SAARLAND UNIVERSITY

Faculty of Mathematics and Computer Science
Department of Computer Science
MASTER THESIS

DECENTRALISED AI: A BLOCKCHAIN-BASED FEDERATED LEARNING IMPLEMENTATION

Submitted by
VÍCTOR MARTÍNEZ PALOMARES
Saarbrücken
May 2024

Advisors:
Vabuk Pahari and Johnnatan Messias
Max Planck Institute for Software Systems
Saarland Informatics Campus
Saarbrücken, Germany

Supervisor and First Examiner: Prof. Dr. Krishna P. Gummadi
Second Examiner: Dr. Yang Zhang

Saarland University
Faculty MI – Mathematics and Computer Science
Department of Computer Science
Campus – Building E1.1
66123 Saarbrücken
Germany

Erklärungen

Erklärung an Eides statt:

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen andere Medien oder Materialien als die in dieser Arbeit genannten benutzt habe.

Einverständniserklärung:

Ich erkläre mich damit einverstanden, dass beide Versionen meiner Masterarbeit (mit bestandener Note) in der Bibliothek des Fachbereichs Informatik öffentlich zugänglich gemacht werden.

Konformitätserklärung der elektronischen und gedruckten Version:

Hiermit bestätige ich die inhaltliche Übereinstimmung der gedruckten Version und der elektronischen Version der Arbeit.

Saarbrücken, den 21. Mai 2024 Víctor Martínez Palomares

Declarations

Statement in Lieu of an Oath:

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Declaration of Consent:

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Declaration of Conformity of Electronic and Printed Version:

I hereby confirm the congruence of the contents of the printed data and the electronic version of the thesis.

Saarbrücken, 21 May 2024 Víctor Martínez Palomares

Acknowledgements

I would like to express my gratitude to Prof. Dr. Krishna P. Gummadi for offering me the opportunity to join the Max Planck Institute for Software Systems as a researcher and write my master's thesis under his supervision. I extend my gratitude to my advisors, Johnnatan Messias and Vabuk Pahari, whose seminar on Blockchains and Decentralized Finance (DeFi) deeply captivated me and made me come up with the idea for this thesis.

I am also grateful to Dr. Yang Zhang for his interest in my work and for taking the time to review my thesis.

I would also like to thank my close friends Pablo Valdunciel and Blanca Ruiz for reviewing this work and making sure it was up to their high standards.

Additionally, I must thank my girlfriend Maitena — who has become an expert on the topic simply by putting up with my rants — for her unconditional support and thorough review of this text.

And last but not least, I could not forget to express my heartfelt gratitude to my parents.

Abstract

In the evolving field of Artificial Intelligence, Federated Learning emerged as a novel approach designed to train Machine Learning models in a distributed fashion without the need to exchange data among the different participating clients. This approach significantly facilitated collective collaboration without compromising data privacy by enabling participants to maintain their sensitive data locally and train individual models that were subsequently aggregated in a central server. Despite the advancements made by FL in addressing data privacy through its distributed nature, challenges associated with centralisation still persist.

In recent years, Blockchain-Based Federated Learning has emerged as a promising intersection of FL and blockchain technology. By utilising smart contracts, BCFL introduces security and transparency, and successfully mitigates centralisation concerns, such as single point of failure, availability attacks and dishonest behaviour. Nevertheless, the intrinsic complexities and limitations of blockchain technology, such as reduced transaction throughput and arithmetic constraints imposed by strict consensus mechanisms, pose challenges for implementing advanced Deep Learning models like transformers for Natural Language Processing.

In this thesis, we bridge these gaps by providing an open-source implementation of a BCFL framework based on transformer models together with a Hyperledger Fabric permissioned blockchain network, offering a decentralised and distributed solution to NLP tasks that does not compromise either performance or data confidentiality. By using lightweight versions of the BERT model and selectively freezing model layers, the framework significantly reduces the computational and storage demands on the blockchain, enabling efficient and precise on-chain aggregation. An extensive evaluation of binary text classification tasks demonstrates that our BCFL framework is able to achieve equivalent performance to its centralised counterparts. This achievement underscores the potential of BCFL to enable advanced DL tasks while retaining the advantages of decentralisation.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AI Artificial Intelligence. 2, 8

ANN Artificial Neural Network. ix, 2, 9, 23, 36, 38

BCFL Blockchain-Based Federated Learning. x, 4–7, 17–22, 24, 26–28, 32, 33, 35, 36, 39, 42, 44–48, 51, 52, 54–58, 60–66, 68–74

BERT Bidirectional Encoder Representations from Transformers. ix, 23, 24, 26, 39–42, 45–48, 51, 53, 55–66, 68–72

CA Certificate Authority. 47

CDP Centralised Differential Privacy. 13, 25

CNN Convolutional Neural Network. 23, 38

DApp decentralised application. 15

DL Deep Learning. ix, 2–5, 7–9, 17, 23, 24, 26, 28, 29, 32, 33, 35, 36, 40, 42, 46, 48, 51–55, 57, 58, 60–67, 71–74

DLG Deep Leakage from Gradients. 5

DoS Denial-of-Service. 4, 17, 34

DP Differential Privacy. x, 5, 12, 13, 24, 25, 28, 42, 46, 48, 51–53, 55, 58, 63–67, 71, 73, 74

FedAvg Federated Averaging. xi, 10–13, 29, 44, 67–69, 71, 73, 74

FedProx Federated Proximal. xi, 11–13, 29, 44, 67–69, 71, 73, 74

FHE Fully Homomorphic Encryption. 13, 14

FL Federated Learning. ix, 3–15, 17–20, 22–26, 28, 29, 31–38, 42–49, 51, 54–58, 60–67, 69, 71–74

FTL Federated Transfer Learning. 11, 36, 73

GDPR European General Data Protection Regulation. 3, 72

GPT Generative Pre-trained Transformer. ix, 23, 39

GPU Graphics Processing Unit. 2, 52, 58

GRU Gated Recurrent Unit. 23, 38

HE Homomorphic Encryption. 5, 12–14, 24–26, 42, 74

HFL Horizontal Federated Learning. 11, 28, 44

HIPAA Health Insurance Portability and Accountability Act. 3, 72

IBFT Istanbul Byzantine Fault Tolerance. 16, 17

IID independent and identically distributed. 11, 12, 44, 46, 51, 55, 67, 68, 71, 73, 74

IoT Internet of Things. 4, 10, 23, 38, 72

LDP Local Differential Privacy. 9, 13, 25, 26, 29, 32, 42

LLM large language model. 2

LSTM Long Short-Term Memory. 23, 24, 26

ML Machine Learning. 2, 3, 7–9, 17, 19, 20, 23, 35, 46, 48, 55, 57, 59, 64, 65, 72, 73

NLP Natural Language Processing. ix, 3–7, 19, 23, 24, 26, 35, 38, 46, 70, 72

P2P peer-to-peer. 14, 15, 19

PBFT Practical Byzantine Fault Tolerance. 16, 22

PHE Partially Homomorphic Encryption. 13

PoS Proof of Stake. 16

PoW Proof of Work. 16, 20, 22, 56

PTSD Post-Traumatic Stress Disorder. 35, 37, 68

QSCC Query System Chaincode. 42

RNN Recurrent Neural Network. ix, 23, 25, 26, 38

SDK Software Development Kit. 47, 49

SMC Secure Multiparty Computation. 12, 42

SPOF Single Point of Failure. 4, 17, 22, 26, 34, 56, 73

SWHE Somewhat Homomorphic Encryption. 13, 14

TLS Transport Layer Security. 47

VFL Vertical Federated Learning. 11

ZKP Zero-Knowledge Proof. 74

Chapter 1
Introduction

1.1 Motivation

Recent advancements in Artificial Intelligence (AI) are largely driven by the increase in computational power and data availability. Some experts refer to it as a data-driven revolution [23], where big efforts have been made to build high-quality datasets to take advantage of this growth. Machine Learning (ML) is the branch of AI that is capitalising the protagonism of this revolution. ML builds systems that learn and draw inferences from the input data by analysing and identifying patterns. Its subfield, Deep Learning (DL), defines complex models called Artificial Neural Networks (ANNs), which are layers of interconnected nodes simulating the human brain's processing capability. These multi-layered structures together with its non-linear processing capability have enabled the modelling of complex patterns in large datasets. Consequently, they are behind most of the advances in the field of ML in the recent years and considered among the most effective AI techniques [11, 22, 39].

The improvement of these ANNs mostly relies on the increase in its number of parameters, often reaching an amount of millions or even billions. Traditionally, computational power has increased following Moore's Law, which predicts exponential growth in computing capabilities by doubling transistors on a microchip about every two years. However, the scaling of Moore's Law has recently slowed down significantly [24]. At the same time, the parameters of cutting-edge models, such as the popular large language models (LLMs), have continued to surge at a remarkable pace of tenfold each year (refer to Figure 1.1). This is largely thanks to advances in efficient algorithms, distributed computing and specialised hardware — e.g., Graphics Processing Unit (GPU) — which have facilitated training increasingly complex models. As these models get more and

more complex, the amount of data required to train and fine-tune them also rises. This leads to a massive increase in the demand for data availability and quality.

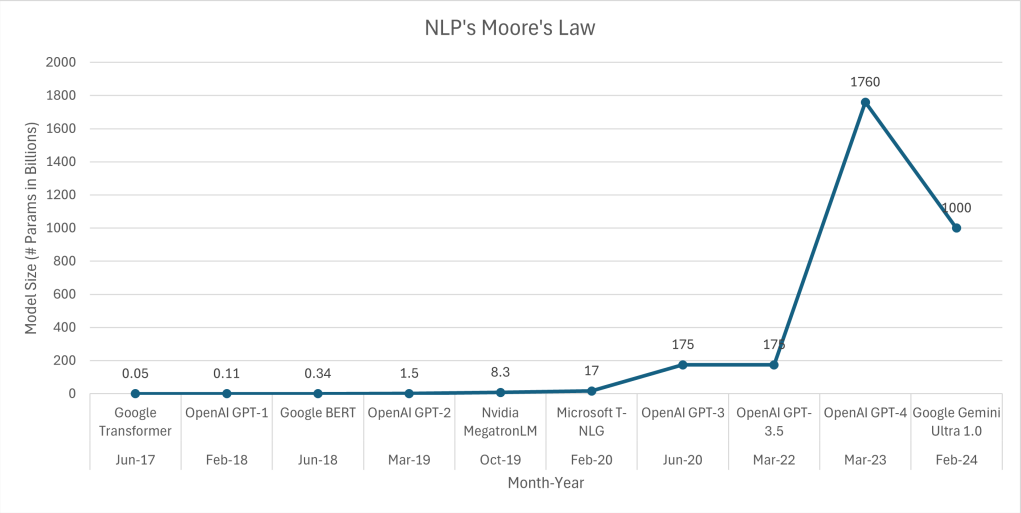

Figure 1.1: Evolution of Natural Language Processing (NLP) model's size parameters

Sectors such as healthcare, telecommunications and finance run the risk of falling behind this revolution as the gathering of such large amounts of data poses challenges in terms of data protection regulations. The European General Data Protection Regulation (GDPR) enforces strict rules on data processing in the context of healthcare and finance sectors [19]. Similarly, in the USA the Health Insurance Portability and Accountability Act (HIPAA) is key in the healthcare sector [54]. It establishes a standard for protecting sensitive patient data and restricts the use of such data without explicit patient consent.

The aforementioned regulations hinder the utilisation of large DL models in such important areas in society as healthcare and finance [73, 70]. This has led to a necessary adaptation to find solutions in the field of ML and largely explains the emergence of Federated Learning (FL), a type of ML proposed by Google in 2016 [53], which builds ML models in a distributed manner [42] motivated by the preservation of data privacy during training.

Unlike traditional ML and DL approaches, which often rely on a centralised data pool, FL enables the participants, i.e., data servers and edge devices, to maintain their own data in their system and create their own local model. These local models are sent to the central server instead of their private data. This central server is responsible for aggregating the local models into a global one. This methodology has attracted interest in many sectors, including finance, retail, telecommunications and healthcare [91, 86, 66, 85, 2], as it provides the benefits of collective collaboration without compromising data privacy and sharing policies. In many of these applications, given the sensitivity of the data being handled, FL is not only an interesting approach, but a necessary one.

Despite its aim of solving data privacy thanks to its distributed nature, FL does indeed

come with some caveats related to centralisation. Some challenges that are not directly addressed by FL may arise, such as Single Point of Failure (SPOF), Denial-of-Service (DoS), inference attacks [32], as well as malicious and lazy behaviour [8]. In such highly sensitive data contexts, these challenges need to be addressed before a FL framework can properly be used. Blockchain technology adds an additional layer of security and transparency, successfully addressing theses challenges [91, 41, 32].

Blockchain-Based Federated Learning (BCFL) is the name given to the framework where FL is enhanced with blockchain. By using smart contracts, this framework allows for the aggregation of the local models trained by FL to take place in a decentralised manner, ensuring trust between the participating parties. Different types of blockchain, depending on their governance, are better suited to different topologies of FL. There is also a further distinction in how the BCFL framework is configured: depending on what role the different nodes of the BCFL system play, the system can be classified as a coupled, decoupled or overlapped model, which has an impact on its operational and resource requirements [92, 79].

In a coupled configuration, all nodes perform both FL and blockchain operations. On the other hand, the decoupled configuration assigns different nodes to handle model training and blockchain management separately. Finally, the overlapped configuration distinguishes between client nodes responsible for FL and blockchain nodes managing transactions, with composite nodes serving dual functions as both a blockchain node and a FL client.

FL also poses some additional challenges related to its computational limitations, especially in its application within cross-device topologies, those involving Internet of Things (IoT) devices such as smartphones and wearables. The inherent computational limitations of such devices hinder the implementation of advanced, complex and high-computational demanding DL models, such as transformers for Natural Language Processing (NLP), as the backbone of the FL frameworks [32].

When it comes to BCFL, the inherent complexities and limitations of blockchain technology also pose challenges for implementing advanced FL frameworks for NLP. Reduced transaction throughput and arithmetic constraints imposed by strict consensus mechanisms are some of its intrinsic limitations that make the implementation of sophisticated DL models such as transformers hard and thus complicate the utilisation of powerful decentralised FL systems for complex NLP tasks.

Permissioned blockchains fit as better solutions to these challenges [79, 32, 57] by providing a semi-decentralised framework with more adaptable consensus mechanisms. This setup allows for better performance in the aggregation of the DL local models, addressing issues present in conventional permissionless blockchain systems [62]. Nevertheless, optimising the configuration of permissioned blockchains is essential to strike a balance between the benefits of decentralisation and the operational efficiency needed for FL implementations.

Additionally, it is important to consider that since the local model updates are registered

in the blockchain, there is still a need for additional privacy measures to prevent any participant in the network from drawing inferences from the local model updates [32, 60] by using Deep Leakage from Gradients (DLG) attacks [84]. Therefore, advanced privacy-enhancing techniques, such as Differential Privacy (DP) [16] or Homomorphic Encryption (HE) [21], become imperative. These methods protect against inference attacks, thus ensuring data confidentiality. However, it is important to note that all privacy-enhancing strategies have an impact either on model performance or computational resources, so a careful evaluation is required to achieve an optimal balance.

1.2 Research Questions

The aim of this thesis is to achieve a deeper understanding of the intersection of FL and blockchain technology and to develop a feasible and functional design and implementation prototype of a BCFL framework, with the objective of investigating the following research questions:

- *RQ1*: How to design a modularised, decentralised FL framework enhanced by blockchain technology to effectively address the centralisation challenges inherent to FL?
- *RQ2*: What is the viability of using transformer-based models as the backbone of a BCFL for NLP tasks?
- *RQ3*: How does a decentralised cross-silo BCFL framework compare in performance with equivalent centralised DL and centralised FL frameworks?
- *RQ4*: What is the effect of the degrees of privacy and the data distribution among clients on the performance of the model in different datasets?

1.2.1 Why? Or Rationale of the Research Questions

Prior research in the field of FL has primarily focused on designing solutions to cross-device topologies, with a particular emphasis on addressing specific security and scalability concerns [26]. However, there is a notable gap in the availability of functional implementations that offer configurable parameters that may be adaptive to diversified operational contexts, especially aimed at cross-silo scenarios, in which the client is characterised by substantial resources.

Similarly, the decentralisation solutions for FL in NLP tasks remain limited. Schumann et al. [68] have thoroughly reviewed the FL applied to NLP literature and reported that out of the seventy-six reviewed studies, only two employed decentralised architectures, with none of them incorporating any blockchain technology enhancements.

There is a marked lack of standardisation in the evaluation of existing implementations in the research. These studies vary significantly, ranging from the use of permissioned

to permissionless blockchain systems. Moreover, not all of these studies provide a comparison of the performance of their framework against centralised FL models, nor do they provide open-source code for community use, with ScaleSFL [50] being a notable exception. To the best of our knowledge, the research that compares the performance of their frameworks — specifically examining the effects of varying configurations, such as degrees of privacy — uses mainly just a single dataset for such evaluations [7]. This approach restricts the generality of findings and provides insights that may not be broadly applicable across different contexts or datasets.

1.2.2 How? Or Research Methodology

For a detailed exploration of RQ1, a comprehensive literature review is conducted to inform the design of a sophisticated, yet functional, decentralised FL framework enhanced with blockchain technology. This framework seeks to solve existing security challenges identified in contemporary studies. Our work places particular emphasis on constructing a modular system that is not only theoretically sound, but also practically viable by taking advantage of current technological advances. The system is, therefore, capable of facilitating trustless collaboration across a spectrum of operational contexts, including coupled, decoupled and overlapped configurations.

RQ2 involves the development of an FL model using the transformer architecture, carefully adjusting the trainable layers and the configuration of the blockchain network. Such adjustments ensure that the aggregation phase can be performed on-chain, in a fully decentralised manner, taking advantage of the performance of transformer models within this innovative framework.

As for RQ3 and RQ4, an extensive ablation study is conducted in multiple phases, each one targeting different aspects of the framework. Experiments are performed on four different NLP datasets to assess performance across different model architectures and hyperparameter settings. This analysis is extended to examine how varying degrees of privacy and variations in data distribution among clients impact the results.

1.3 Contributions

We summarise the contributions of our work as follows:

- We provide an open-source implementation of an entire BCFL framework that can be used and deployed on any permissioned blockchain network that uses general-purpose programming languages.
- We develop our BCFL framework with advanced transformer-based models at its core, offering a novel decentralised and privacy-preserving approach to NLP tasks that achieves a blend of high-end performance and data confidentiality.

- We analyse the empirical performance of the framework on binary text classification over different datasets for various settings, such as different model architectures, differential privacy degrees and data distribution between clients.
- We compare performance and execution time of our BCFL framework against equivalent centralised DL models and centralised FL frameworks with successful results, demonstrating that our BCFL framework can ensure decentralisation control and trustless collaboration without negatively impacting performance.

1.4 Outline of the Thesis

The thesis is structured as follows: the next chapter (Chapter 2), sets out to establish a comprehensive understanding of the fundamental components of our work, namely ML, FL, blockchain technology and BCFL. Subsequently, in Chapter 3, we review the current state of research on BCFL, collecting takeaways from the most comprehensive surveys to the most specific implementations in NLP. Chapter 4 details the design of all the different components and requirements and features of our framework. The technologies used for the implementation of the framework are discussed in Chapter 5, paying special attention to how the different components are integrated to succesfully interact with each other. After the entire framework is built and executed, we perform an ablation study to evaluate it in detail. We discuss the results in Chapter 6, analysing its performance and comparing it to other of centralised ML and FL approaches in order to find an answer to our research questions. In the final chapter of this thesis (Chapter 7), we summarise and draw conclusions from our work, highlighting our inherent limitations and suggesting further lines of work that could leverage our contributions to BCFL, which can be freely accessed through our public repository[1].

[1] https://github.com/vdevictor96/fabric-federated-learning

Chapter 2
Background

In this chapter, we provide a summary of the core concepts and technologies relevant to this thesis. We recommend the reader with a strong background in ML skipping the first section. Likewise, we advise the reader with strong background in blockchain to skip the section dedicated to it.

2.1 Machine Learning

Understanding the mechanics of FL requires a basic knowledge of the principles of ML. At its core, ML is a subset of AI where systems learn to derive insights from data, recognise patterns and make decisions autonomously. Some of the tasks that fall under ML include, but are not limited to, email filtering, speech recognition and image recognition.

While ML can be classified into several types depending on the nature of the input data and the task at hand — supervised, semi-supervised, unsupervised and reinforcement learning [22] — our study focuses exclusively on supervised learning. In supervised learning, the training data consists of pairs containing an input object, often represented as a vector, and a corresponding desired output value, also known as the label. The primary task of the algorithm is to learn an inference function that maps new, unseen data inputs to their correct outputs.

2.1.1 Deep Learning

DL represents a subset of ML methodologies, unique in its ability to process data by using complex networks inspired by the structure and function of the human brain, so-called

ANNs. This approach enables the modelling of complex and non-linear patterns within immense datasets, significantly enhancing the performance of machine learning tasks [22, 39] in cases where there is a large amount of data.

These ANNs consist of interconnected layers of nodes — neurons. Each layer is designed to transform its input data into a more abstract representation, gradually leading to the final output that corresponds to the result of complex pattern recognition or prediction tasks. Figure 2.1 depicts a simple architecture of a neural network. Essentially, these networks are structured with an input layer, several hidden layers and an output layer. The term *deep* learning stems from the depth of these networks, allowing the extraction of complex features from raw data, eliminating the necessity for manual feature engineering.

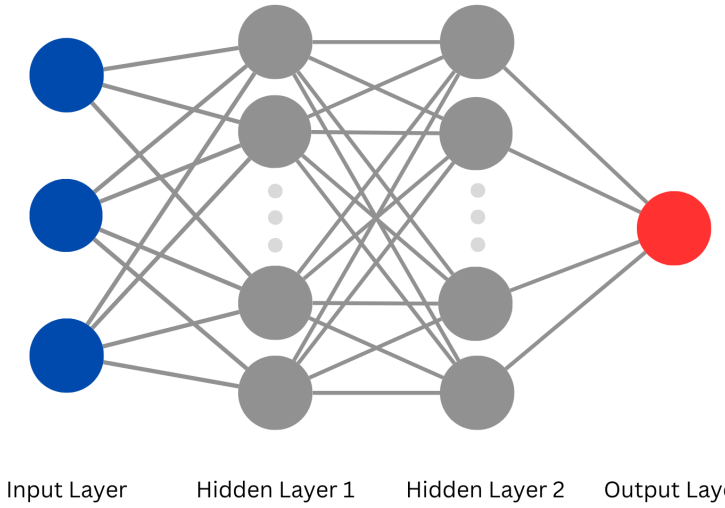

Figure 2.1: Architecture of a simple Artificial Neural Network (ANN) used by Deep Learning (DL) methodologies

2.2 Federated Learning

FL represents a ML paradigm shift that was introduced by Google in 2016 [53] as a novel approach designed to train algorithms across many multiple distributed servers and/or devices without the need to share their data among themselves. This approach not only enhances privacy and security, but also leverages distributed data sources efficiently.

During the FL workflow (see Figure 2.2), clients are first selected based on specific criteria to participate in the training phase. After downloading the unified global model, each of the chosen clients starts training this model locally on its own dataset. Upon completion of local model training, an optional privacy preserving step, such as Local Differential Privacy (LDP), is introduced [32]. Clients can use LDP on their model

updates to safeguard against inference attacks. These secured updates are then sent to a central aggregator server. Finally, the central server computes the global model by executing the aggregation algorithm — e.g., Federated Averaging (FedAvg) [79].

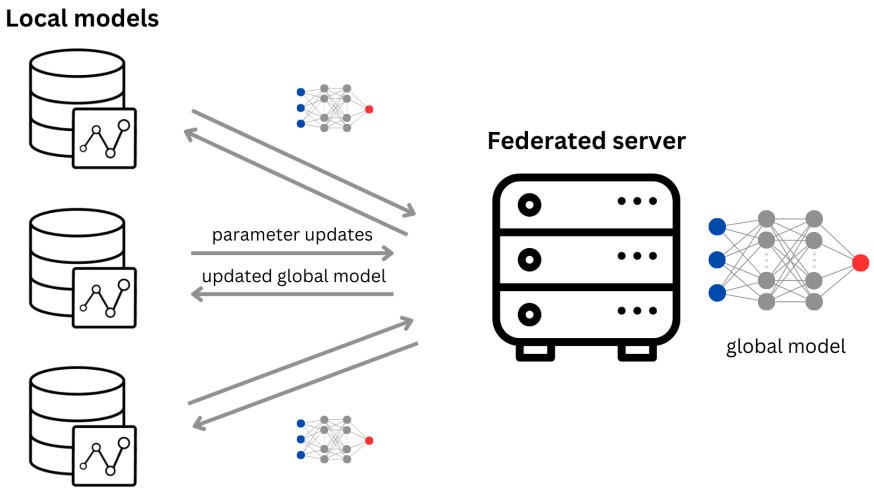

Figure 2.2: Federated Learning (FL) workflow

2.2.1 Categories of Federated Learning

FL can be divided into different forms based on critical aspects such as the network topology [28, 66] or how data is distributed among the participants [86]. Each form of FL presents its own unique challenges and benefits. In the following subsections, we delve into the specifics of each FL architecture variant.

By Topology

FL systems can be classified in terms of their topology into two main types: cross-silo and cross-device FL. This classification depends on factors such as the scale of the participants, the nature of their data contributions and the trust assumptions that can be made. These characteristics of the participating nodes have a significant impact on the design and implementation of the most appropriate FL systems.

- *Cross-silo FL* involves a relatively small number of participants, usually organisations with large datasets. These organisations participate in all training rounds. This setup is characterised by reliable and powerful computing resources compared to cross-device scenarios.
- *Cross-device FL* is defined by its vast number of participants, each contributing a small amount of data from personal devices like smartphones or any type of IoT

device. These devices may not be able to participate in all training rounds. This setup presents clear challenges, including dealing with highly distributed data, unreliable network connections, limited computing power on devices and low levels of trust between participants.

By Data Partition

- *Horizontal Federated Learning (HFL)* is characterised by the collaboration between participants that share different samples within a similar feature space. For example, two healthcare centres in different cities or even countries may have data with the same features (e.g., medical history, diagnosis, treatment outcomes) but different patient groups (e.g., age, wealth, ethnic group). HFL allows these centres to train a model together without directly sharing their patients' data.
- In *Vertical Federated Learning (VFL)* the participants contribute different feature space of the same samples. This scenario can also be easily applied to healthcare, where different centres may have complementary information about the same patient, for example, different medical specialities. VFL allows them to co-develop models, resulting in a more extended and richer feature space without compromising user privacy.
- *Federated Transfer Learning (FTL)* refers to the ability to apply knowledge gained from one domain to another [86]. It is particularly useful in scenarios where direct collaboration between participants is not possible or where the domains are related but not identical. FTL becomes particularly helpful in mitigating the challenges associated with non-independent and identically distributed (IID) data across different participants.

2.2.2 Federated Aggregation Algorithms

We find many algorithms in the literature [66, 78, 36] aimed at the aggregation of the local models trained on the federated network. We briefly discuss two of the most important ones, which are the subject of study in this thesis, namely Federated Averaging (FedAvg) and Federated Proximal (FedProx).

- *Federated Averaging (FedAvg)*, being Google's original algorithm for designing FL systems [53], stands as the foundational algorithm within this learning paradigm. As described in Algorithm 1, local models are trained on the participants' infrastructure, be it devices or data servers. Instead of sharing their raw data, only the model updates are transmitted to a central server. This server aggregates these updates by simple averaging to improve the global model. The improved model is then distributed back to the participants for further training.

- *Federated Proximal (FedProx)* in Algorithm 2 extends FedAvg by introducing a proximal term to the optimisation problem [43], which helps addressing issues related to the heterogeneity of the data distribution across participants (non-IID data).

Algorithm 1: Federated Averaging (FedAvg)

Input: The number of rounds T, client fraction C, total number of clients K, local batch size B, number of local epochs E, learning rate η

Output: Global model weights w

Initialise global model weights w_0
for *each round* $t = 1, 2, \ldots, T$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ random set of m clients
    foreach *client* $k \in S_t$ ***in parallel*** **do**
        $w_{t+1}^k \leftarrow$ **ClientUpdate**(k, w_t)
    $w_{t+1} \leftarrow \sum_{k=1}^{m} \frac{n_k}{n} w_{t+1}^k$
Function ***ClientUpdate***(k, w)**:**
    Split local data into batches of size B
    for *each local epoch* i *from 1 to* E **do**
        foreach *batch* b **do**
            $w \leftarrow w - \eta \nabla \ell(w; b)$
    return w

2.3 Privacy

In the FL framework, clients' data remains on their servers or devices, thus ensuring its confidentiality by only exchanging model parameters with the central server for global model aggregation. However, these model weights are still vulnerable to inference attacks [93], potentially leaking sensitive information, contrary to the goal of FL. The mentioned vulnerability demands the use of appropriate privacy-preserving methods within FL algorithms. Privacy and security in a FL framework cannot be wholly guaranteed without the use of privacy techniques like Differential Privacy (DP), Homomorphic Encryption (HE) or Secure Multiparty Computation (SMC). In this section, we describe the two principal approaches to achieving this goal: DP and HE.

2.3.1 Differential Privacy

DP establishes a mathematical framework that the algorithms follow to preserve a specified degree of privacy. It guarantees that the data remains adequately ambiguous,

Algorithm 2: Federated Proximal (FedProx)

Function ***ClientUpdate***(k, w, μ)**:**

```
// The ClientUpdate function in FedProx includes a proximal
   term μ compared to FedAvg
```

Split local data into batches of size B

for *each local epoch* i *from 1 to* E **do**

  foreach *batch* b **do**

    $w \leftarrow w - \eta(\nabla \ell(w; b) + \frac{\mu}{2}\|w - w_0\|_2^2)$

return w

```
// The rest of the FedProx algorithm structure is identical
   to FedAvg, including initialisation, global aggregation
   and distribution phases.
```

thereby preventing the extraction of identifiable information regarding its origin [16].

In the specific case of FL training, DP is applied through adding computed noise to the local model updates. Common mechanisms include the Laplace, the Gaussian and the exponential. Each of the mechanisms specifically deals with different types of data and satisfies some privacy requirements. DP's versatility extends to settings like Centralised Differential Privacy (CDP), where the noise is added by a central authority or directly in the central server, and Local Differential Privacy (LDP), where the noise is added in the client before data collection, offering stronger privacy guarantees. The effectiveness of LDP is quantified using a parameter ϵ (epsilon), where lower values correspond to more robust privacy guarantees. The formula that encapsulates this criterion is as follows:

Given any two datasets D and D' that differ by at most one element, a randomised algorithm A satisfies ϵ-differential privacy if for all subsets S of A's output range,

$$\Pr[A(D) \in S] \leq e^{\epsilon} \times \Pr[A(D') \in S] \tag{2.1}$$

2.3.2 Homomorphic Encryption

HE is a cryptographic technique that enables computations on encrypted data without having to decrypt it first [86, 89, 6]. This means that operations performed on the encrypted data, when decrypted, yield the same result as if the operations had been performed on the original data. This property allows the local models of the FL system to be aggregated in their encrypted form, thereby ensuring that the central server does not gain any inferential knowledge of the data.

HE can be classified into Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWHE) and Fully Homomorphic Encryption (FHE). PHE supports an unlimited number of operations of a single type — addition or multiplication — while

SWHE allows for the execution of a limited number of operations of both types. FHE, an innovative development achieved by Gentry in 2009 [21], allows for an unlimited number of both addition and multiplication operations on encrypted data.

Despite its interesting advantages, a major drawback of implementing HE is the significant increase in the computational cost and the size of the encrypted objects. Performing operations on encrypted data is inherently more complicated and demands more resources than dealing with unencrypted plain data [36].

2.4 Blockchain

This section introduces the core components of blockchain technology, with a focus on smart contracts, blockchain governance and consensus mechanisms. These components are critical to understanding the potential of blockchain to enhance FL systems, by ensuring data integrity, maintaining privacy and facilitating secure collaboration across distributed networks.

A blockchain is a decentralised distributed ledger that records transactions in a secure, transparent and immutable manner [30]. It utilises cryptographic hashes to interconnect a series of data blocks, each of which contains transactions (see Figure 2.3). Transactions on the blockchain contain data and are also capable of triggering smart contract executions. Once a block is added to the chain, it cannot be altered without changing all the subsequent ones, thereby preserving the integrity of the transaction history.

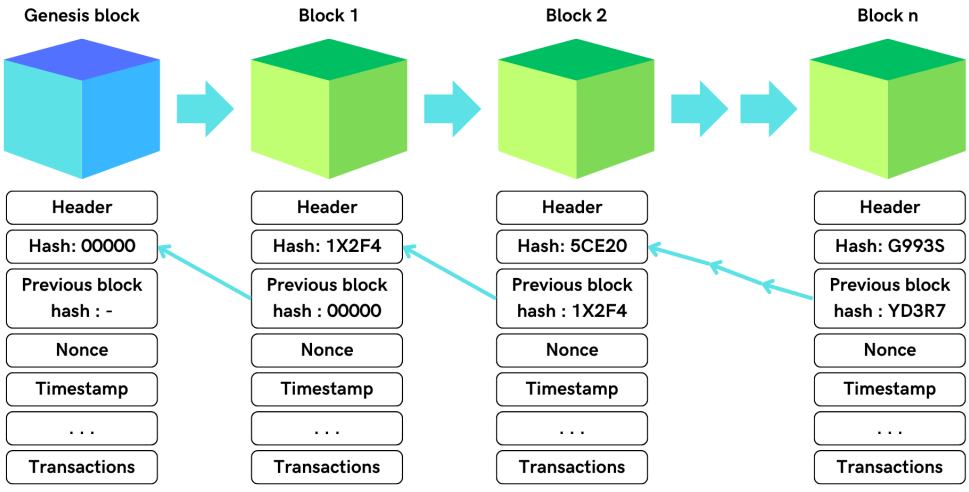

Figure 2.3: Blockchain design structure

Nodes in the blockchain are connected to each other via a peer-to-peer (P2P) network operating without the need for a central authority. Each node holds a complete version

of the ledger and participates in the consensus process to agree on the addition of new blocks. This method of consensus enables trustless cooperation between different participants, even among parties that may not necessarily know or trust each other.

The structure of the blockchain consists of several layers (see Figure 2.4), each serving a specific purpose within the system [5]. The data layer handles the storage of transaction data in the ledger. The network layer manages the P2P network so that nodes can efficiently communicate and exchange data with each other. The consensus layer enables nodes to reach an agreement on the ledger's current state without central oversight. Furthermore, the incentive layer motivates network participants, often referred to as *miners*, by rewarding them for their computational contributions to the network's maintenance. In addition, the contract layer embeds smart contracts that introduce business logic into the blockchain framework. Lastly, the application layer expands blockchain functionality to a wide range of decentralised applications (DApps).

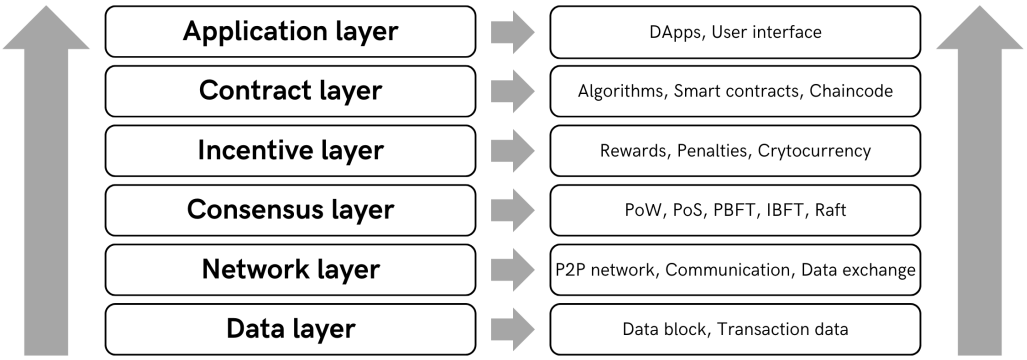

Figure 2.4: Layered blockchain architecture

2.4.1 Smart Contracts

Smart contracts are key to leveraging blockchain technology. They are pieces of software that can be triggered by the transactions or by other smart contracts when certain conditions are met. Smart contracts differ from traditional contracts in that the contractual agreement is written directly into the lines of code so there is no need for a third party to ensure compliance with the agreement. This is commonly referred to as *code is law* [12]. As they are executed within the blockchain, smart contracts can be considered immutable and distributed. Once a smart contract is deployed on the blockchain, it cannot be altered without the consensus of the network, ensuring the integrity and trustworthiness of the contract's execution. This automation has many benefits for the FL systems, as smart contracts could securely and transparently manage rules for federated aggregation of local models, data access and participant incentives.

2.4.2 Types of Blockchain Governance

Blockchain networks can be grouped based on their governance models, which have a strong influence on the use cases of the decentralised system [57], including considerations of network speed, security and transparency.

- *Permissionless blockchains* are open and decentralised networks where anyone can participate without requiring authorisation. They use consensus mechanisms that are secure, but may lead to high latency and significant energy consumption due to the need to protect the network from malicious actors.
- *Permissioned blockchains* restrict network participation to entities approved by the network's governing body. This governing body can be a group of authorities or just a single authority, making the blockchain a *Consortium Blockchain* or a *Private Blockchain*, respectively. This model offers faster transaction speeds and more efficient consensus mechanisms, making it suitable for organisational collaboration where privacy and access control are key factors.

2.4.3 Consensus Mechanisms

Consensus mechanisms are key for the integrity and security of blockchain networks. They ensure that the current state of the blockchain ledger is distributed among diverse, trustless nodes [5]. The election of a consensus mechanism has a great impact on network scalability and efficiency. We briefly describe the most commonly used consensus mechanisms in permissionless and permissioned blockchains:

- *Proof of Work (PoW)* was the first consensus mechanism introduced by Nakamoto in 2008 within the Bitcoin whitepaper [56]. This mechanism demands a significant computational effort from the participating nodes, requiring nodes to solve complex mathematical puzzles to validate transactions and create new blocks. While secure, it is energy-intensive and less scalable.
- *Proof of Stake (PoS)*, on the other hand, relies on the investment commitment of the nodes within the network, as validation and block creation are based on the *stakes* of the nodes in the network. It is more power-efficient than PoW and offers better scalability. It has been adopted over PoW by popular permissionless blockchains like Ethereum in its Ethereum 2.0 update in 2022.
- *Practical Byzantine Fault Tolerance (PBFT)* and *Istanbul Byzantine Fault Tolerance (IBFT)* are both designed for permissioned blockchains, offering high transaction throughput. IBFT improves upon PBFT by reducing the communication steps required for reaching consensus. It also ensures immediate transaction finality, preventing forks in the blockchain.

- *Raft* provides immediate finality, similar to IBFT. It has high transaction throughput at the expense of not being Byzantine fault-tolerant [15], which makes it best suited to environments with a low risk of malicious behaviour. This algorithm elects a leader to manage the transaction ordering, which simplifies the network governance and makes the mechanism more efficient. However, it can also introduce centralisation-related risks if the elected leader gains too much control over the processing of transactions.

2.5 Blockchain-Based Federated Learning

FL is a potent mechanism for training ML distributed models while protecting the privacy of the training data, but it still comes with some caveats. The centralisation nature of FL system has some inherent limitations, which can be directly addressed by incorporating blockchain technology into the system.

BCFL emerges as a promising ML paradigm that intersects FL and blockchain technology to achieve both distribution and decentralisation within the same framework [91, 41, 32]. Depending on the structure of the BCFL framework, the blockchain can be managed by the participating nodes, which can be organisations or devices, or other nodes that are agnostic to the FL training [92]. Participating nodes train their DL model locally on their own data and issue transactions to the blockchain containing their model updates. The aggregation of these local model updates into a global model is done within the blockchain network in a decentralised fashion thanks to smart contracts, in contrast to standard FL.

Below, we list the FL limitations that are addressed by BCFL:

- *Single Point of Failure (SPOF)*. If the central server in charge of the aggregation fails, the framework stops working, which is a clear centralisation vulnerability [79]. In contrast, when using a blockchain, if a node fails, the aggregation can continue to work because any of the nodes can be in charge of the process at any given time.
- *Availability attacks*. Like DoS attacks, where the same reasoning applies as for SPOF. If the central server in FL is targeted by the attackers, the system is vulnerable, whereas in decentralised BCFL there is no single target for the attackers to overwhelm [32].
- *Trust issues*. The global model is aggregated in the central server and used in the subsequent iterations on the local models, which adds opacity to the training process. In BCFL, the aggregation is done transparently via smart contracts, making it easier for the participants to trust this global model [32].
- *Scalability*. The aggregation is not centralised and all participants can be eligible to execute the aggregation in the smart contracts. This prevents overloading a single node in the network [92].

- *Malicious and lazy nodes*. Malicious nodes are participants in the system that engage in deceptive behaviour. The most obvious one is to issue malicious model updates to hinder the global model's overall performance [8]. On the other hand, lazy nodes are clients that participate in learning, but do not actually train their local model [32]. Instead, they copy the parameters of other clients, often perturbing them with artificial noise to hide their behaviour. Smart contracts can validate client contributions before using them in the global model by verifying the model accuracy on test datasets [79]. This can prevent data and model poisoning attacks.

As we have previously mentioned in Chapter 1, BCFL systems can be configured in different setups depending on their operational and resource requirements [92]. To properly understand the capabilities of the BCFL framework developed in this thesis, it is crucial to distinguish between the following types of BCFL structures, depending on how the training and miner nodes are organised:

- *Coupled configuration* includes nodes that perform both FL and blockchain operations.
- *Decoupled configuration* assigns different nodes to handle the model training and the blockchain management separately.
- *Overlapped configuration* comprises nodes responsible for FL training and other blockchain nodes managing transactions. Its peculiarity lies in the introduction of so-called *composite* nodes, which perform dual functions as both a blockchain node and a FL client.

Chapter 3
Related Work

In this chapter, we review the current state of research on BCFL as a decentralised enhancement of FL. We examine the problems that have already been addressed and those that remain unresolved. We pay special attention to decentralised FL research in the context of NLP tasks, as this is the application domain of the framework developed in this thesis.

3.1 Blockchain-Based Federated Learning

Although BCFL is a fairly new paradigm in the field of ML — its first appearance in the literature dates back to late 2018 [64, 81] — the research on the topic is growing at a rapid pace, with some remarkably fruitful outcomes. FL was a well-thought-out strategy for allowing collaborative model training whilst safeguarding the privacy of the data used in this process [42], yet it fell short of achieving a comprehensive solution, raising issues related to centralisation as noted in these extensive surveys [32, 79]. BCFL seeks to extend this framework by incorporating additional security and integrity guarantees. Notably, some research refers to this concept as *swarm learning* [80], although it is worth noting that *swarm learning* does not specifically require a blockchain-based solution, but merely a P2P decentralised one. Integrating blockchain technology together with FL is not a trivial matter and poses many significant challenges inherent to the cryptographic complexity and increased network latency associated with the introduction of blockchain into the training process.

We find very informed reviews that delve into details on the inherent problems of FL centralisation and classify the different BCFL architectures depending on the specific aspects they are trying to improve upon. Hou et al. [26] conducted a review in which

they analysed seventeen distinct approaches targeting the aforementioned concerns. Wang et al. [79] were the first to introduce coupling as an attribute for classification — fully, flexibly and loosely coupled models. Moreover, this work also specifies and discusses other key attributes of the BCFL system, such as the blockchain used, the governance type and the consensus protocol used for the aggregation of the model. Zhou et al. [91] enlist a broad spectrum of applications of both FL and BCFL, covering fields from healthcare [29] to energy prediction [67] and smart financial payments [46]. As a conclusion to this section, we provide a table (refer to Table 3.1) summarising all the work used to support it.

Kim et al. [38] worked on one of the first BCFL designs — named BlockFL — aimed at offering a more reliable solution for distributed learning, resilient to server malfunctions and providing rewards to local devices. BlockFL introduced a validation process for local model updates and rewarded participant contributions in proportion to the training sample sizes. The blockchain used a PoW consensus algorithm, which incurred in a high network latency. However, they addressed this delay by designing an end-to-end latency model that minimised the delay through adjusting the block generation rate. In subsequent research, Li et al. [41] and Ma et al. [49] proposed two BCFL frameworks, also based on public blockchains that used PoW consensus. Both frameworks were named BLADE-FL. Moreover, Li et al. further explored the impact of lazy clients in the network by adding artificial noise to the learning process, although they did not provide a direct solution to this problem. Conversely, Ma et al. proposed a signature mechanism that was resilient to noise perturbation, aimed at addressing the issue of lazy clients. This mechanism was capable of detecting plagiarised models with high detection accuracy, even when perturbed by the malicious agent to hamper detection of this behaviour.

In public blockchains, analogous to the issue with lazy clients, exists the possibility of free-riding attacks, where participants exploit the collaborative global model to their advantage without contributing to local training. Chhikara et al. [10] addressed this problem by incentivising participants with cryptocurrency, awarded proportionally to the volume of training samples they contributed. This approach motivated participants to maximise their contribution. Li et al. [44], in their framework named BFLC, mitigated malicious attacks through the implementation of a committee responsible for validating and scoring local gradient updates. This innovative committee consensus mechanism significantly reduced the power waste associated with PoW. They argued that this mechanism allowed for BFLC to achieve high efficiency while protecting the system from malicious nodes. To validate their approach, they conducted experiments comparing the accuracy of BFLC against standard FL and standalone ML training frameworks, achieving successful results with its performance being only marginally lower — by one percentage point — than the standalone baseline.

Desai et al. [13] defined their ambitious BCFL framework, BlockFLA, as hybrid because it used a private blockchain, based on Hyperledger Fabric, in conjunction with a public blockchain, specifically Ethereum. The private blockchain stored the parameter updates

and generated the global model, while the public blockchain served as a transparent platform for validating participants' actions, detecting potential cheating and orchestrating the penalties. This framework was specifically tailored to protect against backdoor attacks — a type of malicious attack designed to misclassify a particular set of samples — by detecting trojans [47]. Similarly, another BCFL implementation that relied on the permissioned blockchain Hyperledger Fabric for its architecture was ScaleSFL, as named by Madill et al. [50]. They also contributed to the open source community by making their prototype implementation available in a public repository[1]. The researchers presented a sharding mechanism based on two consensus levels: shard level consensus and mainchain consensus. With this addition, they aimed to solve the problem of model update poisoning. They ran several benchmarks to show that the effect of sharding had a positive impact on the accuracy of the model and improved convergence speed. Nevertheless, it is important to note that this comparison was made under the assumption that all clients behaved honestly, as the aggregation occurred off-chain and the blockchain only stored hashes of the global model. This limitation suggests that the insights of the study may not be directly applicable to real-world scenarios.

[1] https://github.com/blockchain-systems/scalesfl

Table 3.1: Summary of Blockchain-Based Federated Learning (BCFL) literature

Ref.	Pub. Type	Name	Issues Addressed	Type of Blockchain	Consensus Mechanism	Key Contributions
[26]	Review	-	-	-	-	Analysis of seventeen BCFL framework designs and how they applied blockchain to solve FL problems.
[79]	Review	-	-	-	-	First to classify fourty different BCFL frameworks based on coupling: fully, flexibly and loosely coupled BCFL.
[91]	Review	-	-	-	-	Includes an investigation on the applied research of BCFL within thirteen different applications.
[38]	Article	BlockFL	SPOF, lack of motivation, network latency	Public	PoW	Introduced a validation process for local model updates and rewarded participant contributions in proportion to the training sample sizes.
[41]	Article	BLADE-FL	SPOF, lazy clients, information leakage	Public	PoW	Explored the impact of lazy clients in the network by adding artificial noise to the learning process.
[49]	Article	BLADE-FL	SPOF, lazy clients, information leakage	Public	PoW	Proposed a signature mechanism resilient to noise perturbation, capable of detecting plagiarised models with high detection accuracy.
[10]	Article	Unnamed	Free-riding attacks	Not specified	Not specified	Addressed free-riding attacks by incentivising participants with cryptocurrency rewards.
[44]	Article	BFLC	SPOF, lack of motivation, poison attacks	Public	Committee consensus	Mitigated malicious attacks through a committee responsible for validating and scoring local gradient updates.
[13]	Article	BlockFLA	SPOF, lack of motivation, poison attacks	Hybrid	PoW and PBFT	Hybrid framework integrating private and public blockchains, tailored to protect against *backdoor* attacks.
[50]	Article	ScaleSFL	SPOF, poison attacks	Private	PBFT	Presented a sharding mechanism based on two consensus levels: shard level consensus and mainchain consensus, aimed at solving the problem of model update poisoning.

3.2 Federated Learning for Natural Processing Language

ML has largely been applied to the field of NLP to process and understand human language [59]. This field can particularly benefit from the usage of DL because of its capability to learn complex patterns beyond any other ML method. DL has been essential in pushing the boundaries of NLP in applications such as machine translation, sentiment analysis and text generation. The inherent requirement of NLP tasks for large datasets aligns well with the strengths of DL networks, which excel in scenarios with abundant data.

There has been a remarkable evolution in the DL architectures employed in NLP, evolving from simple ANNs to the more complex Convolutional Neural Network (CNN) and subsequently to Recurrent Neural Networks (RNNs). RNNs, including their more advanced variants Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks, are designed to process sequential data by being able to capture temporal dependencies, which makes them particularly effective for tasks that require an understanding of context over time, such as tasks related to language processing.

Nonetheless, in the recent years, the usage of these technologies has been eclipsed by transformers. Transformers, introduced by Vaswani et al. [76], improved RNN architectures by the integration of self-attention mechanisms. This innovation allows the models to process entire sequences of data simultaneously, in contrast to the sequential processing inherent to prior models, as depicted in Figure 3.1. This capability makes transformers exceptionally efficient at tasks that involve understanding the context and relationships within large volumes of text. Therefore, transformers are highly effective for a variety of NLP tasks, including machine translation, text summarisation and question-answering systems, thanks to their ability to capture long-range dependencies within text. Popular examples of models that use transformers as their backbone include Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT). BERT, by Devlin et al., is known for its bidirectional context understanding, which makes it a powerful tool for text classification tasks [14]. GPT, by Radford et al. [65], is also very popular for generating contextually rich text sequences.

FL plays an important role in the domain of NLP in cases where the learning task needs to deal with sensitive data such as clinical reports, financial statements and legal documentation, among others. However, research on FL frameworks applied to NLP has predominantly used smaller DL architectures, such as the aforementioned RNNs, LSTM networks and GRU networks, rather than state-of-the-art transformers. We argue that the main reason for these choices was dictated by the constraints of the FL framework designs. Most FL solutions were designed for cross-device FL, where the participants of the distributed training network could be any type of IoT device, with its inherent computational and data accessibility limitations, which hindered or even precluded the use of powerful architectures such as transformers, despite their proven superior performance.

As instances that illustrate this trend of using lightweight models within FL frameworks in the context of NLP, we see that Yang et al. [87] used FL to improve the quality of virtual keyboard search suggestions on mobile phones. This on-device infrastructure influenced on their decision of using LSTM networks for the mobile devices to efficiently process the data within its constrained infrastructure. Thakkar et al. [74] investigated the impact of different components of FL on unintended memorisation in trained models, compared to central learning. Their research also opted for LSTM networks as the model architecture to conduct the experiments. These examples are not isolated, many more implementations following the same argument can be identified in the comprehensive survey of FL in NLP conducted in 2021 by Liu et al [45].

Nevertheless, Basu et al. [7] conducted extensive benchmarks of their FL framework using BERT to explore its efficacy in mental health diagnosis through binary text classification tasks. Their experimental setup involved a comparison between their distributed FL framework and its centralised DL network equivalent. Their benchmarks yielded promising results, with their distributed solution achieving better accuracy metrics than the DL solution. It is important to note that Basu et al. implemented a centralised version of FL, which circumvented the computational and data limitations often associated with decentralised architectures like blockchain technology for the use of complex transformer models. However, this work establishes a viable path for integrating advanced DL architectures into FL for NLP, albeit with certain structural modifications for blockchain integration to achieve a BCFL framework.

Lastly, in their comprehensive literature review conducted in April 2023, Schumann et al. [68] examined the landscape of FL implementations within the field of NLP. Among the seventy-six studies reviewed, only two employed decentralised architectures, highlighting a significant research gap in this area. Notably, neither of these two studies integrated blockchain technology into their FL frameworks to achieve the decentralisation [35, 27], but other decentralised architectures. These findings further underscore the lack of research into the integration of blockchain and FL applied to NLP.

3.3 Privacy Mechanisms

As discussed in Section 2.3, in order to protect the privacy of sensitive data with strong guarantees, FL requires the adoption of additional measures to prevent potential information leakage through inference attacks undertaken by malicious agents. We have previously mentioned the most popular techniques used for this purpose, namely DP and HE. This section is dedicated to analysing the existing research concerning the preferences, arguments and findings related to these two technologies.

Aziz et al. [6] conducted a review in 2023 that explored the challenges associated with these two techniques. They also examined the body of research that explored the fusion of HE and DP in the context of FL, referred to as *hybrid* approaches. Their broader conclusions were that for DP, the main challenge was striking the balance between privacy

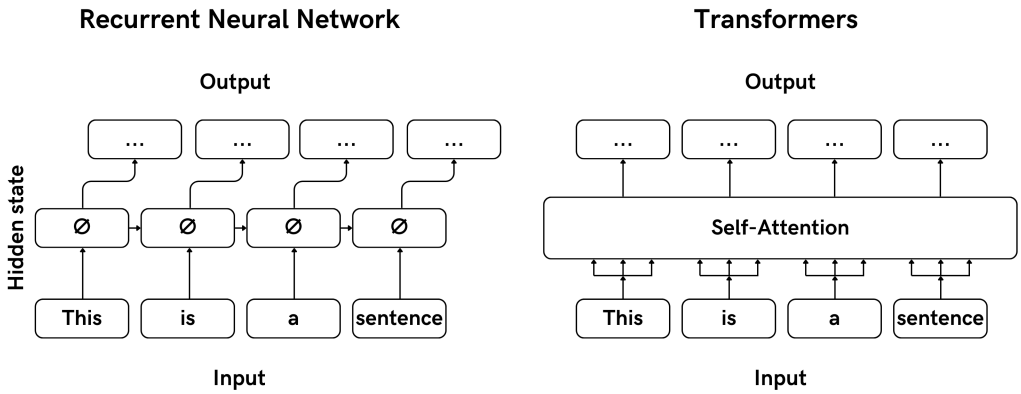

Figure 3.1: Recurrent Neural Network (RNN) vs. transformers architecture. The RNN output is derived from its hidden state, which depends on the previous input, creating a temporal dependency. In contrast, transformers are able to process all the inputs in parallel thanks to their sophisticated self-attention mechanism.

and accuracy, as the introduction of noise always meant a detriment of the performance of the model. As for HE, the challenge was centred on mitigating the trade-off between privacy and computational complexity. HE minimises accuracy loss compared to DP, but the security provided by HE heavily relies on the strength of the encryption, which comes with a significant increase in computational and spatial complexity. Furthermore, the combination of HE and DP was pointed out as an interesting direction for future exploration.

El Ouadrhiri et al. [17] applied many types of DP configurations to analyse the impact on the model's accuracy of the different probability distributions that satisfied the DP mechanism. Of all the probability distributions, they found that with lower levels of ϵ (epsilon) — more protection and less privacy leakage — the Gaussian distribution was the one that was able to maintain the higher accuracy values. Their investigation further explored the differences between CDP and LDP, where LDP was outlined for its purpose of ensuring privacy in cases where individuals did not trust the central data curator. However, a centralised FL mechanism utilising LDP still falls under the need to trust the integrity and availability of the global model to the central server.

Jin et al. [36] presented FedML-HE, an FL system that used HE for secure model aggregation. In the study, they measured the overhead introduced by the HE mechanism and attempted to mitigate it by selectively encrypting only a subset of the parameters which were more sensitive, thereby reducing the computational and communication burden during training. Despite the promising outcomes of their experiments, this approach did not entirely eliminate the challenges associated with HE. The measured overhead of their enhanced HE mechanism was still in general 5x~20x over in computational overhead — being the time taken for the learning process — and 15x over in communication overhead — indicative of the size of the encrypted data. These findings highlight the barriers that

remain in integrating the HE method for architectures based on larger models, such as transformers, particularly within the constraints of block sizes and network latency typical of blockchain technology.

3.4 Conclusions

From the literature review conducted in this chapter, we can draw several key insights into the current state of FL and BCFL. Firstly, it clearly elucidates the significant gap in research on the exploration of transformer architectures in the domain of FL applied to NLP tasks. The majority of the work is focused on cross-device FL, limited by the complexity of the DL models that can be used for the learning process. Simpler models such as RNNs and LSTM networks are preferred for these frameworks. Moreover, there is a lack of research aimed at decentralising FL frameworks to address SPOF and availability attacks, improve scalability and enable trustless collaboration, as shown in the review conducted by Schuman et al. [68] in 2023, where only two out of the seventy-six studies covered in the review addressed these issues.

Beyond the specific context of NLP, we can find some BCFL frameworks integrating blockchain and FL. However, the majority of these approaches involve a reliance on off-chain computations, thereby introducing the need for an additional entity that must be trusted and managed to completely guarantee the confidentiality of the data and the integrity of the global model. This off-chain reliance is frequently attributed to the blockchain's intrinsic limitations in terms of computational and storage capabilities, which significantly complicates the deployment of FL models on the blockchain. The literature reveals that many proposed designs of BCFL frameworks [60, 10] do not offer viable strategies for real-world implementation, highlighting a critical gap in research.

These findings have had a significant impact on the direction of our work. Having identified the existing gaps and challenges in the area of BCFL, particularly as applied to NLP, our study intends to address them by exploring the implementation of a cross-silo BCFL framework using Hyperledger Fabric as a permissioned blockchain and BERT transformer-based models for binary text classification tasks, specifically targeting depression datasets. This approach attempts to solve the two identified gaps together: using advanced transformer architectures in FL for NLP tasks and addressing decentralisation by integrating blockchain and achieving a fully functional BCFL framework.

Furthermore, we have opted to incorporate LDP as the privacy-preserving technique within our framework. LDP is selected to ensure that parameter updates are protected before leaving the client, assuming no trust from the network. We have discarded HE due to its added complexity that results in ten times increased computational demands and fifteen times extended processing times. Such delays and costs pose a considerable threat to FL systems, which are built around fruitful communication and processing among many participants across a wide network. Besides, a larger size of encrypted objects requires more storage and bandwidth, hence taxing BCFL architectures.

The upcoming chapters will offer a detailed analysis of the design choices of our BCFL framework. We will also discuss the different implementation aspects, including the technology stack used and the reasoning behind these choices. In addition, we will present our experimental setup and the evaluation of our experiments, offering insights into the effectiveness of our approach in addressing our research questions and filling the gaps identified in the review of the related work conducted in this chapter.

Chapter 4
System Design

This chapter defines the scope of our BCFL framework to establish a detailed description of the system requirements and design. Comprehending these requirements is crucial for the subsequent integration of the components into a cohesive BCFL framework. Therefore, a solid foundation for the required functionality of the various components that constitute the framework is laid prior to its implementation in Chapter 5.

4.1 Scope of the System

This section provides an overview of the system's scope and functionality and outlines a high-level view of the framework pipeline. Figure 4.1 provides a visual representation of it.

The system embodies a fully functional framework for FL training in a decentralised manner, underpinned by the use of blockchain as the entity responsible for the aggregation of the local models. This framework is designed to enable collaborative training among different organisations within a cross-silo system. The nodes participating in the training are selected beforehand to take part in the decentralised network. These selected nodes have access to sufficient training data and computational resources to adequately fine-tune a complex DL model based on the transformer architecture. All the participants train the same model architecture, which is accessible on the blockchain at the beginning of the training phase. Subsequently, the nodes train their model instances locally with data within the same feature space, following a HFL scheme (see Subsection 2.2.1 for more information about the characteristics of the different schemes). Each node updates their model parameters via the backpropagation mechanism with the gradients resulting from supervised training on its dataset. They can add noise to the model updates with DP

mechanisms in order to prevent inference attacks that could gain insight into their data. The data is fully secured and remains confidential throughout the whole FL process.

The organisations communicate with the blockchain network through an interface known as the *gateway*, which uses a secure and efficient communication channel to connect the FL client with the blockchain. This component abstracts all the communication logic and is responsible for directly issuing transactions containing the local model updates to the blockchain. The blockchain allows the aggregation of multiple instances of the same model, based on the transformer architecture, to complete the final phase of the training iteration. Therefore, it can deploy smart contracts that implement the different federated algorithms, namely FedAvg and FedProx, to be executed at the end of each iteration round. The smart contracts aggregate the local models on-chain without delegating any functionality to off-chain mechanisms. The global model from this aggregation is made available through a query executed by the training nodes on the smart contracts, enabling them to continue with another iteration of the FL process.

In line with the scope of the system described above, the sequence flow of the framework is summarised as follows:

1. *Local training.* Each node — as an organisation — trains its DL model instance locally on its own dataset within the same feature space for a certain amount of iterations.

2. *Noise addition.* Nodes apply — on their choice — LDP to their model parameters to protect the model against inference attacks.

3. *Model upload.* Nodes send their model updates to the *gateway*, which encapsulates the data into the transaction payload and issues these transactions to the blockchain.

4. *Federated aggregation.* Smart contracts aggregate the local models received into a global model using federated algorithms — FedAvg and FedProx — storing the current global model in a new block.

5. *Model update.* All the nodes download the parameters of the new global model to update their model instances and proceed with a new iteration.

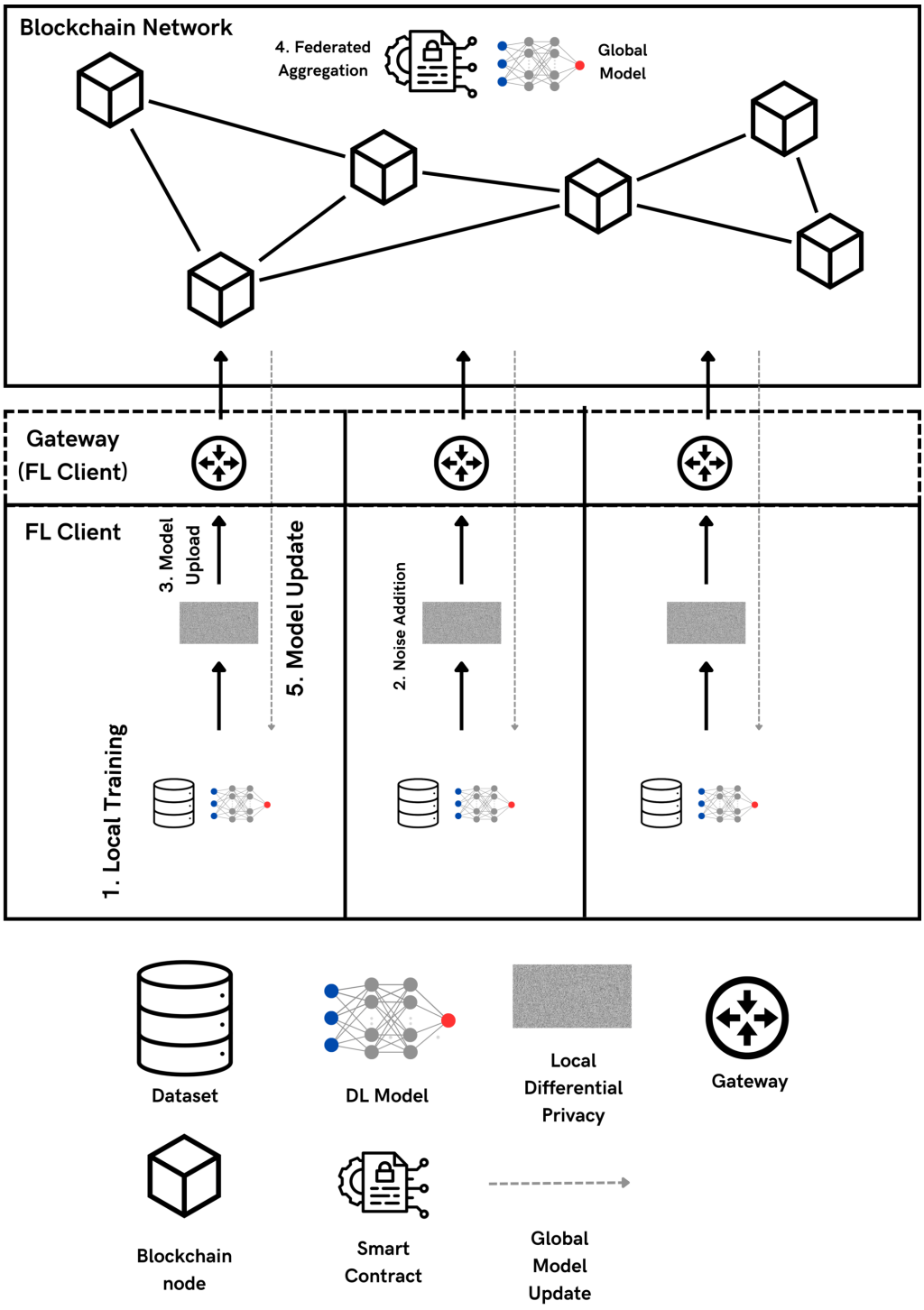

Figure 4.1: Framework's pipeline

4.2 Non-functional Requirements

This section outlines the framework's non-functional requirements, in line with the scope described in the previous section, which provides a broad overview of the framework design. They are as follows:

- *Cross-silo architecture.* Our framework is based on a cross-silo architecture as it is designed to facilitate communication among a limited number of participants with sufficient data and computational capabilities to support large models. In this architecture, various types of organisations engage in collaborative learning in a decentralised manner through the use of blockchain technology. The number of system participants is considerably small, limited to the order of tens, and the members are available for all training rounds, as described for cross-silo architecture in Subsection 2.2.1.
- *Synchronous FL.* The training flow is synchronous. This means that the global model is updated in each iteration once all the local model instances are received in the blockchain. It is possible to integrate a security timeout functionality to safeguard the aggregation in case of participant malfunction. To guarantee the synchronous flow, participants must deploy their FL client on systems with similar resources and train the models with data for approximately the same number of iterations. These requirements can be set up by a governing body or by prior agreement among the participating parties that make up the network.
- *Modular architecture.* The framework is modular, allowing its components to be deployed on different platforms or on the same system. This means that the framework nodes can execute training in its FL client and act as a blockchain node simultaneously. The governance body responsible for framework orchestration can choose from standalone — self-contained — to fully decoupled solutions. The election has no impact on the functionality and capabilities of the framework. This is further discussed in the following section.

4.3 Requirements Analysis

The design of our framework is a modular one, since it can be classified in terms of *coupling* as an overlapped configuration. This means that the nodes participating in model training can also generate blocks for the blockchain. Some nodes within the framework may only perform one of the two tasks, while others may be involved in both. Thanks to this software modularity, the FL network governance body can choose how the different parts of the framework are deployed. The various modules of the software can be divided into the following (see also Figure 4.2 for a more detailed view):

- *FL clients.* They comprise the logic responsible for training the local model instances and injecting the LDP.
- *Blockchain.* This module is divided into a complex layered structure in which the smart contracts are stored, containing the code responsible for aggregating local model instances into a global one.
- *Gateway.* This is the piece of software that enables communication between the FL clients and the blockchain, making the entire process possible.

These components are designed to meet specific requirements that comply with the modular framework, making them independent from each other. This ensures that the functionality of the entire framework is not limited to any specific deployment scenario. To achieve the functionality described in Section 4.1, it is essential for each component to maintain a secure communication channel with the others.

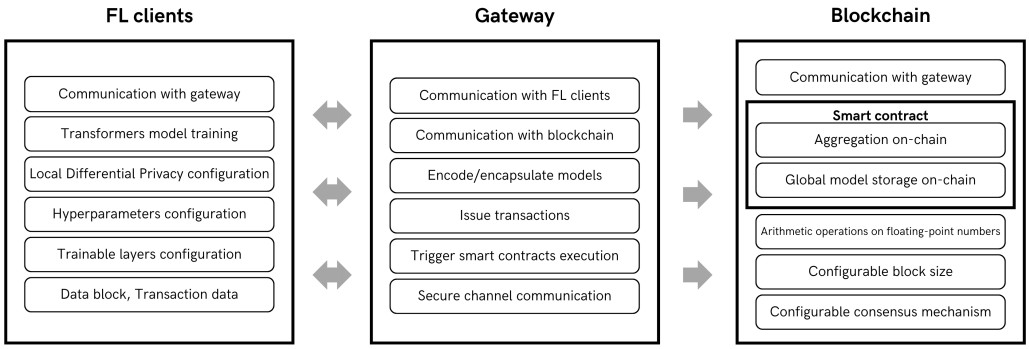

Figure 4.2: Modular framework design

4.3.1 Federated Learning Client

Our BCFL framework is designed to analyse data horizontally. This means that participants' data share a subset of features, enabling us to gain insights by applying the DL model to multiple datasets with the same feature space.

Since one of the research questions (refer to RQ2 in Section 1.2) aims to investigate the feasibility of using transformer-based models as the backbone of the BCFL system, certain requirements must be met by the participants in terms of the computational capacity of the systems they deploy their FL clients. This results in a cross-silo FL system, where participants are audited before accessing the distributed network. Additionally, the complexity of the model used, as well as the layers to be trained, must be tailored to fit the data and resource needs of the blockchain network.

The FL client, responsible for training its local model instance, must allow the configuration of the various training hyperparameters, as well as the different privacy levels,

to tailor them carefully in order to achieve the best balance between data security and model accuracy.

In addition, the FL client must be able to communicate with the blockchain and execute calls to the smart contracts to send its updated model instance, as well as to obtain the global model for subsequent training iterations. This communication is done through a *gateway* that abstracts the entire process of creating transactions and triggering the business logic from the smart contracts. Therefore, FL clients only need to integrate communication with this *gateway* component.

4.3.2 Blockchain

In Section 2.5, we discussed the problems associated with FL that could be solved through integration with the blockchain. In order to guarantee the solution of these problems and the correct functioning of the system, it is necessary to establish specific requirements for the design of this blockchain network.

To ensure the collaboration among participants without assuming trust between them in the network, all the aggregation logic takes place on-chain, which poses significant challenges for the management of the space and network latency of the blockchain. Smart contracts that can perform arithmetic operations on floating-point numbers with a sufficient number of decimal places must be implemented. This is particularly important to overcome any limitations on the number of decimal places within the aggregation, as this can lead to vanishing gradients in DL models [25, 22]. An example is presented in Figure 4.3, which illustrates how information for low value parameters is lost in both integer and fixed-point arithmetic cases, in contrast to floating-point arithmetic. On the other hand, the blockchain must be able to store the different models within its blocks, which limits the choice of blockchains that can be used. Public blockchains, such as Bitcoin and Ethereum, feature block size averages of approximately 1 MB [77] and have severe limitations on arithmetic operations due to the domain languages they use, making these options infeasible for our BCFL framework.

	Floating-point arithmetic		Integer arithmetic		Fixed-point arithmetic (scale 10000)	
Input values	-1.7294e-04	-1.3206e-04	-1.7294e-04	-1.3206e-04	-1.7294e-04	-1.3206e-04
Scaled values	-1.7294e-04	-1.3206e-04	0	0	-1	-1
Averaged value	-1.525e-04		0		-1e-04	

Figure 4.3: Comparative precision of averaging using different arithmetic techniques

Due to these limitations and the fact that our system is designed as a cross-silo framework, our system falls under the requirements of a permissioned blockchain. Permissioned blockchains allows for access control to network members and a specific configuration

of block size and block rate, as well as the choice of the most appropriate consensus mechanism for the correct functioning of the network. This guarantees the scalability of the system and ensures that the entire process is carried out on-chain through smart contracts, transparently and without the need to rely on any off-chain entity.

Regarding the security of the network, to protect it from SPOF and availability attacks, the nodes responsible for consensus and smart contract execution must be sufficient in number and distributed in such a way that the operational functioning of the system is not compromised if one or more of the nodes have an availability problem or are targeted by a DoS attack. Moreover, in the event of a malicious agent gaining access to a node on the blockchain, its decentralised nature ensures that the system remains uncompromised and functional because the transaction data is still accessible to all other nodes and no single node is solely responsible for executing smart contracts. The malicious node cannot take control of the whole system because consensus must be reached to add new blocks of transactions and execute code. Depending on the consensus mechanism, a minimum percentage of participants must agree. Additionally, the data stored in the blocks is immutable, preventing any compromise in the integrity of the global model and the models registry. This ensures that potential attackers cannot gain control over the business logic or manipulate the blockchain data.

4.3.3 Gateway

The *gateway* component of the framework acts as a bridge between the FL clients and the blockchain. Given the modular nature of the framework and the independence of these technologies, it is necessary to have a component that serves as an intermediary. The purpose of the *gateway* is to receive the parameters of the models, encode and/or encapsulate them in the necessary way and send them to the blockchain within the transaction payload. This component is capable of sending transactions directly to the blockchain and executing the code of the smart contracts to integrate the various FL clients participating in the training with the blockchain. To ensure full confidentiality of the data and prevent any potential inference attacks on the local models, this component can be deployed within the same system responsible for training the local models in each organisation of the cross-silo FL system. Otherwise, communication must be conducted through encrypted and secure channels.

Chapter 5
Implementation

This chapter discusses all the implementation aspects of the BCFL framework designed in the previous chapter. It includes a detailed explanation of the choice and rationale for each of the technologies used for each component of the framework. Additionally, it provides information on the NLP use case the framework will focus on and the selected datasets used for training.

5.1 Use Case

Before delving into the technologies used to implement our BCFL framework, we will first outline the specific use case for which it will be tailored.

The chosen use case for this framework is healthcare, specifically binary text classification for depression to aid in mental health diagnosis. As our design is based on cross-silo FL, as described in the scope (Section 4.1), the participating nodes in our BCFL framework are pre-selected hospitals and health centres collaborating on the training process. We can therefore assume that these cross-silo organisations possess the necessary resources to fine-tune complex DL architectures, which allows for the use of more advanced NLP transformer-based models. It should be noted that this choice does not limit the applicability of our BCFL framework to other sectors such as finance, retail and telecommunications [91, 86]. The framework is designed to have configurable DL model choices, hyperparameters and input data sources, so that it can be easily adapted to other tasks in different domains.

The rationale behind our choice is that there is a great variety of surveys and reviews of ML and FL solutions applied to mental health [85, 66, 2] using a wide range of datasets relating to mental health conditions such as depression, Post-Traumatic Stress Disorder

(PTSD), anxiety and anorexia, among others [33, 4, 71, 70]. However, these studies rarely explore the potential advantages of integrating blockchain technology to enhance their framework by reducing the risks associated with centralisation. Remarkably, to the best of our knowledge, only one paper has investigated the intersection of BCFL and mental health. This study by Shukla et al. [69] focused on psychophysiological data, with a dataset comprising 142 patients and utilising a DL model based on an ANN with three dense layers for its architecture. As shown in the systematic review by Myrzashova et al. [55], the application of BCFL in psychiatry remains poor. Due to the sensitive nature of psychiatric data and the complexity of its diagnosis, we believe that a decentralised FL approach based on a complex DL model is not only an interesting approach but also a necessary one.

5.2 Datasets

One of the challenges of this project is establishing a reliable data pipeline [71], which introduces certain limitations to the research. First, leveraging all types of datasets for the aggregation of the global model becomes too ambitious due to the different types of data, be it video, text or neurological data. Each of them requires its own set of preprocessing steps and unique DL model implementation. Furthermore, the aggregation of all these models into a global one would necessitate the use of FTL [58], which is beyond the scope of this work, although it could be under consideration for future research. To manage these constraints, the study will focus on text-based datasets [63, 70, 71, 88, 31, 7, 52, 33, 4], taking advantage of their large number of data points. Most of these datasets are derived from web scraping techniques applied to popular platforms like Reddit[1] and Twitter[2]. Some of them are labelled from inference techniques validated using smaller datasets of around 500 posts labelled by psychologists from the "depression subreddit" — a Reddit subforum. Other datasets are labelled based on the users self-reported diagnoses of depression.

Some of the datasets classifying the state of depression may contain multiple target labels, resulting in different output dimensions. To address this, we will only keep the target label associated with depression and use label binarisation when needed [9]. This approach will ensure that all local models have the same output dimension, thereby enabling aggregation into a global binary classification model for predicting depression.

Each dataset is splitted into a training and a test sets using an 80/20% ratio. The training set is further divided into two parts: train and evaluation, using the same 80/20% ratio as shown in Table 5.1. The training partition is used to feed the network with inputs, while the evaluation partition assesses the best convergence point of the model. Finally, the test set evaluates the generalisation capability of the fine-tuned model on unseen data under a similar data distribution.

[1] https://www.reddit.com/
[2] https://www.twitter.com/

Dataset	Total Size	Train Size	Eval Size	Test Size
twitter_dep	3096	1982	496	618
acl_dep_sad	3267	2085	512	650
mixed_depression	2822	1806	452	564
dreaddit	3553	2270	568	715

Table 5.1: Dataset splits

The purpose of using public datasets to fine-tune our framework is to demonstrate its functionality, rather than serving as the primary data source for the target application of the framework. The actual application of this framework would extend to private datasets consisting of data gathered in hospital and health centre datasets containing clinical textual data, such as transcribed therapy sessions and clinical interviews [83, 34, 60]. These datasets are not available at the time of the study for incorporation into this project, but could be used in a real-world case scenario.

The transformer models in this framework are fine-tuned using the following datasets (see also Figure 5.1):

- *twitter_dep* from Basu et al. [7] comprises over 3,000 tweets that have been labelled as either depression or non-depression.
- *acl_dep_sad* from Roy et al. [70] is a collection of six different datasets, three of which classify depression and three of which classify sadness. The final dataset comprises 3,256 samples, of which 1,914 are labelled as sadness and 1,342 are labelled as depression, representing negative and positive samples, respectively.
- *mixed_depression* from Pirina et al. [61] contains nearly 3,000 posts gathered from Reddit, from a large subreddit devoted to depression.
- *dreaddit* from Elsbeth et al. [18] contains more than 3,000 posts in Reddit from ten different subreddits — in the five domains of abuse, social, anxiety, PTSD and financial need — classified as depression vs. non-depression.

5.3 FL Client

The specific implementation presented in this work adheres to a decoupled configuration, in which the nodes responsible for managing the blockchain are separated from the FL clients. Furthermore, the FL clients are executed on a single Python process, which creates an execution thread for each client. These execution threads posses their own variables, but within the same shared environment. This configuration enables the main Python process to know when all the FL clients have sent their local models to the blockchain, thereby simplifying the synchronisation mechanism. This does not affect

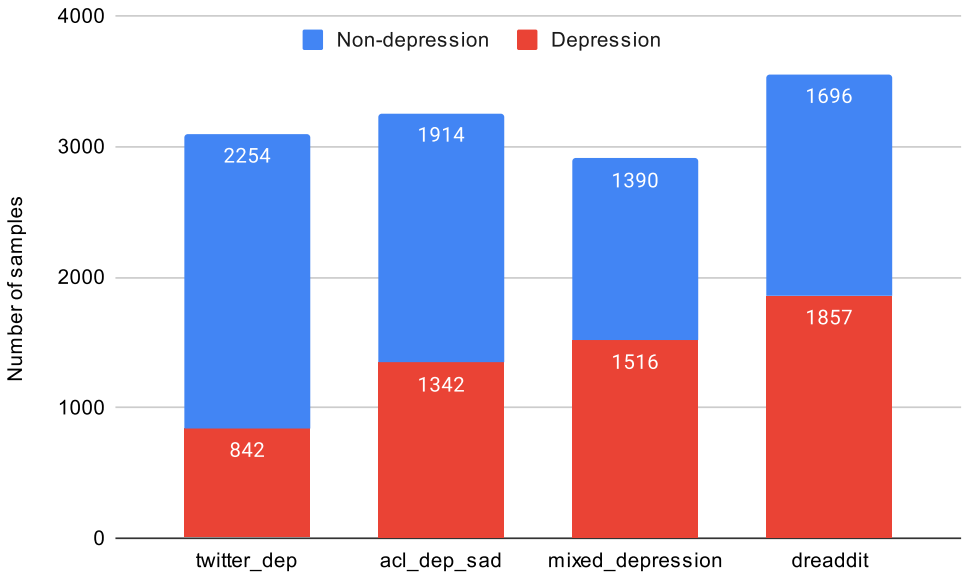

Figure 5.1: Dataset sizes

the assessment of the framework's feasibility, although it should be reconsidered when applying the framework to any production environment.

The FL client is responsible for training the local model instances, sending them to the *gateway* for subsequent aggregation on the blockchain and retrieval of the new global model. This section covers the implementation details of the transformer-based models chosen for the NLP text classification task, as well as the privacy method implemented.

5.3.1 Model

The majority of current research of FL applied to NLP tasks adheres to traditional ANN architectures like CNN, RNN and GRU [33, 4, 71]. We argue that there are two main reasons for this. Firstly, the reduced sample size that they use to feed the models necessitates the use of simpler models that can perform well with limited data. Secondly, they implement cross-device FL architectures, where the clients are IoT devices with low performance expectations and limited computational resources [66, 28]. Our research addresses these issues by benefiting from access to numerous and large text-based datasets, enabling the use of more advanced NLP transformer-based models, which are known for their superior performance in capturing the long-term syntactic and semantic relationships of words [52]. Additionally, our framework is also based on cross-silo architecture, where organisations have the resources to train more computationally intensive models.

FL clients in our framework are implemented using models based on the transformer

architecture. Transformers, thanks to their self-attention mechanism, can process all tokens of the entire sequence, or sentence, in parallel, facilitating a more effective capture of content for each word [63, 7, 52]. We chose to implement our BCFL framework with BERT [14] transformer architecture for its bidirectional nature, which is crucial for understanding the context of words in text, making it excellent for text classification. Other transformer models, such as OpenAI's GPT [65], are better suited for tasks like text generation because they use a left-to-right transformer, only considering the context from the left. The difference between these two transformer-based architecture models is represented in Figure 5.2.

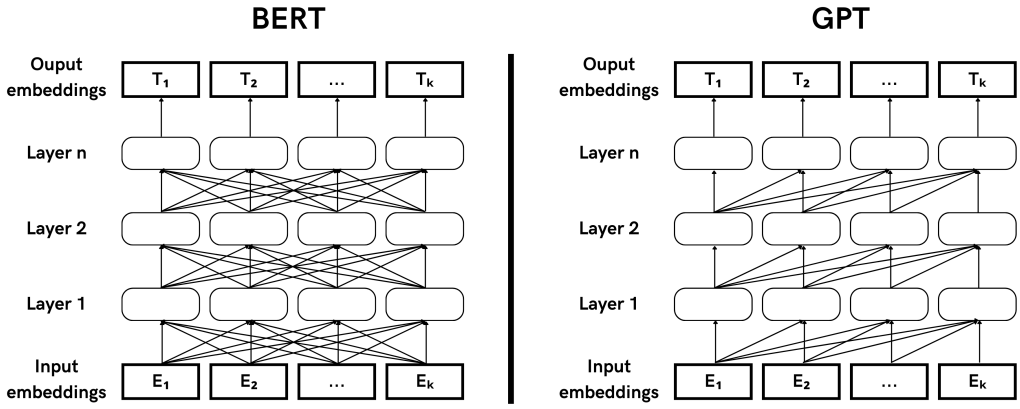

Figure 5.2: BERT vs. GPT architecture. The BERT model stands for its bidirectional architecture, which improves the understanding of the surrounding context of the word. In contrast the GPT model is better suited for text generation because it considers the context of the previous words — so-called left-to-right transformer.

Nevertheless, there are many versions of the BERT architecture, with different sizes and amount of parameters ranging from tens of millions to almost half a billion, as we can see in Figure 5.3. For the sake of project simplicity and considering the computational limitations of our deployment environment, we implement the lighter versions. The versions of BERT that our framework has been tested on are the following: BERT Tiny, BERT Mini, BERT Small and BERT Medium — information about these can be found in the official Google GitHub repository[3].

Figure 5.4 displays the sizes of the four model versions, ranging from 25 MB for BERT Tiny to 240 MB for BERT Medium. These figures correspond to the size of the parameters of the entire model, including embedding layers, after being encoded and compressed to occupy less space. It is evident that there is a significant difference between the lightest and most complex models, by a factor of approximately 10x. The figure also displays the corresponding size of the last three layers of each model, which shows a noticeable difference with respect to the full model. As the BERT models are pre-trained on large corpora and the task at hand involves fine-tuning them for specific datasets, the literature

[3]https://github.com/google-research/bert

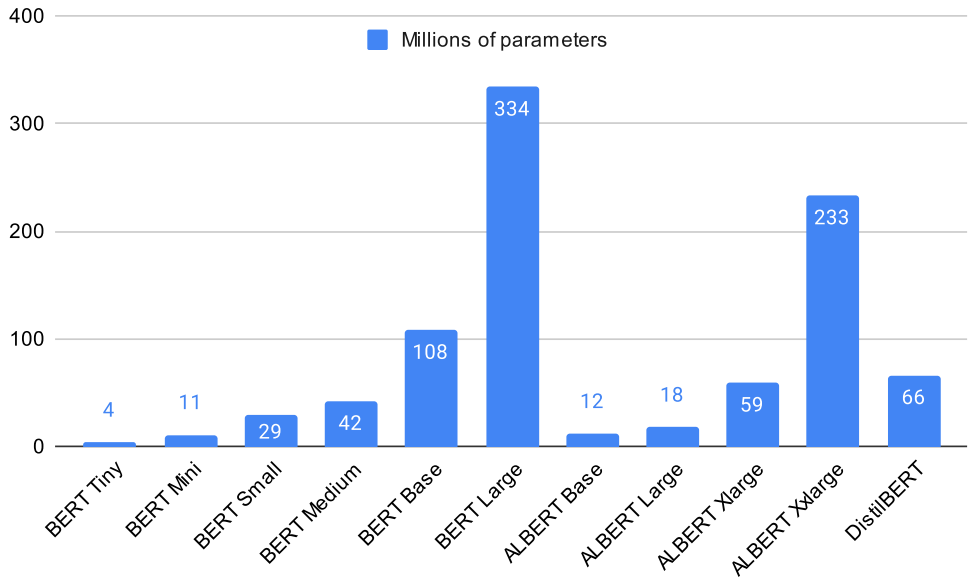

Figure 5.3: Millions of parameters of BERT transformer Deep Learning (DL) models

suggests that training only one fourth of the model layers can achieve a 10% parity with the fully trained model and may even improve the result by preventing overfitting to the training set [40, 72].

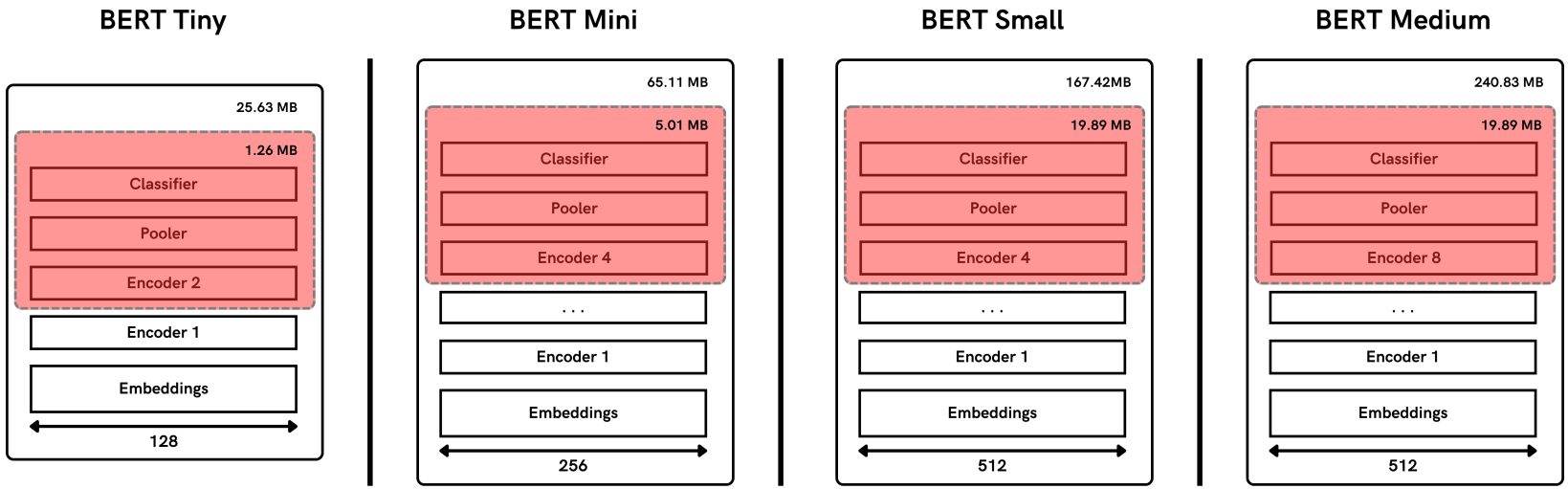

Figure 5.4: BERT model size comparison. Comparison of the sizes of the four different BERT models after encoding and compression. The size varies depending on the number of hidden encoding layers and their height — 128, 256 or 512. The last three trainable layers are highlighted together with their corresponding sizes.

5.3.2 Privacy

Although the data is not shared between the different organisations in our BCFL framework, privacy still poses some challenges due to the potential inference attacks on the local model updates sent to the blockchain [93]. Various ways have been proposed to tackle this, mainly focusing on DP and advanced encryption techniques like SMC and HE [32]. For background information on these different mechanisms refer to Section 2.3.

We implement LDP with Opacus[4], a Python[5] library that offers ready-made solutions to train DL models with DP. LDP adds Gaussian noise to the local model updates before being sent to the *gateway*, thus preventing the possibility of any inference attack from a node of the network. The privacy strength of the models relies on the value of the DP parameter ϵ (epsilon). This parameter is configurable in our framework and can also be deactivated to create solutions that match the required level of privacy in each scenario. In the next chapter, we analyse the impact of different privacy levels on the accuracy of the models used (see Section 6.4).

5.4 Blockchain

Although FL training method and DP mechanism offer certain warranties in terms of data privacy, concerns remain regarding the vulnerabilities of the centralised server, such as those mentioned in Section 2.5.

As stated in the blockchain design — Subsection 4.3.2 of the previous chapter — our framework utilises a permissioned blockchain for decentralisation of the FL process. We have opted to use Hyperledger Fabric[6], one of the most popular permissioned blockchains available, thanks to its numerous advantages over permissionless blockchains. Hyperledger Fabric provides easy configuration of block size, making it capable of storing small instances of BERT models that are tens of MB in size. The consensus mechanisms can be simpler thanks to their reliance on the governance body and the auditing of pre-selected nodes in the network. This also enables a significant increase in transaction throughput. To access block and transaction information directly, Hyperledger Fabric provides a built-in system chaincode called Query System Chaincode (QSCC)[7]. This chaincode enables the retrieval and decoding of information for each block and its transactions.

Apart from its governance model and consensus mechanism, Hyperledger Fabric differs from other permissionless public blockchains, like Ethereum, in its architecture, having a strong positive influence on the smart contracts' capability. Hyperledger Fabric employs an architecture called execute-order-validate, which executes transactions before agreeing

[4]https://opacus.ai/

[5]https://www.python.org/

[6]https://www.hyperledger.org/projects/fabric

[7]https://hyperledger-fabric.readthedocs.io/en/release-2.5/smartcontract/smartcontract.html#system-chaincode

on their order. This process eliminates non-deterministic inconsistencies by filtering them out before ordering, enabling the use of general-purpose programming languages like TypeScript[8]. This provides more flexibility compared to permissionless blockchains such as Ethereum, which uses a domain-specific language called Solidity. As a result, Hyperledger Fabric can handle complex operations, including floating-point arithmetic, which are restricted in Solidity. This is crucial because the precision of the federated aggregation algorithms is highly dependent on the precision of the parameter tensors — floating-point numbers with dozens of decimal points — being maintained during the aggregation. It also improves the transactions throughput not only due to simpler and faster consensus mechanisms, but also because this architecture allows for parallel processing of transactions.

5.4.1 Chaincode

The chaincode — Hyperledger Fabric's name for smart contracts — deployed in our blockchain includes the following features executable via transactions:

- `AggregateModels(ctx, globalModelId, round)`
  Aggregates the local model instances for the given round and creates a new global model with the provided ID. This function can be triggered automatically by the chaincode when all the local model instances have been received, or manually triggered by a FL client via a transaction.
- `CreateModel(ctx, id, modelParams)`
  Creates a new model with the given ID and the model parameters, encoded in a base64 string. This function is used within the chaincode to create the global model after each aggregation round.
- `DeleteAllModels(ctx)`
  Updates the current ledger's state by deleting all existing models.
- `DeleteModel(ctx, id)`
  Deletes the model with the given ID.
- `GetAllModels(ctx)`
  Retrieves all models. Useful for understanding the current state of the decentralised ledger.
- `ReadModel(ctx, id)`
  Retrieves a model stored in the blockchain by the given ID. The FL clients use this feature to retrieve the new global model before beginning a new local training iteration.

[8] https://www.typescriptlang.org/

- `SubmitLocalModel(ctx, modelParams)`
  A local model instance is submitted from an FL client *gateway*. The model ID is assigned internally by the system to identify the corresponding owner, and the local model parameters are stored in base64 string format.

In every feature the `ctx` object is an interface used by the chaincode to interact with the ledger, enabling reading from and writing to the ledger's state. It is not a parameter passed through transactions by the *gateway*.

5.4.2 Federated Aggregation Algorithm

The chaincode is responsible for the aggregation of the local trained models. This aggregation is performed by applying the federated aggregation algorithm to the different local model instances that the blockchain receives from the participating nodes. The basic foundational aggregation algorithm is FedAvg [53], which calculates a global model by taking the average of the local model updates from each participating client.

As we are dealing with an HFL scenario where each party trains different samples on the same feature space, the data is likely to be imbalanced. Each dataset may have a varying amount of data and follow a slightly different distribution. While FedAvg employs weighted averaging based on the number of data samples at each client to handle non-IID data, it can still be sensitive to data imbalance and skewed distributions. In contrast, FedProx is more resilient to these challenges thanks to the inclusion of the proximal term in the objective function [43, 66], as shown in Algorithm 2. It also tolerates client drop-out, but it is more computationally demanding than FedAvg.

Both FedAvg and FedProx algorithms are implemented in our Hyperledger Fabric's chaincode to evaluate their respective performances and determine the potential enhancement introduced by FedProx when working with non-IID data. Section 6.5 in our evaluation chapter is devoted to performing an impact analysis on the performance of the FL model with varying data distributions among the participants.

5.5 Gateway

The previous sections covered the main components of the framework, namely FL client and blockchain. This section provides a detailed description of the *gateway* component, which acts as a bridge that communicates the FL clients — responsible for local model training — and the blockchain — responsible for storing and aggregating the models globally. As explained in Subsection 4.3.3, due to the modular nature of the framework, this component can be completely decoupled from the FL client or it can be part of the same system, depending on the choice of the organisation participating in the cross-silo BCFL system. If the decoupled option is chosen, the organisation's deployment system must ensure that only the corresponding FL client can communicate with the *gateway*, in

order to avoid fraudulent use of it.

To perform its function, the *gateway* has been implemented to maintain a secure and efficient communication channel with both the local FL clients and the blockchain. It consists of a server with a REST interface that receives requests from the client and converts them into transactions that are then sent to the blockchain. It accesses the Hyperledger Fabric blockchain through a certificate-secured gRPC channel that identifies it as a valid network contributor and provides it with the ability to send transactions to be added within the blocks after validation through the corresponding consensus mechanism.

5.6 Technology Stack

This section details the software technologies used for each component. All the technologies described in this section are illustrated in Figure 5.5, which also shows the interconnections between them, giving a detailed overview of the entire BCFL framework implemented in this work.

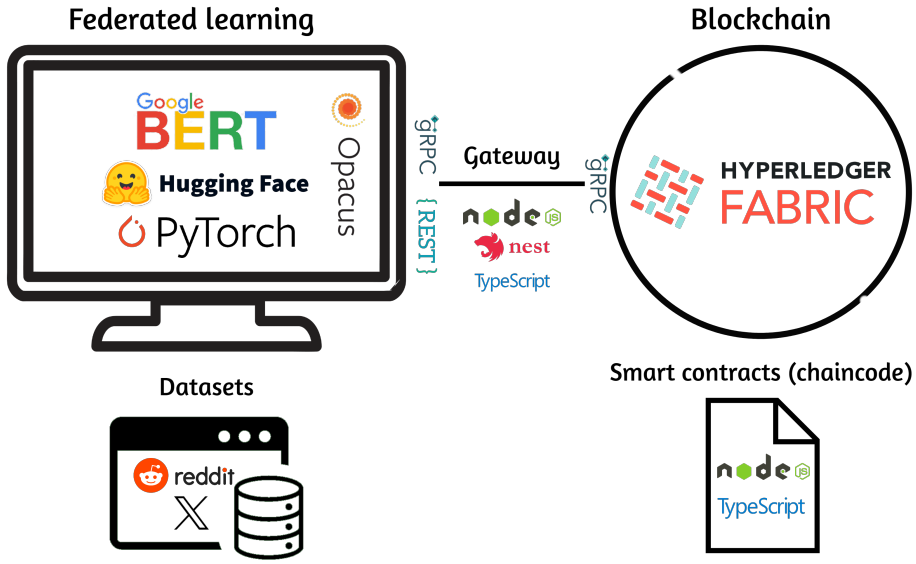

Figure 5.5: Framework's technology stack

As stated in Subsection 5.3.1, the models used for fine-tuning correspond to the lightweight versions of Google's BERT base model: BERT Tiny, BERT Mini, BERT Small and BERT Medium. These models are trained using PyTorch[9], a popular Python framework for

[9]https://pytorch.org/

ML. The models are preloaded from Hugging Face[10], a collaborative platform that hosts models and datasets for the development of ML tools. The implementation of these models can be found at *prajjwal1*[11] Hugging Face repository. They are PyTorch versions of the original models created by Google in their TensorFlow DL development framework. These BERT models are already pre-trained on large data corpora for NLP. We further fine-tune them using the public depression datasets listed in Section 5.2. For the fine-tuning process, we configure the number of BERT model layers that are going to be trained and leave the weights of the rest layers frozen — unchanged. This helps to avoid overfitting the model weights to our comparatively small depression dataset.

Our datasets are loaded from spreadsheets using data processing tools provided by PyTorch. Each dataset is splitted into a training and a test set using an 80/20% ratio. The training partition is divided into as many parts as there are clients participating in the network. The data distribution depends on whether the experiment follows an IID or non-IID data distribution. This enables the simulation of real-world scenarios where varying data distributions among the different clients is likely to occur. Each FL client trains its model using 80% of its data samples and allocates the remaining 20% to the validation set to select the best model weights during its local training. The local training is carried out for several epochs, which are configured before the BCFL system is started and must be consistent across all clients in order to maintain a synchronous flow throughout the framework. If DP is enabled, customers train their local models with Opacus, which adds Gaussian noise to the model parameters in order to strengthen privacy and prevent any type of inference attack.

Prior to sending the local model to the *gateway*, the FL client filters out the parameters of the frozen layers (see Subsection 5.3.1) to save space and computational resources, so that only the parameters of the model that have been fine-tuned are stored on the blockchain and aggregated by the chaincode. Once the updated parameters are obtained, they are serialised using the Python binary serialisation tool msgpack[12]. After serialising the model, the binary file containing the model parameters is compressed using the Python's native zlib library and, finally, encoded as a base64 text string. The base64 string containing the parameter updates resulting from the local training is then sent to the *gateway* via a REST request. This process incurs additional overhead, but guarantees that no parameter data is lost over the communication channel. It also guarantees that the parameters remain compatible across the various technologies comprising the framework — e.g. Python for the FL client, TypeScript for the *gateway* and the chaincode — while also minimising space usage.

The *gateway* component is a REST server built on NestJS[13] framework, based on Express[14], and operates within the Node.js[15] runtime environment. NestJS is built with TypeScript

[10]https://huggingface.co/
[11]https://huggingface.co/prajjwal1/
[12]https://pypi.org/project/msgpack/
[13]https://nestjs.com/
[14]https://expressjs.com/
[15]https://nodejs.org/en/about

as its programming language. This REST server receives requests from its corresponding FL client, intercepts the data sent and generates transactions to be sent to the blockchain. The *gateway* connects to the Hyperledger Fabric blockchain through the Fabric Gateway Software Development Kit (SDK)[16] provided by Hyperledger. This component is a core feature of the Hyperledger Fabric blockchain network and facilitates the coordination of actions required to submit transactions and query ledger state on behalf of client applications. The *gateway* can therefore communicate with the blockchain and submit transactions through a gRPC channel using a certificate issued by a trusted Certificate Authority (CA) over the Transport Layer Security (TLS) protocol. This ensures that only the gRPC channels belonging to the designated organisations are capable of interacting with the permissioned blockchain.

Lastly, the blockchain deployed in our BCFL framework is a test network provided by Hyperledger for sample implementations, namely Hyperledger Fabric Samples[17]. It deploys using Docker Compose[18], which creates a docker[19] container containing each node of the blockchain. The blockchain consists of two peer organisations and an ordering organisation. To simplify the configuration, a single node Raft ordering service is used. Raft has a high transaction throughput due to its leader operation, but it is not tolerant to Byzantine faults. For further details, see Subsection 2.4.3 on consensus mechanisms in the background chapter (Chapter 2).

Our chaincode is deployed on this blockchain and contains the functionality outlined in Subsection 5.4.1. It is programmed in TypeScript using the Fabric Contract API tool[20] provided by Hyperledger Fabric to access the context of the decentralised ledger, performing write and read operations to aggregate the local models and update the global model after each iteration.

Before aggregating the parameters of the trainable layers of the local model instances into the global model, the chaincode decodes their parameters stored in the blockchain in a base64 string, decompresses them using the Node.js zlib library[21] and deserialises them using the @msgpack/msgpack[22] library for TypeScript. Since it is developed in Typescript, the chaincode is able to perform model aggregation using floating-point arithmetic, respecting the decimals of the model parameters, without affecting the accuracy of the model compared to native tensor operations in PyTorch. After creating the new global model, this model undergoes the process of serialisation, compression and encoding before being stored on the blockchain. This enables FL clients to request the global model and load the new global parameters corresponding to the trainable layers into their BERT model instance for a new iteration of training.

[16] https://hyperledger.github.io/fabric-gateway/
[17] https://github.com/hyperledger/fabric-samples
[18] https://docs.docker.com/compose/
[19] https://www.docker.com/
[20] https://hyperledger.github.io/fabric-chaincode-node/main/api/
[21] https://nodejs.org/api/zlib.html
[22] https://www.npmjs.com/package/@msgpack/msgpack

Table 5.2: Framework technology stack summary

Component	Name	Software Type	Version (Date Accessed)	Main Utility
FL client	Python	Programming language	3.11.7	Programming language used for developing DL models within the BCFL framework. Extensive libraries and community support in the fileds of ML and data science.
	PyTorch	Framework	2.1.2	Open-source DL framework known for its flexibility and ease-of-use. Used for creating the training process of our FL clients.
	Hugging Face	Development platform	28/12/2023	Collaborative platform that hosts popular DL models and datasets. We use BERT models from their PyTorch repository.
	BERT Mini	DL model	28/12/2023	BERT model lightweight version used for fine-tuning our BCFL framework.
	Opacus	Library	1.4.0	Python library that offers ready-made solutions to train DL models with DP.
	msgpack	Library	1.0.7	Python binary serialisation library to reduce the size of the model sent to the blockchain and enable proper compression and base64 string encoding.
	zlib	Python native library	3.11.7	Python native library to compress the model sent to the blockchain.
Gateway	TypeScript	Programming language	5.1.3	JavaScript superset programming language that adds types. Used to develop the *gateway* component.
	Node.js	Runtime environment	18.19.3	Based on JavaScript, most popular runtime environment for developing asynchronous tasks.
	NestJS	Framework	10.0.0	Progressive Node.js framework for building efficient server-side applications. Used to create a REST server to allow for the FL client component to communicate with the blockchain.

Component	Name	Software Type	Version (Date Accessed)	Main Utility
Blockchain	@hyperledger/fabric-gateway	SDK	1.4.0	Core feature of the Hyperledger Fabric blockchain network that facilitates the coordination of actions required to submit transactions and query ledger state on behalf of client applications.
	@grpc/grpc-js	Library	1.9.13	Used for efficient and secure interaction between the *gateway* and the blockchain.
	Docker Compose	Multi-container deployment tool	2.21.0	Used to simplify the deployment of the various nodes that form the blockchain, along with their interdependencies.
	Docker	Deployment tool	24.0.7	Used to deploy the blockchain node software inside containers, ensuring they run in their own reliable environment.
	Hyperledger Fabric	Blockchain network	2.5.x	Permissioned blockchain framework implementation that enables the development of decentralised applications with a modular architecture. It is used as a permissioned blockchain for decentralising our FL framework.
	TypeScript	Programming Language	5.3.3	JavaScript superset programming language that adds types. Used to develop the chaincode features. It supports floating-point arithmetic.
	Fabric Contract API	Library	2.5.4	High level API provided by Hyperledger Fabric to access the context of the decentralised ledger, performing write and read operations. It enables the development of chaincode features.
	@msgpack/msgpack	Library	3.0.0-beta2	Library for deserialising the local models received through transactions.
	zlib	Library	5.3.3	Node.js library for decompressing the local models received through transactions. This step is necessary before aggregating the local models.

Moreover, Figure 5.6 displays a comparable image to the framework pipeline outlined in Chapter 4 (see Figure 4.1), illustrating the technologies employed in each stage of the process. This figure offers a good overview of the technologies within the flow of the framework.

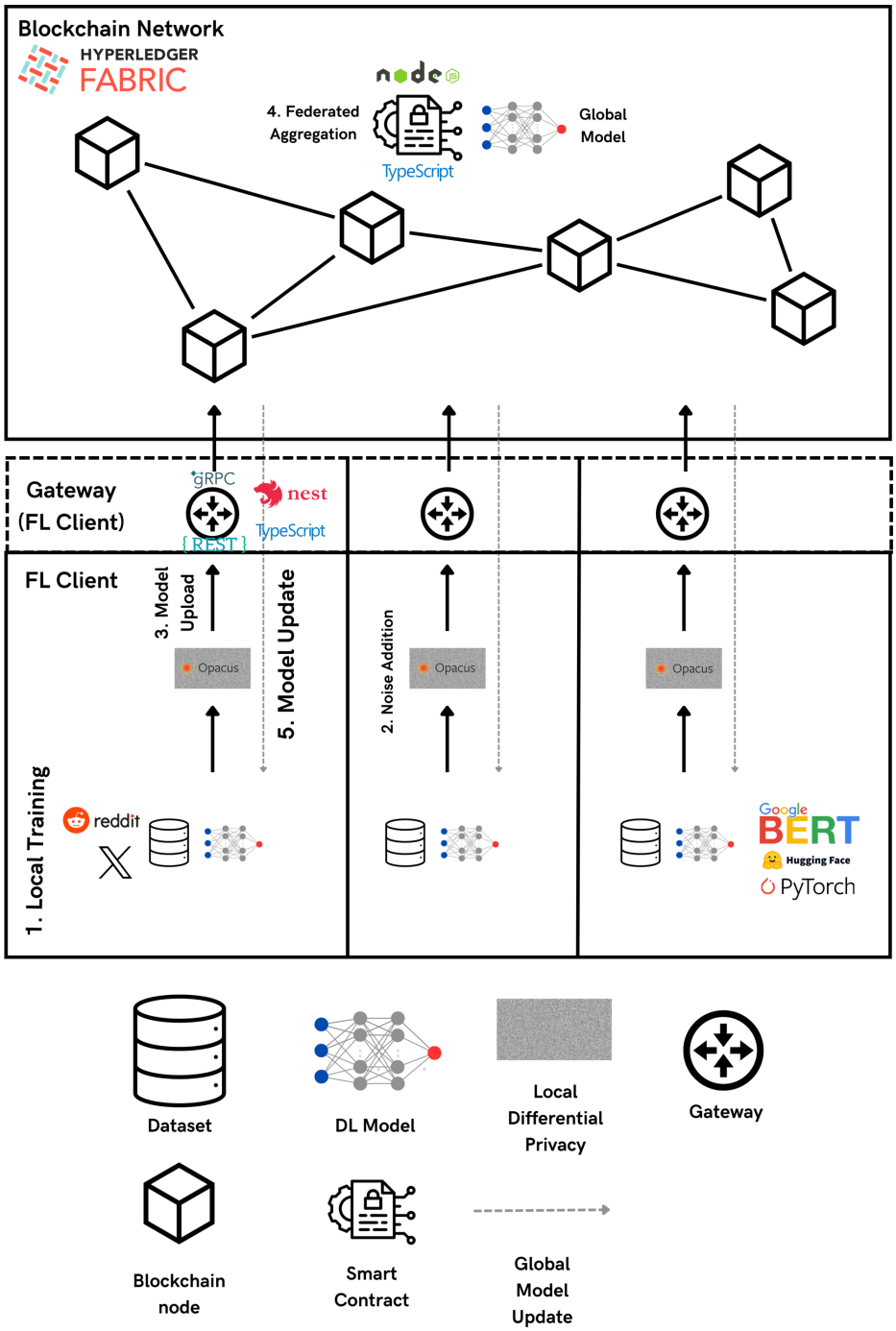

Figure 5.6: Framework's pipeline with technologies

5.7 Configurable parameters

Hyperparameter configuration is a cornerstone of DL, as it is the key to optimal model development. Beyond the configuration of these hyperparameters, our BCFL framework is implemented in such a way that many other aspects can be configured, from the choice of the BERT model to be trained, the dataset to be used for fine-tuning, the number of clients involved on the training, the distribution of data between these clients, the federated algorithm used for aggregation and many others. This enables an extensive ablation study over multiple variables of the framework in order to evaluate the efficiency and functionality of our BCFL framework compared to others, as well as to its centralised FL and centralised DL versions. This section lists the details of the most important parameters, excluding the most typical hyperparameters, which are also configurable in our framework. For a fully detailed list of all the configurable parameters, refer to the software project in our public repository[23].

- `ml_mode: [ml, fl, bcfl]`
  Framework training mode. Useful for performance comparisons between equivalent centralised DL, centralised FL and BCFL pipelines.
- `model: [bert_tiny, bert_mini, bert_small, bert_medium]`
  Choice of the BERT model for the backbone of the FL client.
- `dataset: [twitter_dep, acl_dep_sad, dreaddit, mixed_depression]`
  Selection of the dataset to be used for training the framework.
- `layers: Any valid positive integer from 1 to 3`
  Number of the last encoding and classifier layers in the BERT model that are trainable. The remaining layers are frozen.
- `dp_epsilon: Any valid positive float between 0 and 15`
  ϵ (epsilon) value to control the DP strength. Lower values of ϵ correspond to stronger privacy. Setting the value of ϵ to 0 disables DP.
- `fed_alg: [fedavg, fedprox]`
  Federated learning algorithm for aggregating the local models into a global model.
- `data_distribution: [iid, non_iid]`
  Data distribution among FL clients. The term IID refers to data that is independent and identically distributed.
- `num_clients: [Any valid positive integer between 1 and 15]`
  Number of FL clients participating in the training. Maximum limit set to 15 because of limitation to cross-silo environment.

[23] https://github.com/vdevictor96/fabric-federated-learning/blob/main/federated-learning/client/config/default_config.json

Chapter 6
Evaluation

This chapter presents the ablation study conducted on our BCFL framework implemented in the previous chapter (Chapter 5), including a thorough analysis of the results. In more detail, this chapter begins with the specification of the environment in which the experiments are carried out. We then describe the different hyperparameters and the setup configurations. Subsequently, the results of these experiments are displayed and analysed. Next, the DP strength and the data distribution variables are isolated in order to make a specific impact study analysis of each of thems. Lastly, we conclude with a brief discussion of the lessons learned from the analysis of the different results.

6.1 Experimental Setup

To conduct the ablation study, we deployed our BCFL framework in an environment with the hardware specifications outlined in Table 6.1. It is important to note that these specifications correspond to a collaborative environment, although its use has been isolated to this study, so the performance of the hardware is not be affected by other processes running on the same system. However, the results regarding the execution times of the different framework configurations are highly dependent on the available resources and should not be considered definitively conclusive.

The experiments leverage the rapid processing advantage of using GPUs for DL training, thanks to the implementation of the model training on PyTorch, which supports CUDA interface. CUDA is a programming model and computing toolkit developed by NVIDIA that enables faster compute-intensive operations by parallelising tasks across GPUs.

Hardware	Specifications
CPU	2x Intel Xeon Gold 5317 @ 3.00GHz, 2x 12 Cores / 24 Threads
GPU	NVIDIA A100 Nvidia Tesla 80GB PCIe
RAM	1024GB = 16 x 64GB DDR4 @ 1600MHz
Disk Storage	2x 480GB SSD, 2x 1.9TB SSD
OS	64bit MPI Linux distribution based on Debian

Table 6.1: Hardware and OS specifications

6.2 Hyperparameters

Before running the experiments, a preliminary study is conducted to set the default values for some of the hyperparameters that have a high influence on the training, yet an in-depth study for extensive hyperparameter optimization is beyond the scope of this project. The PyTorch Transformer Trainer tool has been used to set these parameters in order to conduct quality experiments. This tool includes a method called `hyperparameter_search`, which utilizes the Optuna[1] framework to define a hyperparameter space. This method performs trials with random configurations within the defined space to determine the best hyperparameters for the object of study. Furthermore, we have conducted several experiments to evaluate the accuracy of the resulting models empirically and determine the optimal hyperparameters for the DL model in all subsequent experiments. The following hyperparameters have been examined:

- *Learning rate.* The learning rate has been set to a value of $6e-05$ as it resulted in the best performance across the various datasets when fine-tuning the different BERT models. This is in line with the original papers on BERT fine-tuning [72, 14]. It is important to note that this value differs when DP is added, as explained in the corresponding section (Section 6.4).
- *Number of epochs.* The epochs are set to three for each local training round for a total of four rounds, which results on a total of twelve iterations. For experiments trained with the centralised DL training mode, a total of twelve training iterations are performed. As this is a fine-tuning process on pre-trained BERT models, a large number of iterations over the data is unnecessary to achieve good accuracy. Too many iterations would be counterproductive as the model may overfit the dataset [22]. This values are similar to those used in the original papers on BERT fine-tuning.
- *Trainable layers.* The optimal balance between computational and spatial cost and accuracy in our framework is achieved by fine-tuning only the three last layers. As discussed in Subsection 5.3.1, previous work has shown that by fine-tuning

[1] https://optuna.org/

just one fourth of the final layers, the model can achieve 10% parity with the fully fine-tuned model and, in some cases, even outperform it, as shown in the literature [40, 72]. The rest of the model layers remain frozen. These layers correspond to the classification, the pooler and the last encoding layer. This approach yields the best results for the model, preventing overfitting and significantly reducing the number of modified parameters — hence the size — that need to be sent and stored in the blockchain.

- *Batch size.* For the batch size, a value of four is appropriate. This is because the partitions of the dataset trained by each client are small, and a larger batch size could lead to reduced generality of the models [22].
- *Number of clients.* The number of clients is set to five as it is considered a sufficiently representative number for a cross-silo network, without adding complexity to the deployment of the test network developed for the experiments. Furthermore, our datasets are relatively small, with only around three thousand samples each. If partitioned among too many clients, each would have too little data. For completeness, we also conducted experiments with three to fifteen clients to assess the impact of varying client numbers on accuracy and execution time. Details of these experiments can be found in Section 6.6.

6.3 Performance Analysis

This section provides an overview of the experimental results of our BCFL framework on different setups. Later, each of the experiment outcomes is analysed and the main takeaways are highlighted. The first phase of the ablation study involves the performance analysis of the framework by examining in depth on the following three different aspects.

- *Accuracy.* Accuracy (A) is commonly used metric defined as the ratio of true positives (TP) and true negatives (TP) to the sum of all the samples — true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). This metric is applied to the test dataset partition after the model had been successfully fine-tuned using the training data. Formally, the accuracy metric is given by:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

- *Execution time.* We assess the total training time of our BCFL framework, compared to the centralised FL equivalent for a fixed number of aggregation rounds and local training epochs. This evaluation allows us to estimate the overhead introduced by decentralising the framework using blockchain. We also include the execution times for the centralised DL training mode.

- *Convergence.* Besides the test accuracy, we also study the convergence of the models.This metric shows the evolution of the model accuracy on the training data and helps us assess how the different training methods affect the efficacy and speed of the model learning. This metric also shows how the training performance relates to their generalisation on test data.

To facilitate the understanding of the experiment results on this section, the different setup configurations are listed. The impact of the level of DP and the distribution of data among FL clients are discussed in separate sections (Section 6.4 and Section 6.5 respectively), as they are considered distinctively relevant in the context of cross-silo FL. The setup variables are the following:

- *ML training mode.* The framework operates in three distinct training modes, namely BCFL, centralised FL and centralised DL. In centralised FL, the updated parameters of each FL client are gathered in the deployed Python client and aggregated within the same software component, whereas in BCFL mode, the parameters are sent individually to the blockchain. The centralised DL training mode trains a single BERT model with all the data of the dataset, not partitioned as in the federated training modes.
- *BERT model.* The experiments are carried out on the four different lightweight versions of the BERT model specified in Subsection 5.3.1 of the implementation chapter.
- *Dataset.* Each model used in the experiments is fine-tuned on the four datasets specified in Section 5.2 of the implementation chapter. During the BCFL and FL mode training, the datasets are divided into equivalent-sized partitions, one for each of the FL clients. The experiments follow a data partitioning scheme that ensures an IID data distribution. This is done in order to prevent any potential impact of data imbalance on the experimental results presented in this section. The assessment of the impact of data distribution in the framework performance is presented in Section 6.5.

All setup configurations were executed ten times, each with a different random seed. Therefore, the displayed results show the mean and standard deviation of the executions, which gives an insight into the robustness of each different configuration. For the standard deviation, we used the formula for the sample standard deviation instead of the standard formula, because the population of accuracy results is infinite — as many results as there are random seeds to run the model with. Therefore, the formula used in the tables is as follows:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{6.2}$$

The results of the experiments together with all the training logs can be found in the public repository of the project — in the `ablation_study` folder — along with the software implementation[2].

6.3.1 Test Accuracy

Table 6.2 displays the results of fine-tuning the four distinct BERT models on the four depression datasets using the three different training modes listed in the previous section, resulting in a total of forty-eight combinations. We can observe that the best results for all datasets are mostly achieved with the BERT Medium model, as it is the most powerful model. However, the improvement in comparison to BERT Small is minimal and, in some cases, even worse. This is due to BERT Medium complexity being unable to improve generalisation on the limited sized datasets over a lighter version like BERT Small, having the risk of overfitting to the training data. This is made clear in the convergence of the different models (Figures 6.1, 6.2, 6.3 and 6.4), where we can see that the convergence point of BERT Medium and BERT Small is almost the same — sometimes achieving 100% accuracy on training data — regardless of the difference in model complexity.

It is noteworthy that the results obtained by the framework trained with federated aggregation on the blockchain (BCFL) and with centralised FL show equivalent accuracies. This demonstrates that our BCFL framework can ensure decentralisation control, trustless collaboration and protect against limitations inherent to FL, such as SPOF and availability attacks (see Section 2.5 in Chapter 2) without having a negative impact on the accuracy. These findings are consistent with previous work on decentralised FL using permissioned blockchains [51, 48], which demonstrated that the blockchain did not interfere with the FL process. Conversely, the results diverge from those of studies on decentralised FL using a permissionless blockchain based on PoW consensus mechanisms [82]. In this work the authors conducted a performance evaluation and reported that the blockchain-enabled approach achieved 80% of the evaluation accuracies of the centralised baseline approach. Another example of a BCFL framework using a public blockchain, achieving lower accuracies to those of centralised FL can be found in the research [44]. Our BCFL framework is able to achieve these results thanks to the use of Hyperledger Fabric as a permissioned blockchain, which uses general-purpose languages and allows the execution of floating-point number arithmetic (see Subsection 4.3.2 and Section 5.4). Furthermore, as we are fine-tuning only the last three layers of the model, this approach enables sending only the parameters of these layers to the blockchain. This reduces the computational and communication overhead introduced by using the blockchain for aggregation, allowing for precise calculations for the aggregation, equivalent to those of a centralized server. For more information on the model and layer sizes, please refer to the Subsection 5.3.1.

[2] https://github.com/vdevictor96/fabric-federated-learning/tree/main/ablation_study

Table 6.2: Average test accuracy results. Comparison of BERT models across different ML training modes and datasets. The results show the mean and standard deviation accuracy percentages for ten executions with different random seeds.

ML Training Mode	Dataset	BERT Tiny	BERT Mini	BERT Small	BERT Medium
Centralised DL	twitter_dep	79.48 ± 0.93	80.23 ± 0.75	82.17 ± 1.32	82.28 ± 1.03
	acl_dep_sad	91.82 ± 0.50	92.80 ± 1.04	93.77 ± 0.83	93.63 ± 0.50
	dreaddit	72.60 ± 0.88	74.52 ± 0.82	75.51 ± 1.09	76.78 ± 1.07
	mixed_depression	87.14 ± 0.45	87.30 ± 0.76	88.72 ± 0.74	88.64 ± 0.98
Centralised FL	twitter_dep	78.26 ± 0.90	80.62 ± 1.09	82.01 ± 0.64	81.81 ± 0.64
	acl_dep_sad	87.35 ± 0.64	88.89 ± 0.34	92.06 ± 0.48	92.97 ± 0.66
	dreaddit	69.65 ± 0.45	74.19 ± 0.58	75.00 ± 0.87	75.93 ± 0.80
	mixed_depression	84.15 ± 0.75	85.46 ± 0.65	87.91 ± 0.56	86.97 ± 0.85
BCFL	twitter_dep	78.50 ± 0.86	80.74 ± 0.72	81.86 ± 0.68	81.72 ± 0.38
	acl_dep_sad	87.52 ± 0.52	89.02 ± 0.69	91.68 ± 0.54	93.32 ± 0.52
	dreaddit	69.24 ± 0.96	74.01 ± 0.66	74.87 ± 0.81	75.70 ± 0.77
	mixed_depression	84.06 ± 0.75	85.74 ± 0.78	87.92 ± 0.44	87.25 ± 0.87

We can also observe slightly better accuracies for models trained in centralised DL mode compared to their federated version. The reason for this is that for centralised FL and BCFL, we use the same total amount of data as in centralised DL, but partitioned among the FL clients. It is important to note that this is a hypothetical comparison used to assess the performance of the FL paradigm. In a real-world scenario, gathering all the private data from the different organisations on a central server to apply a centralised DL model would not be possible, as this would completely destroy data privacy safeguards. Keeping this in mind, in a real-world application, the FL and BCFL frameworks are able to collect more data — one dataset belonging to each organisation — in a secure manner than a centralised DL, potentially resulting in better accuracy results.

After comparing the sizes of the available BERT models in this study (refer to Figure 5.4 in Subsection 5.3.1) to the average test accuracy results achieved for each one (see Table 6.2), we argue that BERT Small is the best choice in terms of complexity/performance ratio, as it achieves almost as good accuracy results on the test partition for all datasets as BERT Medium while being $\frac{7}{10}$ of the size after encoding and compression. Therefore, we have selected this model as the default option in the subsequent sections on the studies on the impact of DP strength (Section 6.4), data distribution (Section 6.5) and number of clients (Section 6.6).

6.3.2 Execution Time

Table 6.3 displays the average duration and the standard deviation of the ten runs for each of the forty-eight different combinations tested in this performance analysis. The times are shown in minutes and seconds. These trainings are considerably fast as they use lightweight BERT models (see sizes in Subsection 5.3.1) and only the last three layers are fine-tuned. However, it is important to note that the execution was carried out in an isolated environment using an NVIDIA Tesla A100 with 80GB (refer to Section 6.1). The GPU's numerous CUDA cores enable a high level of parallelisation. Therefore, in the centralised training modes, there is barely any increase in training time among the different BERT models, as the wider layers can still be executed in parallel thanks to the combination of the transformer technology and the hardware computational capabilities.

Table 6.3: Average test execution times. Comparison of BERT models across different ML training modes and datasets. The results show the mean and standard deviation execution time for ten executions with different random seeds. The times correspond to minutes and seconds.

ML Training Mode	Dataset	BERT Tiny	BERT Mini	BERT Small	BERT Medium
Centralised DL	twitter_dep	01:21 ± 00:07	01:24 ± 00:03	01:39 ± 00:01	02:23 ± 00:01
	acl_dep_sad	01:37 ± 00:33	01:25 ± 00:10	01:42 ± 00:03	02:28 ± 00:02
	mixed_depression	01:19 ± 00:06	01:23 ± 00:02	01:42 ± 00:01	02:23 ± 00:01
	dreaddit	01:29 ± 00:11	01:30 ± 00:11	01:53 ± 00:00	02:44 ± 00:00
Centralised FL	twitter_dep	01:45 ± 00:02	01:59 ± 00:02	02:06 ± 00:02	02:43 ± 00:01
	acl_dep_sad	01:38 ± 00:01	01:51 ± 00:02	02:00 ± 00:01	02:38 ± 00:01
	mixed_depression	01:22 ± 00:02	01:35 ± 00:02	01:42 ± 00:02	02:16 ± 00:02
	dreaddit	01:52 ± 00:02	02:08 ± 00:02	02:14 ± 00:02	02:51 ± 00:01
BCFL	twitter_dep	02:00 ± 00:03	02:52 ± 00:02	06:03 ± 00:07	06:38 ± 00:07
	acl_dep_sad	01:54 ± 00:02	02:52 ± 00:04	05:58 ± 00:09	06:34 ± 00:11
	mixed_depression	01:38 ± 00:04	02:34 ± 00:05	05:38 ± 00:10	06:12 ± 00:08
	dreaddit	02:09 ± 00:03	03:04 ± 00:05	06:24 ± 00:10	06:45 ± 00:05

The framework in BCFL mode adds a communication overhead, which involves sending the local models to the blockchain and the subsequent communication between nodes on the blockchain to reach a consensus. The aggregation algorithm does not add any overhead compared to aggregation on a central server. The larger the model, the more noticeable the communication overhead between blockchain nodes becomes, as it multiplies by several factors while propagating to multiple nodes before reaching consensus. However, this increase in execution time is promising as it does not exceed a factor of x3 compared to the centralised FL mode on the more complex models, while maintaining the same accuracy. It is important to note that this blockchain was deployed for testing purposes with only three nodes and a Raft consensus mechanism (see Section 5.4). In blockchains with a greater number of nodes, the execution time of the framework could potentially be multiplied by a larger factor.

In our prototype framework, the centralised federated learning does not add any communication overhead to a central server because the aggregation occurs within the same system. However, in a real-world application, some communication overhead may be expected. Although, it is important to note that, in our experiments, the FL clients are trained using the same Python process in separate threads, which cannot be considered a fully parallelised process. Therefore, the potential parallelisation in the case where the FL clients run in separate processes would have a positive impact on the duration of the fine-tuning in FL mode and could compensate for the delay introduced by the communication with the central aggregation server.

6.3.3 Convergence

To compare the convergence of the centralised model with the model trained in a decentralised and distributed fashion using BCFL, we flattened the local training epochs of the five clients from each federated training round in order to have a comparable time sequence to the twelve iterations on centralised DL. Centralised FL training mode is ignored as its accuracy per epoch is equivalent to that of the BCFL.

We can see that in most cases in BCFL there is a small drop in accuracy every three iterations. The reason for this drop is that aggregation takes place every round — which involves three local epochs — and the local models are then replaced by the global averaged one. This averaged model generalises better to the classification task in hand, but is less refined to each partition of the training dataset, resulting in a drop in training accuracy. This effect is more pronounced in the more complex models (BERT Small and BERT Medium) because they overtrain more on their small sized local partition of the dataset, therefore the aggregation has a bigger effect on theses models.

When observing the evolution per epoch from the convergence plots presented in this section (Figures 6.1, 6.2, 6.3 and 6.4), it becomes clear that the complexity of the BERT model has a direct relationship with its maximum accuracy in the training set. However, this does not necessarily mean that more complex models will generalise better, as

demonstrated by the accuracy in the test set in Table 6.2. We can also observe that the BCFL aggregation compensates for overtraining in comparison to centralised DL training, as the training accuracy is more closely aligned with the test accuracy.

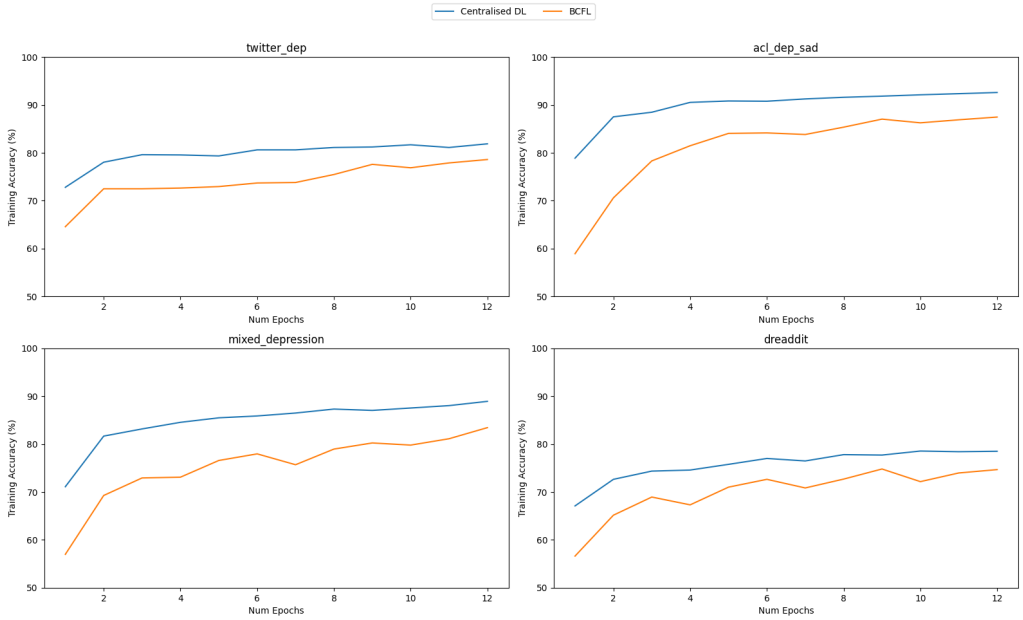

Figure 6.1: BERT Tiny training accuracy per epoch. The x-axis represents the number of epochs for the training. The y-axis displays the training accuracy value achieved by the BERT model at the given epoch. The plot compares the convergence speed between centralised DL and BCFL for the four fine-tuned datasets. Centralised FL training mode is ignored as its accuracy per epoch is equivalent to that of the BCFL.

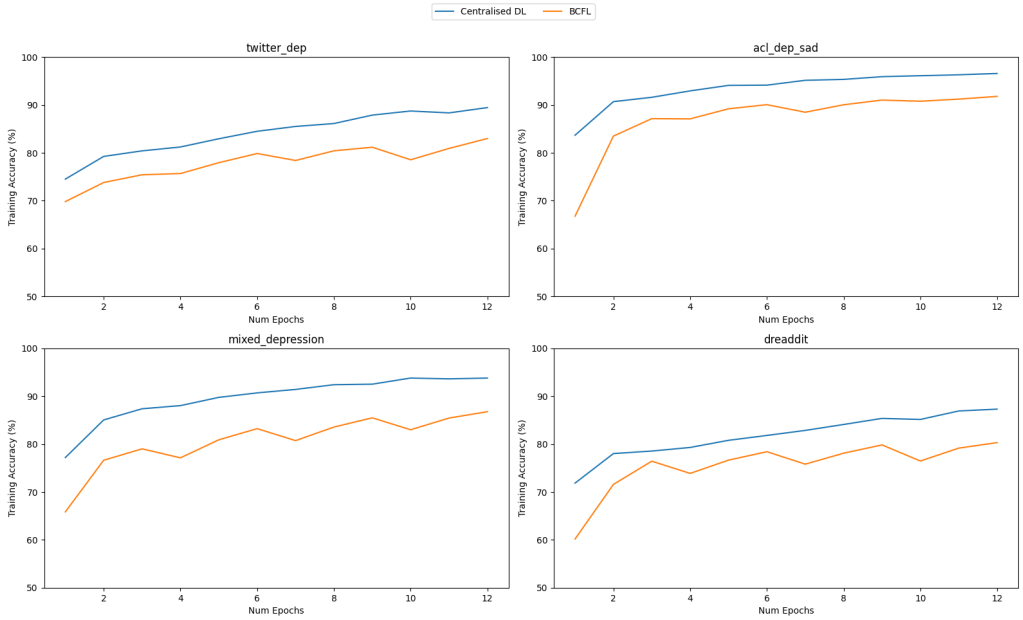

Figure 6.2: BERT Mini training accuracy per epoch. The x-axis represents the number of epochs for the training. The y-axis displays the training accuracy value achieved by the BERT model at the given epoch. The plot compares the convergence speed between centralised DL and BCFL for the four fine-tuned datasets. Centralised FL training mode is ignored as its accuracy per epoch is equivalent to that of the BCFL.

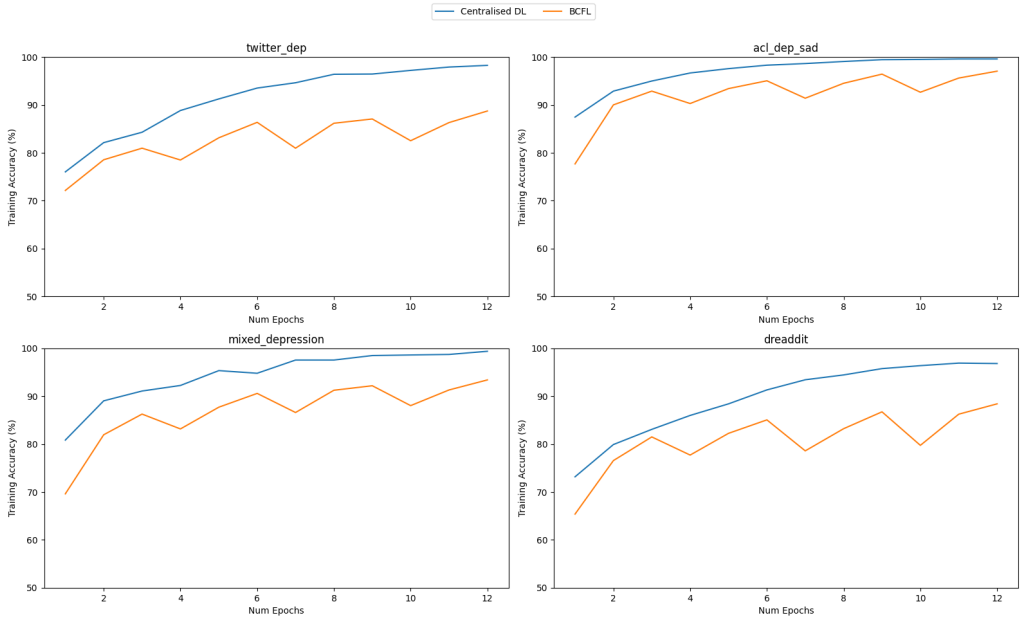

Figure 6.3: BERT Small training accuracy per epoch. The x-axis represents the number of epochs for the training. The y-axis displays the training accuracy value achieved by the BERT model at the given epoch. The plot compares the convergence speed between centralised DL and BCFL for the four fine-tuned datasets. Centralised FL training mode is ignored as its accuracy per epoch is equivalent to that of the BCFL.

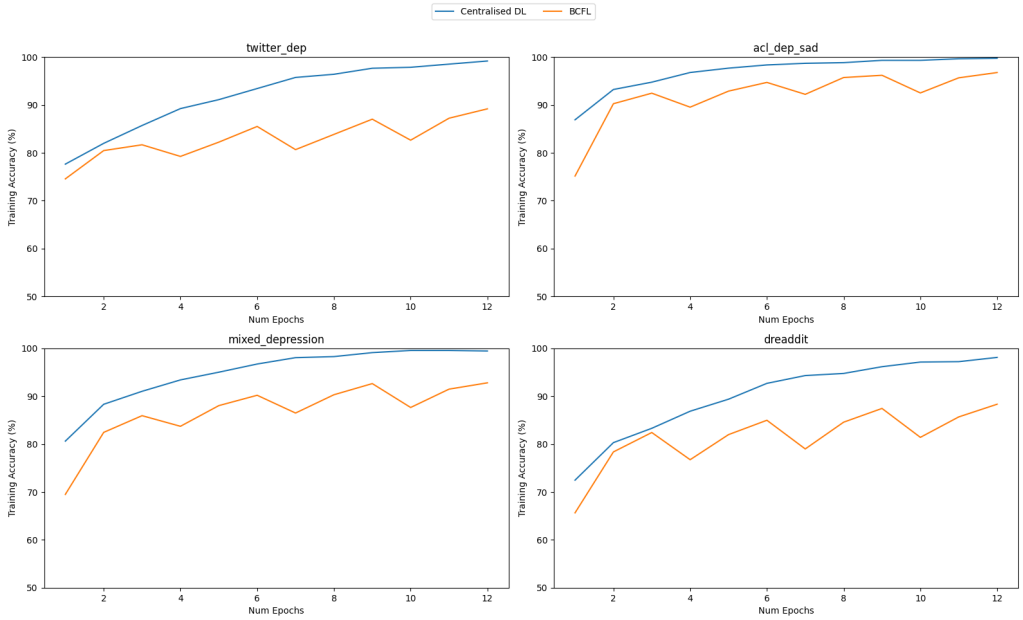

Figure 6.4: BERT Medium training accuracy per epoch. The x-axis represents the number of epochs for the training. The y-axis displays the training accuracy value achieved by the BERT model at the given epoch. The plot compares the convergence speed between centralised DL and BCFL for the four fine-tuned datasets. Centralised FL training mode is ignored as its accuracy per epoch is equivalent to that of the BCFL.

6.4 Impact Analysis of Differential Privacy

This section is devoted to evaluate the impact on model performance of the DP mechanism discussed previously in Subsection 2.3.1 and Subsection 5.3.2 of Chapter 2 and Chapter 5, respectively. As discussed in these previous sections, DP is a mechanism used by the FL clients to introduce noise to their model parameters before being sent to the blockchain — or any other aggregation system — in order to prevent potential inference attacks on their private data by identifying patterns on their iterative model updates. This privacy mechanism does not come without a price, as introducing noise in the model parameters often leads to a reduction in the performance of the model. Therefore, it is crucial to analyse different configurations to find an appropriate trade-off between security and performance.

In Table 6.4 we present the performance results for different values of ϵ (epsilon) — parameter that controls the privacy strength of the DP — with lower values of epsilon conveying higher degrees of privacy. We also compare them with the models trained without adding any DP. The impact analysis is conducted on the BERT Small transformer model fine-tuned with the four different datasets available on the three different training modes that our framework allows — centralised DL, centralised FL and BCFL. For completeness, the average execution times for the ten runs of each combination are also displayed in Table 6.5.

Table 6.4: Differential Privacy (DP) impact average accuracy results. Comparison of the impact of DP ϵ (epsilon) values across different ML training modes and datasets on BERT Small. Lower values of ϵ imply higher privacy guarantees. The results show the mean and standard deviation accuracy percentages for ten different executions with different random seeds.

ML Training Mode	**Epsilon**	**twitter_dep**	**acl_dep_sad**	**mixed_depression**	**dreaddit**
Centralised DL	0.5	71.47 ± 1.63	82.24 ± 1.81	77.23 ± 2.33	65.70 ± 1.45
	10	71.31 ± 2.08	84.52 ± 1.20	76.72 ± 0.82	65.42 ± 1.15
	∞ (No noise)	82.02 ± 1.17	93.74 ± 0.51	88.71 ± 0.60	76.11 ± 1.20
Centralised FL	0.5	61.15 ± 7.52	59.77 ± 1.47	60.04 ± 6.40	55.45 ± 3.84
	10	61.67 ± 6.60	79.06 ± 5.33	73.14 ± 4.19	57.97 ± 4.74
	∞ (No noise)	81.86 ± 0.70	91.79 ± 0.60	88.05 ± 0.50	75.24 ± 0.87
BCFL	0.5	60.68 ± 6.94	58.84 ± 0.19	61.49 ± 6.49	53.54 ± 2.71
	10	59.74 ± 6.69	78.14 ± 3.50	74.33 ± 2.73	57.90 ± 4.88
	∞ (No noise)	81.83 ± 0.88	91.90 ± 0.33	87.93 ± 0.59	74.80 ± 1.16

Table 6.5: Differential Privacy (DP) impact average execution times. Comparison of the impact of DP ϵ (epsilon) values across different ML training modes and datasets on BERT Small. Lower values of ϵ imply higher privacy guarantees. The results show the mean and standard deviation execution time for ten executions with different random seeds. The times correspond to minutes and seconds.

ML Training Mode	**Epsilon**	**twitter_dep**	**acl_dep_sad**	**mixed_depression**	**dreaddit**
Centralised DL	0.5	03:57 $\pm$ 00:11	04:13 $\pm$ 00:10	03:44 $\pm$ 00:07	04:42 $\pm$ 00:04
	10	04:20 $\pm$ 00:16	04:55 $\pm$ 00:08	04:12 $\pm$ 00:06	05:15 $\pm$ 00:06
	∞ (No noise)	01:38 $\pm$ 00:02	01:42 $\pm$ 00:00	01:43 $\pm$ 00:01	01:54 $\pm$ 00:00
Centralised FL	0.5	05:00 $\pm$ 00:07	05:02 $\pm$ 00:09	04:29 $\pm$ 00:07	05:24 $\pm$ 00:04
	10	06:23 $\pm$ 00:07	06:10 $\pm$ 00:05	05:52 $\pm$ 00:06	06:48 $\pm$ 00:09
	∞ (No noise)	02:03 $\pm$ 00:03	01:59 $\pm$ 00:02	01:37 $\pm$ 00:02	02:08 $\pm$ 00:02
BCFL	0.5	08:49 $\pm$ 00:08	08:47 $\pm$ 00:11	08:19 $\pm$ 00:11	09:15 $\pm$ 00:15
	10	10:07 $\pm$ 00:07	09:51 $\pm$ 00:12	09:37 $\pm$ 00:06	10:29 $\pm$ 00:11
	∞ (No noise)	06:10 $\pm$ 00:08	06:12 $\pm$ 00:11	05:43 $\pm$ 00:06	06:25 $\pm$ 00:09

For the models where no DP is applied, the default hyperparameter configuration defined by the Trainer tool is used, specified in the hyperparameter section of this chapter (Section 6.2). However, the models with DP for both centralised DL training and FL required a higher learning rate, ranging from $6e-04$ to $6e-03$, to cope with the noise and be able to learn from the data. Fine-tuning on `twitter_dep` dataset also required smaller training steps by reducing the training batch size from 4 to 2 [75].

In addition to the epsilon value to control the amount of noise, the selection of the gradient clipping threshold is also crucial when configuring DP [1]. Gradient clipping limits the sensitivity of the algorithm to any single data point by restricting the magnitude of the gradients for each data point before averaging them together. For our experiments, we set the maximum gradient norm for gradient clipping to 1.5, as recommended by Arasteh et al. [3] in their comprehensive ablation study on DP applied to DL on medical data. The specific hyperparameter settings for each experiment can be found in the `ablation_study` section of the public repository, along with the implementation of the BCFL framework.

Similar to the results in the previous section, when no noise is added, i.e. when no DP mechanism is applied, the accuracy results for centralised FL and BCFL are very close to those for centralised DL. This is consistent with the rationale presented in the previous section on general performance analysis. Equally, the BCFL training mode produces equivalent accuracy results as its centralised equivalent, thanks to the use of floating-point number arithmetic, which allows for decentralised and trustless aggregation without having a negative impact on accuracy — see Section 6.3 for more details.

Considering the case where DP is applied, we can see that the accuracy of the model does indeed degrade with the addition of a certain level of noise. The degradation of model accuracy follows a trend proportional to the level of security applied. The models trained with lower epsilon values are the most secure but also the least accurate across the four datasets. This is a typical outcome when adding Gaussian noise, as models with higher privacy guarantees — more noise — are less capable of generalising to the data they are trained on, resulting in poorer performance when tested on the dataset's test partitions.

Additionally, we observe that models trained with FL generally degrade more than centrally trained models when DP is applied. Basu et al. [7] — from which we obtained the `twitter_dep` dataset — also evaluated various BERT model architectures on this specific dataset. Similar to our findings, their results on the Depression Dataset for FL-IID training also show higher degradation of the model accuracy for FL scenarios compared to those on centralised DL. It is noticeable that the reduction in their accuracy results does not always show a directly proportional relationship with the level of DP, with loose privacy guarantees giving worse performance in some cases than when more noise is applied. However, no explanation is provided for this. We hypothesise that this more pronounced deterioration when adding DP in FL scenarios is due to the fact that in these

scenarios the noise is added to smaller amounts of data. The noise is added in each FL client, which are only fine-tuned on a partition of the dataset, making this training method less resilient to DP, as studies such as Tramer et al. [75] suggest that more data benefits the utility of the model for a fixed DP ϵ (epsilon) value by allowing training on more steps.

Finally, with regard to the average execution times shown in Table 6.5, we can observe that training with DP introduces a considerable overhead compared to the case where no noise is added. In a counterintuitive manner, the models trained with a higher epsilon value, which provides less privacy guarantees, take longer than when a lower value is used. For the centralised DL on the `twitter_dep` dataset, the results show higher times compared to others due to the influence of other processes running on the shared environment used for the experiments. Therefore, this should be ignored.

6.5 Impact Analysis of Data Distribution

FedProx is a federated aggregation algorithm designed to maintain performance in cases where the distribution of data across clients is not IID. It is based on FedAvg, but adds a proximal term to the loss function that penalises local models that are too far away from the global model of the previous iteration, thus achieving a more homogeneous parameter distribution (see Subsection 2.2.2).

During this impact study, we observed that FedAvg performed well even when the data was imbalanced — with some clients having only 10% of samples from one class. In these cases, FedAvg yielded accuracy results that were close to, if not equal to, those obtained with balanced data. Therefore, FedProx was unable of improving upon the baseline results set by FedAvg algorithm. Similarly, in this study by Katz et al. [37], the performance of FedAvg and FedProx was tested in four different data distribution scenarios. The study revealed that FedProx outperformed FedAvg only when some of the clients did not have samples of certain classes. Additionally, their data was divided across fifty clients, but only seven clients were involved in each training round, which meant that only $\frac{7}{50}$ of the data was used in each round, resulting in a significant reduction in the amount of training data in addition to the increased imbalance. We emulated a similar scenario by reducing the amount of data in the `twitter_dep` dataset. FedProx performed better than FedAvg on less extreme data distributions, where all clients had at least 10% of samples from each class, with FedAvg achieving 61.88% accuracy and FedProx achieving 69.86%.

To conduct a thorough study on the impact of data distribution and the performance of FedAvg and FedProx algorithms, we compare balanced data escenarios with extreme non-IID scenarios where the dataset is divided into as many partitions as clients, with some clients having no samples of one or more classes. Although this is less common in binary classification — which is the case for our datasets — such scenarios have significant applicability to multi-class classification, for which our framework is also fully

functional. Our framework could also be applied to datasets that classify multiple classes of mental health diseases, such as depression, anxiety, PTSD and anorexia, among others. In these cases, it is interesting to determine whether FedProx would be an effective method for protecting model performance against imbalanced data.

Table 6.6 shows the different accuracy results for the BERT Small transformer model trained on BCFL training mode across the four datasets. Each dataset is tested on four different settings, using FedAvg and FedProx for IID and non-IID data distributions. For completeness, the average execution times for the ten runs of each combination are also displayed in Table 6.7.

Table 6.6: Data distribution impact average accuracy results. Comparison of the impact of data distribution among clients across different datasets and federated aggregation algorithms on BERT Small over BCFL training mode. The results show the mean and standard deviation accuracy percentages for ten executions with different random seeds.

Data Dist.	**Fed. Agg. Algorithm**	**twitter_dep**	**acl_dep_sad**	**mixed_depression**	**dreaddit**
IID	FedAvg	82.18 ± 0.66	91.55 ± 0.77	88.26 ± 0.76	75.02 ± 1.05
	FedProx	82.11 ± 0.49	89.95 ± 0.49	86.93 ± 0.46	74.62 ± 0.65
Non-IID	FedAvg	75.20 ± 1.43	86.84 ± 1.15	75.46 ± 2.19	55.66 ± 1.36
	FedProx	71.33 ± 5.48	80.70 ± 3.15	82.09 ± 0.95	63.65 ± 2.87

For IID data distribution, FedAvg yields better results than FedProx. However, there is not much difference in accuracy between the two federated aggregation algorithms across the four different datasets. This suggests that FedProx could be a valuable option to be used in general case scenarios where there is no certainty about the distribution of the data, as it improves on non-IID data, whilst its performance is resilient in other cases of IID.

When dealing with non-IID data, FedProx clearly outperforms FedAvg in `mixed_depression` and `dreaddit` dataset — by a minimum of 7%. We find that FedProx struggles particularly with imbalanced datasets, such as `twitter_dep` and `acl_dep_sad`. This is because when creating extreme non-IID scenarios with some partitions lacking samples of a certain class, the imbalanced nature of the dataset increases the total number of clients that do not receive the less representative classes. As a result, the proximal term added by FedProx is unable to improve upon baseline FedAvg. Simply put, an excessive imbalance in client data inhibits the improvement of FedProx over FedAvg. To confirm this hypothesis, we fine-tuned the BERT Model in the same non-IID scenario

using a balanced version of `twitter_dep` — achieved by removing negative samples. We found that for this balanced version of the dataset, FedProx indeed achieved better accuracy than FedAvg — 54.36% and 31.39% accuracy respectively — in contrast to the results with the original `twitter_dep` data, which we can find in Table 6.6.

Table 6.7 shows the mean execution times for the ten runs of each combination. We can observe that the introduction of the FedProx function results in an overhead of approximately 25% due to the additional time required for the calculation of the proximal term in the loss function.

Table 6.7: Data distribution impact average execution times. Comparison of the impact of data distribution among clients across different datasets and federated aggregation algorithms on BERT Small over BCFL training mode. The results show the mean and standard deviation execution time for ten executions with different random seeds. The times correspond to minutes and seconds.

Data Dist.	**Fed. Agg. Algorithm**	**twitter_dep**	**acl_dep_sad**	**mixed_depression**	**dreaddit**
IID	FedAvg	05:50 ± 00:11	06:05 ± 00:13	05:42 ± 00:07	06:16 ± 00:10
	FedProx	07:37 ± 00:19	07:54 ± 00:04	07:29 ± 00:07	08:21 ± 00:06
Non-IID	FedAvg	05:38 ± 00:13	05:55 ± 00:06	05:14 ± 00:19	06:00 ± 00:05
	FedProx	07:13 ± 00:23	07:42 ± 00:07	07:13 ± 00:03	07:52 ± 00:07

6.6 Impact Analysis of Number of Clients

In the previous experiments, five FL clients were used within the framework. This number was considered appropriate for the assessment of a cross-silo scenario with a relatively limited dataset — a partitioned dataset comprising approximately 2200 training samples, which resulted in 440 samples for each of the clients. This section is devoted to analyse the impact of varying number of clients on the framework's performance, while maintaining the framework's design specifications and requirements for a cross-silo scenario and the limitations of the prototype implementation. The results are showed in Table 6.8. The selected range of clients for this impact study comprises three, five, ten and fifteen clients. For completeness, the average execution times for the ten runs of each combination are also displayed in Table 6.9.

Table 6.8: Number of clients impact average accuracy results. Comparison of the impact of varying client numbers across different datasets on BERT Small over BCFL training mode.

Number of Clients	twitter_dep	acl_dep_sad	mixed_depression	dreaddit
3	81.96 ± 0.64	92.83 ± 0.41	88.23 ± 0.28	75.17 ± 0.73
5	81.86 ± 0.68	91.68 ± 0.54	87.92 ± 0.44	74.87 ± 0.81
10	82.22 ± 0.76	89.96 ± 0.42	86.44 ± 0.75	74.80 ± 0.58
15	80.68 ± 0.67	89.27 ± 0.57	84.57 ± 0.59	73.20 ± 0.46

Table 6.9: Number of clients impact average execution times. Comparison of the impact of varying client numbers across different datasets on BERT Small over BCFL training mode. The results show the mean and standard deviation execution time for ten executions with different random seeds. The times correspond to minutes and seconds.

Number of Clients	twitter_dep	acl_dep_sad	mixed_depression	dreaddit
3	$05{:}38 \pm 00{:}06$	$05{:}34 \pm 00{:}14$	$04{:}33 \pm 00{:}02$	$05{:}34 \pm 00{:}06$
5	$06{:}03 \pm 00{:}07$	$05{:}58 \pm 00{:}09$	$05{:}38 \pm 00{:}10$	$06{:}24 \pm 00{:}10$
10	$08{:}34 \pm 00{:}07$	$08{:}46 \pm 00{:}22$	$08{:}15 \pm 00{:}07$	$08{:}41 \pm 00{:}07$
15	$11{:}21 \pm 00{:}07$	$11{:}26 \pm 00{:}09$	$11{:}03 \pm 00{:}15$	$11{:}33 \pm 00{:}11$

The results show that the performance of the framework is consistent across varying client numbers, with similar accuracies achieved when the same data is partitioned into three to fifteen clients. With slightly greater accuracies observed for fewer clients, the framework consistently demonstrated comparable results for larger numbers of clients. With regard to the execution times, we observe that the communication time significantly increases due to the increase in the number of local models — one for each client — that the blockchain network has to transmit among the managing nodes.

6.7 Discussion

The evaluation of our BCFL framework demonstrates equivalent performance to its centralised counterpart. The ablation study conducted shows that it is feasible to design a decentralised FL model using blockchain technology. The implementation can be achieved successfully using state-of-the-art transformer architecture models for NLP tasks. This satisfactorily answers research questions 1, 2 and 3 (Section 1.2). This equivalent performance is primarily achieved through two factors. Firstly, the use of Hyperledger Fabric as a permissioned blockchain, which utilises general-purpose

language, unlike other permissionless blockchains. This allows for the precision of the parameters to be maintained during the aggregation algorithm, thanks to the use of floating-point number operations. Secondly, optimising the fine-tuning process and sending only the final layers of the model to the blockchain makes it feasible to store and aggregate local models on-chain, without the need of any off-chain participant during the process. This incurs an evident communication overhead, but manageable. Moreover, reducing the number of model layers modified during the fine-tuning process does not harm the framework's performance. The convergence plots presented in this chapter show that fine-tuning overly complex models or modifying too many layers leads to generalisation issues due to overfitting to the training data.

The execution times of our models are significantly fast for all four variants of the BERT architecture. This is due to the parallelisation capability of the transformer models, which allows for the training of more complex models with similar times to their lighter versions.

Additionally, our study on the impact of different strength levels of DP applied during the learning process yielded two main takeaways. Firstly, in general, more data is needed to achieve good performance [75], as shown by our results from FL where each client was trained with a partition of the data — roughly 400 samples — and the degradation of performance with DP was much more considerable than that of centralised DL. Secondly, the standard deviation values of the model accuracies showed that even a slight data imbalance had a significant negative impact on the robustness of the model outcomes when DP was applied, even for low amounts of noise. This finding is consistent with previous research on DP [20].

Furthermore, the study on the impact of data distribution on model performance showed that while FedAvg performed well in slightly imbalanced scenarios, FedProx was able to match its performance and improve upon it in more extreme imbalanced data cases. This makes it an interesting option for general cases where there is uncertainty about the data distribution. Moreover, research has shown that in cases where FedProx was unable to outperform FedAvg, other algorithms such as FedYogi [37] and CSFedAvg [90] were able to achieve better performance. These algorithms are worth considering for future research to assess their performance on binary text classification tasks with non-IID data distributions.

Finally, the results of our BCFL framework demonstrated minimal variation in performance when varying the number of clients using the same data in a cross-silo scenario. With slightly greater accuracies observed for fewer clients, the framework consistently demonstrated comparable results for larger numbers of clients.

Chapter 7
Conclusion

Data security and confidentiality are essential to ensure individual's privacy in the age of interconnectivity and big data on the internet. Sectors such as healthcare, telecommunications and finance are particularly vulnerable and must take extra precautions with their data and comply with strict rules imposed by governments such as GDPR or HIPAA. ML is a powerful technology that can provide solutions to many problems thanks to the use of data, but it must be done in a way that protects sensitive information. FL emerges as a solution that enables distributed ML while maintaining data security and privacy. However, it does not offer a fully functional solution for the aforementioned sectors as it depends on a central server to oversee the final development of the ML model. This centralisation of control poses inherent problems and limitations.

BCFL was introduced in 2018 as an improvement to the FL mechanism. It provides a decentralised, immutable, and persistent alternative by using smart contracts on the blockchain. This allows for trustless and decentralised collaboration between different organisations or IoT devices. The literature presents various designs that use different types of blockchain with their respective consensus mechanisms. However, there is a scarcity of frameworks that offer full functionality for training complex DL models. Classification tasks for NLP — our chosen application for this thesis — particularly lacks attention in the literature for decentralised approaches.

To address the research questions posed at the beginning of this thesis, we created a modular and fully functional BCFL framework that allows for a wide range of configurations. Its architecture is based on lightweight versions of the BERT transformer model, designed for NLP tasks. By freezing some of the model layers, we reduce the number of model parameters that needs to be transferred to the blockchain for the on-chain federated aggregation by an approximated factor of 10x. This way, the entire federated aggregation process can be carried out within the space and computational constraints

intrinsic to a decentralised blockchain system.

To satisfactorily answer our research questions, an in-depth ablation study was conducted to evaluate the performance of the implemented framework. This ablation study highlights the decentralised framework's ability to achieve equivalent results to its centralised counterpart. This is made possible by the use of Hyperledger Fabric blockchain, which supports general-purpose language and enables floating-point number arithmetic. The blockchain network prototype introduces a communication overhead of approximately 3x, which is significant but manageable. However, further testing is required to determine the framework's effectiveness in a production blockchain.

With respect to the last research question, we observe in the experiments results that moderate DP values (ϵ = 10) do not negatively affect the performance of the model as much as stricter values (ϵ = 0.5). However, this degradation is highly dependent on the amount of training data, as reported in the literature, and could be mitigated by using larger datasets. Furthermore, the baseline FedAvg aggregation algorithm showed resilience to data imbalances. However, alternative algorithms such as FedProx demonstrate their higher effectiveness in extreme cases of non-IID data distributions.

In conclusion, the work developed in this thesis significantly contributes to the literature by designing and implementing a fully functional BCFL framework with promising results that encourage further research in the field of decentralised distributed ML. In summary, our results show that a decentralised FL framework using blockchain is able to achieve equivalent results to an FL framework, with the added value of the guarantees that decentralisation provides, such as trustless collaboration and resilience against SPOF and availability attacks.

7.1 Future Work

In this section we present potential avenues for future research into the context of BCFL, some of which address the limitations of this work.

One limitation of our thesis is the lack of access to real-world clinical data to fine-tune the DL model in the BCFL framework. Future research could provide interesting insights by applying this framework to clinical interviews or therapy transcriptions, such as the datasets used by Xenozaki et al. [83] and Jan et al. [34]. Additionally, to assess the FL training method within the same distribution and feature space, each client uses a partition of a dataset. To create a more realistic scenario, each FL client could use a different dataset with a prior preprocessing work to ensure that the datasets follow a similar distribution and share the same feature space.

Furthermore, the investigation of the application of FTL to create a global model that learns from datasets with different samples and feature space shows great promise for expanding the application of this framework. The combination of health data from different specialties could facilitate cross-cutting data insights that could help with an

integral diagnosis of diseases in the healthcare sector.

Another limitation of our implementation is the lack of asynchronous support for the federated aggregation. In our framework, the blockchain waits to obtain all local model instances before performing aggregation with the federated algorithm. This requires FL clients to be executed in environments with similar resources and evenly sized datasets. With the implementation of asynchronous flow, FL clients could use datasets of different sizes and training times without negatively affecting the global model development.

In the study of the impact of DP on model performance we see that, although DP is the most widely used mechanism to ensure data confidentiality and prevent inference and information leakage attacks, models trained on small amounts of data experience significant accuracy degradation with the addition of random noise. Future research could explore alternative mechanisms, such as HE [89] — presented in Subsection 2.3.2 of Chapter 2 — or more innovative approaches, such as the hybrid integration of HE and DP. This combination holds great promise, as research suggests that combining these two techniques could leverage the strengths of each, mitigate their respective drawbacks and enhance privacy protection [6].

When studying the impact of data distribution on the performance of the framework, previous research has shown that in some non-IID data scenarios, where FedProx was unable to outperform FedAvg, other algorithms such as FedYogi [37] and CSFedAvg [90] were able to achieve better performance. In the future, one avenue for research could be to evaluate the performance of these federated aggregation algorithms in non-IID data distributions for binary and multi-class text classification.

Lastly, one of the main advantages of our framework is its ability to perform all the operations of the federated operations on-chain, without needing to entrust any critical data or processes to an off-chain entity. However, this approach imposes certain limitations to our framework, such as the need to use permissioned blockchains with flexible block and transaction size or, instead, using a simple DL model. A promising approach is to use Zero-Knowledge Proofs (ZKPs) schemes. These schemes can provide guarantees that the federated aggregation algorithm is correctly executed by an off-chain entity, thanks to the use of cryptographic mechanisms. Although this technology shows great promise, it is still limited in that, due to its reliance on encrypted operations, it does not allow floating-point number arithmetic. Moreover, it is not able to certify that the algorithm has been run on the data provided, so its guarantees are currently insufficient for the task performed by our BCFL framework.

Bibliography

[1] ABADI, MARTIN; CHU, ANDY; GOODFELLOW, IAN; MCMAHAN, H BRENDAN; MIRONOV, ILYA; TALWAR, KUNAL AND ZHANG, LI. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (2016), pp. 308–318.

[2] ANTUNES, RODOLFO STOFFEL; ANDRÉ DA COSTA, CRISTIANO; KÜDERLE, ARNE; YARI, IMRANA ABDULLAHI AND ESKOFIER, BJÖRN. Federated Learning for Healthcare: Systematic Review and Architecture Proposal. *ACM Trans. Intell. Syst. Technol. 13*, 4 (05 2022).

[3] ARASTEH, SOROOSH TAYEBI; LOTFINIA, MAHSHAD; NOLTE, TERESA; SAEHN, MARWIN; ISFORT, PETER; KUHL, CHRISTIANE; NEBELUNG, SVEN; KAISSIS, GEORGIOS AND TRUHN, DANIEL. Preserving privacy in domain transfer of medical AI models comes at no performance costs: The integral role of differential privacy. *arXiv preprint arXiv:2306.06503* (2023).

[4] ARJI, GOLI; ERFANNIA, LEILA; ALIREZAEI, SAMIRA AND HEMMAT, MORTEZA. A systematic literature review and analysis of deep learning algorithms in mental disorders. *Informatics in Medicine Unlocked 40* (2023), p. 101284.

[5] ARYA, JASWANT; KUMAR, ARUN; SINGH, AKHILENDRA PRATAP; MISHRA, TAPAS KUMAR AND CHONG, PETER HJ. Blockchain: Basics, applications, challenges and opportunities.

[6] AZIZ, REZAK; BANERJEE, SOUMYA; BOUZEFRANE, SAMIA AND LE VINH, THINH. Exploring homomorphic encryption and differential privacy techniques towards secure federated learning paradigm. *Future internet 15*, 9 (2023), p. 310.

[7] BASU, PRIYAM; ROY, TIASA SINGHA; NAIDU, RAKSHIT; MUFTUOGLU, ZUMRUT; SINGH, SAHIB AND MIRESHGHALLAH, FATEMEHSADAT. Benchmarking differential privacy and federated learning for bert models. *arXiv preprint arXiv:2106.13973* (2021).

[8] BHAGOJI, ARJUN NITIN; CHAKRABORTY, SUPRIYO; MITTAL, PRATEEK AND CALO, SERAPHIN. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning* (2019), PMLR, pp. 634–643.

[9] BISHOP, CHRISTOPHER. Pattern recognition and machine learning. *Springer google schola 2* (2006), pp. 5–43.

[10] CHHIKARA, PRATEEK; SINGH, PRABHJOT; TEKCHANDANI, RAJKUMAR; KUMAR, NEERAJ AND GUIZANI, MOHSEN. Federated Learning Meets Human Emotions: A Decentralized Framework for Human–Computer Interaction for IoT Applications. *IEEE Internet of Things Journal 8*, 8 (2021), pp. 6949–6962.

[11] DARGAN, SHAVETA; KUMAR, MUNISH; AYYAGARI, MARUTHI ROHIT AND KUMAR, GULSHAN. A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering 27* (2020), pp. 1071–1092.

[12] DE FILIPPI, PRIMAVERA AND HASSAN, SAMER. Blockchain technology as a regulatory technology: From code is law to law is code. *arXiv preprint arXiv:1801.02507* (2018).

[13] DESAI, HARSH BIMAL; OZDAYI, MUSTAFA SAFA AND KANTARCIOGLU, MURAT. Blockfla: Accountable federated learning via hybrid blockchain architecture. In *Proceedings of the eleventh ACM conference on data and application security and privacy* (2021), pp. 101–112.

[14] DEVLIN, JACOB; CHANG, MING-WEI; LEE, KENTON AND TOUTANOVA, KRISTINA. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[15] DRISCOLL, KEVIN; HALL, BRENDAN; SIVENCRONA, HÅKAN AND ZUMSTEG, PHIL. Byzantine fault tolerance, from theory to reality. In *International Conference on Computer Safety, Reliability, and Security* (2003), Springer, pp. 235–248.

[16] DWORK, CYNTHIA. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation* (2008), Springer, pp. 1–19.

[17] EL OUADRHIRI, AHMED AND ABDELHADI, AHMED. Differential privacy for deep and federated learning: A survey. *IEEE access 10* (2022), pp. 22359–22380.

[18] ELSBETH, TURCAN. Dreaddit: A reddit dataset for stress analysis in social media. In *Proceedings of the 10th International Workshop on Health Text Mining and Information Analysis* (2019).

[19] EUROPEAN COMMISSION. Data protection in the EU, Nov 2023.

[20] FARRAND, TOM; MIRESHGHALLAH, FATEMEHSADAT; SINGH, SAHIB AND TRASK, ANDREW. Neither private nor fair: Impact of data imbalance on utility and fairness in differential privacy. In *Proceedings of the 2020 workshop on privacy-preserving machine learning in practice* (2020), pp. 15–19.

[21] GENTRY, CRAIG. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[22] GOODFELLOW, IAN; BENGIO, YOSHUA AND COURVILLE, AARON. *Deep learning*. MIT press, 2016.

[23] HAN, XU; ZHANG, ZHENGYAN; DING, NING; GU, YUXIAN; LIU, XIAO; HUO, YUQI; QIU, JIEZHONG; YAO, YUAN; ZHANG, AO; ZHANG, LIANG ET AL. Pre-trained models: Past, present and future. *AI Open 2* (2021), pp. 225–250.

[24] HENNESSY, JOHN L AND PATTERSON, DAVID A. A new golden age for computer architecture. *Communications of the ACM 62*, 2 (2019), pp. 48–60.

[25] HOCHREITER, SEPP. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6*, 02 (1998), pp. 107–116.

[26] HOU, DONGKUN; ZHANG, JIE; MAN, KA LOK; MA, JIEMING AND PENG, ZITIAN. A systematic literature review of blockchain-based federated learning: Architectures, applications and issues. In *2021 2nd Information communication technologies conference (ICTC)* (2021), IEEE, pp. 302–307.

[27] HUA, YIFAN; MILLER, KEVIN; BERTOZZI, ANDREA L; QIAN, CHEN AND WANG, BAO. Efficient and reliable overlay networks for decentralized federated learning. *SIAM Journal on Applied Mathematics 82*, 4 (2022), pp. 1558–1586.

[28] HUANG, CHAO; HUANG, JIANWEI AND LIU, XIN. Cross-silo federated learning: Challenges and opportunities. *arXiv preprint arXiv:2206.12949* (2022).

[29] HUANG, LI; SHEA, ANDREW L; QIAN, HUINING; MASURKAR, ADITYA; DENG, HAO AND LIU, DIANBO. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of biomedical informatics 99* (2019), p. 103291.

[30] IGLESIAS, MANUEL JOSÉ FERNÁNDEZ. Introduction to Blockchain, Smart Contracts and Decentralized Applications.

[31] ILIAS, LOUKAS; MOUZAKITIS, SPIROS AND ASKOUNIS, DIMITRIS. Calibration of Transformer-Based Models for Identifying Stress and Depression in Social Media. *IEEE Transactions on Computational Social Systems* (2023).

[32] ISSA, WAEL; MOUSTAFA, NOUR; TURNBULL, BENJAMIN; SOHRABI, NASRIN AND TARI, ZAHIR. Blockchain-Based Federated Learning for Securing Internet of Things: A Comprehensive Survey. *ACM Comput. Surv. 55*, 9 (01 2023).

[33] IYORTSUUN, NGUMIMI KAREN; KIM, SOO-HYUNG; JHON, MIN; YANG, HYUNG-JEONG AND PANT, SUDARSHAN. A Review of Machine Learning and Deep Learning Approaches on Mental Health Diagnosis. *Healthcare 11*, 3 (2023).

[34] JANG, JIHEE; YOON, SEOWON; SON, GAEUN; KANG, MINJUNG; CHOEH, JOON YEON AND CHOI, KEE-HONG. Predicting personality and psychological distress using natural language processing: a study protocol. *Frontiers in Psychology 13* (2022), p. 865541.

[35] JEON, BEOMYEOL; FERDOUS, SM; RAHMAN, MUNTASIR RAIHAN AND WALID, ANWAR. Privacy-preserving decentralized aggregation for federated learning. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2021), IEEE, pp. 1–6.

[36] JIN, WEIZHAO; YAO, YUHANG; HAN, SHANSHAN; JOE-WONG, CARLEE; RAVI, SRIVATSAN; AVESTIMEHR, SALMAN AND HE, CHAOYANG. FedML-HE: An Efficient Homomorphic-Encryption-Based Privacy-Preserving Federated Learning System. *arXiv preprint arXiv:2303.10837* (2023).

[37] KATZ, ROY. Performance comparison of different federated learning aggregation algorithms: How does the performance of different federated learning aggregation algorithms compare to each other?

[38] KIM, HYESUNG; PARK, JIHONG; BENNIS, MEHDI AND KIM, SEONG-LYUN. Blockchained on-device federated learning. *IEEE Communications Letters 24*, 6 (2019), pp. 1279–1283.

[39] LECUN, YANN; BENGIO, YOSHUA AND HINTON, GEOFFREY. Deep learning. *nature 521*, 7553 (2015), pp. 436–444.

[40] LEE, JAEJUN; TANG, RAPHAEL AND LIN, JIMMY. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090* (2019).

[41] LI, JUN; SHAO, YUMENG; WEI, KANG; DING, MING; MA, CHUAN; SHI, LONG; HAN, ZHU AND POOR, H VINCENT. Blockchain assisted decentralized federated learning (BLADE-FL): Performance analysis and resource allocation. *IEEE Transactions on Parallel and Distributed Systems 33*, 10 (2021), pp. 2401–2415.

[42] LI, TIAN; SAHU, ANIT KUMAR; TALWALKAR, AMEET AND SMITH, VIRGINIA. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine 37*, 3 (2020), pp. 50–60.

[43] LI, TIAN; SAHU, ANIT KUMAR; ZAHEER, MANZIL; SANJABI, MAZIAR; TALWALKAR, AMEET AND SMITH, VIRGINIA. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems 2* (2020), pp. 429–450.

[44] LI, YUZHENG; CHEN, CHUAN; LIU, NAN; HUANG, HUAWEI; ZHENG, ZIBIN AND YAN, QIANG. A blockchain-based decentralized federated learning framework with committee consensus. *IEEE Network 35*, 1 (2020), pp. 234–241.

[45] LIU, MING; HO, STELLA; WANG, MENGQI; GAO, LONGXIANG; JIN, YUAN AND ZHANG, HE. Federated learning meets natural language processing: A survey. *arXiv preprint arXiv:2107.12603* (2021).

[46] LIU, YUAN; AI, ZHENGPENG; SUN, SHUAI; ZHANG, SHUANGFENG; LIU, ZELEI AND YU, HAN. Fedcoin: A peer-to-peer payment system for federated learning. In *Federated learning: privacy and incentive*. Springer, 2020, pp. 125–138.

[47] LIU, YINGQI; MA, SHIQING; AAFER, YOUSRA; LEE, WEN-CHUAN; ZHAI, JUAN; WANG, WEIHANG AND ZHANG, XIANGYU. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)* (2018), Internet Soc.

[48] LU, YUNLONG; HUANG, XIAOHONG; ZHANG, KE; MAHARJAN, SABITA AND ZHANG, YAN. Communication-efficient federated learning and permissioned blockchain for digital twin edge networks. *IEEE Internet of Things Journal 8*, 4 (2020), pp. 2276–2288.

[49] MA, CHUAN; LI, JUN; SHI, LONG; DING, MING; WANG, TAOTAO; HAN, ZHU AND POOR, H VINCENT. When federated learning meets blockchain: A new distributed learning paradigm. *IEEE Computational Intelligence Magazine 17*, 3 (2022), pp. 26–33.

[50] MADILL, EVAN; NGUYEN, BEN; LEUNG, CARSON K AND ROUHANI, SARA. ScaleSFL: a sharding solution for blockchain-based federated learning. In *Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure* (2022), pp. 95–106.

[51] MARTINEZ, ISMAEL; FRANCIS, SREYA AND HAFID, ABDELHAKIM SENHAJI. Record and reward federated learning contributions with blockchain. In *2019 International conference on cyber-enabled distributed computing and knowledge discovery (CyberC)* (2019), IEEE, pp. 50–57.

[52] MARTÍNEZ-CASTAÑO, RODRIGO; HTAIT, AMAL; AZZOPARDI, LEIF AND MOSHFEGHI, YASHAR. BERT-based transformers for early detection of mental health illnesses. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 12th International Conference of the CLEF Association, CLEF 2021, Virtual Event, September 21–24, 2021, Proceedings 12* (2021), Springer, pp. 189–200.

[53] MCMAHAN, BRENDAN; MOORE, EIDER; RAMAGE, DANIEL; HAMPSON, SETH AND Y ARCAS, BLAISE AGUERA. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (2017), PMLR, pp. 1273–1282.

[54] MOORE, WILNELLYS AND FRYE, SARAH. Review of HIPAA, part 1: history, protected health information, and privacy and security rules. *Journal of nuclear medicine technology 47*, 4 (2019), pp. 269–272.

[55] MYRZASHOVA, RAUSHAN; ALSAMHI, SAEED HAMOOD; SHVETSOV, ALEXEY V; HAWBANI, AMMAR AND WEI, XI. Blockchain meets federated learning in healthcare: A systematic review with challenges and opportunities. *IEEE Internet of Things Journal* (2023).

[56] NAKAMOTO, SATOSHI. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review* (2008).

[57] NGUYEN, DINH C; DING, MING; PHAM, QUOC-VIET; PATHIRANA, PUBUDU N; LE, LONG BAO; SENEVIRATNE, ARUNA; LI, JUN; NIYATO, DUSIT AND POOR, H VINCENT. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet of Things Journal 8*, 16 (2021), pp. 12806–12825.

[58] NGUYEN, DINH C; PHAM, QUOC-VIET; PATHIRANA, PUBUDU N; DING, MING; SENEVIRATNE, ARUNA; LIN, ZIHUAI; DOBRE, OCTAVIA AND HWANG, WON-JOO. Federated learning for smart healthcare: A survey. *ACM Computing Surveys (CSUR) 55*, 3 (2022), pp. 1–37.

[59] OTTER, DANIEL W; MEDINA, JULIAN R AND KALITA, JUGAL K. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems 32*, 2 (2020), pp. 604–624.

[60] PAPADOPOULOS, PAVLOS; ABRAMSON, WILL; HALL, ADAM J; PITROPAKIS, NIKOLAOS AND BUCHANAN, WILLIAM J. Privacy and trust redefined in federated machine learning. *Machine Learning and Knowledge Extraction 3*, 2 (2021), pp. 333–356.

[61] PIRINA, INNA AND ÇÖLTEKIN, ÇAĞRI. Identifying depression on reddit: The effect of training data. In *Proceedings of the 2018 EMNLP workshop SMM4H: the 3rd social media mining for health applications workshop & shared task* (2018), pp. 9–12.

[62] PONGNUMKUL, SUPORN; SIRIPANPORNCHANA, CHAIYAPHUM AND THAJCHAYAPONG, SUTTIPONG. Performance analysis of private blockchain platforms in varying workloads. In *2017 26th international conference on computer communication and networks (ICCCN)* (2017), IEEE, pp. 1–6.

[63] POURKEYVAN, ALIREZA; SAFA, RAMIN AND SOROURKHAH, ALI. Harnessing the Power of Hugging Face Transformers for Predicting Mental Health Disorders in Social Networks. *arXiv preprint arXiv:2306.16891* (2023).

[64] PREUVENEERS, DAVY; RIMMER, VERA; TSINGENOPOULOS, ILIAS; SPOOREN, JAN; JOOSEN, WOUTER AND ILIE-ZUDOR, ELISABETH. Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences 8*, 12 (2018), p. 2663.

[65] RADFORD, ALEC; NARASIMHAN, KARTHIK; SALIMANS, TIM; SUTSKEVER, ILYA ET AL. Improving language understanding by generative pre-training.

[66] RANA, NEETA AND MARWAHA, HITESH. Role of federated learning in healthcare systems: A survey. *Mathematical Foundations of Computing* (2023), pp. 0–0.

[67] SAPUTRA, YURIS MULYA; HOANG, DINH THAI; NGUYEN, DIEP N; DUTKIEWICZ, ERYK; MUECK, MARKUS DOMINIK AND SRIKANTESWARA, SRIKATHYAYANI. Energy demand prediction with federated learning for electric vehicle networks. In *2019 IEEE global communications conference (GLOBECOM)* (2019), IEEE, pp. 1–6.

[68] SCHUMANN, GERRIT; AWICK, JAN-PHILIPP AND GÓMEZ, JORGE MARX. Natural Language Processing using Federated Learning: A Structured Literature Review. In *2023 IEEE International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)* (2023), IEEE, pp. 1–7.

[69] SHUKLA, MANAN AND SENEVIRATNE, OSHANI. MentalHealthAI: Utilizing Personal Health Device Data to Optimize Psychiatry Treatment. *arXiv preprint arXiv:2307.04777* (2023).

[70] SINGHA ROY, TIASA; BASU, PRIYAM; PRIYANSHU, AMAN AND NAIDU, RAKSHIT. Interpretability of Fine-grained Classification of Sadness and Depression, 03 2022.

[71] SU, CHANG; XU, ZHENXING; PATHAK, JYOTISHMAN AND WANG, FEI. Deep learning in mental health outcome research: a scoping review. *Translational Psychiatry 10* (12 2020).

[72] SUN, CHI; QIU, XIPENG; XU, YIGE AND HUANG, XUANJING. How to fine-tune bert for text classification? In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18* (2019), Springer, pp. 194–206.

[73] SURULIRAJ, BANUCHITRA AND ORJI, RITA. Federated Learning Framework for Mobile Sensing Apps in Mental Health. In *2022 IEEE 10th International Conference on Serious Games and Applications for Health(SeGAH)* (2022), pp. 1–7.

[74] THAKKAR, OM; RAMASWAMY, SWAROOP; MATHEWS, RAJIV AND BEAUFAYS, FRANÇOISE. Understanding unintended memorization in federated learning. *arXiv preprint arXiv:2006.07490* (2020).

[75] TRAMER, FLORIAN AND BONEH, DAN. Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660* (2020).

[76] VASWANI, ASHISH; SHAZEER, NOAM; PARMAR, NIKI; USZKOREIT, JAKOB; JONES, LLION; GOMEZ, AIDAN N; KAISER, ŁUKASZ AND POLOSUKHIN, ILLIA. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[77] VUJIČIĆ, DEJAN; JAGODIĆ, DIJANA AND RANĐIĆ, SINIŠA. Blockchain technology, bitcoin, and Ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)* (2018), IEEE, pp. 1–6.

[78] WANG, HONGYI; YUROCHKIN, MIKHAIL; SUN, YUEKAI; PAPAILIOPOULOS, DIMITRIS AND KHAZAENI, YASAMAN. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440* (2020).

[79] WANG, ZHILIN AND HU, QIN. Blockchain-based federated learning: A comprehensive survey. *arXiv preprint arXiv:2110.02182* (2021).

[80] WARNAT-HERRESTHAL, STEFANIE; SCHULTZE, HARTMUT; SHASTRY, KRISHNAPRASAD LINGADAHALLI; MANAMOHAN, SATHYANARAYANAN; MUKHERJEE, SAIKAT; GARG, VISHESH; SARVESWARA, RAVI; HÄNDLER, KRISTIAN; PICKKERS, PETER; AZIZ, N AHMAD ET AL. Swarm learning for decentralized and confidential clinical machine learning. *Nature 594*, 7862 (2021), pp. 265–270.

[81] WENG, JIASI; WENG, JIAN; ZHANG, JILIAN; LI, MING; ZHANG, YUE AND LUO, WEIQI. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing 18*, 5 (2019), pp. 2438–2455.

[82] WILHELMI, FRANCESC; GIUPPONI, LORENZA AND DINI, PAOLO. Blockchain-enabled server-less federated learning. *arXiv preprint arXiv:2112.07938* (2021), pp. 1–14.

[83] XEZONAKI, DANAI; PARASKEVOPOULOS, GEORGIOS; POTAMIANOS, ALEXANDROS AND NARAYANAN, SHRIKANTH. Affective conditioning on hierarchical networks applied to depression detection from transcribed clinical interviews. *arXiv preprint arXiv:2006.08336* (2020).

[84] XU, CHENHAO; GE, JIAQI; LI, YONG; DENG, YAO; GAO, LONGXIANG; ZHANG, MENGSHI; XIANG, YONG AND ZHENG, XI. Scei: A smart-contract driven edge intelligence framework for iot systems. *IEEE Transactions on Mobile Computing* (2023).

[85] XU, JIE AND WANG, FEI. Federated Learning for Healthcare Informatics. *CoRR abs/1911.06270* (2019).

[86] YANG, QIANG; LIU, YANG; CHEN, TIANJIAN AND TONG, YONGXIN. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol. 10*, 2 (01 2019).

[87] YANG, TIMOTHY; ANDREW, GALEN; EICHNER, HUBERT; SUN, HAICHENG; LI, WEI; KONG, NICHOLAS; RAMAGE, DANIEL AND BEAUFAYS, FRANÇOISE. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903* (2018).

[88] ZANWAR, SOURABH; WIECHMANN, DANIEL; QIAO, YU AND KERZ, ELMA. The Best of Both Worlds: Combining Engineered Features with Transformers for Improved Mental Health Prediction from Reddit Posts. In *Proceedings of The Seventh Workshop on Social Media Mining for Health Applications, Workshop & Shared Task* (2022), pp. 197–202.

[89] ZHANG, LI; XU, JIANBO; VIJAYAKUMAR, PANDI; SHARMA, PRADIP KUMAR AND GHOSH, UTTAM. Homomorphic encryption-based privacy-preserving federated learning in iot-enabled healthcare system. *IEEE Transactions on Network Science and Engineering* (2022).

[90] ZHANG, WENYU; WANG, XIUMIN; ZHOU, PAN; WU, WEIWEI AND ZHANG, XINGLIN. Client selection for federated learning with non-iid data in mobile edge computing. *IEEE Access 9* (2021), pp. 24462–24474.

[91] ZHOU, JIEHAN; ZHANG, SHOUHUA; LU, QINGHUA; DAI, WENBIN; CHEN, MIN; LIU, XIN; PIRTTIKANGAS, SUSANNA; SHI, YANG; ZHANG, WEISHAN AND HERRERA-VIEDMA, ENRIQUE. A Survey on Federated Learning and its Applications for Accelerating Industrial Internet of Things. *CoRR abs/2104.10501* (2021).

[92] ZHU, JUNCEN; CAO, JIANNONG; SAXENA, DIVYA; JIANG, SHAN AND FERRADI, HOUDA. Blockchain-empowered federated learning: Challenges, solutions, and future directions. *ACM Computing Surveys 55*, 11 (2023), pp. 1–31.

[93] ZHU, LIGENG; LIU, ZHIJIAN AND HAN, SONG. Deep leakage from gradients. *Advances in neural information processing systems 32* (2019).