viable option. Indeed, empirical evidence shows that the impact of client participation fades away very slowly when data is homogeneous, and it remains concretely noticeable for many subsequent rounds. In the particular case of a large-scale federation with a low participation rate of clients and non-IID local datasets, natural forgetting can help the unlearning process.

3. **Requirements and Metrics** As we have detailed from the requirements in Sec. 6.4.1, an unlearning algorithm should be more efficient than a retraining alternative, and the (re-tuned) unlearned global model should achieve performance of generalization comparable to the original model. At the same time, with effective unlearning, the unlearned global model should not respond suspiciously confidently when presented with forgetting data. Then, how to verify those requirements?

While for improved efficiency and recovered performance there is quite a consensus on the metrics to be used (respectively speed-up ratio and test accuracy are most often employed), the works in literature are less aligned. Forgetting verification, MIA and backdoor success rates emerged as the most used assessment metrics, often being arranged differently work by work.

4. **Methods**. The majority of the presented solutions center around completely removing a specific client's contributions, i.e., client unlearning. However, the algorithms designed for a particular FU objective may have broader applicability. Specifically, client unlearning coincides with sample unlearning when a singular client exclusively provides the samples marked for forgetting and encompasses all of their data. Furthermore, client unlearning can be viewed as a specialized instance of class unlearning, applicable when only a single client contributes to the category targeted for erasure. Similarly, these considerations can be extended to sample unlearning, wherein the subset marked for forgetting corresponds to a class to be unlearned.

6.7 FedUP: Efficient Federated Unlearning Through Pruning

The literature review in the previous section highlighted a significant gap in addressing unlearning in malicious contexts. In these scenarios, unlearning becomes more challenging due to the lack of cooperation from clients who need to be unlearned, as they may act maliciously or refuse to participate. For an unlearning algorithm to be effective in a DLT-based FL framework, it must be capable of operating in the presence of malicious clients.

To address this gap, in this section, we present FedUP, an unlearning algorithm specifically designed for adversarial settings. Our solution ensures the secure and efficient removal of data contributions without relying on client cooperation while maintaining the integrity of the global model and ensuring compliance with privacy regulations.

6.7.1 Architecture and Workflow

Our algorithm leverages one-shot soft pruning [202, 203], which allows the model to zero out specific weights while retaining the potential to retrain and recover them if needed. For simplicity, from this point, we will refer to soft pruning as pruning. However, unlike traditional pruning techniques that focus on identifying and removing less relevant connections to make the neural network more efficient, our algorithm reverses this principle entirely. Instead of discarding the less critical connections, we eliminate the most important ones that significantly influence the network's output, contributing more to the model's predictive power. Specifically, FedUP zeroes out the salient weights identified in the local models sent by malicious clients from the global model. This process effectively removes any potentially malicious contributions contained in these connections, thereby neutralizing the influence of the malicious clients. Let us notice that FedUP can be applied to remove multiple malicious clients' contributions simultaneously. The process remains unchanged regardless of the number of malicious clients.

Figure 6.3 presents the unlearning procedure. It is important to outline that the FedUP unlearning algorithm does not affect traditional training when not activated. The unlearning process is executed by the server only when it detects the presence of one or more malicious clients. First, the server creates two separate reference models: one by averaging benign models and the other by averaging malicious ones (1). The step is designed to enhance efficiency, especially when dealing with a large number of clients, by minimizing the need for extensive pairwise comparisons. Then, it computes the difference between the averaged models to identify the conflicting weights between the benign and malicious clients (2). However, weights in neural networks are not equally important. To account for this, weight differences are scaled by their global model

Figure 6.3: Unlearning procedure after detecting malicious clients.

values (3). After that, it selects the highest magnitude weights that differ the most to create an unlearning mask (4). The mask is then applied to the model produced by averaging the benign models (5). This step allows for the exclusion of the malicious clients' models and their contribution in that round as well. However, some pruned weights might include benign information from non-malicious clients. As a result, pruning can generally lead to a reduction in the model's performance. To recover the performance lost, FedUP incorporates additional training rounds after pruning. Finally, the server sends the pruned global model to the remaining clients to perform recovery training rounds to restore the performance (6).

FedUP seamlessly works whether clients send models, like in FedAVG [5], or gradients, like in FedSGD, as the information remains consistent in both cases. The most dissimilar weights are the same ones with the most dissimilar gradients. Moreover, our method does not require saving or receiving any additional information apart from the models or gradients from the clients. This allows for easy integration into any FL framework. The same goal of FedUP can be achieved through catastrophic forgetting [204] but at the cost of numerous additional round overhead that make the catastrophic forgetting process inefficient when the model has already converged [205]. Therefore, the intuition behind our work is to push the model outside its current configuration by pruning specific connections. This accelerates catastrophic forgetting and forces the model to reconfigure itself while discarding the data to be forgotten during the recovery rounds.

Identifying Most Important Connections

Our objective is to identify the most important weights for a specific client using only the latest updates sent by the clients. Many studies on weight importance are designed for centralized contexts [206, 207]. Identifying critical connections for specific clients in FL is challenging because multiple clients can update the same weight simultaneously. The variation in weights between rounds helps to understand the direction in which a client aims to drive the global model [208, 209]. By

6 Federated Unlearning: Taxonomy, Approaches, and Solutions

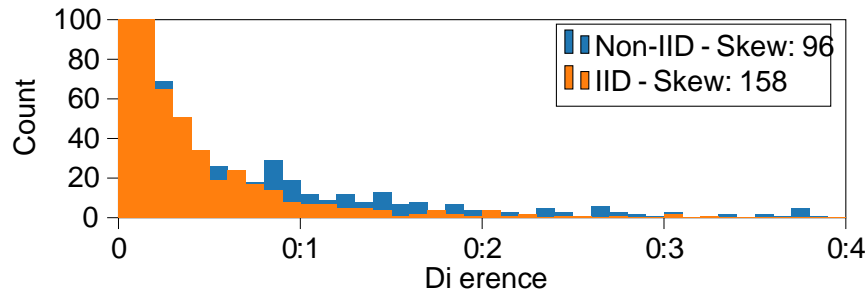


Figure 6.4: Normalized squared difference between averaged benign and malicious weights.

applying this principle to detect which weights are primarily influenced by each group of clients, we compute the squared difference between benign and malicious updates (line 5 of Algorithm 3). Figure 6.4 illustrates the squared differences between the averaged benign and malicious models in a CNN after one training round under both IID and Non-IID settings. In both cases, malicious clients also possess benign data that makes them more stealthy. As shown, most weights experience minimal changes. In the IID setting, where clients have similar data, the weight differences are minor, with the distribution skewed towards zero and only a few weights showing notable divergence. In contrast, in the Non-IID setting, where even benign clients possess different data distributions, the weight differences become more pronounced. Malicious clients are more likely to have weights that diverge significantly from benign clients, making their influence more noticeable. As said, weights in a neural network contribute differently. Those with higher magnitudes typically have a greater influence on the network's behavior [210]. Therefore, the magnitude of the weight being modified must also be considered. For this reason, the difference between the weights is multiplied by the weight value in the global model (line 6 of Algorithm 3). This adjustment prioritizes the most significant weights both for the client to unlearn and for the model itself, making the pruning process more efficient and effective.

Unlearning Procedure

This procedure can be executed as soon as the server detects the presence of one or more malicious clients. This can also happen multiple times during the FL process. First, the server calculates the unlearning mask by selecting each layer's most important weights. The steps are presented in Algorithm 3. The percentage of weights to be pruned could be estimated based on the complexity of the model and the data distribution. More details are discussed in Section 6.7.2. Then, the server uses the unlearning mask to zero out the weights in the model produced by averaging the benign models (line 8 of Algorithm 4). As said, applying the mask on this model already removes the contribution sent by malicious clients in the last training round. Algorithm 4 shows the

Algorithm 3 Generation of the Unlearning Mask

Input : localModels^t; globalModel^{f-1}; P

Output : mask

```

1: maliciousModels = [localModelj | j is malicious ]
2: avgMaliciousModels = Avg(maliciousModels )
3: benignModels = [localModelj | j is not malicious ]
4: avgBenignClients = Avg(benignModels)
5: difference = (avgMaliciousModels - avgBenignModels)2
6: rank = difference / globalModelf-1
7: mask = []
8: for i = 0 to len(rank) do
9:   rankSorted = sortDescending(rank [i])
10:  weights = choose(rankSorted; P)
11:  mask.append(weights)
12: end for
13: return mask

```

Algorithm 4 Apply Mask on Global Model

Input : mask; avgBenignClients

Output : prunedGlobalModel

```

1: prunedGlobalModel = avgBenignClients.copy()
2: layers = prunedGlobalModel.getLayers()
3: for i = 0 to len(layers) do
4:   if layer[i] ∈ [Dense; Convolutional ] then
5:     layer[i][mask[i]] = 0
6:   end if
7: end for return prunedGlobalModel

```

procedure to apply the mask. Another important consideration is regarding the layers on which the mask is applied. Not all layers have the same characteristics, and indiscriminately inhibiting weights in every layer can significantly degrade the network's performance. For this reason, pruning is limited to fully connected and convolutional layers (line 7 of Algorithm 4), as seen in other works [211, 212, 213]. The result of this operation is a network that has become more or less sparse, depending on the percentage of the pruned weights. In centralized contexts, it has already been shown that sparsity can facilitate data unlearning [214]. At this stage, the network has lost part of its prediction capabilities and shifted away from the potential local minimum it had reached after convergence.

Restoring Performance

Pruning is inherently destructive, resulting in the loss of information stored in the zeroed out connections. Each connection stores information from multiple clients in a distributed training context such as FL. Therefore, pruning the weights, even if related to malicious clients, will likely erase some of the information learned from other clients. To mitigate this effect, a few additional training rounds performed by the remaining benign clients are sufficient to restore the performance lost during the unlearning procedure. The clients use the same local dataset and configuration as in the standard FL training process, and these recovery rounds remain transparent to them. This operation further strengthens the removal of the excluded client's contribution, as the model will be more finely tuned to the remaining clients. Estimating the number of recovery rounds is a complex task, as it is equivalent to estimating the number of rounds a model needs to converge in a standard FL process. However, in Section 6.7.2, we provide some considerations that offer insight into the estimation.

6.7.2 Technical Insights

In this section, we give technical insights and practical guidelines to tune the algorithm properly.

Pruning Most Dissimilar Weights

The choice to prune the most dissimilar weights is supported by mathematical evidence, as detailed in the following section. In FL, each client starts a training round with the global model from the previous round $W^{GM;t-1}$. The client trains this model on its local data, computing updates W^{C_i} , which adjust the global model to the client's data. The updated client weights are:

$$W^{C_i;t} = W^{GM;t-1} + W^{C_i} \quad (6.6)$$

The server aggregates these updates, typically using weighted averaging, to create the new global model for the next round. Here, W^{C_i} reflects the client-specific contributions. In a converged model, $W^{GM;t-1}$ is near-optimal for the global data distribution, minimizing global loss. For weights already optimized globally, updates W^{C_i} are small. In contrast, updates for weights critical to reducing a client's local loss $\mathcal{L}_i^{C_i}$, are larger. At convergence, the difference between benign and malicious weights stabilizes, manifesting two distinct patterns:

1. Both parties are satisfied with the weight value, so the updates are nearly zero. These weights usually capture the common knowledge shared by all clients. Malicious clients,

6.7 FedUP: Efficient Federated Unlearning Through Pruning

for example, may possess benign data to complicate their identification. In this case, the difference is minimal.

- Both parties have updated the weights in a conflicting manner. These weights are the contested ones, used by clients to shape the network and minimize the loss on their data. In this case, the difference is more significant.

During training, clients will naturally adjust the weights to align with each other's updates, promoting consensus on those parameters. However, when clients possess malicious data, their update will diverge on a particular weight [209]. This divergence can persist at convergence, given that malicious clients are typically in the minority, as most of the model's knowledge reflects benign clients' contributions. To identify the weights most targeted by malicious clients, the server computes the differences, w_j^{di} , between all the weights from benign and malicious clients. Weights with the largest values w_j^{di} are the most likely to have been influenced by malicious clients.

Threshold Choice

One of the most crucial aspects for precise unlearning is choosing an appropriate percentage of weights to prune P . Selecting a value that is too low risks leaving malicious knowledge intact within the model. Conversely, a value that is too high risks pruning too many weights, thereby completely deteriorating the model. The percentage can be fine-tuned based on the following considerations:

- Number of weights: In larger networks, there is more room for all clients' contributions, which makes it easier to detect targeted weights. This reduces the need for aggressive pruning. Conversely, in smaller networks, client contributions are more likely to overlap, necessitating more pruning to remove malicious clients' influence effectively.
- Skewness of the distances between benign and malicious weights: High right skewness in the distribution of weight differences suggests that many weights are similar, indicating more overlap between benign and malicious contributions. This scenario is often associated with an IID setup, where more pruning is needed. On the other hand, low skewness indicates dissimilar weights, with less overlap and potentially a Non-IID setup, requiring less pruning. These scenarios have already been shown in Figure 6.4.

Considering these factors, the percentage value could be approximated by:

$$P = \frac{1}{W} + \text{Skew} \quad (6.7)$$

6 Federated Unlearning: Taxonomy, Approaches, and Solutions

where W is the total number of weights in the network, $Skew$ represents the skewness, and α and β are scaling parameters that ensure the percentage is between 0 and 1.

Number of Recovery Rounds

The recovery process involves retraining the model to restore performance after zeroing out specific weights. While predicting the exact number of recovery rounds is challenging, the upper bound is given by the number of rounds needed for retraining from scratch, denoted as R_{full} . We hypothesize that an optimal pruning percentage, P_{opt} , exists, which effectively removes the influence of malicious data and ensures $R_{pruned} < R_{full}$. Not all weights contribute equally to predicting a sample, implying a subset of weights, $W_{malicious}$, is responsible for predicting malicious samples $D_{malicious}$. We can determine P_{opt} by pruning one weight at a time, progressively increasing the pruning percentage with each iteration. As we prune more weights, the model's performance will degrade, increasing the number of recovery rounds. Once $W_{malicious}$ is no longer predictable. At P_{opt} , all malicious weights are pruned, leaving only the benign ones, which still retain useful knowledge. As a result, the model requires fewer recovery rounds than retraining from scratch because it already possesses some knowledge.

6.7.3 Experimental Evaluation

In this section, we present the experiments conducted to evaluate the effectiveness and efficiency of our algorithm. This evaluation aims to demonstrate how effectively FedUP removes the contribution of one or more malicious clients and efficiently restores the original performance in as few recovery rounds as possible. It also presents a comparison with a SOTA solution in terms of storage requirements and the number of recovery rounds needed to restore performance.

Experimental Setup

Malicious Setting. To evaluate the effectiveness of our algorithm in a SOTA malicious setting, we replicate the experimental setup outlined in [215], which provides a robust framework for simulating adversarial clients in FL. We utilize standard datasets such as CIFAR-10 [83] and FashionMNIST [216], which are commonly used in the literature for benchmarking FL algorithms. This enables an assessment of unlearning performance across varying degrees of difficulty. The neural network models used in our experiments are the same as those in the reference setting. They are standard CNNs similar to the LeNet architecture [217]. We employ the Adam optimizer with a learning rate of $1e-3$. Local training is conducted over 1 epoch with a batch size of 32. We repeat each experiment in various FL settings, using FedAVG as the aggregation strategy.

6.7 FedUP: Efficient Federated Unlearning Through Pruning

We vary the number of clients (5, 10, 20) and the data distribution (IID and Non-IID, using a Dirichlet distribution with $\alpha = 0.5$). Malicious clients execute a label-flipping attack, an adversarial attack where the attacker deliberately changes the labels of the training data to incorrect values. This attack aims to degrade the performance of the global model by misleading the training process, causing the model to learn incorrect patterns and make erroneous predictions. In our experiments, malicious clients swap 10% of the labels while keeping the remaining data correctly labeled. The labels to flip are chosen following the same criteria as the reference paper, namely, the most wrongly classified labels in a non-malicious scenario [215]. This approach is designed to make malicious activities more challenging to detect. Furthermore, we consider various percentages of malicious clients: 30% (with 10 and 20 clients) and 40% (across all client numbers). We choose these proportions because at least 30% of malicious clients are required to execute a successful label-flipping attack [215]. To evaluate the efficacy of the unlearning performance, we measure the accuracy on the to-be-forgotten data before and after the unlearning procedure. This metric is a SOTA way to understand how effectively the model has unlearned the specific data [14]. After unlearning, the new model should perform similarly to a retrained model on the forgotten malicious data, that we use a baseline. To assess the restoration performance, we evaluate the model on the same test dataset before and after the unlearning process. The new model should maintain equivalent performance. To evaluate efficiency, we measure the number of recovery rounds, required to restore the global model's performance, compared to the rounds needed for a retrain from scratch R . Lower values indicate faster performance recovery and, consequently, greater efficiency.

SOTA Setting: FedEraser. To better show the capabilities of our algorithm, we also present a comparison with a SOTA solution, FedEraser [169]. In FedEraser, performing unlearning requires the server's participation, which retains all historical updates, along with the remaining clients. However, it cannot operate in a malicious context, and removing multiple clients simultaneously is not supported. To ensure the fairest comparison, we use the same setup in FedEraser: a small custom CNN and the same dataset (CIFAR-10) split in an IID manner among 10 clients. While FedEraser is designed to forget specific benign clients and FedUP focuses on removing contributions from malicious clients, this comparison is essential to evaluate the unlearning effectiveness of FedUP. FedEraser was chosen for this analysis as it is one of the few solutions with publicly available and verifiable code. For additional details, please refer to Section 6.5.

Results

This section shows the results of FedUP in a malicious setting and compared with a SOTA solution.

6 Federated Unlearning: Taxonomy, Approaches, and Solutions

No. Clients (Malicious)	IID									Non-IID								
	Test Data		Malicious Data			R	R _{rec}	P	Test Data		Malicious Data			R	R _{rec}	P		
	Before	After	B	Before	After				Before	After	B	Before	After					
5 (2)	78	79	4	57	5	40	1	10	72	70	4	44	3	32	2	1		
10 (3)	79	77	3	51	2	26	1	11	78	78	8	30	6	37	2	4		
10 (4)	78	78	4	59	3	27	1	11	77	77	10	23	9	43	2	4		
20 (6)	80	78	4	60	5	42	2	11	79	77	11	37	11	75	4	3		
20 (8)	79	79	5	69	6	54	3	12	77	77	15	37	15	52	4	2		

Table 6.3: Accuracy (%) before and after unlearning on test and malicious clients' data, the number of recovery rounds, and the percentage of weights pruned (%) used with CIFAR-10.

No. Clients (Malicious)	IID									Non-IID								
	Test Data		Malicious Data			R	R _{rec}	P	Test Data		Malicious Data			R	R _{rec}	P		
	Before	After	B	Before	After				Before	After	B	Before	After					
5 (2)	84	88	4	73	5	22	1	15	83	82	21	54	20	19	2	6		
10 (3)	85	87	4	65	6	34	1	15	89	88	9	22	7	25	1	5		
10 (4)	84	88	5	72	5	36	1	15	86	87	11	36	12	19	1	5		
20 (6)	86	88	6	58	5	51	1	15	90	89	5	16	4	64	2	5		
20 (8)	83	87	5	73	6	57	1	15	89	88	3	12	2	33	2	5		

Table 6.4: Accuracy (%) before and after unlearning on test and malicious clients' data, the number of recovery rounds, and the percentage of weights pruned (%) used with Fashion-MNIST.

Performance with Malicious Clients. Tables 6.3 and 6.4 show the results of FedUP with CIFAR-10 and Fashion-MNIST in both IID and Non-IID settings. These tables present the performance metrics of the global model before and after applying FedUP, specifically the accuracy obtained on the test set and on the malicious data. They also highlight the number of recovery rounds, R_{rec} , required to achieve these results. Table 6.3 shows the results of FedUP on the CIFAR-10 dataset. As shown, FedUP effectively eliminates the influence of malicious data, as evidenced by the significant drop in the accuracy on their data. Meanwhile, the performance on the test set is almost completely restored in nearly all cases. The performance achieved is comparable to that of the baseline. Table 6.4 illustrates the performance of the global model on the Fashion-MNIST dataset. The results are consistent with those observed with the CIFAR-10 dataset. Specifically, a notable decrease in performance on data from malicious clients is evident, while the performance on the test set remains essentially unchanged. Regarding the number of recovery rounds, the recovery process is generally very fast in both cases. However, the Non-IID case typically requires more recovery rounds on average. This is due to the highly heterogeneous data distribution across clients in Non-IID settings, which makes the updates less stable. As a result, the model experiences more variability in the updates, requiring additional iterations to recover the model effectively. Nevertheless, the number of recovery rounds is considerably smaller than those needed to retrain from scratch. The selection of the percentage of weights to prune follows the guidelines presented in Section 6.7.2, demonstrating their effectiveness. The IID setting requires higher percentages of

6.7 FedUP: Efficient Federated Unlearning Through Pruning

Work	Test Data Before/After	Forgetting Data Before/After	R_{rec}	Storage (MB)
FedEraser	56 / 56	76 / 56	20	54
FedUP	56 / 56	74 / 56	3	2.7

Table 6.5: Comparison with FedEraser showing the accuracy (%) on test and forgetting data before and after unlearning, the number of recovery rounds, and the storage needed (MB).

pruning to remove more weights in both cases, as many are likely similar. In contrast, the Non-IID setting demands lower percentages of pruning due to the presence of more conflicting weights. Regarding the number of weights, the CNN used in the CIFAR-10 experiment contains approximately 500k parameters, while the model used for Fashion-MNIST has 50k parameters. This difference is reflected in the choice of pruning percentages. A larger number of parameters provides more room for weights to differ, leading to less overlap and, consequently, requiring lower percentages for pruning. Conversely, a smaller number of parameters results in a more significant overlap between weights, necessitating higher percentages to effectively remove the conflicting weights.

Comparison with SOTA. Table 6.5 presents the comparison with FedEraser using CIFAR-10 and IID data in a non-malicious setting with a custom CNN. It shows the accuracy of the global model on the test dataset and on the client's data to be forgotten, both before the unlearning procedure and after just one recovery round. Both algorithms succeed in effectively unlearning the client's data, as evidenced by the significant drop in the global model's performance on that client's data. Additionally, they can recover the performance lost during the unlearning process, restoring the model's effectiveness on retained data. However, the key difference is that while FedEraser fails to recover the original performance rapidly, FedUP fully restores the model's performance in just a few recovery rounds. In contrast, FedEraser requires at least 20 additional training rounds to achieve similar results. Moreover, FedUP requires only the storage needed by FedEraser to execute, as it only requires the updates and the global model from the last training round. The pruning percentage selected for this setting is 30%. This is a particularly unique configuration, as it is both IID and does not involve malicious clients or clients with distinct data. Additionally, the CNN used in this experiment has only 62k parameters. Even in this context, the percentage of weights to prune is consistent with expectations. This comparison highlights how our algorithm is significantly faster and more storage-efficient than a SOTA solution while remaining equally effective. Furthermore, this shows that, despite being specifically designed for malicious environments, FedUP can be effectively applied in benign settings as well.

6.7.4 Conclusion

In this work, we introduce FedUP, a novel FU algorithm that can efficiently remove malicious contributions through pruning techniques. FedUP works by zeroing out the highest magnitude weights that differ the most between benign and malicious clients. To recover the performance lost during pruning, FedUP performs a few additional training rounds. Our extensive experiments, conducted on both IID and Non-IID settings with multiple datasets, demonstrate that FedUP effectively reduces the influence of malicious contributions while efficiently restoring the model's performance on benign data. Furthermore, FedUP proves to be faster than a SOTA FU solution, achieving full recovery in significantly fewer rounds and using considerably less storage. These results showcase FedUP as an efficient approach to securing FL models in environments where adversaries attempt to poison the global model.

7 Conclusion and Future Work

This dissertation explored how DLT can serve as an enabling backbone for fostering trust and security in FL. As FL continues to evolve as a popular method for training models across distributed data sources, trustworthiness, transparency, and security become essential. It has been analyzed how DLT, with its decentralized, immutable, and verifiable properties, addresses the key challenges that FL faces, extensively described in Chapter 3.

The first major contribution of this dissertation was the introduction of novel architectures for integrating FL with blockchain and DAGs in Chapter 4. These architectures provide distinct advantages, offering scalable solutions that can adapt to the diverse needs of both enterprise environments and resource-constrained edge systems. In particular, this dissertation proposed FlowChain, an enterprise-ready blockchain-based solution for environments requiring high levels of trust and transparency, and FETA, a DAG-based architecture designed to enhance scalability and performance. Extensive experiments demonstrated that both architectures can be effectively deployed in real-world scenarios through a detailed analysis of resource consumption. Specifically, with FlowChain, an actual deployment in the healthcare domain was presented, introducing FRAMH, a FL risk-based authorization middleware.

Another essential contribution of this dissertation was the presentation of TruFLaaS, a framework designed to deliver trustworthy FLaaS, in Chapter 5. Built on FlowChain as the enabling technology, TruFLaaS ensures the integrity, transparency, and accountability of the entire FL process. This is further reinforced by a novel model validation technique directly executed through a smart contract. The framework not only aims to simplify the setup of FL processes but also introduces incentives to encourage active and honest participation.

In Chapter 6, this dissertation explored FU, a crucial concept to ensure privacy and regulatory compliance within FL systems. A comprehensive review of the existing approaches was presented, focusing on metrics and design guidelines. This dissertation then introduced a novel unlearning algorithm tailored for adversary settings to remove malicious contributions from the global model.

As future work, the first step will be integrating the unlearning algorithm, FedUP, into the proposed DLT-based FL architecture. This integration will provide a comprehensive solution

7 Conclusion and Future Work

that addresses the challenges outlined at the beginning of this dissertation. By leveraging the decentralized and immutable characteristics of DLT, we aim to enhance the trustworthiness and integrity of model updates in FL while ensuring full compliance with regulatory requirements. Specifically, to assess and certify the unlearning process, DIDs and VCs will play a crucial role. These technologies will be used to verify and certify that unlearning actions have been properly executed, offering proof that specific data contributions have been removed from the model in accordance with regulations such as GDPR. This certification process will ensure that unlearning actions are accurate and verifiable, providing transparency and confidence for both participants and regulators.

Furthermore, we will explore metrics to effectively evaluate client contributions within the FL framework. Developing robust evaluation criteria will allow us to quantify the quality and impact of each client's data contributions on the overall model performance. This assessment will help refine the unlearning process and enable more informed decisions about which contributions should be retained or discarded.

Implementing these metrics will significantly strengthen the system's rewards and accountability mechanisms. Establishing a clear framework for evaluating client contributions will create a more transparent reward system that incentivizes high-quality data sharing while discouraging malicious or low-value inputs. This approach will foster greater participant engagement and enhance the overall security and reliability of the FL system. By encouraging a culture of accountability, all clients will better understand their roles and responsibilities, reducing the risks associated with malicious contributions and boosting participant trust.

Overall, these efforts will contribute to a more resilient and secure distributed learning environment, leading to the effective application of FL in sensitive and complex data-driven scenarios.

Bibliography

- [1] K. Sha que, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios, *IEEE Access*, vol. 8, pp. 23022–23040, 2020.
- [2] Statista, Number of internet of things (iot) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2033. URL <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>, 2024. Accessed: 2024-09-18.
- [3] European Union, General data protection regulation compliance. URL <https://gdpr.eu/>, 2016. Accessed: 2024-07-01.
- [4] State of California Department of Justice, California consumer privacy act (ccpa). URL <https://oag.ca.gov/privacy/ccpa>, 2018. Accessed: 2024-07-01.
- [5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.
- [6] K. Sameera, S. Nicolazzo, M. Arazzi, A. Nocera, R. R. KA, P. Vinod, and M. Conti, Privacy-preserving in blockchain-based federated learning systems, *Computer Communications*, 2024.
- [7] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, Practical secure aggregation for privacy-preserving machine learning, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, ACM, 2017.
- [8] Q. Yang, Y. Liu, T. Chen, and Y. Tong, Federated machine learning: Concept and applications, *ACM Transactions on Intelligent Systems and Technology (TIST)*, no. 2, pp. 1–19, 2019.

Bibliography

- [9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, Federated optimization in heterogeneous networks, *Proceedings of Machine learning and systems* no. 3, pp. 429–450, 2020.
- [10] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, *Foundations and trends in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [11] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, How To Backdoor Federated Learning, in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics* (S. Chiappa and R. Calandra, eds.), vol. 138, *Proceedings of Machine Learning Research*, pp. 2938–2948, PMLR, PMLR, 26–28 Aug 2020.
- [12] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, Inverting gradients - how easy is it to break privacy in federated learning?, *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 16937–16947, Curran Associates, Inc., 2020.
- [13] G. Ye, Q. V. H. Nguyen, and H. Yin, Heterogeneous Decentralized Machine Unlearning with Seed Model Distillation, *arXiv preprint arXiv:2308.13260*, vol. 56, no. 3, pp. 1–44, 2023.
- [14] N. Romandini, A. Mora, C. Mazzocca, R. Montanari, and P. Bellavista, Federated unlearning: A survey on methods, design guidelines, and evaluation metrics, *Transactions on Neural Networks and Learning Systems*, 2024.
- [15] P. Bellavista, L. Foschini, and A. Mora, Decentralised Learning in Federated Deployment Environments: A System-Level Survey, *ACM Comput. Surv.*, vol. 54, pp. 1–38, feb 2021.
- [16] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, A review of applications in federated learning, *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.
- [17] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, Scalable: Stochastic controlled averaging for federated learning, *International conference on machine learning*, pp. 5132–5143, PMLR, 2020.
- [18] C. Xu, Y. Qu, Y. Xiang, and L. Gao, Asynchronous federated learning on heterogeneous devices: A survey, *Computer Science Review*, vol. 50, p. 100595, 2023.

- [19] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, A survey on federated learning: challenges and applications, *International Journal of Machine Learning and Cybernetics* vol. 14, no. 2, pp. 513–535, 2023.
- [20] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, Federated learning for smart healthcare: A survey, *ACM Computing Surveys (CSUR)* vol. 55, no. 3, pp. 1–37, 2022.
- [21] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, Vertical federated learning: Concepts, advances, and challenges, *IEEE Transactions on Knowledge and Data Engineering*, 2024.
- [22] P. Chen, X. Du, Z. Lu, J. Wu, and P. C. Hung, Ev : An explainable vertical federated learning for data-oriented artificial intelligence systems, *Journal of Systems Architecture* vol. 126, p. 102474, 2022.
- [23] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, A secure federated transfer learning framework, *IEEE Intelligent Systems* vol. 35, no. 4, pp. 70–82, 2020.
- [24] P. Kairouz et al, Advances and Open Problems in Federated Learning, preprint arXiv:1912.04977, 2019.
- [25] C. Dwork, Differential privacy, *Encyclopedia of Cryptography and Security*, pp. 338–340, 2011.
- [26] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, *Decentralized business review* 2008.
- [27] R. Beck, Beyond Bitcoin: The Rise of Blockchain World, *Computers* vol. 51, no. 2, pp. 54–58, 2018.
- [28] Ethereum, Proof-of-stake (pos), July 2023.
- [29] S. Popov, The tangle, *White paper* vol. 1, no. 3, p. 30, 2018.
- [30] W3 Recommendation, Decentralized Identifiers (DIDs) v1.0, 2022.
- [31] W. Recommendation, Verifiable Credentials Data Model v1.1, 2022.
- [32] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, Analyzing federated learning through an adversarial lens, *International conference on machine learning*, pp. 634–643, PMLR, 2019.

Bibliography

- [33] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, A survey on security and privacy of federated learning, *Future Generation Computer Systems* vol. 115, pp. 619–640, 2021.
- [34] J. Zhang, H. Zhu, F. Wang, J. Zhao, Q. Xu, and H. Li, Security and privacy threats to federated learning: Issues, methods, and challenges, *Security and Communication Networks* vol. 2022, no. 1, p. 2886795, 2022.
- [35] B. Biggio, B. Nelson, and P. Laskov, Poisoning attacks against support vector machines, *arXiv preprint arXiv:1206.6389*, 2012.
- [36] M. Fang, X. Cao, J. Jia, and N. Gong, Local Model Poisoning Attacks to Byzantine-Robust Federated Learning, *20th USENIX Security Symposium (USENIX Security 20)* pp. 1605–1622, USENIX Association, Aug. 2020.
- [37] T. Gu, B. Dolan-Gavitt, and S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain, *arXiv preprint arXiv:1708.06733*, 2017.
- [38] Y. Fraboni, R. Vidal, and M. Lorenzi, Free-rider attacks on model aggregation in federated learning, in *International Conference on Artificial Intelligence and Statistics*, pp. 1846–1854, PMLR, 2021.
- [39] L. Lyu, H. Yu, and Q. Yang, Threats to federated learning: A survey, *arXiv preprint arXiv:2003.02133*, 2020.
- [40] Y. Zeng, H. Chen, and K. Lee, Improving fairness via federated learning, *arXiv preprint arXiv:2110.15548*, 2021.
- [41] R. Zeng, C. Zeng, X. Wang, B. Li, and X. Chu, A comprehensive survey of incentive mechanism for federated learning, *arXiv preprint arXiv:2106.15426*, 2021.
- [42] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, Federated learning for mobile keyboard prediction, *arXiv preprint arXiv:1811.03602*, 2018.
- [43] R. Shokri and V. Shmatikov, Privacy-preserving deep learning, in *Proceedings of the 22nd ACM SIGSAC conference on computer and communication security*, pp. 1321–1329, ACM, 2015.

- [44] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, Membership inference attacks against machine learning models, 2017 IEEE Symposium on Security and Privacy (SP) 18, IEEE, 2017.
- [45] N. Truong, K. Sun, S. Wang, F. Guitton, and Y. Guo, Privacy preservation in federated learning: An insightful survey from the gdpr perspective, Computers & Security, vol. 110, p. 102402, 2021.
- [46] L. Lao, Z. Li, S. Hou, B. Xiao, S. Guo, and Y. Yang, A survey of iot applications in blockchain systems: Architecture, consensus, and traffic modeling, ACM Computing Surveys (CSUR), vol. 53, no. 1, pp. 1–32, 2020.
- [47] Z. Wang and Q. Hu, Blockchain-based Federated Learning: A Comprehensive Survey, ArXiv, vol. abs/2110.02182, 2021.
- [48] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, Blockchain-enabled federated learning: A survey, ACM Computing Surveys, vol. 55, no. 4, pp. 1–35, 2022.
- [49] U. Majeed and C. S. Hong, Flchain: Federated learning via mec-enabled blockchain network, in 2019 20th Asia-Pacific Network Operations and Management Symposium (AP-NOMS), pp. 1–4, IEEE, 2019.
- [50] Y. Qu, L. Gao, T. H. Luan, Y. Xiang, S. Yu, B. Li, and G. Zheng, Decentralized privacy using blockchain-enabled federated learning in fog computing, IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5171–5183, 2020.
- [51] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory, IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10700–10714, 2019.
- [52] W. Issa, N. Moustafa, B. Turnbull, N. Sohrabi, and Z. Tari, Blockchain-Based Federated Learning for Securing Internet of Things: A Comprehensive Survey, ACM Comput. Surv., vol. 55, pp. 1759–1799, jan 2023.
- [53] C. Mazzocca, N. Romandini, M. Mendula, R. Montanari, and P. Bellavista, TruFLaaS: Trustworthy Federated Learning as a Service, IEEE Internet of Things Journal, vol. 10, no. 24, pp. 1–11, 2023.
- [54] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT, IEEE Transactions on Industrial Informatics, vol. 16, no. 6, pp. 4177–4186, 2020.

Bibliography

- [55] H. Kim, J. Park, M. Bennis, and S.-L. Kim, Blockchained on-device federated learning, *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019.
- [56] S. Liu, X. Wang, L. Hui, and W. Wu, Blockchain-Based Decentralized Federated Learning Method in Edge Computing Environment, *Applied Sciences*, vol. 13, no. 3, 2023.
- [57] Y. Liu, Y. Qu, C. Xu, Z. Hao, and B. Gu, Blockchain-Enabled Asynchronous Federated Learning in Edge Computing, *Sensors*, vol. 21, no. 10, 2021.
- [58] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, Low-Latency Federated Learning and Blockchain for Edge Association in Digital Twin Empowered 6G Networks, *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5098–5107, 2021.
- [59] R. Schmid, B. P. tzner, J. Beilharz, B. Arnrich, and A. Polze, Tangle Ledger for Decentralized Learning, in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 852–859, 2020.
- [60] M. Cao, L. Zhang, and B. Cao, Toward On-Device Federated Learning: A Direct Acyclic Graph-Based Blockchain Approach, *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.
- [61] L. Jiang, H. Zheng, H. Tian, S. Xie, and Y. Zhang, Cooperative Federated Learning and Model Update Verification in Blockchain-Empowered Digital Twin Edge Networks, *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11154–11167, 2022.
- [62] J. Lee and W. Kim, DAG-Based Blockchain Sharding for Secure Federated Learning with Non-IID Data, *Sensors*, vol. 22, no. 21, 2022.
- [63] P. Bellavista, L. Foschini, R. Montanari, and N. Romandini, Flowchain: the playground for federated learning in industrial internet of things environment, *IEEE Internet of Things Magazine*, vol. 5, no. 2, pp. 78–83, 2022.
- [64] H. Foundation, Hyperledger fabric, Jul 2023.
- [65] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability, in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 536–540, IEEE, 2019.
- [66] Hyperledger, Trustid: A new approach to fabric user identity management, hyperledger foundation, Apr 2020.

- [67] F. Chollet et al, Keras, 2015.
- [68] nodeca, Github - nodeca/pako: High speed zlib port to javascript, works in browser and node.js, 2014.
- [69] P. Deutsch, DEFLATE Compressed Data Format Specification version RFC, vol. 1951, pp. 1-17, 1996.
- [70] S. Cai, S. Bileschi, and E. Niebler, Deep Learning with JavaScript: Neural networks in TensorFlow.js, Manning, 2020.
- [71] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference, 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2704-2713, 2018.
- [72] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão et al, Flower: A friendly federated learning research framework, arXiv preprint arXiv:2007.14390, 2020.
- [73] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks, IEEE Access, vol. 7, pp. 22328-22370, 2019.
- [74] J. Sedlmeir, H. U. Buhl, G. Fridgen, and R. Keller, The Energy Consumption of Blockchain Technology: Beyond Myths, Business & Information Systems Engineering, vol. 62, pp. 599-608, Dec 2020.
- [75] H. Vranken, Sustainability of bitcoin and blockchain, Current Opinion in Environmental Sustainability, vol. 28, pp. 1-9, 2017. Sustainability governance.
- [76] M. J. Krause and T. Tolaymat, Quantification of energy and carbon costs for mining cryptocurrencies, Nature Sustainability, vol. 1, pp. 711-718, Nov 2018.
- [77] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, Energy Efficient Federated Learning Over Wireless Communication Networks, IEEE Transactions on Wireless Communications, vol. 20, no. 3, pp. 1935-1949, 2021.
- [78] C. W. Zaw, S. R. Pandey, K. Kim, and C. S. Hong, Energy-Aware Resource Management for Federated Learning in Multi-Access Edge Computing Systems, IEEE Access, vol. 9, pp. 34938-34950, 2021.

Bibliography

- [79] M. Uddin, Y. Darabidarabkhani, A. Shah, and J. Memon, Evaluating power efficient algorithms for efficiency and carbon emissions in cloud data centers: A Renewable and Sustainable Energy Reviews, vol. 51, pp. 1553–1563, 2015.
- [80] A. Van de Ven, Powertop, Aug 2022.
- [81] jdrouet, Github - jdrouet/docker-activity: Tool to monitor the statistics and the energy consumption of docker containers, Jan 2022.
- [82] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. cite arxiv:1708.07747 Comment: Dataset is freely available at <https://github.com/zalando-research/fashion-mnist> Benchmark is available at <http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/>.
- [83] A. Krizhevsky, Learning multiple layers of features from tiny images, tech. rep., 2009. CIFAR-10 dataset, available at <http://www.cs.toronto.edu/~kriz/cifar.html>.
- [84] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520, June 2018.
- [85] NVIDIA, Nvidia jetson nano, Oct 2024.
- [86] C. Mazzocca, N. Romandini, M. Colajanni, and R. Montanari, Framh: A federated learning risk-based authorization middleware for healthcare, IEEE Transactions on Computational Social Systems, vol. 10, no. 4, pp. 1679–1690, 2022.
- [87] H. F. Atlam, M. A. Azad, M. O. Alassa, A. A. Alshdadi, and A. Alenezi, Risk-based access control model: A systematic literature review, Future Internet, vol. 12, no. 6, p. 103, 2020.
- [88] D. Choi, D. Kim, and S. Park, A framework for context sensitive risk-based access control in medical information systems, Computational and mathematical methods in medicine, vol. 2015, 2015.
- [89] O. Standard, Extensible Access Control Markup Language (XACML) version 3.0 (2013), January 2013). URI: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html> 2013.
- [90] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, Attack of the Tails: Yes, You Really Can Backdoor Federated Learn-

- ing, in *Advances in Neural Information Processing Systems*, pp. 16070–16084, Curran Associates, Inc., 2020.
- [91] Hangzhou Internet Court of the People's Republic of China, *Hangzhou Huatai Yimei Culture Media Co., Ltd. v. Shenzhen Daotong Technology Development Co., Ltd.*, *Digital Evidence and Electronic Signature Law Review* 70, 12 2019.
- [92] I. Silva, G. B. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, Predicting in-hospital mortality of ICU patients: The PhysioNet/Computing in cardiology challenge 2012, *Computing in Cardiology*, pp. 245–248, 2012.
- [93] A. E. Johnson et al., Patient specific predictions in the intensive care unit using a Bayesian ensemble, in *Computing in Cardiology (CinC)*, 2012, pp. 249–252, IEEE, 2012.
- [94] G. L. Aguiar, B. Krawczyk, and A. Cano, A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework, *ArXiv*, vol. abs/2204.03719, 2022.
- [95] Y. Bengio, Practical Recommendations for Gradient-Based Training of Deep Architectures, in *Neural Networks: Tricks of the Trade*, 2012.
- [96] A. F. Agarap, Deep Learning using Rectified Linear Units (ReLU), *ArXiv*, vol. abs/1803.08375, 2018.
- [97] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, *CoRR*, vol. abs/1412.6980, 2015.
- [98] K. You, M. Long, J. Wang, and M. I. Jordan, How Does Learning Rate Decay Help Modern Neural Networks?, *arXiv: Learning* 2019.
- [99] S. B. Baker, W. Xiang, and I. M. Atkinson, Continuous and automatic mortality risk prediction using vital signs in the intensive care unit: a hybrid neural network approach, *Scientific Reports*, vol. 10, 2020.
- [100] R. Sadeghi, T. Banerjee, and W. L. Romine, Early Hospital Mortality Prediction using Vital Signals, *Smart health*, vol. 9-10, pp. 265–274, 2018.
- [101] L. Brand, A. Patel, I. Singh, and C. Brand, Real Time Mortality Risk Prediction: A Convolutional Neural Network Approach, in *HEALTHINF*, 2018.

Bibliography

- [102] C. Mazzocca, N. Romandini, R. Montanari, and P. Bellavista, Enabling federated learning at the edge through the iota tangle, *Future Generation Computer Systems*, vol. 152, pp. 17 29, 2024.
- [103] H. R. Hasan, K. Salah, I. Yaqoob, R. Jayaraman, S. Pesic, and M. Omar, Trustworthy IoT Data Streaming Using Blockchain and IEEE Access, vol. 10, pp. 17707 17721, 2022.
- [104] iotaledger, Github - iotaledger/identity.rs: Implementation of the decentralized identity standards such as did and verifiable credentials by w3c for the iota tangle., Sep 2024.
- [105] Y. LeCun, C. Cortes, and C. Burges, Mnist handwritten digit database, *AT&T Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist/>, 2010.
- [106] N. Kourtellis, K. Katevas, and D. Perino, FLaaS: Federated Learning as a Service, in *Proceedings of the 1st Workshop on Distributed Machine Learning*, (New York, NY, USA), p. 7 13, Association for Computing Machinery, 2020.
- [107] R. K. Mobley, *An introduction to predictive maintenance*. Elsevier, 2002.
- [108] R. Philipp, A. Mladenow, C. Strauss, and A. Völz, Machine Learning as a Service: Challenges in Research and Applications, in *Proceedings of the 22nd International Conference on Information Integration and Web-Based Applications and Services*, (New York, NY, USA), p. 396 406, Association for Computing Machinery, 2020.
- [109] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, Federated learning in smart city sensing: Challenges and opportunities, *Sensors*, vol. 20, no. 21, p. 6230, 2020.
- [110] Y. Wang and B. Kantarci, A novel reputation-aware client selection scheme for federated learning within mobile environments, *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* pp. 1 6, IEEE, 2020.
- [111] Y. Wang and B. Kantarci, Reputation-enabled federated learning model aggregation in mobile platforms, in *ICC 2021-IEEE International Conference on Communications* pp. 1 6, IEEE, 2021.
- [112] Y. Wang, B. Kantarci, and W. Mardini, Aggregation of incentivized learning models in mobile federated learning environments, *IEEE Networking Letters*, vol. 3, no. 4, pp. 196 200, 2021.

- [113] M. H. u. Rehman, A. M. Dirir, K. Salah, E. Damiani, and D. Svetinovic, TrustFed: A Framework for Fair and Trustworthy Cross-Device Federated Learning in IEEE, Transactions on Industrial Informatics, vol. 17, no. 12, pp. 8485–8494, 2021.
- [114] H. Chen, S. A. Asif, J. Park, C.-C. Shen, and M. Bennis, Robust blockchained federated learning with model validation and proof-of-stake inspired consensus, preprint arXiv:2101.03300, 2021.
- [115] S. K. Lo, Y. Liu, Q. Lu, C. Wang, X. Xu, H.-Y. Paik, and L. Zhu, Towards Trustworthy AI: Blockchain-based Architecture Design for Accountability and Fairness of Federated Learning Systems, IEEE Internet of Things Journal, pp. 1–11, 2022.
- [116] M. Abdel-Basset, N. Moustafa, and H. Hawash, Privacy-Preserved Cyberattack Detection in Industrial Edge of Things (IEoT): A Blockchain-Orchestrated Federated Learning Approach, IEEE Transactions on Industrial Informatics, vol. 18, no. 11, pp. 7920–7934, 2022.
- [117] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning, In FOCOM 2019 - IEEE Conference on Computer Communications, pp. 2512–2520, 2019.
- [118] J. C.-W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, and M. Aloqaily, Privacy-Preserving Multiobjective Sanitization Model in 6G IoT Environments, IEEE Internet of Things Journal, vol. 8, no. 7, pp. 5340–5349, 2021.
- [119] C.-W. Lin, T.-P. Hong, and H.-C. Hsu, Reducing Side Effects of Hiding Sensitive Itemsets in Privacy Preserving Data Mining, The Scientific World Journal, vol. 2014, p. 235837, Apr 2014.
- [120] J. Park and H. Lim, Privacy-Preserving Federated Learning Using Homomorphic Encryption, Applied Sciences, vol. 12, no. 2, 2022.
- [121] J. Geng, N. Kanwal, M. G. Jaatun, and C. Rong, DID-EFed: Facilitating Federated Learning as a Service with Decentralized Identities, Evaluation and Assessment in Software Engineering, EASE 2021, (New York, NY, USA), p. 329–335, Association for Computing Machinery, 2021.
- [122] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, A Learning-Based Incentive Mechanism for Federated Learning, IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6360–6368, 2020.

Bibliography

- [123] Y. Li, Y. Chen, K. Zhu, C. Bai, and J. Zhang, An Effective Federated Learning Verification Strategy and Its Applications for Fault Diagnosis in Industrial IoT Systems, *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16835–16849, 2022.
- [124] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, Reliable Federated Learning for Mobile Networks, *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.
- [125] X. Cao, M. Fang, J. Liu, and N. Z. Gong, FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping, 2022.
- [126] L. Wang, X. Zhao, Z. Lu, L. Wang, and S. Zhang, Enhancing privacy preservation and trustworthiness for decentralized federated learning, *Information Sciences*, vol. 628, pp. 449–468, 2023.
- [127] H. Gao, N. He, and T. Gao, SVeriFL: Successive verifiable federated learning with privacy-preserving, *Information Sciences*, vol. 622, pp. 98–114, 2023.
- [128] G. Caldarelli, Understanding the Blockchain Oracle Problem: A Call for Action, *Information*, vol. 11, no. 11, 2020.
- [129] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, et al., Chainlink 2.0: Next steps in the evolution of decentralized oracle networks, *Chainlink Labs*, 2021.
- [130] D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, and A. Miller, CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability, 2021 IEEE Symposium on Security and Privacy (SP), pp. 1348–1366, 2021.
- [131] O. Avellaneda, A. Bachmann, A. Barbir, J. Brennan, P. Dingle, K. H. Du y, E. Maler, D. Reed, and M. Sporny, Decentralized Identity: Where Did It Come From and Where Is It Going?, *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 10–13, 2019.
- [132] D. D. S. Braga, M. Niemann, B. Hellingrath, and F. B. D. L. Neto, Survey on Computational Trust and Reputation Models, *ACM Comput. Surv.*, vol. 51, nov 2018.
- [133] D. Smilkov, N. Thorat, Y. Assogba, C. Nicholson, N. Kreeger, P. Yu, S. Cai, E. Nielsen, D. Soegel, S. Bileschi, M. Terry, A. Yuan, K. Zhang, S. Gupta, S. Sirajuddin, D. Sculley,

- R. Monga, G. Corrado, F. Viegas, and M. M. Wattenberg, TensorFlow.js: Machine Learning For The Web and Beyond, *Proceedings of Machine Learning and Systems* (F. T. Asch, S. Kulkarni, V. Smith, and M. Zaharia, eds.), vol. 1, pp. 309–321, 2019.
- [134] Provable, Provable things, Jun 2024.
- [135] A. Saxena, K. Goebel, D. Simon, and N. Eklund, Damage propagation modeling for aircraft engine run-to-failure simulation, *2008 International Conference on Prognostics and Health Management*, pp. 1–9, 2008.
- [136] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, N-BaloT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders, *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.
- [137] H. Xu, T. Zhu, L. Zhang, W. Zhou, and P. S. Yu, Machine unlearning: A survey, *Computing Surveys*, vol. 56, no. 1, pp. 1–36, 2023.
- [138] Y. Cao and J. Yang, Towards Making Systems Forget with Machine Unlearning, in *IEEE Symposium on Security and Privacy*, pp. 463–480, 2015.
- [139] L. Zhu, Z. Liu, and S. Han, Deep Leakage from Gradients, in *Proceedings of Conference on Neural Information Processing Systems*, pp. 4747–4756, 2019.
- [140] A. Golatkar, A. Achille, and S. Soatto, Eternal sunshine of the spotless net: Selective forgetting in deep networks, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9304–9312, 2020.
- [141] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges, *Information fusion*, vol. 58, pp. 52–68, 2020.
- [142] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, Measuring catastrophic forgetting in neural networks, *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.
- [143] G. Legate, L. Caccia, and E. Belilovsky, Re-weighted softmax cross-entropy to control forgetting in federated learning, *arXiv preprint arXiv:2304.05260*, 2023.
- [144] D. Caldarola, B. Caputo, and M. Ciccone, Improving Generalization in Federated Learning by Seeking Flat Minima, *Proc. of European Computer Vision Conference*, pp. 654–672, Springer, 2022.

Bibliography

- [145] G. Lee, M. Jeong, Y. Shin, S. Bae, and S.-Y. Yun, Preservation of the global knowledge by not-true distillation in federated learning, *Advances in Neural Information Processing Systems*, 2022.
- [146] X. Gao, X. Ma, J. Wang, Y. Sun, B. Li, S. Ji, P. Cheng, and J. Chen, Veri : Towards verifiable federated unlearning, *arXiv preprint arXiv:2205.12709*, 2022.
- [147] C. Dwork, A. Roth et al, The algorithmic foundations of differential privacy, *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3 4, pp. 211 407, 2014.
- [148] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, Learning differentially private recurrent language models, *arXiv preprint arXiv:1710.06968*, 2017.
- [149] R. C. Geyer, T. Klein, and M. Nabi, Differentially private federated learning: A client level perspective, *arXiv preprint arXiv:1712.07527*, 2017.
- [150] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, Differential privacy has disparate impact on model accuracy, *Advances in Neural Information Processing Systems*, 15453 15462, 2019.
- [151] L. Wu, S. Guo, J. Wang, Z. Hong, J. Zhang, and Y. Ding, Federated unlearning: Guarantee the right of clients to forget, *IEEE Network*, vol. 36, no. 5, pp. 129 135, 2022.
- [152] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, A Hybrid Approach to Privacy-preserving Federated Learning, *Proc. of the ACM Workshop on Artificial Intelligence and Security*, pp. 1 11, 2019.
- [153] C. Zhang, S. Ekanut, L. Zhen, and Z. Li, Augmented multi-party computation against gradient leakage in federated learning, *IEEE Transactions on Big Data*, 2022.
- [154] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli, Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 7210 7217, 2023.
- [155] T.-M. H. Hsu, H. Qi, and M. Brown, Measuring the effects of non-identical data distribution for federated visual classification, *arXiv preprint arXiv:1909.06325*, 2019.
- [156] Q. Li, Y. Diao, Q. Chen, and B. He, Federated Learning on Non-iid Data Silos: An Experimental Study, *Proc. of IEEE International Conference on Data Engineering, (ICDE)* pp. 965 978, IEEE, 2022.

- [157] F. Wang, B. Li, and B. Li, “Federated unlearning and its privacy threats,” *IEEE Network*, pp. 1–7, 2023.
- [158] M. P. Naeni, G. Cooper, and M. Hauskrecht, “Obtaining well calibrated probabilities using bayesian binning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 29, 2015.
- [159] Y. Liu, Z. Ma, X. Liu, and J. Ma, “Learn to forget: User-level memorization elimination in federated learning,” *arXiv preprint arXiv: 2205.14102*, vol. 1, 2020.
- [160] P. W. Koh and P. Liang, “Understanding black-box predictions via influence functions,” in *International conference on machine learning*, pp. 1885–1894, PMLR, 2017.
- [161] T. Shaik, X. Tao, L. Li, H. Xie, T. Cai, X. Zhu, and Q. Li, “FRAMU: Attention-based Machine Unlearning using Federated Reinforcement Learning,” *arXiv preprint arXiv: 2305.12345*, 2023.
- [162] A. Golatkar, A. Achille, A. Ravichandran, M. Polito, and S. Soatto, “Mixed-privacy forgetting in deep networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 792–801, June 2021.
- [163] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, “Privacy risk in machine learning: Analyzing the connection to overfitting,” in *IEEE 31st computer security foundations symposium (CSF)*, pp. 268–282, IEEE, 2018.
- [164] H. Hu, Z. Salcic, G. Dobbie, J. Chen, L. Sun, and X. Zhang, “Membership inference via backdooring,” in *The 31st International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.
- [165] A. Halimi, S. Kadhe, A. Rawat, and N. Baracaldo, “Federated unlearning: How to efficiently erase a client in fl?,” *arXiv preprint arXiv: 2205.14102*, 2022.
- [166] N. Su and B. Li, “Asynchronous federated unlearning,” in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pp. 1–10, 2023.
- [167] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, “The right to be forgotten in federated learning: An efficient realization with rapid retraining,” in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pp. 1749–1758, IEEE, 2022.

Bibliography

- [168] J. Wang, S. Guo, X. Xie, and H. Qi, “Federated Unlearning via Class-Discriminative Pruning,” in *Proceedings of the ACM Web Conference*, WWW ’22, (New York, NY, USA), p. 622–632, Association for Computing Machinery, 2022.
- [169] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, “Federaser: Enabling efficient client-level data removal from federated learning models,” in *IEEE/ACM 13th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, 2021.
- [170] J. Gong, O. Simeone, and J. Kang, “Bayesian variational federated learning and unlearning in decentralized networks,” in *IEEE 32nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 216–220, 2021.
- [171] Y. Zhao, P. Wang, H. Qi, J. Huang, Z. Wei, and Q. Zhang, “Federated unlearning with momentum degradation,” *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 8860–8870, 2024.
- [172] W. Yuan, H. Yin, F. Wu, S. Zhang, T. He, and H. Wang, “Federated unlearning for on-device recommendation,” in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 393–401, 2023.
- [173] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, “Fedrecover: Recovering from poisoning attacks in federated learning using historical information,” in *IEEE Symposium on Security and Privacy (SP)*, pp. 1366–1383, 2023.
- [174] X. Guo, P. Wang, S. Qiu, W. Song, Q. Zhang, X. Wei, and D. Zhou, “Fast: Adopting federated unlearning to eliminating malicious terminals at server side,” *IEEE Transactions on Network Science and Engineering*, pp. 1–14, 2023.
- [175] C. Wu, S. Zhu, and P. Mitra, “Federated unlearning with knowledge distillation,” *arXiv preprint arXiv:2208.10111*, 2022.
- [176] G. Li, L. Shen, Y. Sun, Y. Hu, H. Hu, and D. Tao, “Subspace based federated unlearning,” *arXiv preprint arXiv:2301.08111*, 2023.
- [177] M. Alam, H. Lamri, and M. Maniatakos, “Get rid of your trail: Remotely erasing backdoors in federated learning,” *arXiv preprint arXiv:2301.08111*, 2023.
- [178] J. Gong, J. Kang, O. Simeone, and R. Kassab, “Forget-svgd: Particle-based bayesian federated unlearning,” in *IEEE Data Science and Learning Workshop (DSLW)*, pp. 1–6, 2022.

- [179] J. Gong, O. Simeone, and J. Kang, “Compressed particle-based federated bayesian learning and unlearning,” *IEEE Communications Letters*, vol. 27, no. 2, pp. 556–560, 2023.
- [180] W. Wang, Z. Tian, C. Zhang, A. Liu, and S. Yu, “BFU: Bayesian Federated Unlearning with Parameter Self-Sharing,” in *Proceedings of the ACM Asia Conference on Computer and Communications Security*, ASIA CCS’23, (New York, NY, USA), p. 567–578, Association for Computing Machinery, 2023.
- [181] L. Zhang, T. Zhu, H. Zhang, P. Xiong, and W. Zhou, “FedRecovery: Differentially Private Machine Unlearning for Federated Learning Frameworks,” *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4732–4746, 2023.
- [182] Y. Liu, Z. Ma, Y. Yang, X. Liu, J. Ma, and K. Ren, “RevFRF: Enabling Cross-Domain Random Forest Training With Revocable Federated Learning,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3671–3685, 2022.
- [183] C. Pan, J. Sima, S. Prakash, V. Rana, and O. Milenkovic, “Machine unlearning of federated clusters,” *arXiv preprint arXiv: 2212.12345*, 2022.
- [184] Y. Tao, C.-L. Wang, M. Pan, D. Yu, X. Cheng, and D. Wang, “Communication efficient and provable federated unlearning,” *arXiv preprint arXiv: 2401.12345*, 2024.
- [185] A. Dhasade, Y. Ding, S. Guo, A.-m. Kermarrec, M. De Vos, and L. Wu, “Quick-Drop: Efficient Federated Unlearning by Integrated Dataset Distillation,” *arXiv preprint arXiv: 2312.12345*, 2023.
- [186] P. Wang, Z. Yan, M. S. Obaidat, Z. Yuan, L. Yang, J. Zhang, Z. Wei, and Q. Zhang, “Edge Caching with Federated Unlearning for Low-latency V2X Communications,” *IEEE Communications Magazine*, pp. 1–7, 2023.
- [187] R. Jin, M. Chen, Q. Zhang, and X. Li, “Forgettable Federated Linear Learning with Certified Data Removal,” *arXiv preprint arXiv: 2312.12345*, 2023.
- [188] H. Xia, S. Xu, J. Pei, R. Zhang, Z. Yu, W. Zou, L. Wang, and C. Liu, “FedME2: Memory Evaluation & Erase Promoting Federated Unlearning in DTMN,” *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 11, pp. 3573–3588, 2023.
- [189] Z. Xiong, W. Li, Y. Li, and Z. Cai, “Exact-Fun: An Exact and Efficient Federated Unlearning Approach,”

Bibliography

- [190] T. Che, Y. Zhou, Z. Zhang, L. Lyu, J. Liu, D. Yan, D. Dou, and J. Huan, “Fast federated machine unlearning with nonlinear functional theory,” in *Proceedings of the 37th International Conference on Machine Learning* (A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds.), vol. 202 of *Proceedings of Machine Learning Research*, pp. 4241–4268, PMLR, 23–29 Jul 2023.
- [191] X. Zhu, G. Li, and W. Hu, “Heterogeneous Federated Knowledge Graph Embedding Learning and Unlearning,” in *Proceedings of the ACM Web Conference* , WWW ’23, (New York, NY, USA), p. 2444–2454, Association for Computing Machinery, 2023.
- [192] J. Nocedal, “Updating quasi-newton matrices with limited storage,” *Mathematics of Computation*, vol. 35, no. 151, pp. 773–782, 1980.
- [193] A. Mora, I. Tenison, P. Bellavista, and I. Rish, “Knowledge distillation for federated learning: a practical guide,” *arXiv preprint arXiv:* , 2022.
- [194] A. Hoecker and V. Kartvelishvili, “Svd approach to data unfolding,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 372, no. 3, pp. 469–481, 1996.
- [195] Q. Liu and D. Wang, “Stein variational gradient descent: A general purpose bayesian inference algorithm,” *Advances in neural information processing systems*, vol. 29, 2016.
- [196] R. Kassab and O. Simeone, “Federated generalized bayesian learning via distributed stein variational gradient descent,” *IEEE Transactions on Signal Processing*, vol. 70, pp. 2180–2192, 2022.
- [197] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:* , 2014.
- [198] J. Geng, Z. Chen, Y. Wang, H. Woisetschlaeger, S. Schimmler, R. Mayer, Z. Zhao, and C. Rong, “A Survey on Dataset Distillation: Approaches, Applications and Future Directions,” *arXiv preprint arXiv:* , 2023.
- [199] Z. Zhang, Y. Zhou, X. Zhao, T. Che, and L. Lyu, “Prompt certified machine unlearning with randomized gradient smoothing and quantization,” in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 13433–13455, Curran Associates, Inc., 2022.

- [200] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, “A Survey on Knowledge Graphs: Representation, Acquisition, and Applications,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, 2022.
- [201] H. Peng, H. Li, Y. Song, V. Zheng, and J. Li, “Differentially Private Federated Knowledge Graphs Embedding,” in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM ’21*, (New York, NY, USA), p. 1416–1425, Association for Computing Machinery, 2021.
- [202] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, “Soft filter pruning for accelerating deep convolutional neural networks,” *arXiv preprint arXiv: 1808.05868*, 2018.
- [203] L. Cai, Z. An, C. Yang, and Y. Xu, “Softer pruning, incremental regularization,” in *31st international conference on pattern recognition (ICPR)*, pp. 224–230, IEEE, 2021.
- [204] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.
- [205] X. Gao, X. Ma, J. Wang, Y. Sun, B. Li, S. Ji, P. Cheng, and J. Chen, “Verifi: Towards verifiable federated unlearning,” *IEEE Transactions on Dependable and Secure Computing*, 2024.
- [206] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, “Nisp: Pruning networks using neuron importance score propagation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9194–9203, 2018.
- [207] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, “Importance estimation for neural network pruning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11264–11272, 2019.
- [208] W. Gill, A. Anwar, and M. A. Gulzar, “Provfl: Client-driven interpretability of global model predictions in federated learning,” *arXiv preprint arXiv: 2305.12345*, 2023.
- [209] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, “Lfighter: Defending against the label-flipping attack in federated learning,” *Neural Networks*, vol. 170, pp. 111–126, 2024.
- [210] Y. LeCun, J. Denker, and S. Solla, “Optimal brain damage,” in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 2, Morgan-Kaufmann, 1989.

Bibliography

- [211] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” *Advances in neural information processing systems*, vol. 28, 2015.
- [212] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *arXiv preprint arXiv:1803.03069*, 2018.
- [213] T. Wu, X. Li, D. Zhou, N. Li, and J. Shi, “Differential evolution based layer-wise weight pruning for compressing deep neural networks,” *Sensors*, vol. 21, no. 3, p. 880, 2021.
- [214] J. Liu, P. Ram, Y. Yao, G. Liu, Y. Liu, P. SHARMA, S. Liu, *et al.*, “Model sparsity can simplify machine unlearning,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [215] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, “Data poisoning attacks against federated learning systems,” in *Computer security–ESORICS 2020: 15th European symposium on research in computer security, ESORICS 2020, guildford, UK, September 14–18, 2020, proceedings, part I*, pp. 480–501, Springer, 2020.
- [216] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [217] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.