الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة قاصدي مرباح ورقلة
كلية التكنولوجيات الحديثة للمعلومات والاتصال
قسم الإعلام الآلي وتكنولوجيا المعلومات

Republic of Algeria Democratic and People's
Ministry of Higher Education and Scientific Research
University of KASDI Merbah – Ouargla
Faculty of New Technologies of Information and Communication
Department of Computer Science and Information Technologies

Professional Master Thesis

Domain: Mathematics and Computer Science
Sector: Computer Science
Specialty: Network Administration and Security

Theme

A Privacy-Preserving Federated Learning Framework Based on Homomorphic Encryption

Presented by:
Belmesmar Abdelalim and Korichi Mohammed Moussa

Publicly discussed: 11/06/2025

Jury Members:

Dr. Boukhamla Akram	President
Dr. BENKADDOUR Mohammed Kamel	Supervisor
Dr. Kahlessenane Fares	Examiner

Academic year: 2024/2025

Acknowledgements

First and foremost, we would like to thank Allah Almighty for giving us the strength, patience, and health to complete this thesis.

We express our deepest gratitude to our supervisor ***Dr. BENKADDOUR Mohammed Kamel*** *for his continuous guidance, valuable advice, and encouragement throughout this work.*

We would also like to thank the members of the jury for accepting to examine and evaluate our work.

Dedication

I dedicate this work to the one who has always been my guiding light.
To **my mother**, whose endless love, quiet strength,
and unwavering belief in me carried me through every challenge.
To **my father**, whose steady presence and lifelong sacrifices
gave me the foundation I needed to grow and succeed.
To **my family**, whose support, warmth, and encouragement
have been a constant source of strength.
And to **everyone** who offered me kindness,
lifted me up, or believed in me when I needed it most.
This achievement belongs to all of you.

- Abdelalim Belmesmar-

Abstract

The development of artificial intelligence has been significantly influenced by the growing availability of large and diverse datasets, which has enabled the rapid evolution of machine learning and deep learning techniques. Initially, centralized learning was adopted as the dominant approach, requiring data to be collected and stored in a single location for model training. While effective in performance, this method raises serious concerns regarding data privacy and security, particularly in domains such as healthcare. To overcome these limitations, federated learning has emerged as a distributed alternative that allows collaborative model training while keeping data localized. In this setting, only model updates are exchanged between clients and the central server. However, despite not sharing raw data, these updates can still leak sensitive information through heuristic attacks, such as gradient inversion techniques that reconstruct original inputs. In response to this threat, this thesis presents a federated learning framework enhanced with homomorphic encryption to secure the exchange of model updates. The proposed approach combines structural and sensitivity-based strategies to apply encryption where it is most needed, aiming to achieve a balance between data protection and learning efficiency. The framework was evaluated through a series of controlled experiments designed to assess its effectiveness in preserving privacy while maintaining model accuracy. The results demonstrate the practical potential of the framework as a secure solution for privacy-aware collaborative learning.

Keywords: Federated Learning, Privacy Preservation, Inference Attacks, Homomorphic Encryption, Selective Encryption

ملخص

شهد الذكاء الاصطناعي تطوراً كبيراً بفضل التوفر المتزايد للبيانات الضخمة والمتنوعة، مما أدى إلى تقدم ملحوظ في تقنيات التعلم الآلي والتعلم العميق. وقد اعتمدت الأساليب التقليدية على التعلم المركزي، الذي يتطلب تجميع البيانات في موقع واحد لتدريب النماذج. وعلى الرغم من فعاليته من حيث الأداء، إلا أن هذا النهج يثير مخاوف كبيرة تتعلق بخصوصية البيانات، خصوصاً في المجالات الحساسة مثل الرعاية الصحية. لمواجهة هذه التحديات، برز التعلم الفيدرالي كحل بديل يسمح بتدريب النماذج بطريقة تعاونية دون مشاركة البيانات الخام، حيث يقتصر تبادل المعلومات على تحديثات النموذج فقط بين العملاء والخادم المركزي. وعلى الرغم من ذلك، فإن هذه التحديثات قد تكشف عن معلومات خاصة من خلال هجمات استنتاجية تعتمد على تحليل التدرجات، مثل هجمات استعادة البيانات. لمعالجة هذه المشكلة، يقترح هذا البحث إطار عمل للتعلم الفيدرالي مدعّماً بتقنية التشفير المتماثل الكامل، بهدف تأمين تبادل تحديثات النماذج بين الأطراف. وقد تم اعتماد استراتيجية تشفير انتقائي، تعتمد على بنية النموذج وحساسية طبقاته، لتحقيق توازن بين حماية الخصوصية وكفاءة الأداء. وتم تقييم الإطار المقترح من خلال مجموعة من التجارب المخبرية، لقياس فعاليته في تعزيز الخصوصية مع الحفاظ على دقة النموذج. وقد أظهرت النتائج أن الإطار يوفر حلاً عملياً وآمناً للتعلم التعاوني في بيئات تتطلب حماية صارمة للبيانات.

الكلمات المفتاحية: التعلم الفيدرالي، حماية الخصوصية، هجمات الاستدلال، التشفير المتماثل، التشفير الانتقائي

Résumé

Le développement de l'intelligence artificielle a été fortement stimulé par la disponibilité croissante de données volumineuses et diversifiées, ce qui a permis l'évolution rapide des techniques d'apprentissage automatique et d'apprentissage profond. L'approche d'apprentissage centralisé, initialement adoptée, exige la collecte des données dans un seul emplacement pour l'entraînement des modèles, soulevant ainsi des préoccupations importantes en matière de confidentialité, notamment dans des domaines sensibles comme la santé. Pour répondre à ces limitations, l'apprentissage fédéré est apparu comme une alternative distribuée, permettant l'entraînement collaboratif des modèles tout en maintenant les données localement. Dans ce contexte, seuls les paramètres mis à jour sont partagés entre les clients et le serveur central. Toutefois, ces mises à jour peuvent révéler des informations sensibles à travers des attaques heuristiques, telles que l'inversion de gradients. Pour atténuer ce risque, ce mémoire propose un cadre d'apprentissage fédéré renforcé par le chiffrement homomorphe afin de sécuriser les échanges entre les clients et le serveur. L'approche adoptée repose sur une stratégie de chiffrement sélectif guidée par la structure du modèle et la sensibilité des gradients, permettant de concilier protection des données et efficacité de l'apprentissage. Le cadre proposé a été évalué à travers une série d'expériences contrôlées afin d'analyser sa capacité à préserver la confidentialité tout en maintenant une performance élevée. Les résultats obtenus démontrent le potentiel pratique de ce cadre comme solution sécurisée pour l'apprentissage collaboratif respectueux de la vie privée.

Mots-clés : Apprentissage fédéré, Préservation de la vie privée, Attaques par inférence, Chiffrement homomorphe, Chiffrement sélectif

Contents

List of Figures

List of Tables

List of Algorithms

List of abbreviation

AI	Artificial Intelligence
FL	*Federated Learning*
ML	*Machine Learning*
DL	*Deep Learning*
ANN	*Artificial Neural Network*
MLP	*Multilayer Perceptron*
RNN	*Recurrent Neural Network*
FedAvg	*Federated averaging*
VFL	*Vertical Federated Learning*
FTL	*Federated Transfer Learning*
DFL	*Decentralized Federated Learning*
IID	*Independent and Identically Distributed*
Non-IID	*Non-Independent and Identically Distributed*
GDPR	*General Data Protection Regulation*
IoT	*Internet of Things*
IoV	*Internet of Vehicles*
EHRs	*Electronic Health Records*
HIPAA	*Health Insurance Portability and Accountability Act*
HFL	*Horizontal Federated Learning*
CPS	*Cyber-Physical Systems*
NLP	*Natural Language Processing*
SSL	*Secure Sockets Layer*
MIAs	*Membership Inference Attacks*
GAN	*Generative Adversarial Network*
DLG	*Deep Leakage from Gradients*
iDLG	*Improved Deep Leakage from Gradients*
DLM	*Deep Leakage from Model*
LLL	*Linear Layer Leakage*
DP	*Differential Privacy*
CDPFL	*Central Differential Privacy Federated Learning*
LDPFL	*Local Differential Privacy Federated Learning*
SMPC	*Secure Multi-party Computation*
HE	*Homomorphic Encryption*
PHE	*Partially Homomorphic Encryption*
SHE	*Somewhat Homomorphic Encryption*
FHE	*Fully Homomorphic Encryption*
CKKS	*Cheon-Kim-Kim-Song*
MKHE	*Multi-Key Homomorphic Encryption*
CNN	*Convolutional Neural Network*

SGD	*Stochastic Gradient Descent*
RGB	*Red Green Blue*
PSNR	*Peak Signal-to-Noise Ratio*
SSIM	*Structural Similarity Index*

General Introduction

Context and Motivation

Artificial intelligence, particularly Machine Learning and Deep Learning, has proven itself in analyzing large volumes of data in today's technological environment. Traditionally, this sophisticated model-building technique relied on centralized learning, where data from different sources was compiled into one single central repository. This centralized approach to model training has yielded landmark performance results in many applications. However, such an aggregated nature of data makes risk possible and hence enormous challenges regarding data security, governance, and most importantly personal privacy. In domains handling sensitive information, such as healthcare, finance, and personal communications, the prerequisite of centralizing raw data introduces profound vulnerabilities and often conflicts with stringent regulatory frameworks (e.g., General Data Protection Regulation (GDPR) , Health Insurance Portability and Accountability Act (HIPAA)), thereby creating significant impediments to data utilization and collaborative innovation.

The tension between the usefulness of large-scale data analysis and the fundamental right to privacy has called for a shift in paradigm towards collaborative learning methodologies where data confidentiality is maintained. The aim is to leverage collective intelligence spread across many data silos as not to have perilous data aggregation. This necessity has led to growing interest in distributed and decentralized computing architectures wherein the computation happens near the data sources. General distributed systems deal with parallelism of computation or management of resources that are dispersed over geography, whereas a specific subset deals with the issue of collaborative model training under privacy constraints.

Federated Learning (FL) [1] has materialized as a prominent and specialized decentralized machine learning paradigm. FL provides a structured framework that enables multiple entities (clients) to collaboratively train a shared global model, coordinated by a central server. Crucially, this process avoids exchanging local raw datasets. The participants train the model on their local partitions of data and send only the updated models to the server for aggregation. This keeps data in its place, complies with the principle of data minimization, and reduces privacy risk as well as communication cost compared to centralized approach. Since FL separates model training from direct access to data, it can be regarded as an alternative means for exploiting sensitive distributed datasets and promoting cooperative research in AI developments where privacy is predominantly concerned.

Problem and Research Challenges

Despite the architectural advantages of federated learning in preserving data locality, the protocol itself is not inherently immune to privacy infringements. Although the exchange of parameters between models may appear abstract in theory, it exposes a potential attack surface in practice. A large body of work has shown that the updates shared by clients (e.g., gradients or weights) leak sensitive information about the clients' private training data and that such information can even be inferred or reconstructed. Potential adversaries, including a compromised central server or malicious participants, can perform powerful inference attacks that can compromise individual dataset-level privacy.

Objective

The main goal of this study is to explore strong ways for improving the privacy guarantees of Federated Learning (FL) systems. Taking into account the known weaknesses linked to the sharing of model updates and their likelihood to allow sensitive data inference, this thesis aims to identify and evaluate effective methods for mitigating these privacy risks, thereby enabling secure and trustworthy collaborative machine learning, especially within domains demanding stringent confidentiality, without fundamentally compromising the collaborative nature of FL

Research Contributions

In line with the objective, this study finds Homomorphic Encryption (HE) to have cryptographic properties ideally suited for the requirements of secure Federated Learning (FL). The basic ability of HE to allow certain mathematical operations on ciphertexts without necessitating decryption is directly relevant to the FL paradigm since it enables key collaborative operations especially aggregation of model parameter updates from server sides to be performed while keeping the underlying plaintext values secret.

The main proposition of this thesis involves the systematic integration of HE within the FL architecture to establish robust end-to-end privacy for sensitive model information. While the theoretical benefits of HE are compelling, its real-world application introduces significant computational and communication overhead, which can slow down wide adoption. So, this study mainly tackles the issue of balancing enhanced privacy guarantees with the constraints of operational efficiency

To address this balance, this research evaluates two distinct HE application strategies:

Full Encryption: Using HE fully on all shared model parameters to give robust cryptographic security.

Selective Encryption: A targeted strategy in which HE is applied only to the most sensitive subsets of model layers , aiming to reach a practical balance between the level of security and the use of resources.

Roadmap of the Thesis

- **Chapter 1** introduces the fundamental concepts, from general artificial intelligence to Machine Learning then deep learning. It further explores the move towards decentralized and collaborative learning paradigms, leading to explanation of Federated Learning, its core architecture, functionalities, different Types, and its advantages especially in respect to privacy-critical applications

- **Chapter 2** examines privacy risks in standard federated learning frameworks, discusses commonly used attack vectors such as inference attacks, and explores major defense strategies. It also reviews relevant research combining FL with privacy-enhancing technologies.
- **Chapter 3** This chapter details the architectural design of the privacy-preserving federated learning system proposed in this research. It focuses on the integration strategy for homomorphic encryption within the FL workflow and also covers both the full and selective encryption approaches, including the methodology employed for identifying sensitive model components in the selective strategy.
- **Chapter 4** This chapter describes the practical implementation of the designed system, outlining the tools, environment, dataset utilized, and the specific experimental setup. It presents the empirical results obtained from evaluating the different system configurations (federated learning basline, federated learning with full encryption and also with selective encryption). Finally, it analyzes these results, focusing on model performance metrics, computational overhead, and the achieved level of privacy protection.

Chapter 1

Overview of Federated Learning

1.1 Introduction

The modern digital age is significantly characterized by the ascendancy of artificial intelligence (AI), driven by rapid growth in data generation and computational power. AI-driven methodologies, especially machine learning (ML) and its advanced subfield as deep learning,have been instrumental in identifying complex patterns and uncovering actionable insights. The performance and generalization of such systems are usually dependent on high-quality, large-scale, and diverse datasets, which fuel the learning process.

Traditionally, the predominant training paradigm involved centralizing such data, aggregating information from diverse origins into a singular central repository. While conducive to achieving high model performance, this centralized architecture inherently introduces substantial impediments concerning data privacy, security vulnerabilities, and adherence to increasingly stringent regulatory mandates, particularly when sensitive information is involved.

Research into alternative learning paradigms has been motivated by the basic conflict between the analytic possibilities attained by means of broad data access and the needs of privacy and security. Driven by the need to leverage collective intelligence resident in scattered data sources without requiring centralized data pooling, there is clearly a visible shift towards distributed and cooperative methods. This change meets the demand for the kind of structure that lets models grow together while maintaining data confidentiality and locality, preserving their development. Knowing this shift from centralized processing towards distributed, privacy-conscious cooperation helps one to appreciate the evolution of specialized techniques; Federated Learning (FL) is a major paradigm meant especially to enable such secure, cooperative model training across distributed datasets.

1.2 Artificial Intelligence: An Overview

Artificial intelligence (AI) embodies the creation of systems that execute tasks normally requiring human intelligence, such as reasoning, learning, decision-making, and perception [2] . Although the concept of AI has existed since the 1950s, its development has rapidly become practical in the past few years due to the growth in data, accessible powerful computing infrastructure, and advancements in learning algorithms.

The original conceptualization of AI systems relied on rule-based logic and symbolic reasoning, where the intelligence was manually encoded by experts [2]. Although these systems performed well in certain bounded tasks, they were brittle in their generalizability and struggled with uncertainty. The field evolved to embrace data-driven methods of intelligence ; machines did not have to rely on experts to construct an intelligent system based on predefined rules. This marked an inflection point in the movement from expert-defined intelligence to intelligence defined by the ability to learn from experience.

As a result, two major subfields emerged under AI: Machine Learning (ML) and Deep Learning (DL). These learning-based methods have become essential components of modern AI systems due to their ability to process large-scale datasets, extract patterns, and make accurate predictions [3].These learning-based techniques have enabled AI systems to power real-world applications such as image recognition, speech assistants, autonomous vehicles, and diagnostic tools. As challenges grow in scale and complexity, ML and DL continue to play a central role in extending the reach and capability of intelligent systems.

Figure 1.1 illustrates that artificial intelligence is a broad discipline encompassing both machine learning and deep learning.

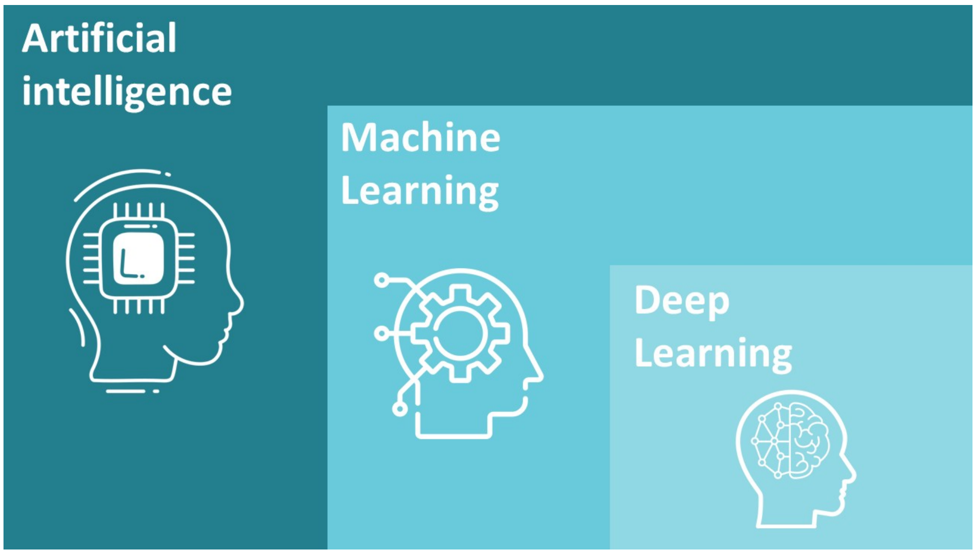


Figure 1.1: Relationship between Artificial Intelligence, Machine Learning and Deep Learning

1.2.1 Machine learning

machine learning (ML) is a scientific field dedicated to the creation of algorithms and statistical models that allow computer systems to perform particular tasks without explicit programming. It uses learning algorithms to examine and interpret difficult datasets, enabling systems to gradually increase their performance by spotting trends and guiding decisions with the least human involvement. Machine learning's main goal is to enable the meaningful information extraction from data, facilitating automated decision-making and predictive analytics in many fields including data mining, image processing, and predictive modeling. [4]

1.2.1.1 The Methods of Machine Learning

Machine learning algorithms are generally classified into the following types based on the nature of the data and the learning objective:

- **Supervised Learning:** This is an approach in which the algorithm is trained on a labeled dataset, where a known output is associated to each input The model learns to map inputs to outputs and is used for tasks such as classification and regression [5].
- **Unsupervised Learning:** This method involves data without labeled outputs. The goal is to discover hidden patterns or groupings within the data. Common techniques include clustering and dimensionality reduction [5].
- **Semi-supervised learning** is a machine learning approach that merges supervised learning (using labeled data) and unsupervised learning (using unlabeled data). It leverages abundant unlabeled data alongside limited labeled data to improve prediction accuracy and uncover patterns in the entire dataset, particularly when labeling data is resource-intensive or time-consuming [5].

- **Reinforcement Learning**This paradigm emphasizes the decision-making process to be utilized in interactive environments, where an agent makes decisions, takes actions, and receives a feedback signal in the form of reward or penalty. Reinforcement learning is commonly used in robotics, gaming, and autonomous systems [5].

1.2.2 Deep learning

Deep learning is a advanced branch of machine learning that utilizes artificial neural networks (ANN) with multiple layers, referred to as deep neural networks, which are Inspired by how the human brain processes information. These networks process data through an architecture composed of an input layer, where raw data is fed into the system; multiple hidden layers, which hierarchically extract and transform increasingly complex features by simulating the brain's ability to interpret patterns; and an output layer, which produces the final prediction or decision. This layered structure enables the automatic learning of data representations, eliminating the need for manual feature engineering and allowing the system to tackle very complex problems effectively. [6]

1.2.3 Deep learning approaches

1.2.3.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) functions as a core neural network topology which uses an input layer combined with multiple hidden fully connected layers and a final output layer as shown in (Figure 1.2). The learning capacity of the model depends on two structural components: hidden layer width indicating neuron number and depth which establishes layer count because they enable increased feature capture and hierarchical representation learning respectively. Theoretical research shows a single hidden layer MLP has the ability to approximate all continuous functions. The standard MLP functions well for healthcare and finance yet its main challenge arises from needing one-dimensional structured input data which restricts its direct application to unstructured patterns such as images text and speech before extraction or transformation occurs.[6]

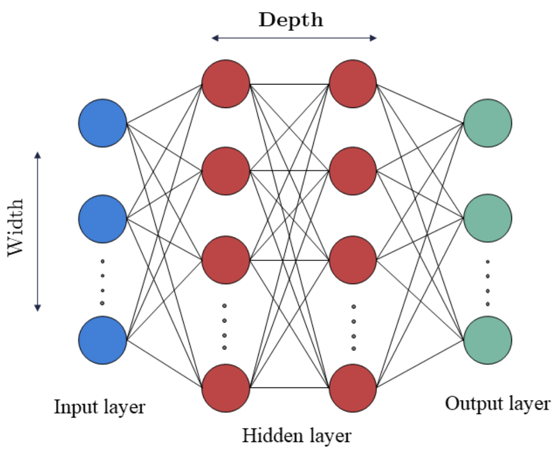


Figure 1.2: A fully connected neural network

1.2.3.2 Recurrent neural network

A recurrent neural network (RNN) is a type of neural network architecture intended to process sequential data by means of recurrent connections spanning time steps. By means of shared weights, it preserves an internal state (memory) parameterized and models temporal dependencies in data including time series, text, or voice. RNNs can detect patterns depending on past context in the input sequence by means of this sequential processing. [6]

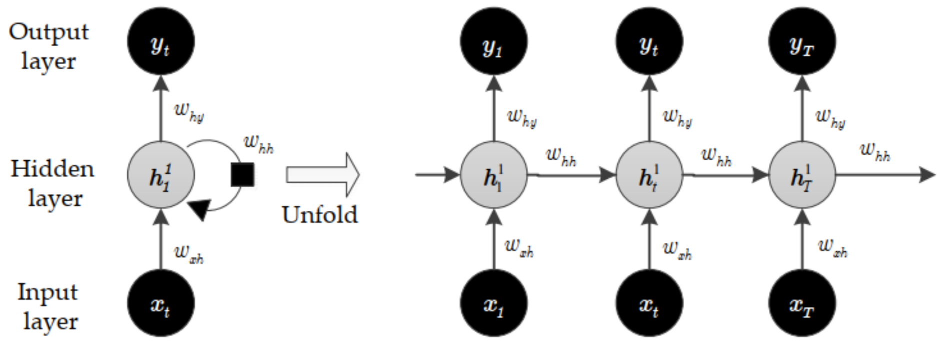


Figure 1.3: Recurrent neural network (RNN).

1.3 Centralized learning

1.3.1 Definition

Centralized learning is a machine learning paradigm in which data collected from various sources is streamed to a central location, typically a cloud-based infrastructure (Figure 1.4), for processing, analysis, and model training . In this approach, high-performance servers housed in centralized data centers perform the computational tasks required to extract features, train models, and generate predictions or decisions.[7]

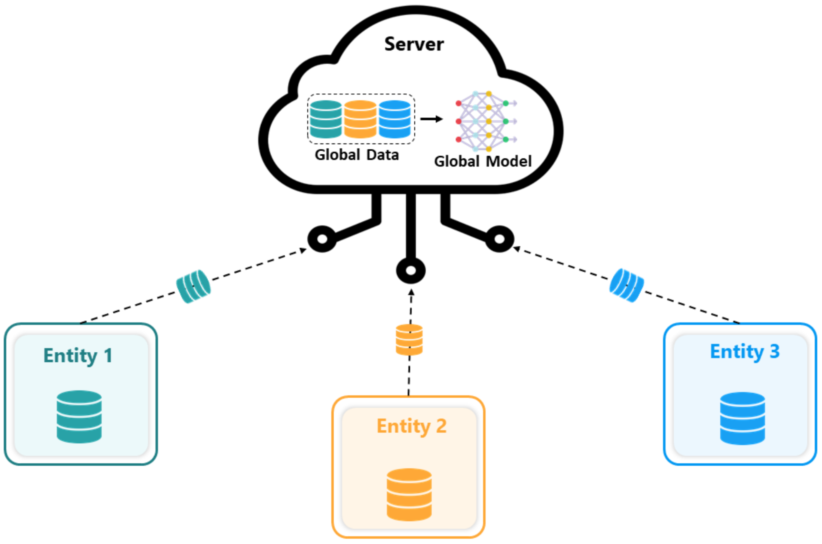


Figure 1.4: Centralized learning architecture

1.3.2 Centralized learning limitation

The paradigm of centralized machine learning has undeniably demonstrated its utility, enabling significant advancements by harnessing large, consolidated datasets for model training, thereby yielding robust predictive capabilities instrumental in numerous applications.
Despite these strengths and its widespread adoption, the centralized approach is confronted with several inherent and increasingly critical limitations that can impede its applicability, security, and long-term viability, particularly as data scales and privacy concerns intensify [8]. These limitations include:

- **Amplified Risk Exposure:** The practice of aggregating extensive datasets, particularly those containing sensitive information, into a single repository inherently creates a focal point of vulnerability. Such concentration significantly increases the susceptibility to data breaches, unauthorized access, and the potential for malicious actors to compromise the integrity of the model or its underlying data.
- **Single-Point Dependency:** The operational continuity of the entire machine learning system becomes critically reliant on the stability and security of this central infrastructure. Any disruption, failure, or compromise at this single junction can cascade, potentially leading to a complete cessation of operations and jeopardizing data accessibility.
- **Obstacles to Data Sharing and Collaborative Endeavors:** The necessity for data to be transferred to and housed within an external central system can present considerable challenges. Organizations frequently exhibit caution regarding the surrender of direct control over proprietary or sensitive information, a stance that can consequently limit the scope for collaborative learning initiatives and the richness of data available for developing more comprehensive models.
- **Potential for Scalability and Performance Impasses:** With the escalating volume and complexity of data, a singular, centralized architecture may eventually encounter practical limitations in its capacity for efficient processing. This can result in diminished performance or necessitate significant, often resource-intensive, enhancements to the central processing capabilities.

1.4 Decentralized learning

Decentralized machine learning emerges as an alternative approach that goes beyond the necessity of a single unified point of data aggregation and computation. This model entails the distribution of both data resources and computational workloads across a network composed of multiple, autonomous devices, or nodes (Figure 1.5). This framework facilitates collaborative model training, frequently allowing raw data to remain within its original environment, thereby intrinsically aiming to improve data privacy, enhance system security, and improve overall operational robustness.[8]

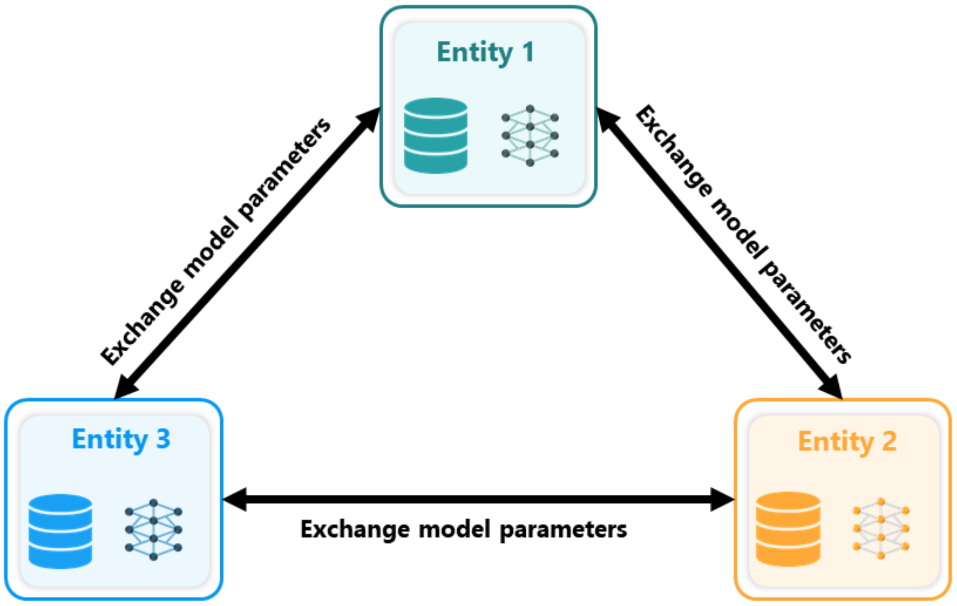

Entity 1
Entity 2
Entity 3
Exchange model parameters
Exchange model parameters
Exchange model parameters

Figure 1.5: Decentralized learning architecture

1.5 Federated Learning

1.5.1 What is Federated Learning?

Federated learning is a decentralized machine learning approach that enables multiple clients such as devices or institutions, to collaboratively develop a common model without sharing their local datasets. Each client retains its own dataset and performs training locally, while a central server coordinates the learning process by aggregating model updates contributed by participating clients [9]. This approach allows for collective intelligence to emerge from distributed data sources without compromising the privacy or ownership of that data .

By keeping data on-device or on-premises and exchanging only model-related information, federated learning offers a practical solution for scenarios where data sharing is restricted by privacy concerns, legal regulations, or infrastructure limitations.

Initially introduced by Google researchers, FL has since become a leading framework for the development of collaborative models that are sensitive to privacy in sensitive sectors such as healthcare, finance, and mobile applications [1].

1.5.2 Federated Learning Process

Ttypical process of federated learning can be summarized in the following steps :

- **Step 1: Client Selection and Initialization of the Global Model**
  The central server initiates the process by selecting a subset of clients based on specific criteria, such as availability, computational resources, and network conditions. These criteria ensure that the selected clients are capable of participating effectively in the training process. Once the clients are chosen, a global model is initialized on the central server. This global model serves as the starting point for the federated

training process, providing a common baseline for all clients to begin their local training.[10]

- **Step 2: Local Training**
  After each client receives the global model, it begins training it on its local dataset using optimization algorithms such as stochastic gradient descent (SGD) (Algorithm 1). This local training allows the model to learn from the unique characteristics of the data available at each client. By keeping the data on the client's device, this step ensures enhanced privacy and security.[10]

Algorithm 1 Federated Learning (FedAvg) Client-Side Training at Federated Round T

```
1: function ClientUpdate($w$)
2:     Input: Local dataset $\mathcal{D}_k$, global model $w$, local epochs $E$, batch size $B$, learning rate $\eta$
3:     Split $\mathcal{D}_k$ into mini-batches $\{b_1, b_2, \dots\}$
4:     for each local epoch $e = 1$ to $E$ do
5:         for each batch $b$ in $\mathcal{D}_k$ do
6:             $w \leftarrow w - \eta \nabla \ell(w; b)$        ▷ Local update (With SGD optimizer)
7:         end for
8:     end for
9:     return updated model $w$ and number of samples $|\mathcal{D}_k|$
10: end function
```

- **Step3 : Model Upload**
  After completing local training, each client computes the updates to the model, such as gradients or parameter changes, and uploads these updates back to the central server. No raw data is shared during this exchange.[10]

- **Step4: Aggregation and Broadcast**
  The server aggregates the updates received from the clients. This can be done using simple weighted averaging (e.g., FedAvg proposed by McMahan et al.[1]), or more sophisticated strategies that adaptively weight contributions based on factors like training efficiency or anomaly scores (e.g., FedProx [11], FedAdam [12], and Krum [13]). The aggregated updates result in an updated global model. The server then broadcasts the updated global model parameters or gradients back to the clients, ensuring that all clients have the latest version of the model and can continue to contribute to the federated training process. [10]

Algorithm 2 Federated Learning (FedAvg) - server-side aggregation

```
1: **Input:** Number of rounds $T$, number of clients $K$, initial model $w_0$
2: **for** each round $t = 1, 2, \ldots, T$ **do**
3:     Select subset of clients $\mathcal{C}_t \subseteq \{1, \ldots, K\}$
4:     **for** each client $k \in \mathcal{C}_t$ **in parallel do**
5:         Send global model $w_{t-1}$ to client $k$
6:     **end for**
7:     **for** each client $k \in \mathcal{C}_t$ **in parallel do**
8:         Receive updated model $w_k^t$ and sample size $n_k$ from client $k$
9:     **end for**
10:    $w_t \leftarrow \sum_{k \in \mathcal{C}_t} \frac{n_k}{\sum_{j \in \mathcal{C}_t} n_j} w_k^t$        ▷ Weighted aggregation
11: **end for**
12: **return** final global model $w_T$
```

- **Step 5: Iteration and Convergence**
  Steps 1 to 4 are repeated across multiple communication rounds. With each iteration, the global model improves, becoming more accurate and robust as it learns from a diverse set of data sources. The training process continues until the model reaches a satisfactory level of performance or a predefined stopping criterion is met. Recent advancements aim to enhance training efficiency and accelerate convergence through various optimization techniques, further improving the effectiveness of federated learning.[10]

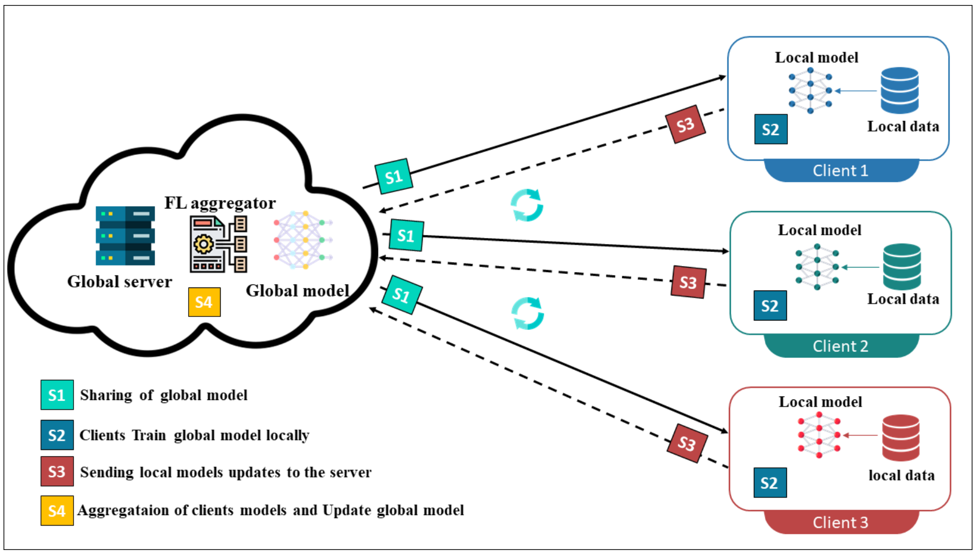

FL aggregator
Global server
Global model
S4
S1
S3
Local model
Local data
S2
Client 1
Client 2
local data
Client 3
S1 Sharing of global model
S2 Clients Train global model locally
S3 Sending local models updates to the server
S4 Aggregataion of clients models and Update global model

Figure 1.6: Federated learning workflow

1.5.3 Types of federated learning

Various types of FL exist, each suited for different applications and scenarios. This section outlines the primary types of FL, categorized by Data Partitioning, system architecture, and Federation Scope.

1.5.3.1 Categories Based on Data Partitioning

As introduced by Yang et al. [14], Federated Learning can be classified based on the way data is partitioned across clients. Specifically, the categorization depends on the relationship between the feature space *(X)*, label space *(Y)*, and sample ID space *(I)* of the local datasets

$$D_i = (X_i, Y_i, I_i) \tag{1.1}$$

Depending on whether clients share the same features, labels, or sample identities, FL can be divided into three main categories:

- **Horizontal federated learning**

  or Sample-based Federated Learning applies when datasets across different clients share the same feature space *(X)*, but are different in the sample ID space *(I)* .[14]. We can represent horizontal federated learning by Equation 1.2:

  $$X_i = X_j, \quad Y_i = Y_j, \quad I_i \neq I_j, \quad \forall D_i, D_j,\ i \neq j \tag{1.2}$$

  This scenario is common in applications where multiple entities collect similar types of data about different individuals. For example, several hospitals use the same patient records format but treat different patients.[15]

- **Vertical Federated Learning (VFL)**

  Vertical federated learning, also known as feature-based FL, is used when datasets at different clients share the same sample ID (I) but contain different types of features (X).[14]. a relationship is depicted in Equation 1.3:

  $$X_i \neq X_j, \quad Y_i \neq Y_j, \quad I_i = I_j, \quad \forall D_i, D_j,\ i \neq j \tag{1.3}$$

  This setting arises in cases where organizations hold complementary information about the same group of individuals. For example, one organization may have demographic information about individuals, while another has purchasing behavior data [16].

- **Federated Transfer Learning (FTL)**

  Federated transfer learning is suitable when clients have both different feature spaces *(X)* and different sample spaces *(I)* (see Equation 1.4).[14]. It combines federated learning with transfer learning techniques [17] to enable knowledge sharing across heterogeneous datasets. This type is particularly useful in scenarios with limited data overlap, such as international collaboration between institutions in different domains.

  $$X_i \neq X_j, \quad Y_i \neq Y_j, \quad I_i \neq I_j, \quad \forall D_i, D_j,\ i \neq j \tag{1.4}$$

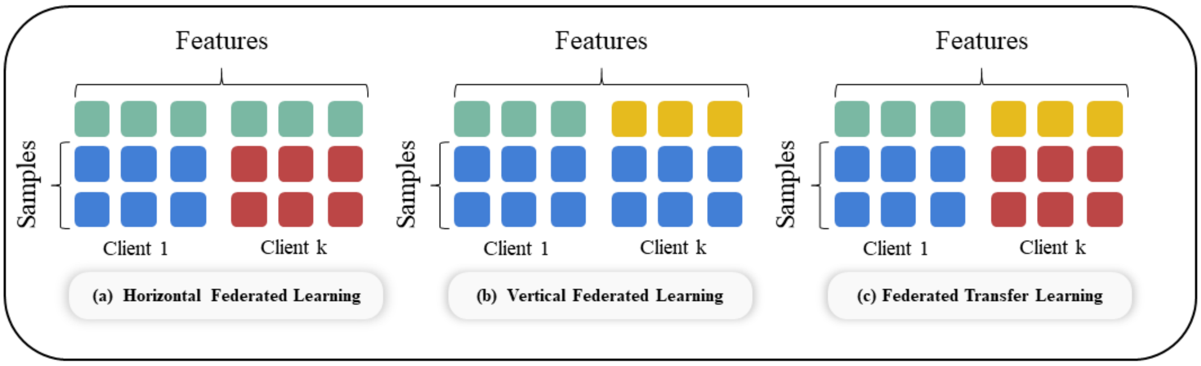

Features
Samples
Client 1
Client k
(a) Horizontal Federated Learning
(b) Vertical Federated Learning
(c) Federated Transfer Learning

Figure 1.7: Types of Federated learning based on data Partitioning

1.5.3.2 Categories Based on System Architecture

Another way to categorize Federated Learning is the type of the system's architecture which is used to control the communication, coordination, and aggregation. There are two main architectures introduced and experimented mostly in the literature

- **Centralized federated learning**

  In centralized federated learning, the entire training process is coordinated by one central server [18]. This server is the one that takes care of the global model's initialization, the selection of the clients for each training round, the collection of the local model updates of the clients and their aggregation, and the rebroadcasting of the updated model to all clients. This architecture is simple to implement and efficient in terms of convergence, and it forms the backbone of most practical FL deployments.

  However, with all its effectiveness, this architectural design poses a single point of failure and requires trust in the central server, which can become a target for attacks or a bottleneck in communication.

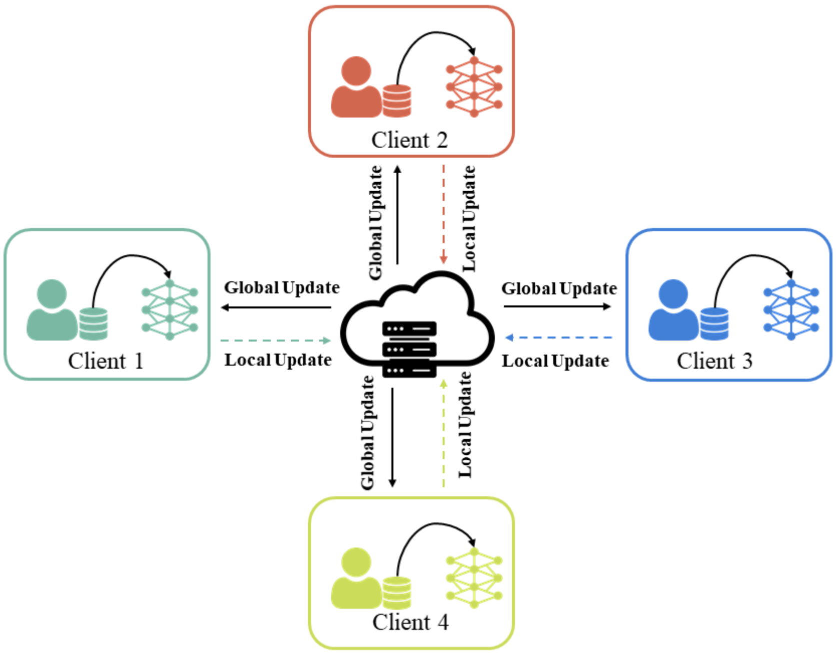


Figure 1.8: Centralized federated learning architecture

- **Decentralized Federated Learning (DFL)**
  In decentralized federated learning, there is no central server. Instead, clients collaborate with each other in a peer-to-peer manner as shown in (Figure 1.9) to train the global model [19]. Each client communicates only with a subset of other clients, forming a communication graph or overlay network. Aggregation and model updates are performed locally or collaboratively, often requiring techniques from consensus algorithms, such as gossip protocols [20] or blockchain-based [21] coordination.
  This architecture improves fault tolerance, removes the central trust assumption, and enhances robustness to failure or adversarial behavior. However, it introduces greater complexity in synchronization, convergence guarantees, and communication overhead.

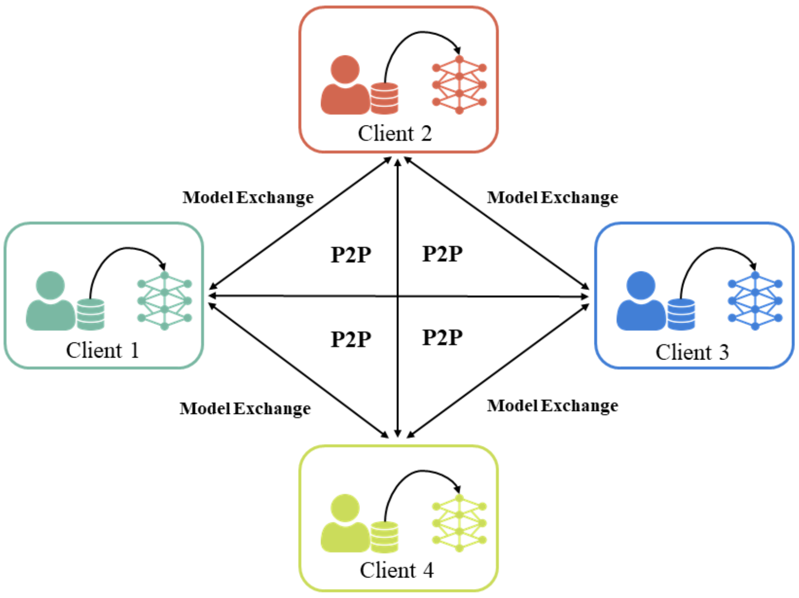


Figure 1.9: Decentralized federated learning architecture

1.5.3.3 Categories Based on Federation Scope

Federated Learning can also be categorized based on the scope of participation and the nature of the federation. This classification addresses the question of who the clients are, how frequently they participate, and how reliable or resource-constrained their environments tend to be. The two primary categories under this scope are cross-device federated learning and cross-silo federated learning.

- **Cross-device FL**
  Cross-device FL refers to scenarios in which the federation is composed of a large population of personal or edge devices, such as smartphones, tablets, or IoT sensors [22] (Figure1.10). These clients typically contribute small, non-iid (non-independent and identically distributed) datasets and may be intermittently available due to constraints in power, connectivity, or user behavior. Cross-device FL is often used in consumer-facing applications such as mobile keyboard prediction, voice assistants, or wearable health monitoring

- **Cross-silo FL**
  Cross-silo FL involves collaboration between a small number of stable and reliable institutions, such as hospitals (as shown in Figure 1.11), banks [23]. These entities typically have larger datasets, consistent availability, and well-maintained infrastructure. The collaboration is usually governed by formal agreements or shared goals.

  Cross-silo FL is highly applicable in domains like healthcare and finance, where data sharing is restricted, but model collaboration is beneficial.

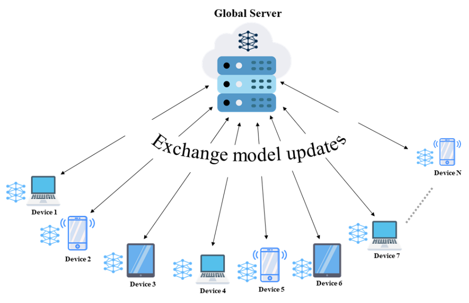


Figure 1.10: Cross-device FL

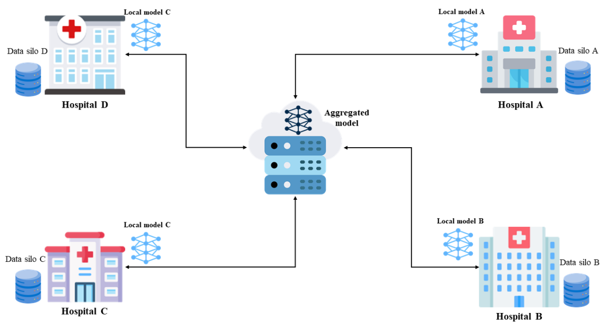


Figure 1.11: Cross-silo FL

1.5.4 Data Distribution in Federated Learning

In Federated Learning , the performance and behavior of the global model are heavily influenced by the way data is distributed across participating clients. The two primary modes of data distribution are Independent and Identically Distributed (IID) and Non-Independent and Identically Distributed (Non-IID). These modes represent fundamentally different learning conditions and present unique challenges in optimization and generalization.

1.5.4.1 Independent and Identically Distributed (IID)

- **Independence:**
  Definition: The generation of data is consistent. Formally, for any class y and a feature set X, the joint probability *P(X,y)* can be decomposed into the product of the marginal probabilities *P(X)* and *P(y)* [24]:

$$P(X, y) = P(X) \cdot P(y) \tag{1.5}$$

- **Identical Distribution:**
  Definition: The datasets from different clients fall under the same probability distribution. Formally, for any class y, the conditional probability P(y|Di) is the same across all clients [24]:

$$P(y \mid D_1) = P(y \mid D_2) = \ldots = P(y \mid D_n) \tag{1.6}$$

In an IID setting, the data across clients is both consistent in its generation process and drawn from the same underlying probability distribution. This ensures that the local data distribution of each client closely resembles the global data distribution, enabling all clients to contribute equally to the learning process. As a result, the aggregated model converges faster and achieves higher accuracy. For example, if participants in a federated learning task are training a model to classify images of cats, dogs, and rabbits, each participant would have a balanced and representative dataset of these classes, allowing the aggregated model to generalize well [24].

1.5.4.2 Non-Independent and Identically Distributed (Non-IID)

- **Non-Independence:**
  Definition: Clients with different behaviors generate conflicting or inconsistent data. For example, in an object identification model, each client might take images from varying angles, leading to a feature distribution skew[24].
- **Non-Identical Distribution:**
  Definition: Clients' local data is drawn from different data distributions. For example, in a classification task, each client might have a bias towards a specific class, leading to a label distribution skew[24].

In a Non-IID setting, the data across clients is either inconsistent in its generation process or drawn from different underlying probability distributions. This creates significant challenges in the federated learning process, as the global model struggles to generalize due to the heterogeneity of the data. To address this issue, specific aggregation strategies have been developed. For instance, the Scaffold algorithm proposed by Karimireddy et al [25]. introduces a control variate mechanism to correct for client drift caused by Non-IID data during the federated averaging process. Despite such advancements, Non-IID data remains a critical challenge in FL, as it often leads to slower convergence, reduced accuracy. For example, if one participant has mostly images of cats, another mostly dogs, and a third mostly rabbits, the aggregated model may fail to capture the statistical patterns of all classes effectively, resulting in suboptimal performance [24].

Figure 1.12 illustrates the difference between IID and Non-IID data distributions across clients in a federated learning setup, highlighting how IID ensures uniform class distribution, while Non-IID results in significant data imbalance and heterogeneity among clients.

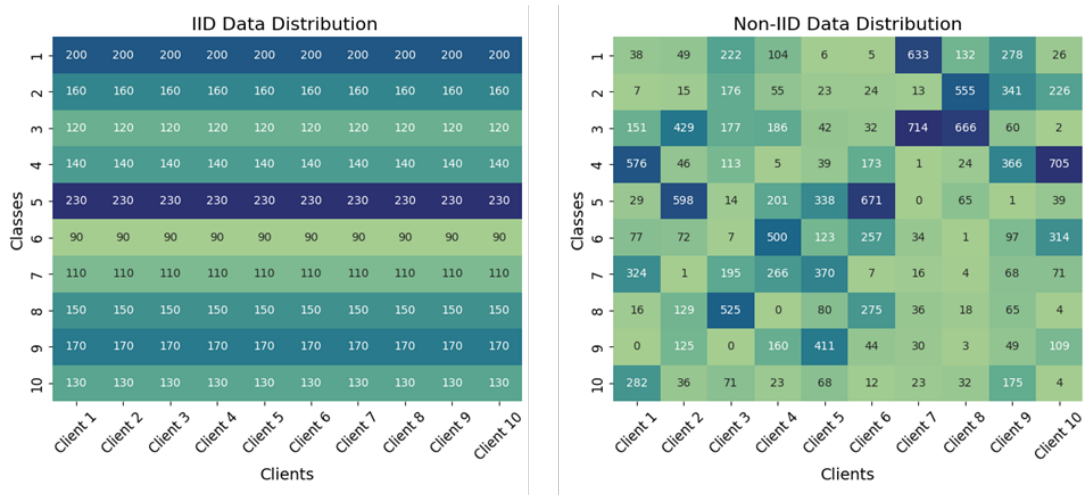


Figure 1.12: Federated Learning Data Distribution IID and non-IID

1.5.5 Benefits of federated learning

Federated Learning introduces a transformative learning paradigm that brings computation to the edge, enabling collaborative model training while preserving data privacy and system decentralization. Its design provides several key advantages over traditional centralized machine learning approaches, making it especially suitable for privacy-sensitive and distributed environments.

1. **Privacy Preservation**: Federated learning enables privacy-preserving model training by retaining raw data on local devices, thereby eliminating the need for centralized data aggregation. This approach mitigates privacy risks inherent in traditional centralized learning paradigms and aligns with privacy regulations (e.g., GDPR), as sensitive information never leaves the device [26].

2. **Efficient Communication and Edge Resource Utilization**: Federated Learning improves system performance by reducing communication overhead and optimizing the use of edge device resources. Instead of sending raw data to a central server, clients only share model updates such as gradients or weights. This approach lowers bandwidth usage and supports data privacy. Several techniques help reduce the volume and frequency of communication, including gradient sparsification, quantization [27], and adaptive aggregation strategies like FedCAMS [28]. In parallel, FL benefits from edge computing, where training is performed directly on devices with limited resources. This reduces dependency on cloud infrastructure and supports real-time applications[29]. Huang et al. [30] show that optimizing communication topology can speed up training while lowering bandwidth demands in edge-based FL systems.

3. **Scalability :** Federated Learning inherently supports scalability by enabling distributed model training across numerous devices or nodes . This decentralized approach allows the system to efficiently handle increasing data volumes and participant numbers, making it particularly suitable for large-scale applications such as IoT networks and mobile devices. By distributing the computational load and minimizing communication overhead, FL systems can expand their capacity without overburdening central servers or compromising performance.

4. **Increased Generalizability of AI Models** Federated Learning enhances the generalizability of AI models by leveraging data from a multitude of sources without compromising individual data privacy. This approach allows models to learn from a broader spectrum of data distributions, capturing diverse patterns and variations that improve their performance across different tasks and populations. Consequently, FL-trained models are better equipped to generalize to new, unseen data, making them more robust and effective in real-world applications.

1.5.6 Application of federated learning

1. **Healthcare**
   Federated Learning is particularly relevant to the healthcare field due to the critical need for large-scale data in training effective machine learning models alongside stringent data privacy requirements. Electronic Health Records (EHRs) constitute the main data source for such applications [31], but models trained solely on local

data often suffer from bias and lack generalizability. Achieving the requisite data diversity typically involves multi-institutional collaboration. However, the highly sensitive character of health information, coupled with rigorous regulations governing its use(e.g., HIPPA), severely restricts traditional data-sharing approaches. Standard de-identification methods frequently fail to adequately protect patient privacy [32], and the practical challenges associated with high-quality dataset creation and maintenance further discourage data pooling [33]. Federated Learning addresses these constraints by enabling distributed model training directly at the data source. This approach facilitates the development of more generalized and robust healthcare models by leveraging diverse datasets across institutions while adhering to critical privacy preservation mandates.

2. **Finance and Insurance**
   The financial and insurance sectors are increasingly recognizing the imperative for inter-institutional collaboration to effectively leverage the power of big data. This has catalyzed a transformative trend towards adopting Federated Learning paradigms. Recent advancements demonstrate FL's nascent application across critical financial functions, notably in refining risk control mechanisms, optimizing marketing strategies, and bolstering anti-money laundering efforts. Both Horizontal FL, used by banks for credit forecasting, and Vertical FL, used by different institutions for loan repayment prediction, help improve understanding of customer investment ability and credit scores. Participating institutions share model training results without exposing raw data. However, FL in finance is still largely in the early research phase, with limited large-scale deployment. A notable application is WeBank's use of FL for risk management in small business and personal loans, improving credit scoring and insurance claims prediction. Research is also advancing, with frameworks like privacy-preserving boosting trees based on VFL being developed for collaborative credit scoring [34]. The combination of FL and finance is seen as a promising way to overcome data silos and drive intelligent applications in the privacy-sensitive financial and insurance sectors.[35]

3. **Recommendation System**
   Traditional recommender systems are significantly challenged by inherent user privacy concerns and the constraints imposed by data silos, which impede the conventional aggregation and analysis of extensive user data. Federated Learning presents a compelling paradigm shift to address these limitations, particularly in facilitating cross-domain recommendation tasks. This involves the federalization of established recommendation algorithms, including those based on collaborative filtering [36], deep learning, and meta-learning. Within this framework, the vertical federated recommendation architecture is noteworthy for its capacity to leverage data from disparate domains, thereby enriching the quality of content recommendations provided to users. Substantial research efforts are dedicated to augmenting the accuracy and deepening the integration of FL within recommendation systems [[37], [38]]. These initiatives encompass the development of novel approaches, such as FL-based recommendation algorithms designed for accelerated training of distributed models, exemplified by methods like FedFast [39] which builds upon and enhances the classical FedAvg algorithm.
   the core objective of integrating FL with recommendation systems is to deliver accurate and practical recommendations while strictly safeguarding user privacy and

trade secrets, thereby advancing recommendation technology across various commercial entities.[35]

4. **Autonomous Systems and the Internet of Things**
   Autonomous Systems, IoT, Cyber-Physical Systems (CPS), and Edge Computing, empowered by AI/ML/DL, are central to Industry 4.0. Their growing deployment necessitates integrating data privacy and smart processing into model training. Federated Learning (FL) is being adopted to address this need. For instance, the ”FengHuoLun” FL framework [40] is used to train ML models for adding trustworthiness to CPS at edge devices and implementing smart services. Personalized FL models [41] have shown promise in handling the heterogeneity inherent in IoT environments, as demonstrated in Human Activity Recognition Systems leveraging edge computing. Overall, the application of FL in various IoT and CPS scenarios on Edge networks is considered a trustworthy, smart, and efficient alternative to traditional centralized approaches.

5. **Natural Language Processing (NLP)**
   Training accurate Natural Language Processing models requires large datasets, often from user devices, but centralized collection poses a significant privacy risk due to sensitive textual data. Federated Learning provides a solution by enabling model training on decentralized data, addressing this privacy bottleneck while maintaining feasibility for NLP tasks [42]. Consequently, FL is being applied to various NLP applications, including improving mobile keyboard suggestions (like Google’s GBoard [43]), predicting the next word [44], which can outperform server-based methods. Frameworks like FedBERT [45] further demonstrate FL’s use with pre-trained models, limiting raw data access and yielding better results. Overall, FL allows for effective NLP model development while preserving user privacy.

1.6 Conclusion

This chapter established the context for Federated Learning, beginning with the foundational principles of Artificial Intelligence , Machine Learning , and Deep Learning . It highlighted how the conventional centralized training paradigm, despite its achievements, presents significant challenges related to data privacy, security, and logistical feasibility. In response, the field has shifted toward distributed and collaborative learning frameworks capable of extracting insights from siloed data. Federated Learning stands out as a pivotal development in this evolution, offering a practical architecture that enables multiple parties to train shared models without exchanging raw data. The chapter further examined its classifications, based on data partitioning, system architecture, and federation scope, explored its ability to handle diverse data distributions, and emphasized its advantages, such as privacy preservation, scalability, and communication efficiency. Applications in critical sectors such as healthcare, finance, and mobile systems underscore their real-world impact. Ultimately, Federated Learning emerges not merely as a technical innovation, but as a foundational step toward realizing large-scale machine learning systems that uphold data privacy, autonomy, and ethical collaboration.

Chapter 2

State-of-Art : Privacy Threats and Defense Mechanisms in Federated Learning

2.1 Introduction

This chapter explores the important facets of Federated Learning privacy landscape, building on the fundamental ideas covered in the previous chapter. FL eliminates the need for centralizing raw data, which has many benefits, but the protocol is not impervious to privacy violations. Despite being supposedly abstract, the iterative exchange of model parameters, like weights or gradients, could be a source of unintentional information leakage. As a result, maintaining participant data confidentiality is still a top priority and a topic of ongoing research. The state of the art is thoroughly reviewed in this chapter, which also critically examines the known privacy flaws in FL systems and surveys the well-known defenses created to lessen these risks. Cryptographic techniques such as homomorphic encryption are given special attention. In order to put the current state of research in context, a review of relevant related works that combine these techniques will also be presented.

2.2 Privacy Threat Landscape in Federated Learning

In order to examine the privacy implications of Federated Learning, it is necessary to go beyond the protocol's fundamental tenet of data decentralization and examine the information flow that it incorporates. Despite being abstract, the model updates that are shared during training may be subject to a number of privacy threats. This section takes a multifaceted approach to methodically comprehend these potential risks, taking cues from well-known taxonomies like the 5W scenario model put forth by Yin et al.[46]. By considering who might attack , their level of knowledge and capability, the strategies they use, the time and location of the attack within the FL lifecycle, and the motivation behind the attack (i.e., the specific inference goal), we will investigate potential privacy leaks. The foundation for assessing suitable defence mechanisms is laid by this methodical analysis.

2.2.1 Potential Adversaries

Analyzing the privacy concerns related to Federated Learning first requires determining who might endanger the privacy. Depending on their relative position to the FL system, adversaries can be generally classified as internal actors engaged in the learning process or external actors functioning outside of it. Assessing vulnerabilities depends on knowing the possible access and reasons behind every kind.

2.2.1.1 Internal Adversaries

These are entities directly involved in the FL training protocol and typically have privileged access to certain components of the system.

- **Central Server**:
  In common client-server FL architectures (required for standard horizontal FL or coordinated vertical FL), the central server orchestrates the training process. It receives model updates (e.g., gradients or weights) from participating clients, performs aggregation, and distributes the updated global model. This central position grants the server access to intermediate training updates and the evolving global

model. Consequently, a server acting in an "honest-but-curious" manner, or one that is fully compromised or malicious, represents a significant internal threat. For instance, a curious server might analyze client updates to reconstruct sensitive input data.

- **Participating Clients**:
  Clients hold the local datasets and perform local training computations. While they do not see other clients' raw data, they receive the global model parameters in each round and contribute their own updates. Malicious clients participating in the federation can pose several threats. They could send maliciously designed updates designed to probe the data of other honest clients or manipulate the global model to leak specific information [47].
  Furthermore, a malicious entity might employ a Sybil attack[48], creating numerous pseudo-client identities to gain disproportionate influence over the training process or increase the opportunities for inference.

2.2.1.2 External Adversaries

These entities are not direct participants in the FL training loop but may interact with the system or its outputs.

- Model Consumers:
  These actors interact with the final, trained FL model, typically after deployment, often through query APIs [49]. If malicious, or controlled by an attacker, they can leverage their access to the model's predictions or potentially its parameters (if publicly released) to launch inference attacks

- Eavesdroppers:
  These adversaries attempt to intercept the communication between clients and the server (or between peers) [50]. If communication channels are not adequately secured (e.g. via TLS/SSL), eavesdroppers might steal intermediate updates (gradients/weights) or the final model parameters. This stolen information could then potentially be used for reconstruction attacks While requiring more effort than direct model access and potentially mitigated by cryptographic network security, eavesdropping remains a potential vector, especially for sensitive intermediate updates.

Therefore, evaluating FL privacy requires considering threats originating both from within the federation (server, clients) and from external entities interacting with the system or its communication channels.

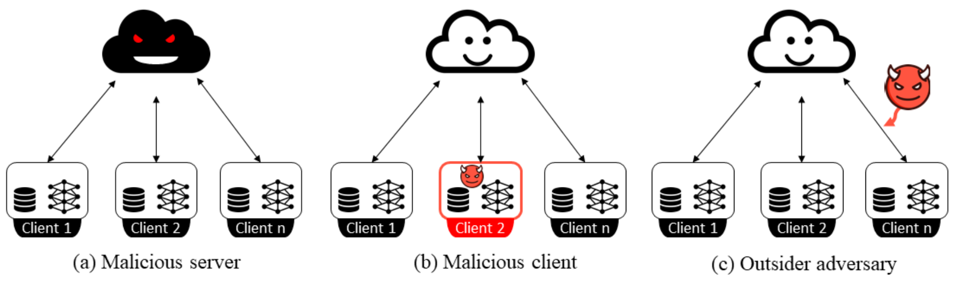


Figure 2.1: Different potential adversaries in vulnerable FL systems.

2.2.2 Adversary Knowledge and Capabilities

Beyond the adversary's position within the system, their potential effectiveness is heavily influenced by their level of knowledge regarding the target system's internals and their capability to manipulate communications or computations. These aspects define the sophistication and potential impact of an attack.

2.2.2.1 White-box Access

This model assumes the adversary possesses significant knowledge about the internal workings of the FL system or the target model . In the context of FL this could include access to:

- The specific model architecture being trained (e.g., layers, activation functions).
- The training algorithm and its hyperparameters (e.g., learning rate, optimizer,loss function).
- Intermediate values generated during training, most notably the gradients or model weights exchanged.

An honest-but-curious or malicious central server often operates under a white-box assumption concerning the updates it receives from clients. Similarly, participating clients (honest or malicious) typically have white-box access to the global model parameters distributed by the server in each round. Attacks leveraging white-box knowledge, such as many gradient inversion techniques [51], can be particularly powerful due to the detailed information available to the adversary.

2.2.2.2 Black-box Access

In this model, the adversary interacts with the FL system or the resulting model solely through its defined external interfaces, without knowledge of its internal structure or parameters. Typical black-box interactions include :

- Submitting queries to a deployed model and observing the output predictions.

External model consumers typically operate under a black-box assumption when targeting a deployed model via its prediction API. Although potentially less informative than white-box access, black-box attacks such as membership inference based on query responses can still pose significant privacy risks.

2.2.3 Attack Strategies

The strategies used by adversaries in federated learning can be broadly categorized based on whether they merely observe the system or actively interfere with its operation.

2.2.3.1 Passive Attacks

A passive attacker aims to learn or infer sensitive information by observing the FL process and its communications without modifying the system's resources or operations. In the context of FL, passive attackers typically monitor legitimate computations and data exchanges, such as model weights, gradients, or the final aggregated model, during both the training and inference phases [46]. The specific information obtainable depends heavily on the adversary's knowledge level:

- **Passive Black-box:**
  Assumes the adversary can only access the system's outputs, such as predictions obtained by querying a deployed model API, without visibility into internal parameters or intermediate updates. While the risk may be comparatively limited, techniques like membership inference attacks have been studied under this model[52].

- **Passive White-box:** Assumes the adversary has access to internal information, which could include intermediate training updates (gradients/weights), model parameters, and potentially query results. This level of access generally enables more potent inference attacks, potentially revealing significantly more sensitive information about the training data [52].

Passive attacks primarily consist of various inference techniques aimed at extracting unintended information simply by observing the legitimate flow of the FL protocol.

2.2.3.2 Active Attacks

In contrast, active attacks involve adversaries who deliberately attempt to alter the resources of the system or affect its operations to achieve their goals. Within FL, active attackers seek to influence the training process itself, often to extract sensitive information more effectively or to compromise the integrity of the model. Active strategies are generally considered more powerful than passive ones, because the adversary can directly manipulate parts of the system. Examples of active attack manifestations in FL include:

- **Malicious Client Uploads:**
  A malicious participating client might send deliberately crafted or modified gradients/weights instead of honestly computed updates. This can be done to mislead the global model into learning specific, potentially sensitive features related to other clients' data [47] or to manipulate the learning process (e.g., stochastic gradient descent) to enhance the leakage of information like membership status [52].

- **Malicious Server Actions:**
  A malicious central server could manipulate the aggregation process or selectively interact with clients (e.g., isolating them) to gain more information or control the model outcome [53].

- **Strategic Client Control:**
  An adversary controlling multiple clients could potentially adjust their training data or updates strategically to embed specific patterns or weaknesses into the global model [54].

Active attacks aim to exploit the interactive nature of FL, turning protocol steps into opportunities for manipulation to achieve adversarial objectives, often related to extracting private data more effectively than passive observation would allow.

2.2.4 Attack Timing and Vector

Privacy leakages in federated learning can manifest during distinct phases of the system's lifecycle, targeting specific types of information exchanged or generated throughout the process. The two primary phases are the training phase, where the collaborative model building occurs, and the inference phase, involving the use of the trained model[46]. Both phases present unique vulnerabilities.

2.2.4.1 Training Phase

This phase encompasses the iterative model building process, including local client computations, update communication, server aggregation, and global model distribution . The primary vectors for privacy leakage during this crucial phase are the intermediate model updates communicated between participants. This includes

- **Local Model Updates (Gradients/Weights):**
  Clients compute these updates directly based on their own local data. When these updates are sent to the server for aggregation, they inherently contain sensitive information about the local data. Gradients, in particular, have been shown to be vulnerable to reconstruction attacks that can restore training data with high accuracy [55]. This could also be extended to exploiting weight updates alone, as discussed in [56]. Exposing these local updates to external parties, such as a curious server or malicious peers, poses a significant privacy risk.

- **Aggregated Gradients and Weights:**
  Even the aggregated updates, often broadcast back to clients after server processing, can leak information. Malicious participants receiving these may attempt inference attacks targeting the contributions of others within the aggregate. Research has shown that aggregated gradients can be exploited to perform attacks, such as recovering private labels of individual clients, thereby compromising privacy in federated learning systems [57].

Therefore, any parameter information exchanged during the iterative training is a potential point of vulnerability

2.2.4.2 Inference Phase

This phase occurs after collaborative training concludes and the final model is deployed for use, often as a service. Attacks during this phase target the finalized model itself through two main vectors:

- **Final Model Parameters:**
  If an adversary gains white-box access to the parameters of the fully trained model after deployment, they can directly analyze this information. Such access allows for potent attacks; for example, model inversion techniques have been demonstrated to extract sensitive information used during training, such as reconstructing recognizable facial images from a face recognition model by leveraging the model's parameters [58]. This highlights the risk associated with releasing or exposing the internal details of a trained FL model.

- **Model Query Outputs:**
  Even without access to the internal model parameters (i.e., in a black-box setting), adversaries can attack the deployed model by observing its input-output behavior. This is particularly relevant when models are deployed as a service, potentially via a Machine Learning-as-a-Service API [59]. By carefully crafting queries and analyzing the model's predictions or confidence scores, adversaries can still infer private information. Notably, membership inference attacks can be conducted in this manner, potentially determining whether a specific data record was part of the original training set by observing how the model responds to queries involving that record [60].

2.2.5 Attack Objectives: Inference Goals

Beyond understanding the actors, their capabilities, strategies, and timing, a complete threat analysis requires identifying the specific adversarial objectives, typically centered around inferring sensitive information about the private training datasets used in Federated Learning (FL). These objectives fall under the umbrella of inference attacks, which can be categorized based on the type of information the adversary seeks to extract [61].

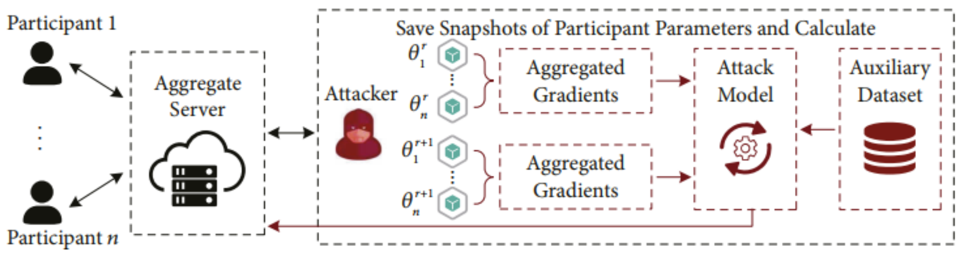


Figure 2.2: Overview of inference attacks in FL. The attacker saves the snapshots of the aggregated model parameters in each round and performs inference attacks by employing the difference between the continuous snapshots.[62]

2.2.5.1 Membership Inference Attack

Membership inference seeks to accurately determine if a specific sample contributed to the training of the network, operating simultaneously on all target samples [60].For instance, verifying a patient's clinical record utilized to train a model linked to a certain ailment would disclose the patient's health status.
Formally, given a query input x and a target model $F(\theta)$, an MIA algorithm A attempts to infer the *membership status* m of x, indicating whether or not x was included in the training dataset D_{tr} [63]. This inference task is commonly modeled as a binary classification problem:

$$m = A(x, F(\theta)) = \begin{cases} 1, & \text{if } x \in D_{\text{tr}} \\ 0, & \text{if } x \notin D_{\text{tr}} \end{cases}$$

In this formulation, $m = 1$ implies that the adversary believes the input x was used to train the model, whereas $m = 0$ indicates otherwise.

In the context of Federated Learning (FL), where multiple clients collaboratively train a shared model without exchanging raw data, Membership Inference Attacks (MIAs) pose a significant privacy risk. These attacks exploit the iterative nature of FL to determine whether specific data points were used in the training process. A seminal work by Nasr et al. [52] demonstrated that both passive and active MIAs could be effectively launched against FL models, with adversaries achieving high accuracy in inferring membership status by analyzing model updates. Melis et al. [47] further explored inference attacks in collaborative settings, showing that malicious clients could leverage differences in the global model over time to deduce membership information from other clients' data. Additionally, Truex et al. [49] provided a systematic framework for constructing black-box MIAs, which can be adapted to FL scenarios. These studies collectively highlight the susceptibility of FL to MIAs and emphasize the critical need for enhanced privacy protections in distributed learning environments.

2.2.5.2 Property Inference Attack

Property Inference Attacks seek to derive particular features or characteristics of the training dataset that may not be directly associated with the model's principal learning objective. An adversary may seek to deduce if individuals in the training dataset were wearing glasses, despite the model being only trained for gender categorization. Such assaults pose significant privacy problems, as they facilitate the inadvertent disclosure of information regarding the data. Moreover, with adequate prior knowledge, an adversary may ascertain the existence of particular samples within the training dataset.[64] Melis et al. [47] were among the first to demonstrate that shared gradients can expose unintended properties, such as demographic attributes, using auxiliary classifiers. Expanding on the general concept of PIAs, Ganju et al. [65] showed that fully connected neural networks can leak high-level characteristics of the training dataset by exploiting permutation-invariant representations. Their approach inferred global attributes—such as the proportion of data from a specific class or environmental conditions during data collection—that were never part of the training objective. Kerkouche et al. [66] later demonstrated that even with secure aggregation protocols in place, adversaries could still mount client-specific property inference attacks by analyzing updates (Gradients) across rounds. Collectively, these studies reveal that PIAs remain a significant privacy threat in FL systems and highlight the need for mitigation strategies.

2.2.5.3 Class Representative Inference Attack

In federated learning are a form of privacy leakage wherein an adversary aims to reconstruct representative samples of specific classes from the training data. These attacks exploit the information shared during the collaborative training process, particularly gradients, to generate synthetic data that closely resembles the original training samples.
Fredrikson et al. [67] introduced the concept of model inversion attacks, demonstrating that adversaries could reconstruct input data by exploiting the outputs of machine learning models. Building upon this, Hitaj et al. [68] proposed a more potent attack in the context of FL. They showed that a malicious participant could train a Generative Adversarial Network (GAN) [69] during the FL process to generate class representatives that closely resemble the training data of other participants (Figure 2.3). This approach is particularly effective when the data within a class is homogeneous; for instance, in a facial recognition task where all images pertain to the same individual, the reconstructed images can be strikingly similar to the original ones.Further advancing this line of research, Wang et al. [70] developed a GAN-based framework with a multi-task discriminator ca-

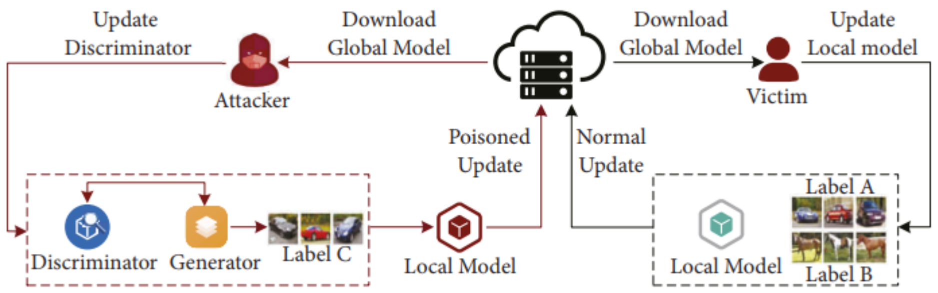


Figure 2.3: By using the client-side GAN attacks, the attacker can reconstruct sensitive information from the victim.[62]

pable of simultaneously distinguishing data category, authenticity, and client identity. This enhancement enables the generation of user-specific data representatives, posing a significant threat to individual privacy in FL systems.

2.2.5.4 Data Reconstruction Attack

Data Reconstruction Attacks (Training Sample and Label Inference) in Federated learning pose a substantial privacy risk, as adversaries aim to retrieve original training data samples and their associated labels by leveraging information disseminated during the training process[71]. Classifying data reconstruction attacks in federated learning based on their underlying reconstruction mechanism offers valuable insight into their operational differences. Proposing a distinction between ”Optimization-Based Attacks” and ”Closed-form Attacks” appears logical based on current literature [64]

- **Optimization-based attacks**
  Attacks based on optimization try to retrieve original training data by iteratively optimizing inverse problems [64]. One of the most notable approaches is Deep Leakage from Gradients (DLG), introduced by Zhu et al. [72]. This method initializes dummy data and labels, then iteratively updates them to minimize the distance between the gradients computed on the dummy data and the actual gradients shared by clients (as shown in Figure 2.4). As a result, it can reconstruct input samples with high pixel-wise accuracy. An improved version, known as Improved Deep Leakage from Gradients (iDLG), was proposed by Zhao et al. [73]. This attack exploits the mathematical relationship between the gradient direction and the ground-truth label, allowing the adversary to directly infer the true label from gradient signs under common loss functions such as cross-entropy, thereby improving reconstruction efficiency. Unlike these approaches that rely on gradients, the Deep Leakage from Model (DLM) attack [74] reconstructs input data by leveraging weight updates. This method assumes access to the model before and after a training step, and it optimizes dummy inputs to match the observed weight changes. DLM shows that even in the absence of gradient information, model weights alone can leak sensitive information when carefully analyzed.

- **Closed-form attacks**
  In contrast to iterative optimization methods, closed-form attacks aim to analytically recover input data by inverting model computations. A prominent example is the Linear Layer Leakage (LLL) attack [75], which targets fully connected layers.

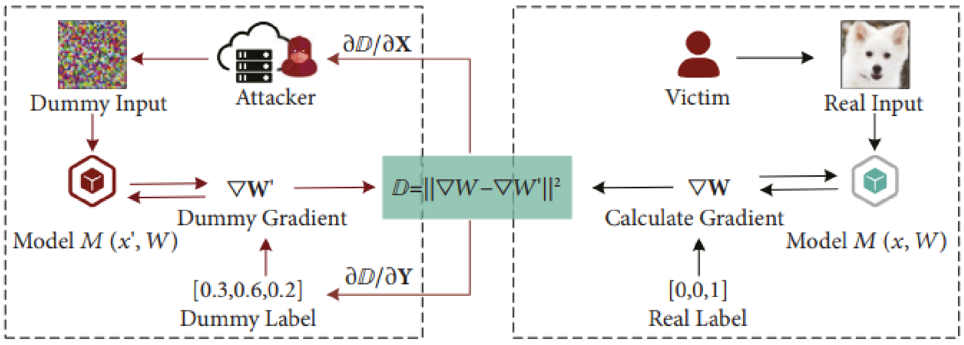


Figure 2.4: Overview of DLG attacks against FL based on an image classification task.[62]

This method assumes knowledge of the model weights and biases and uses linear algebra to invert the transformations applied by the linear layer. When conditions such as small batch size and well-conditioned matrices are met, this approach can accurately reconstruct the input without requiring iterative optimization.

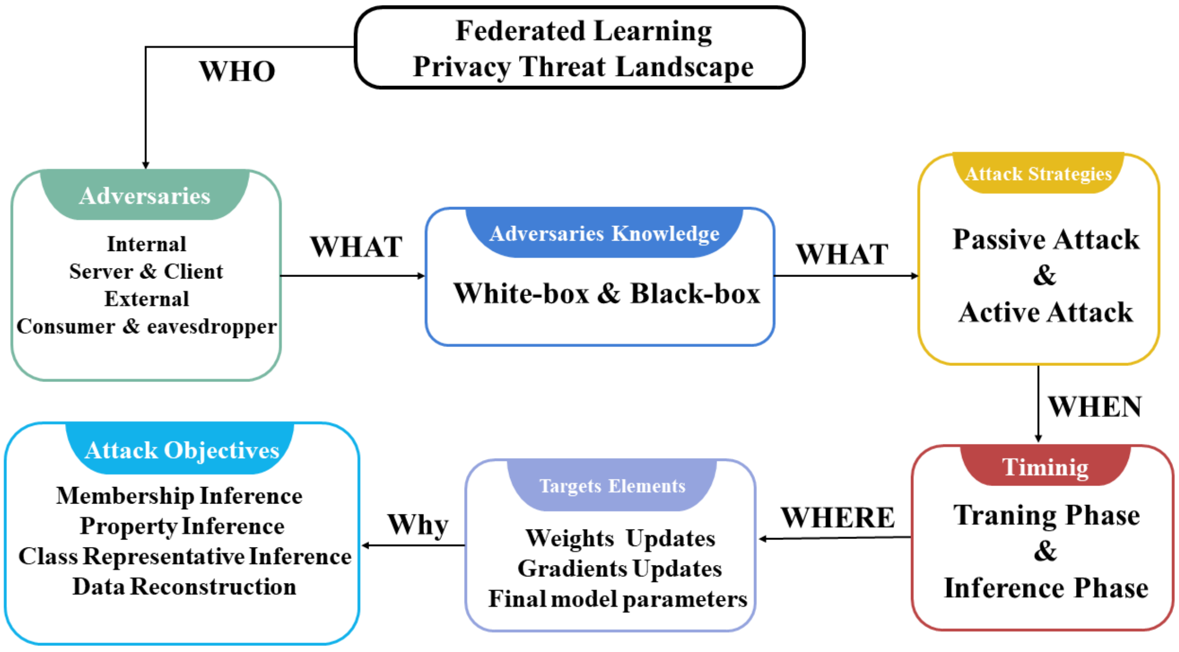


Figure 2.5: Taxonomy of Privacy Threats in Federated Learning

In conclusion, this analysis underscores the complex and multifaceted nature of privacy threats within federated learning systems. Vulnerabilities surpass the basic protocol's safeguarding of raw data, emerging through several attack routes at both training and inference stages. The likelihood of privacy violation is heavily contingent upon the particular opponent, including their status, expertise, abilities, and intentions, which might vary from membership inference to complete data reconstruction. Having established a comprehensive understanding of the various methods by which privacy can be compromised in federated learning, the discussion now shifts to the essential countermeasures and defense strategies devised by the research community to mitigate these risks and enhance the security of collaborative learning.

2.3 Defense Mechanisms for Privacy Preservation in FL

Beyond the fundamental idea of data localization, the study shown in the previous section emphasizes the major privacy hazards included in ordinary federated learning systems. Acknowledging these weaknesses, from membership inference to complete data reconstruction via gradient leakage, has encouraged a lot of study into building strong defenses. These approaches aim to reduce the recognized hazards and improve the privacy guarantees of the collaborative learning process, thereby building confidence and allowing the safe implementation of FL in sensitive fields. Reviewing the main kinds of defense measures used: mostly Differential Privacy (DP), safe Multiparty Computation (SMC) techniques (frequently used for safe aggregation), and Homomorphic Encryption (HE), this part analyzes the state of the art in privacy preservation for FL. Every method has different benefits and presents different difficulties with relation to privacy assurances, effect on model utility, and system overhead.

2.3.1 Differential Privacy

Differential Privacy has emerged as a rigorous mathematical framework for quantifying and limiting the privacy leakage associated with releasing aggregate information about datasets containing sensitive individual records .The core principle of differential privacy is to guarantee that the results of any analysis or query conducted on a dataset remain statistically indistinguishable, regardless of the inclusion of any individual's data within the dataset. This is generally accomplished by incorporating meticulously adjusted random noise into the computational process or its outcomes.[76]
Formally, a randomized algorithm M satisfies (ε, δ)-Differential Privacy if, for any two adjacent datasets D and D' (differing by only one individual's record), and for any possible set of outputs S, the following inequality holds [77]:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \cdot \Pr[M(D') \in S] + \delta \tag{2.1}$$

Here, $\Pr[\cdot]$ denotes probability. The parameter ε (epsilon) represents the privacy budget: a smaller ε value signifies stronger privacy protection, as it bounds the ratio of probabilities more tightly, making the outputs on D and D' very similar. A value of $\varepsilon = 0$ would imply perfect privacy (output independent of any single record). The parameter δ (delta) represents the probability that the strict ε-privacy guarantee might be broken; typically, δ is set to a very small value, making the guarantee hold with high probability. Common mechanisms used to achieve DP include adding noise drawn from specific distributions, such as the Laplace or Gaussian distributions, scaled according to the sensitivity of the function being computed and the desired privacy budget ε. [76]

2.3.1.1 Differential Privacy in Federated Learning (DPFL)

As mentioned before, although federated learning essentially provides a level of privacy by preserving the locality of raw data, model updates (gradients or parameters) exchanged during training can still leak sensitive information. Differential privacy provides an effective technique to further enhance privacy guarantees within the FL framework by adding noise to obscure the exact contribution of any individual client's data [76]. Implementing differential privacy in federated learning presents a basic challenge: the intrinsic trade-off between privacy preservation and model efficacy. Incorporating noise safeguards privacy, although it may adversely affect the accuracy and convergence rate of the trained global model . Extensive research concentrates on enhancing this equilibrium.[76]

2.3.1.2 Central Differential Privacy Federated Learning vs Local Differential Privacy Federated Learning

In federated learning, differential privacy is implemented in two forms: Central Differential Privacy Federated Learning (CDPFL) and Local Differential Privacy Federated Learning (LDPFL) (see Figure 2.6). Their distinction lies in the location of noise introduction and the degree of trust assigned to the server.

- **Central Differential Privacy Federated Learning**
  In CDPFL , clients send exact model updates to a trusted central server, which then aggregates them and adds noise post-aggregation to ensure privacy. This approach allows for better model utility than local DP, as the noise is scaled to the sensitivity of the aggregated data rather than individual contributions. However, its effectiveness is highly dependent on trust in the central server, which has access to unprotected client updates before noise is applied.[76]

- **Local Differential Privacy Federated Learning**
  In Local Differential Privacy Federated Learning , each client adds noise to its model updates before sending them to the server, removing the need to trust the central server. This ensures strong client-side privacy, as the server never sees raw updates. However, the independent noise addition by all clients leads to higher cumulative noise, which can significantly degrade model accuracy and convergence.[76]

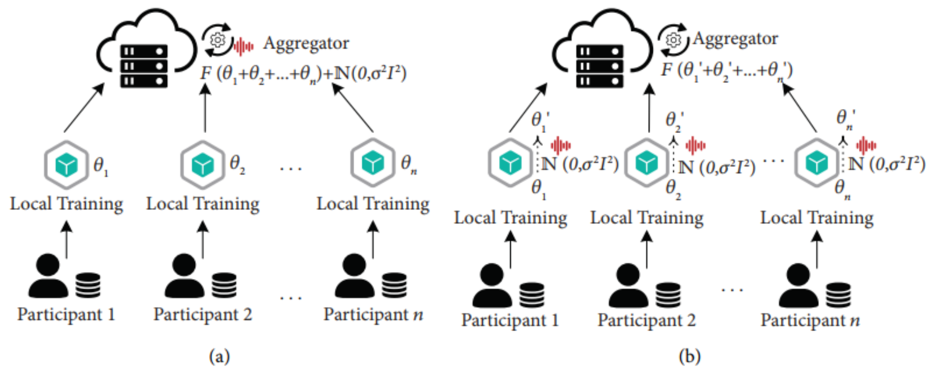


Figure 2.6: Architecture of Differential Privacy in FL. (a) CDPFL, (b) LDPFL.[62]

In essence, the choice between CDPFL and LDPFL involves a trade-off between the trust placed in the central server and the potential impact on the final model's performance due to noise injection.

2.3.2 Secure Multi-party Computation

Secure Multi-Party Computation (SMC or MPC) refers to protocols within cryptography that allow a set of distinct parties to jointly compute a function over their private inputs. The core requirements for such protocols are privacy and correctness: the privacy requirement ensures that parties learn nothing more than their prescribed output from the computation, keeping their individual inputs hidden from other participants; the correctness requirement ensures that each party receives the correct output corresponding to the agreed-upon function applied to all inputs. Essentially, SMPC aims to replicate the result of a computation performed by a trusted third party, but without requiring the actual presence or trust in such an entity.[78]

2.3.2.1 Secure multiparty computation in Federated Learning

Secure Multi-Party Computation techniques are integrated into Federated Learning (FL) frameworks primarily as a method to enhance data privacy during the collaborative training process . This integration typically focuses on securing the communication and computation involved in the model aggregation phase. Instead of clients sending potentially revealing model updates directly, SMC protocols are employed for secure aggregation . Mainstream multi-party computation techniques, such as homomorphic encryption or secret sharing, allow clients' local model updates to be obfuscated or encrypted before being sent . The FL server, or participants in a decentralized configuration, can execute the aggregation computation, such as averaging gradients, directly on these secured updates without seeing the specific, confidential contributions from each client. This guarantees that sensitive local data remains confidential throughout aggregation, providing enhanced privacy assurances relative to conventional federated learning methods.[79]

2.3.3 Homomorphic Encryption (HE)

Homomorphic Encryption (HE) is a sophisticated cryptographic method that enables computations to be executed directly on encrypted data without necessitating decryption. Grounded in the algebraic notion of homomorphism, wherein a function maintains the intrinsic structure between algebraic sets, HE schemes guarantee that particular operations performed on ciphertexts correspond to analogous operations on plaintexts during decryption.This distinctive attribute facilitates secure data processing in untrusted situations, such as cloud computing platforms, where privacy issues typically necessitate data decryption prior to utilization. In contrast to traditional encryption techniques that need decryption prior to computing, hence risking data exposure, homomorphic encryption preserves data confidentiality during the entire computational procedure.[80]
Let $\mathcal{M}$ (resp., $\mathcal{C}$) denote the set of plaintexts (resp., ciphertexts). An encryption scheme is said to be *homomorphic* if, for any given encryption key k, the encryption function E satisfies [81]:

$$\forall m_1, m_2 \in \mathcal{M}, \quad E(m_1 \odot_{\mathcal{M}} m_2) \to E(m_1) \odot_{\mathcal{C}} E(m_2)$$

for some operators $\odot_{\mathcal{M}}$ in $\mathcal{M}$ and $\odot_{\mathcal{C}}$ in $\mathcal{C}$, where $\to$ means "can be directly computed from," that is, without any intermediate decryption.

2.3.3.1 Types of Homomorphic Encryption

Homomorphic encryption schemes are broadly categorized based on the nature and extent of operations they support on encrypted data [80]. The primary types are:

- **Partially Homomorphic Encryption (PHE)**
  Partially homomorphic encryption schemes allow only a single type of operation,

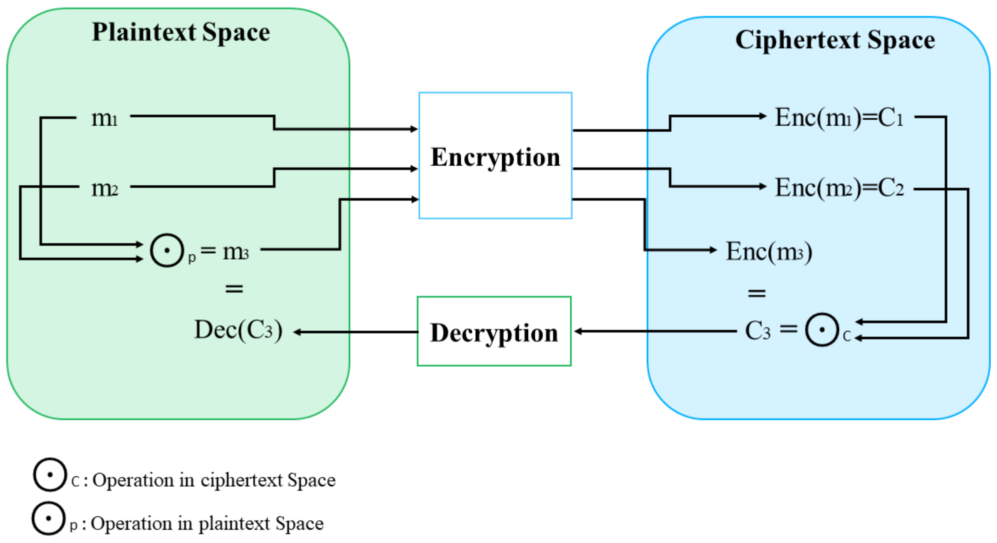


Figure 2.7: Homomorphic Property of Encryption: Correspondence Between Plaintext and Ciphertext Operations

either addition or multiplication, to be performed on ciphertexts an arbitrary number of times[80]. Although limited in scope, PHE schemes laid the foundation for more expressive homomorphic constructions.

- **Somewhat Homomorphic Encryption (SHE)**
  Somewhat Homomorphic Encryption supports both addition and multiplication operations but with limitations on the number of operations that can be performed. These limitations arise from noise growth in the ciphertext, which eventually makes decryption impossible after a certain number of computations[80]. SWHE is useful for applications requiring a small, fixed number of operations.

- **Fully Homomorphic Encryption(FHE)**
  Fully Homomorphic Encryption is the most powerful type, allowing unlimited addition and multiplication operations on encrypted data. This enables the evaluation of arbitrary computable functions on ciphertexts, making it suitable for complex applications like privacy-preserving machine learning and secure outsourcing of computations.
  The concept was first introduced by Gentry in 2009 [82] and has since evolved into more efficient and practical schemes.

2.3.3.2 Homomorphic Encryption in Federated Learning

Integrating homomorphic encryption into federated learning framework is a pivotal approach to enhancing data privacy without compromising model performance. To address the issue of privacy leakage due to local client updates, HE offers an effective solution. In this approach, clients first encrypt their local model updates. HE then allows a central server to aggregate these encrypted updates directly without the need to decrypt them. This mechanism ensures that the server can integrate client contributions without accessing the underlying sensitive information, preserving data confidentiality throughout the training process. This integration protects the privacy of individual data while preserving the overall utility of the federated model. Importantly, choosing appropriate HE schemes is essential to balance computational efficiency with security requirements, ensuring that integration avoids significant overhead costs. HE-enhanced FL systems therefore represent a promising direction for future research and applications, especially in scenarios where these privacy concerns are critical.[83]

2.4 Related works

This section examines previous studies on privacy preservation in federated learning, focusing on three primary paradigms: differential privacy, secure multi-party computation, and homomorphic encryption. Each of these techniques offers a distinct trade-off between privacy guarantees, computational efficiency, and communication overhead. We present key contributions from selected studies under each category, highlighting their methodologies, strengths, and practical relevance to our work.

2.4.1 Differential privacy based Federated learning

Weiet al. (2020) [84] In their work proposed a globally differentially private frameworke for federated learning. The study implemented procedures to balance the trade-off between privacy assurances and model efficacy through the calibration of Gaussian noise. The authors performed comprehensive experiments illustrating the impact of privacy budgets and sampling rates on model efficacy.

Zhiqiang et al. (2024) [85] presented an adaptive global differential privacy method for federated learning that dynamically adjusts the privacy budget. The allocation is influenced by key training indicators such as model accuracy, training loss, communication round, and client data volume. This adaptive approach aims to enhance the privacy-utility trade-off during training.

Truex et al. (2020) [86] introduced LDP-Fed, a federated learning framework incorporating local differential privacy by injecting noise directly at the client side. Their approach ensures privacy without requiring a trusted aggregator, preserving data confidentiality during communication. The study also analyzes the impact of LDP noise on model performance and utility.

Fan et al (2024).[87] proposed a federated learning algorithm based on Gaussian local differential privacy, combined with the SCAFFOLD optimization method. Their technique aims to reduce client drift while maintaining convergence stability by adding calibrated Gaussian noise to client updates. Experimental results show improved model performance under LDP constraints.

2.4.2 Secure Multi-Party Computation-based Federated Learning

Zhu et al (2020) [88]. proposed a weighted FL model secured via an oracle-aided SMPC protocol, separating cryptographic operations from aggregation logic. Their method ensures privacy-preserving model updates without a trusted third party. The work introduces a practical way to embed MPC in federated optimization.

Li et al (2021)[89].developed a Chain-PPFL framework using single-masking and chained communication mechanisms to protect data. Clients pass masked data serially, ensuring high security with low overhead. The method preserves privacy without significantly affecting learning accuracy.

Kanagavelu et al (2022) [90] This study tackles communication inefficiency in SMPC by combining update compression with encryption. CE-Fed lowers bandwidth requirements while preserving privacy using secure computations. It's especially suitable for resource-constrained FL settings.

chen et al (2024) [91] This recent study integrates SMPC with compressed sensing to improve both security and scalability. The model protects client data through secure aggregation while reducing communication costs using compression. Experiments confirm its robustness and efficiency for modern FL systems.

2.4.3 Homomorphic Encryption based Federated Learning

Zhang et al (2020) [92] introduces BatchCrypt, a novel system for efficient aggregation in cross-silo federated learning using leveled CKKS homomorphic encryption. It reduces communication and computation overhead through batching and model quantization while ensuring strong privacy guarantees.

Jayaram et al (2020) [93]. propose MYSTIKO, a federated learning framework utilizing the Paillier encryption scheme to perform secure aggregation of gradients without the use of noise. Their system is designed for settings where a semi-honest cloud server coordinates training, while ensuring that gradients and model updates remain hidden. This approach emphasizes exact computation and eliminates the trade-off between accuracy and privacy.

Ma et al (2022) [94].,in "Privacy-preserving Federated Learning Based on Multi-Key Homomorphic Encryption", develop a privacy-preserving scheme where each client encrypts its data using its own public key under a multi-key homomorphic encryption (MKHE) framework. The server can then aggregate encrypted updates without decrypting them, and without needing all clients to share the same key. This improves both security and scalability in multi-client settings.

Jin et al (2024) [95].this work presents FedML-HE, a useful federated learning system integrating homomorphic encryption (HE) to improve privacy preservation. Understanding the computational and communication overheads connected to HE, the authors offer Selective Parameter Encryption—a method that encrypts just the most privacy-sensitive model parameters. This selected strategy greatly lowers the encrypted model update size, hence increasing efficiency. The system also enables encryption key management and has a universal overhead optimisation engine, therefore enabling scalable deployment over distributed edge devices.

Pan et al [96].introduce a method that divides model parameters based on sensitivity and selectively encrypts them. By applying adaptive segmentation using the CKKS scheme, the system significantly lowers encryption and aggregation costs while maintaining privacy protection for critical parameters. This balances performance and security in large-scale FL.

Korkmaz et al [97]. in the recent study introduce a Selective Homomorphic Encryption method within their framework FAS to enhance privacy-preserving federated learning. Their method encrypts only a subset of sensitive layers using homomorphic encryption, rather than encrypting all model parameters, while employing noise and bitwise scrambling on the remainder. This technique markedly diminishes computing expenses and training duration—attaining up to a 90 % speedup—while maintaining model accuracy, and is corroborated across various medical imaging datasets.

Table 2.1: A most prominent works in Privacy-Preserving Federated Learning

Technique	Reference	Key Contribution	Year
Global Differential Privacy	Zhiqiang et al [85].	Proposed an adaptive global differential privacy method that dynamically adjusts the privacy budget based on training indicators like accuracy and data volume to optimize the privacy-utility trade-off.	2024
Local Differential Privacy	Fan et al [87].	Combined Gaussian LDP with SCAFFOLD optimization to mitigate client drift and preserve convergence.	2024
Secure Multi-Party Computation (SMPC)	Li et al [89]. (Chain-PPFL)	Designed a single-masking, chain-based secure FL protocol with minimal overhead and high data security.	2021
Secure Multi-Party Computation (SMPC)	Chen et al [91].	Enhanced SMPC by integrating compressed sensing to reduce communication while maintaining privacy.	2024
Homomorphic Encryption	Zhang et al [92]. (BatchCrypt)	Employed leveled CKKS encryption and batching for efficient model aggregation in cross-silo FL.	2020
Homomorphic Encryption	Ma et al [94].	Developed a multi-key homomorphic encryption scheme allowing each client to encrypt using its own key, enabling secure aggregation without requiring key sharing.	2022
Homomorphic Encryption	Jin et al [95]. (FedML-HE)	Proposed Selective Parameter Encryption under HE to reduce overhead while maintaining privacy.	2024
Homomorphic Encryption	Korkmaz el al [97]. (FAS Framework)	Introduced selective HE with layer-level encryption and scrambling, achieving 90% speedup in FL tasks.	2025

2.5 Summary and Comparative Analysis

Selecting an appropriate privacy-preserving technique in federated learning requires a careful assessment of multiple factors, including privacy guarantees, computational efficiency, and integration complexity. Each method, differential privacy, secure multi-party computation, and homomorphic encryption, provides distinct advantages and limitations depending on the use case and system constraints. Their effectiveness also varies with respect to model accuracy, communication cost, and deployment feasibility. To support an informed comparison, the table below summarizes the key characteristics of these techniques in the context of federated learning.

Table 2.2: Comparative Analysis of Privacy-Preserving Techniques in Federated Learning

Criteria	**Differential Privacy**	**Secure Multi-Party Computation**	**Homomorphic Encryption**
Privacy Guarantee	Statistical privacy through noise injection	Exact computation without revealing data	Encryption-based computation without decryption
Accuracy Impact	Moderate due to added noise	None (exact results)	None to low (approximate schemes may introduce error)
Computational Overhead	Low to moderate	High	Moderate to High
Communication Cost	Low	Very high due to multiple rounds of interaction	High due to large ciphertext sizes
Suitability for FL	High (lightweight)	Limited (requires synchrony and trust setup)	High (secure and decentralized)
Implementation Complexity	Simple to moderate	Complex (protocol design and coordination)	Moderate to high (requires encryption library integration)

2.6 Conclusion

This chapter provided a comprehensive overview of the state of the art concerning privacy in Federated Learning. It began by dissecting the threat landscape, revealing that despite data localization, vulnerabilities persist through the leakage of information from exchanged model updates and the final trained model, enabling various inference attacks. Subsequently, the primary defense paradigms: Difference Privacy, Secure Multiparty Computation (primarily for secure aggregation), and Homomorphic Encryption were surveyed. Each technique presents a distinct approach to mitigating risks, accompanied by inherent trade-offs involving privacy guarantees, impact on model utility, and system overhead. The review of related works further contextualized the current research landscape, highlighting ongoing efforts, particularly in leveraging cryptographic methods such as homomorphic encryption. This exploration establishes the complex interplay between privacy threats and defenses in FL, underscoring the ongoing challenge of developing solutions that are simultaneously secure, accurate, and efficient.

Chapter 3

Federated Learning Framework Based on Homomorphic Encryption

3.1 Introduction

This chapter presents the proposed Federated Learning framework designed to enable secure collaborative training while protecting sensitive data exchanged between clients and the central server. Based on the analysis of privacy risks and defense mechanisms in Chapter 2, this design uses Homomorphic Encryption . The unique property of HE allowing computations such as model aggregation to be performed directly on encrypted updates without requiring decryption is central to this approach, aiming to ensure data confidentiality throughout the learning process while striving to maintain model utility. While other techniques like differential privacy and secure multi-party computation offer distinct privacy paradigms, HE was selected for its potential to provide strong cryptographic guarantees without inherently introducing noise that might degrade model accuracy or system complexity and incur additional computational costs. The subsequent sections will detail the architecture, data flow and the specific HE scheme used, including the exploration of both full and selective encryption strategies designed to manage computational overhead.

3.2 Overview of the Proposed Framework

The proposed framework is built on the standard federated learning paradigm, extended with homomorphic encryption to prevent privacy leakage through model updates. It allows a group of clients to collectively train a common model without sharing raw data or revealing their model updates in unencrypted form. The server orchestrates the learning process without accessing the sensitive content of client updates, even during aggregation, which preserves data confidentiality even in a semi-honest threat model.
The overall framework workflow is structured as follows:

1. The central server initializes the global model and distributes it to all clients.
2. Each client trains the model locally on its data.
3. After training, clients encrypt the model updates.
4. Encrypted updates are sent to the server.
5. The server performs aggregation directly on the encrypted data.
6. The aggregated result is returned to the clients for decryption and further training.

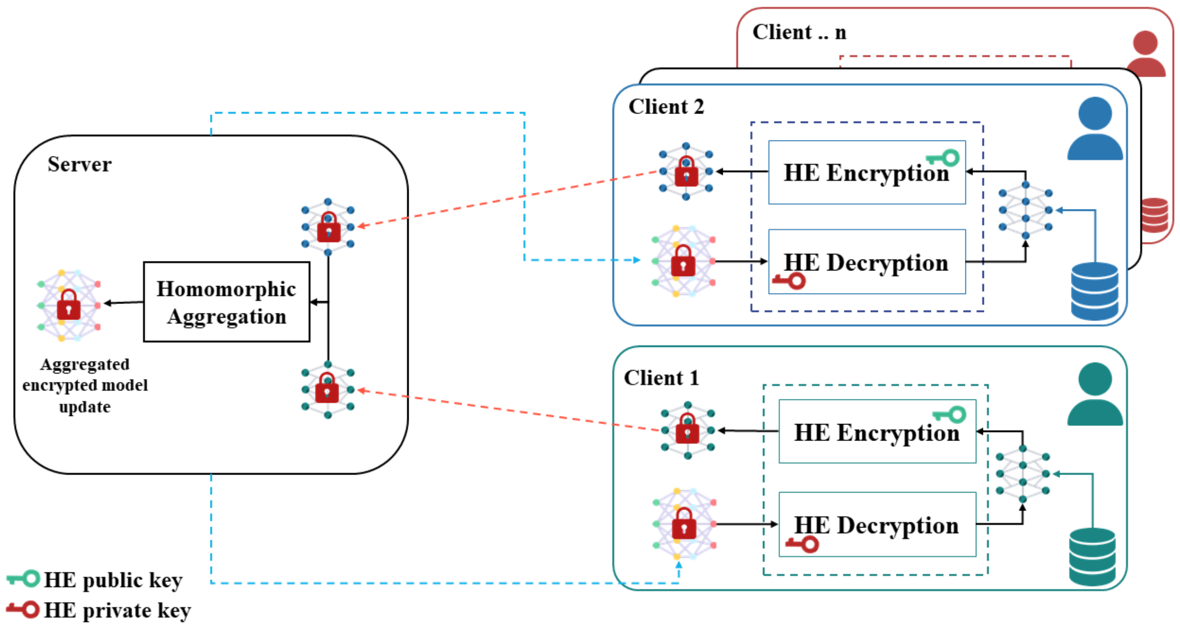


Figure 3.1: Architecture Of The Proposed Privacy-Preserving FL Framework Based On Homomorphic Encryption

3.3 Local Model Architecture

In a federated learning setting, each client trains a model locally on its private dataset and transmits only model updates to the central server. The design of the core model architecture is essential, as it directly influences the learning performance, convergence behavior, and communication efficiency. In this framework, a convolutional neural network (CNN) was adopted as the core model architecture, due to its effectiveness in capturing spatial features from image data and its strong performance in image classification tasks.

3.3.1 Convolutional Neural Network

Convolutional Neural Network is a class of artificial neural networks and a deep learning model, especially intended to process data having a grid-like architecture, such as images. CNNs are fundamentally driven by automatic and adaptive learning of spatial hierarchies of features from incoming data through a process that includes backpropagation. This is achieved using a structure composed of multiple distinct layers, primarily convolution layers, pooling layers, and fully connected layers. The initial layers, convolution and pooling, perform the crucial task of feature extraction, while subsequent fully connected layers map these extracted features to the final output, typically for tasks like classification.[98]

3.3.2 Convolutional neural network components

1. **The input layer**
   The input layer serves as the entry point of a CNN, tasked with receiving raw input data, typically in the form of an image. The input is represented as a multidimensional array: a 2D array for grayscale images, capturing height and width, or a

3D array for color images, incorporating an additional dimension for color channels (e.g., red, green, blue). This layer forwards the unaltered data to subsequent layers for processing.[98]

2. **Convolution Layer**
   A basic element of CNN architecture, the convolution layer extracts features. Usually comprising both linear and nonlinear operations—more especially, a convolution operation and an activation function—it [98]

   - **Convolution Operation :** This is a specific form of linear operation employed for feature extraction. The process entails utilizing a diminutive matrix of values, referred to as a kernel, over the input, which is likewise a matrix of values (a tensor). The element-wise product is computed between each kernel element and the input tensor at each point, and these products are aggregated to derive the output value at the corresponding place in the output tensor, referred to as a feature map. This procedure is reiterated with various kernels to generate several feature maps, each reflecting distinct attributes of the input. Various kernels may be regarded as distinct feature extractors [98].

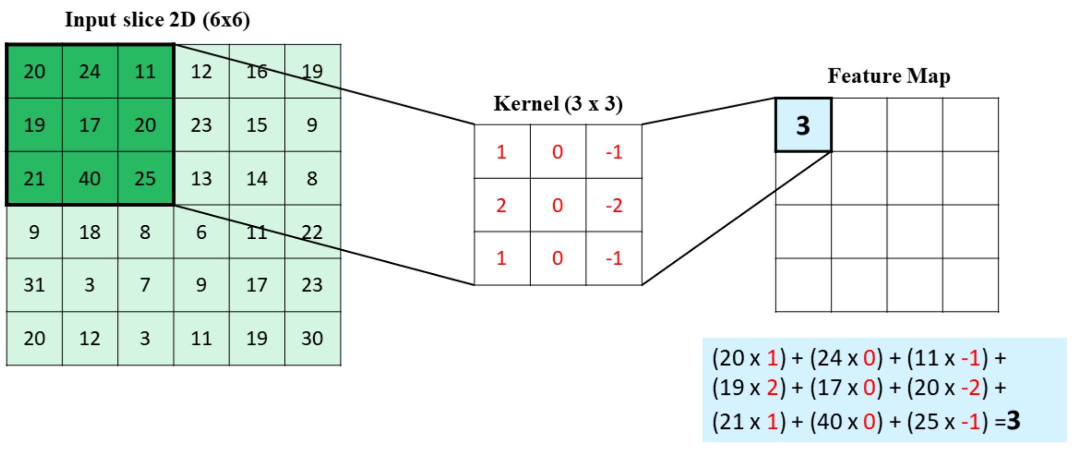


Figure 3.2: Process of convolution operation

The convolution operation encompasses several key components, including: **Kernels**: small arrays of trainable parameters utilized in convolution procedures.
Feature Maps: The output tensors produced by the application of kernels to the input.
Hyperparameters: The principal hyperparameters that characterize the convolution operation are the dimensions and number of kernels. Typical kernel sizes are 3×3, although they may also be 5×5 or 7×7. The number of kernels is arbitrary and dictates the depth of the resultant feature maps.
Padding: Techniques like zero padding are used to address the reduction in the height and width of the output feature map, allowing the center of a kernel to fit on the outermost element of the input and helping to retain in-plane dimensions.
Stride: The distance between two successive kernel positions, which also delineates the convolution operation. The standard stride is 1,however, greater

strides may be employed for downsampling purposes.
Weight Sharing: Kernels are shared across all image positions, contributing to the efficiency of CNNs for image processing. This property also helps learned local feature patterns to be translation invariant.
The training process for the convolution layer involves identifying the kernels that work best for a given task based on the training dataset. Kernels are the only parameters automatically learned in this layer; size, number of kernels, padding, and stride are hyperparameters set before training

- **Activation function** The activation layer introduces non-linearity into the network by applying an activation function to the convolutional layer's output. This non-linearity is pivotal for modeling complex data patterns. The predominant activation function in CNNs is the Rectified Linear Unit (ReLU) [98], defined as :

$$f(x) = \max(0, x) \tag{3.1}$$

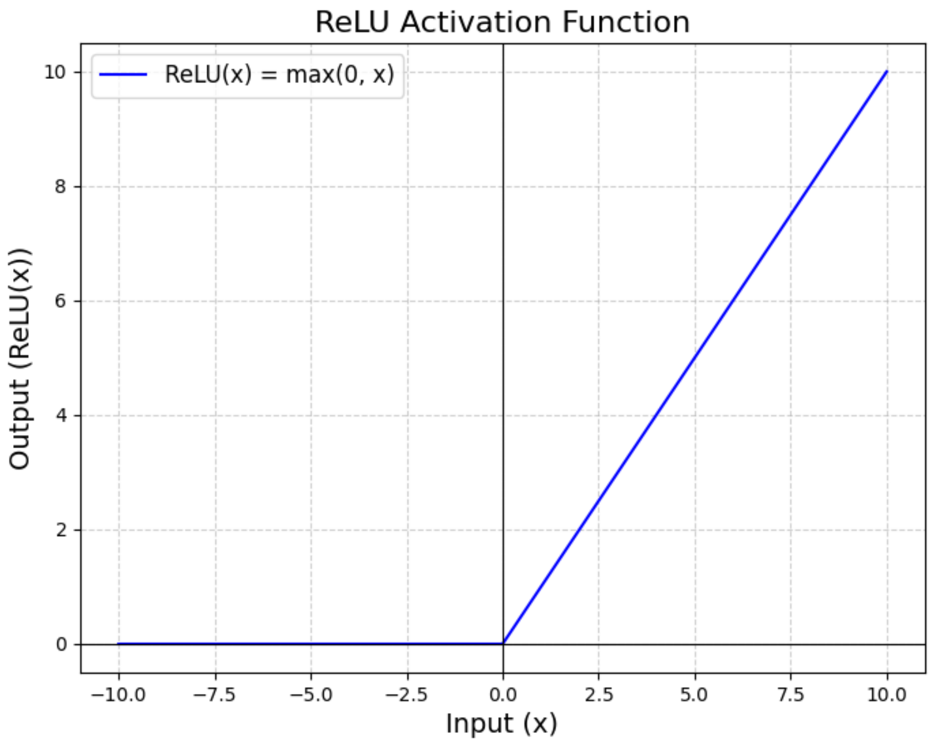


Figure 3.3: Activation Function ReLU

ReLU nullifies negative values while retaining positive ones (Figure 3.3), enhancing training efficiency and mitigating the vanishing gradient issue. This layer empowers the network to discern intricate features critical for tasks like medical image analysis.

3. **Poling layer**
A pooling layer typically provides a downsampling operation that reduces the in-plane dimensionality of the feature maps. This is done to introduce a translation invariance to small shifts and distortions and to decrease the number of subsequent

learnable parameters. There are no learnable parameters in pooling layers, but filter size, stride, and padding are hyperparameters. [98] Two types of pooling operations are mentioned:

- **Max Pooling:** The most popular form of pooling. It extracts patches from the input feature maps, outputs the maximum value in each patch, and discards the other values. A max pooling with a filter size of 2×2 with a stride of 2 is commonly used, which downsamples the in-plane dimension by a factor of 2 (as shown in Figure 3.4). The depth dimension of feature maps remains unchanged. [98]

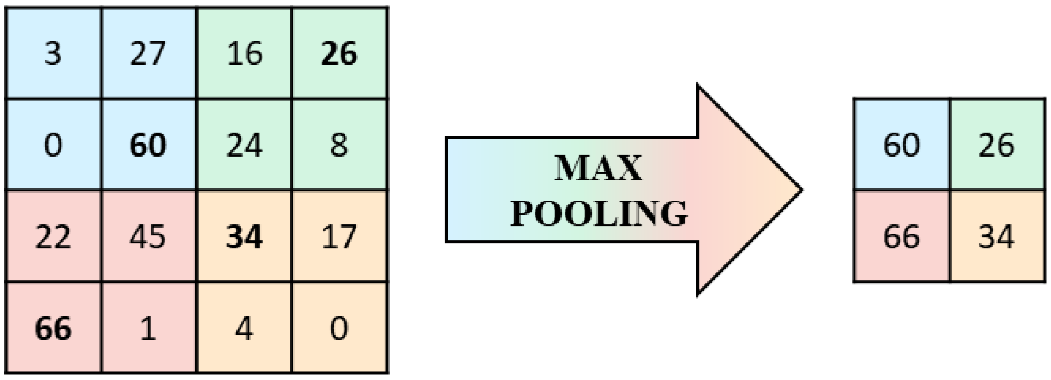


Figure 3.4: Process of max pooling layer

- **Global Average Pooling:** An extreme type of downsampling where a feature map is downsampled into a 1×1 array by taking the average of all elements in each feature map. This operation is typically applied once before the fully connected layers. Its advantages include reducing the number of learnable parameters and enabling the CNN to accept inputs of variable size.[98]

4. **Flattening Layer**
The flattening layer reshapes multi-dimensional feature maps into a one-dimensional vector (as illustrated in Figure 3.5), bridging the feature extraction and classification phases of the CNN. Following convolutional and pooling operations, this layer transforms, for example, a 4×4×32 feature map into a 512-element vector. This reformatted data is then suitable for processing by fully connected layers.[98]

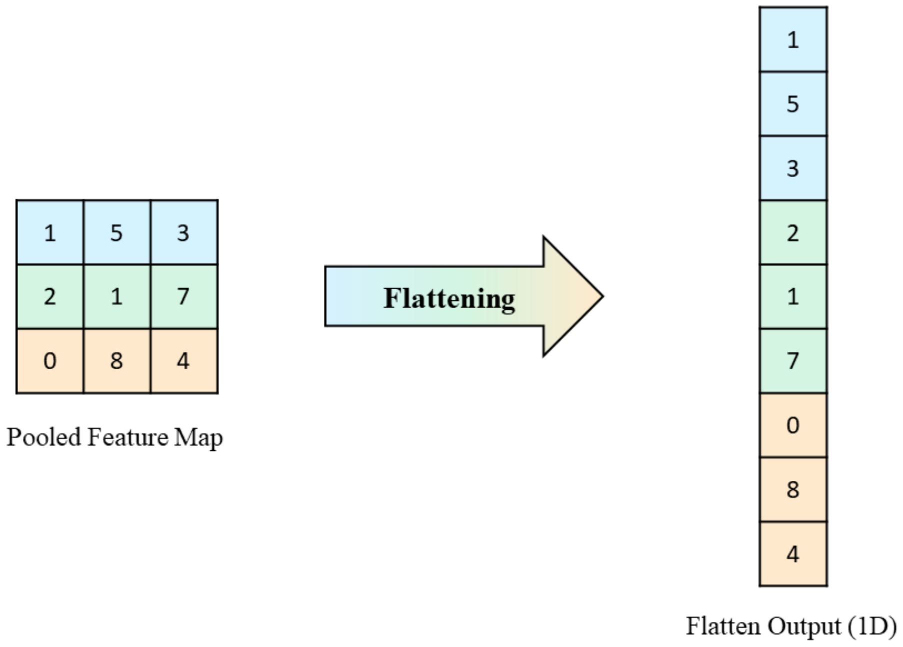


Figure 3.5: Process of Flattening

5. **Fully connected layers**
   Fully connected layers, or dense layers, leverage the extracted features to perform high-level reasoning and prediction. Each neuron in these layers connects to every neuron in the preceding layer, with weights learned during training. These layers synthesize the spatial features into a cohesive representation, typically culminating in a layer with neurons equal to the number of output classes (e.g., two for binary classification). An activation function is applied to refine the output for the subsequent layer.[98]

6. **Output Layer**
   This is typically the final fully connected layer. For classification tasks, it usually has the same number of output nodes as the number of classes. The activation function applied to this layer is usually different from the others and is selected according to the specific task [98] :

   - **Sigmoid:** Used in binary classification, yielding a probability between 0 and 1.
     $$alpha(x) = \frac{1}{1+e^{-x}} \quad \forall x \in R \tag{3.2}$$
   - **Softmax:** Applied in multi-class classification, producing a probability distribution over classes.It can be expressed as follows:
     $$G(e^j) = \frac{e^{ej}}{\sum_i e^{ei}} \tag{3.3}$$

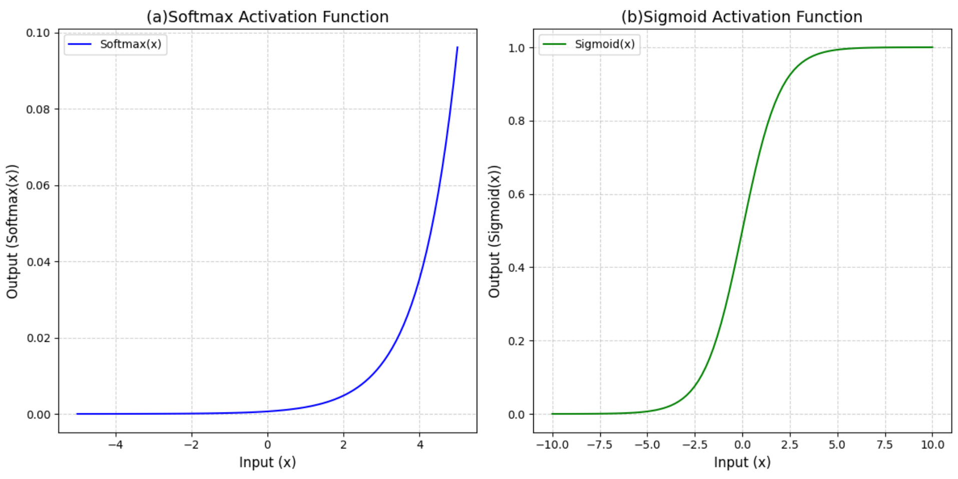


Figure 3.6: Activation function.(a) Softmax ,(b) sigmoid

3.4 Homomorphic scheme used in our framework

Among the various homomorphic encryption schemes available, selecting an appropriate scheme is essential for ensuring compatibility with deep learning workflows while maintaining an acceptable balance between privacy, computational efficiency, and accuracy. While schemes such as Brakerski/Fan-Vercauteren (BFV) and Brakerski-Gentry-Vaikuntanathan (BGV) are designed for exact arithmetic over integers, they are less suitable for deep learning models, which primarily operate on real-valued parameters and gradients represented as floating-point numbers. The Cheon-Kim-Kim-Song (CKKS) scheme, in contrast, supports approximate arithmetic over real numbers, making it a more natural fit for neural network training. Its ability to perform encrypted computations directly on floating-point representations allows for more efficient and seamless integration with federated learning tasks, such as encrypted weight aggregation and gradient updates. Moreover, CKKS offers a flexible trade-off between precision and performance, enabling encrypted training without significant loss in model quality or computational scalability. For these reasons, the CKKS scheme was selected as the encryption backbone of the proposed privacy-preserving federated learning framework.

3.4.1 The Cheon-Kim-Kim-Song (CKKS)

Introduced in 2017 by Cheon, Kim, Kim, and Song [99],the CKKS scheme is a leveled homomorphic encryption system based on the Ring Learning With Errors (RLWE) problem. It is uniquely designed to support efficient approximate arithmetic on encrypted real and complex numbers. Unlike traditional homomorphic encryption schemes that focus on exact operations over integers or finite fields, CKKS enables homomorphic computations directly on encrypted vectors of real and complex values by encoding them into polynomial plaintexts and carefully managing the associated computational noise.While initially presented as a leveled scheme efficient for computations of limited complexity, subsequent developments in compatible bootstrapping techniques have enabled its extension to function as a fully homomorphic encryption scheme capable of handling computations of arbitrary complexity.[100]

3.4.1.1 Mathematical Foundations

- Let N be the degree of the cyclotomic polynomial $\Phi_M(X) = X^N + 1$, where N is a power of 2 and $M = 2N$.
- The plaintext ring is $R = \mathbb{Z}[X]/(X^N + 1)$, a ring of polynomials.
- The ciphertext ring is $R_q = \mathbb{Z}_q^2[X]/(X^N + 1)$, where q is a large prime modulus.
- The message space is a subspace $H \subset \mathbb{C}^N$, which is isomorphic to $\mathbb{C}^{N/2}$, allowing efficient encoding of complex vectors.
- Two important maps:
  σ: canonical embedding from $R \to H$
  π: projection from $H \to \mathbb{C}^{N/2}$

3.4.1.2 Workflow the CKKS Scheme

The workflow of the CKKS encryption scheme can be described as follows[99]:

1. **Key Generation**
   The Key Generation step in the CKKS scheme produces the keys needed for encryption, decryption, and homomorphic operations.
   - Secret key: A random polynomial $s(X)$ is chosen from R_q.
   - Public key: The public key is generated by selecting a random polynomial $a(X)$ from R_q and computing $pk = (a(X) \cdot s(X) + e(X), a(X)) = (pk_1, pk_2)$, where $e(X)$ is a small error polynomial.
   - Relinearization key: The relinearization key, denoted $r(X)$, is used to reduce the size of ciphertexts after multiplication, allowing them to be efficiently decrypted after homomorphic operations.
2. **Encoding**: The CKKS scheme operates over a polynomial ring $R = \mathbb{Z}[X]/(X^N+1)$, and thus cannot directly handle input vectors $\mathbf{z} \in \mathbb{C}^{N/2}$. To bridge this gap, a specialized encoding step is used to map these vectors into plaintext polynomials $m(X) \in R$ that are compatible with the scheme's arithmetic.[99]
   - First , expend the vector $\mathbf{z}$ into $\mathbf{H}$ via the inverse of π :
   $$\pi^{-1}(\mathbf{z}) \in H \tag{3.4}$$
   - Second, scale it by multiplying with a factor Δ to preserve precision:
   $$\Delta \cdot \pi^{-1}(\mathbf{z}) \tag{3.5}$$
   - Finally, round the result and apply the inverse of the canonical embedding:
   $$m(X) = \sigma^{-1}\left(\lfloor \Delta \cdot \pi^{-1}(\mathbf{z}) \rceil\right) \in R \tag{3.6}$$
3. **Encryption :**Now that we have the encoded message polynomial $m(X) \in R$, encryption proceeds by generating the ciphertext using the public key. We construct the ciphertext as a pair of polynomials [99]:
   $$c(X) = (c_0(X),\ c_1(X)) = (m(X) - a(X) \cdot s(X) + e(X),\ a(X)) \tag{3.7}$$

- $c_0(X)$ includes the message, noise, and interaction with the secret key.
- $c_1(X)$ is simply the random polynomial a(X), already part of the public key.
- $e(X) \in R_q$ is a small error polynomial, typically sampled from a discrete Gaussian or uniform distribution with small coefficients (to control decryption noise). $e(X)$ term ensures the security of the encryption by hiding the message within a noise layer, making it hard to recover without the secret key.

As a result, the ciphertext:

$$c(X) = (c_0(X),\ c_1(X)) \in R_q^2 \tag{3.8}$$

now it can be used in homomorphic operations like addition and multiplication.

4. **Homomorphic operation:**
   we have two ciphertexts:

   $$c(X) = (c_0(X),\ c_1(X)), \quad c'(X) = (c'_0(X),\ c'_1(X))$$

   - **Homomorphic addition** :
     We can add them component-wise:

     $$\text{add}(c,\ c') = (c_0(X) + c'_0(X),\ c_1(X) + c'_1(X)) \tag{3.9}$$

     Addition in CKKS is cheap and does not increase noise significantly.
   - **Homomorphic Multiplication** :
     We can multiply the ciphertext to become a 3-component ciphertext:

     $$c_{\text{mul}}(X) = (d_0(X),\ d_1(X),\ d_2(X)) \tag{3.10}$$

     where:
     - $d_0(X) = c_0(X) \cdot c'_0(X)$
     - $d_1(X) = c_0(X) \cdot c'_1(X) + c_1(X) \cdot c'_0(X)$
     - $d_2(X) = c_1(X) \cdot c'_1(X)$

     CKKS Multiplication operations increase noise and scale.

5. **Relinearization**:
   It turns out that when you multiply two ciphertexts, the size of the ciphertext increases from 2 components to 3 (show Eq 3.10).
   This form cannot be decrypted directly because decryption only works with ciphertexts of the form (3.8).[99]
   To restore the ciphertext to a decryptable state, we employ relinearization, utilizing the relinearization key $r(X)$. The process roughly computes:

   $$(c_0, c_1) + \left\lfloor \frac{c_2 \cdot r(X)}{b} \right\rceil \mod q \tag{3.11}$$

   where:

   - c_2 is the extra term from multiplication

- b is a large base (used in key switching)
- r(X) is the relinearization key
- q is the ciphertext modulus

The result is a new ciphertext of size 2 that still encodes the same approximate message, just in a decryptable form.

6. **Rescaling**:
In CKKS, rescaling is needed after a multiplication to control the size of values and keep the precision correct.[99]
When we multiply two ciphertexts, the scale (the factor used to preserve decimal precision during encoding) grows quadratically. If both operands are scaled by Δ, the new scale becomes approximately: $\Delta_{\text{new}} = \Delta^2$
This large scale can cause problems in future computations. So, after multiplication, we perform rescaling to bring the scale back down to a manageable level:

$$\text{Rescale}(c) = \left\lfloor \frac{c}{\Delta} \right\rceil \mod q_{\ell-1} \tag{3.12}$$

- c is a ciphertext
- Δ is the scale (a power of 2)
- $q_{\ell-1}$ is the next smaller modulus level in the chain

This does two things:

(a) Reduces the scale (divides by Δ) to bring it back under control.
(b) Drops to a smaller modulus, moving to a lower encryption level (level $\ell - 1$ if we were at level ℓ)

7. **Decryption :**
To retrieve the message polynomial from the ciphertext $c(X) \in \mathbb{R}_q$, we perform RLWE-based decryption using the secret key polynomial $s(X)$. This yields:

$$m'(X) \approx c_0(X) + c_1(X) \cdot s(X) = m(X) + e(X) \mod q \tag{3.13}$$

where $e(X)$ represents the small error introduced during encryption and computation.

8. **Decoding :**
Once we obtain the decrypted message polynomial $m(X) \in R$, we need to convert it back into a complex vector in $\mathbb{C}^{N/2}$. The decoding process involves the inverse of the operations used during encoding:

- First, we apply the canonical embedding σ to map the polynomial $m(X)$ into the complex subspace H.

$$\sigma(m(X)) \approx \pi^{-1}(z) \cdot \Delta \tag{3.14}$$

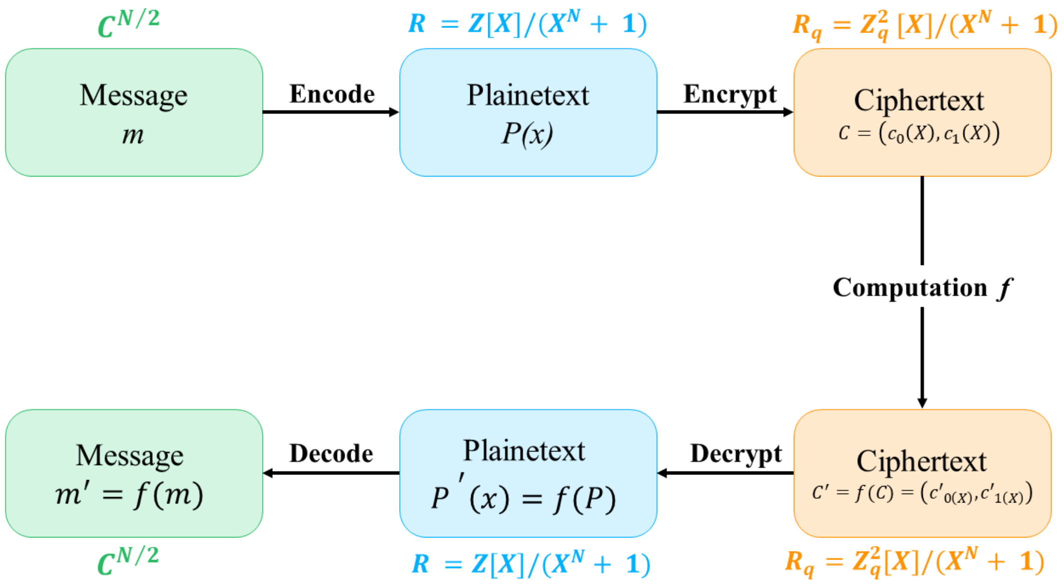
- Second, since the message was scaled during encoding by a factor Δ, we now scale it back by dividing:

$$\Delta^{-1} \cdot \pi^{-1}(z) \cdot \Delta \tag{3.15}$$

- Finally, we apply the projection map π to retrieve the message vector:

$$z \approx \pi(\pi^{-1}(z)) \in \mathbb{C}^{N/2} \tag{3.16}$$

This gives us the approximate original vector of real or complex numbers.



$C^{N/2}$
Message m
Encode
$R = Z[X]/(X^N + 1)$
Plainetext $P(x)$
Encrypt
$R_q = Z_q^2[X]/(X^N + 1)$
Ciphertext $C = (c_0(X), c_1(X))$
Computation f
Ciphertext $C' = f(C) = (c'_{0(X)}, c'_{1(X)})$
Decrypt
Plainetext $P'(x) = f(P)$
Decode
Message $m' = f(m)$
$C^{N/2}$
$R = Z[X]/(X^N + 1)$
$R_q = Z_q^2[X]/(X^N + 1)$

Figure 3.7: Workflow of the CKKS Scheme

3.5 Homomorphic Encryption Strategies used in a Framework

To protect sensitive model updates in the federated learning process, this framework supports two homomorphic encryption strategies: full encryption and selective encryption. The first strategy encrypts all model parameters to maximize privacy, while the second focuses on encrypting only the most sensitive layers to reduce computational and communication overhead.

3.5.1 Full Encryption Strategy

In the full encryption strategy, each client encrypts all model parameters using the CKKS homomorphic encryption scheme before sending them to the server. This method ensures that no part of the model update is transmitted in plaintext, providing complete protection against inference or reconstruction attacks at the communication level.

The core procedure of our privacy-preserving framework is summarized in the following algorithm.

Algorithm 3 Federated Learning with Full Homomorphic Encryption

```
    Server executes:
 1: Initialize global model $W_0$
 2: total rounds $T$
 3: for $t \leftarrow 1$ to $T$ do
 4:     $m \leftarrow \max(\lfloor \rho \cdot K \rfloor, 1)$                ▷ Select number of clients
 5:     $\mathcal{S}_t \leftarrow$ random subset of $m$ clients from $K$
 6:     Initialize list $\mathcal{U}_t \leftarrow \emptyset$                        ▷ Encrypted updates
 7:     for all $c \in \mathcal{S}_t$ in parallel do
 8:         $C_{t+1}^{(c)}, n_c \leftarrow$ ClientUpdate$(c, W_t)$
 9:         Append $(C_{t+1}^{(c)}, n_c)$ to $\mathcal{U}_t$
10:     end for
11:     $C_{t+1}^{\text{agg}} \leftarrow$ EncryptedFedAvg$(\mathcal{U}_t, Pk)$                ▷ Algorithm 6
12:     $W_{t+1} \leftarrow C_{t+1}^{\text{agg}}$                   ▷ Server stores aggregated ciphertext
13:     deliver $W_{t+1}$ to all clients
14: end for

    ClientUpdate$(c, W_t)$:                                 // Executed on client $c$
15: $Sk, Pk \leftarrow$ client's secret and public keys
16: $W_t^{\text{dec}} \leftarrow$ DecryptModel$(W_t, Sk, \mathcal{S})$                     ▷ Algorithm 4
17: Load $W_t^{\text{dec}}$ into client model
18: for $i \leftarrow 1$ to $S$ do                                         ▷ Local epochs
19:     for each batch $b$ in local dataset $\mathcal{D}_c$ do
20:         $W_b \leftarrow W_b - \eta \cdot \nabla \mathcal{L}(W_b; b)$                     ▷ Gradient descent
21:     end for
22: end for
23: $W_{t+1}^{(c)} \leftarrow$ locally updated model
24: $C_{t+1}^{(c)} \leftarrow$ EncryptModel$(W_{t+1}^{(c)}, Pk)$                        ▷ Algorithm 5
25: $n_c \leftarrow |\mathcal{D}_c|$                                          ▷ Local dataset size
26: return $(C_{t+1}^{(c)}, n_c)$
```

The process begins with the server initializing a global model and selecting a subset of clients to participate in each training round. Once selected, the global model is sent to the clients in encrypted form. Each client begins by decrypting the received model using its private key and reshaping the values into their original tensor forms (as shown in Algorithm 4).

Algorithm 4 Decrypt_Model

```
Require: Encrypted parameters $\mathcal{C} = \{c_1, \dots, c_n\}$, secret key $Sk$, shapes $\mathcal{S} = \{s_1, \dots, s_n\}$
Ensure: Decrypted model parameters $\mathcal{P} = \{p_1, \dots, p_n\}$
 1: $\mathcal{P} \leftarrow \emptyset$                          ▷ Initialize list for plaintext parameters
 2: for $i \leftarrow 1$ to $n$ do
 3:     $\tilde{p}_i \leftarrow$ CKKS_Decrypt$(Sk, c_i)$                        ▷ Decrypt ciphertext
 4:     $p_i \leftarrow$ Reshape$(\tilde{p}_i, s_i)$                   ▷ Restore original tensor shape
 5:     Append $p_i$ to $\mathcal{P}$
 6: end for
 7: return $\mathcal{P}$                                      ▷ Return plaintext model
```

Local training is then performed using the decrypted model over multiple epochs and batches of the client's private data. Once training concludes, the updated model parameters are encrypted using the client's public key (as detailed in Algorithm 5),ensuring that no raw data or plaintext model information is exposed. These encrypted updates, along with the size of the client's local dataset, are returned to the server.

Algorithm 5 Encrypt_Model

```
Require: Model parameters $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, public key $Pk$
Ensure: Encrypted parameters $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$
 1: $\mathcal{C} \leftarrow \emptyset$                          ▷ Initialize empty list for ciphertexts
 2: for $i \leftarrow 1$ to $n$ do
 3:     $p_i^{\text{flat}} \leftarrow \text{Flatten}(p_i)$      ▷ Flatten parameter tensor
 4:     $c_i \leftarrow \text{CKKS\_Encrypt}(Pk, p_i^{\text{flat}})$      ▷ Encrypt using CKKS
 5:     Append $c_i$ to $\mathcal{C}$
 6: end for
 7: return $\mathcal{C}$                                ▷ Return encrypted model
```

The server then performs a homomorphic aggregation of the encrypted updates from all clients using a secure weighted averaging strategy "EncyptedFedAvg" (outlined in Algorithm 6). The resulting encrypted global model is then broadcast to clients for the next round. This iterative process continues until a convergence criterion is satisfied. This approach guarantees end-to-end encryption during communication and aggregation, minimizing the risk of data leakage while maintaining the collaborative nature of federated learning.

Algorithm 6 Encrypted_FedAvg

```
Require: Encrypted models $\{(\mathcal{C}^{(j)}, n_j)\}_{j=1}^{m}$, public key $Pk$
Ensure: Aggregated encrypted model $\mathcal{C}^{\text{agg}}$
 1: $N \leftarrow \sum_{j=1}^{m} n_j$                      ▷ Total number of samples
 2: $\mathcal{C}^{\text{agg}} \leftarrow \emptyset$                          ▷ Initialize result list
 3: for $i \leftarrow 1$ to number of parameters do
 4:     $c_i^{\text{agg}} \leftarrow 0$          ▷ Initialize aggregated ciphertext for parameter $i$
 5:     for $j \leftarrow 1$ to $m$ do
 6:         $c_i^{(j)} \leftarrow \mathcal{C}^{(j)}[i]$             ▷ Get $i$th param from $j$th client
 7:         $w \leftarrow \frac{n_j}{N}$                   ▷ Weight based on client data size
 8:         $c_i^{\text{agg}} \leftarrow c_i^{\text{agg}} + (c_i^{(j)} \cdot w)$      ▷ Weighted sum in ciphertext space
 9:     end for
10:     Append $c_i^{\text{agg}}$ to $\mathcal{C}^{\text{agg}}$
11: end for
12: return $\mathcal{C}^{\text{agg}}$
```

3.5.2 Selective Encryption Strategy

Full homomorphic encryption, as discussed in the previous section, offers a high level of privacy by encrypting the entire set of model parameters before transmission. This ensures that no part of the client's model update is exposed in plaintext during communication or aggregation. While this approach provides robust security against inference and reconstruction attacks, it introduces significant computational and communication overhead. Specifically, encrypting the entire model leads to large ciphertext sizes, and homomorphic operations on these encrypted values are inherently slower and more resource-intensive than operations on plaintext.

To alleviate these limitations, we introduce a Selective Encryption Strategy that provides a privacy-efficiency trade-off by encrypting only the sensitive portions of the model. The core idea is to analyze the model's structure and selectively encrypt parameters that are more likely to reveal sensitive information, while allowing the remaining parameters to be shared in plaintext. This strategy appreciably reduces computational cost, memory usage, and communication time, withoutcompromising data privacy. the effectiveness of this strategy depends heavily on accurately determining which parts of the model are truly sensitive and for that, we rely on gradient-based sensitivity analysis.

3.5.2.1 How to Determine the Sensitivity

The sensitivity of a model layer reflects the extent to which its parameters contribute to the learning process and how much private information they may encode. To measure this, we rely on gradient magnitude, as gradients are direct indicators of how strongly each parameter influences the model's loss during training.

In deep learning, model training relies on minimizing a loss function that quantifies the discrepancy between predicted outputs and actual labels. This optimization is performed using gradient-based algorithms such as Stochastic Gradient Descent (SGD), where gradients represent how much each parameter contributes to the loss.

Formally, consider a model with parameters θ, input x ,and label y the training objective is to minimize the loss function $\mathcal{L}(\theta; x, y)$ .During backpropagation, the gradient of the loss with respect to each parameter in the model is computed. For a given layer l its parameters θ_l produce a vector of partial derivatives:

$$\nabla_{\theta_l}\mathcal{L}(x,y) = \left[\frac{\partial \mathcal{L}}{\partial \theta_l^{(1)}}, \frac{\partial \mathcal{L}}{\partial \theta_l^{(2)}}, \ldots, \frac{\partial \mathcal{L}}{\partial \theta_l^{(n_l)}}\right] \tag{3.17}$$

Each component in this vector indicates the rate of change of the loss with respect to a specific parameter. A larger magnitude implies that modifying the parameter would significantly impact the model's output, potentially revealing more about the training data. Hence, layers with larger gradients are considered more informative and sensitive.

To quantify this sensitivity, we compute the L2 norm (Euclidean norm) of the gradient vector for layer l, which gives a single scalar representing its overall influence:

$$\|\nabla_{\theta_l}\mathcal{L}(x,y)\|_2 = \sqrt{\sum_{i=1}^{n_l}\left(\frac{\partial \mathcal{L}}{\partial \theta_l^{(i)}}\right)^2} \tag{3.18}$$

This computation is typically performed not on single samples but on mini-batches $B = \{(x_i, y_i)\}_{i=1}^{m}$ of training data. Thus, the average gradient vector for a layer over a mini-batch is:

$$\bar{\nabla}_{\theta_l}^{(B)} = \frac{1}{m}\sum_{i=1}^{m}\nabla_{\theta_l}\mathcal{L}(x_i, y_i) \tag{3.19}$$

And the corresponding batch-level L2 norm becomes:

$$G_l^{(B)} = \left\| \bar{\nabla}_{\theta_l}^{(B)} \right\|_2 = \left\| \frac{1}{m} \sum_{i=1}^{m} \nabla_{\theta_l} \mathcal{L}(x_i, y_i) \right\|_2 \tag{3.20}$$

To ensure robustness against noise in gradient estimation, we repeat this over T mini-batches and compute the average gradient norm:

$$\bar{G}_l = \frac{1}{T} \sum_{t=1}^{T} G_l^{(B_t)} \tag{3.21}$$

where:

- T: total number of sampled batches,
- $G_l^{(B_t)}$: L2 norm computed on batch t.

After calculating $\bar{G}_l$ for all layers, the next step is to identify which layers are deemed sensitive and should be encrypted . This decision is guided by two criteria:

1. **Mandatory Sensitivity:**
   Certain layers are manually flagged as sensitive based on prior knowledge of their role in the model architecture. For example, layers involved in final classification are always marked for encryption because they are closely tied to label-specific representations and are more prone to leaking label-related information.

2. **Conditional Sensitivity :** For the remaining layers, we rely on the computed sensitivity scores $\bar{G}_l$ to guide selection. These scores are sorted in descending order to rank layers by their gradient magnitude. We then select the top $\gamma\%$ of layers as conditionally sensitive:

   $$\mathcal{S}_{\text{sensitive}} = \left\{ l \mid \bar{G}_l \text{ is among the top } \gamma\% \text{ of all layers} \right\} \tag{3.22}$$

   the union of both mandatory and conditionally selected layers constitutes the final set of layers marked for encryption. This hybrid approach ensures that the most informative and privacy-critical parts of the model are protected, while the remaining layers remain unencrypted to optimize communication and computation.
   By tailoring the selection to model behavior and data characteristics, this strategy achieves a fine balance between privacy assurance and system efficiency.

Algorithm 7 explains how to calculate sensitivity and determine sensitive layers based on the gradient criterion.

Algorithm 7 Determine_Sensitivity

```
Require:
    Model $M$ with parameters $\theta = \{\theta_1, \dots, \theta_L\}$
    Dataset $\mathcal{D}$, batch size $m$, number of batches $T$
    Mandatory prefixes list $\mathcal{P}$, top fraction $\gamma$
Ensure: Annotated list $\mathcal{I} = \{(name_l, shape_l, sensitive_l)\}_{l=1}^{L}$
 1: $\mathcal{I} \leftarrow \emptyset$                                    ▷ Final parameter info list
 2: $\mathcal{N} \leftarrow \emptyset$                       ▷ Store conditional norms: (index, norm)
 3: for $l \leftarrow 1$ to $L$ do
 4:     $name_l \leftarrow$ NameOf($\theta_l$)
 5:     $shape_l \leftarrow$ ShapeOf($\theta_l$)
 6:     $is\_mandatory \leftarrow$ False
 7:     for all $prefix \in \mathcal{P}$ do
 8:         if $name_l$ starts with $prefix$ then
 9:             Append $(name_l, shape_l, \textbf{True})$ to $\mathcal{I}$
10:             $is\_mandatory \leftarrow$ True
11:             break
12:         end if
13:     end for
14:     if $\neg is\_mandatory$ then
15:         $total\_norm \leftarrow 0$
16:         for $t \leftarrow 1$ to $T$ do
17:             $B_t \leftarrow$ sample mini-batch from $\mathcal{D}$
18:             Compute loss $\mathcal{L}_t \leftarrow \mathcal{L}(M(B_t))$
19:             Backpropagate to get gradient $\nabla_{\theta_l}\mathcal{L}_t$
20:             $g_t \leftarrow \|\nabla_{\theta_l}\mathcal{L}_t\|_2$
21:             $total\_norm \leftarrow total\_norm + g_t$
22:         end for
23:         $\bar{G}_l \leftarrow total\_norm/T$
24:         Append $(l, \bar{G}_l)$ to $\mathcal{N}$
25:     end if
26: end for
27: Sort $\mathcal{N}$ by $\bar{G}_l$ in descending order
28: $k \leftarrow \lfloor \gamma \cdot |\mathcal{N}| \rfloor$
29: $\mathcal{T} \leftarrow$ indices of top-$k$ entries from $\mathcal{N}$
30: for $l \leftarrow 1$ to $L$ do
31:     Check if $(name_l, shape_l, \_)$ already in $\mathcal{I}$
32:     if not already added (i.e., non-mandatory) then
33:         if $l \in \mathcal{T}$ then
34:             Append $(name_l, shape_l, \textbf{True})$ to $\mathcal{I}$
35:         else
36:             Append $(name_l, shape_l, \textbf{False})$ to $\mathcal{I}$
37:         end if
38:     end if
39: end for
40: return $\mathcal{I}$
```

Following the sensitivity analysis, the overall federated learning process remains largely consistent with the full homomorphic encryption approach. However, the key distinction lies in the encryption strategy, which becomes selective rather than comprehensive. On the client side, only the layers marked as sensitive are encrypted using the CKKS scheme, while non-sensitive parameters remain in plaintext, as illustrated in Algorithm 8.This adjustment reduces the size of transmitted ciphertexts and the computational cost associated with encryption operations.

Algorithm 8 Selective_Encrypt_Model

```
Require:
    Plaintext parameters $\mathcal{P} = \{p_1, \ldots, p_n\}$
    Public key $Pk$
    Parameter info list $\mathcal{I} = \{(name_i, shape_i, sensitive_i)\}$
Ensure:
    Selectively encrypted parameters and plaintext parameters $\mathcal{C} = \{c_1, \ldots, c_n\}$
 1: $\mathcal{C} \leftarrow \emptyset$                                         ▷ Initialize result list
 2: for $i \leftarrow 1$ to $n$ do
 3:     $(\_, \_, s_i) \leftarrow \mathcal{I}[i]$                              ▷ Get sensitivity flag
 4:     if $s_i =$ True then
 5:         $p_i^{\text{flat}} \leftarrow \text{Flatten}(p_i)$
 6:         $c_i \leftarrow \text{CKKS\_Encrypt}(p_i^{\text{flat}}, Pk)$      ▷ Encrypt sensitive layer
 7:     else
 8:         $c_i \leftarrow p_i$                              ▷ Keep non-sensitive layer in plaintext
 9:     end if
10:     Append $c_i$ to $\mathcal{C}$
11: end for
12: return $\mathcal{C}$
```

On the server side, aggregation is performed using a hybrid method that handles encrypted and plaintext parameters separately (Algorithm 10), and upon receiving the updated model, clients decrypt only the encrypted subset (Algorithm 9). This selective approach offers a practical trade-off between efficiency and privacy by minimizing resource consumption without compromising the protection of sensitive information.

Algorithm 9 Selective_Decrypt_Model

```
Require:
    Input $\mathcal{C} = \{c_1, \ldots, c_n\}$ (encrypted or plaintext)
    Secret key $Sk$, info list $\mathcal{I} = \{(name_i, shape_i, sensitive_i)\}$
Ensure:
    Decrypted parameters $\mathcal{P} = \{p_1, \ldots, p_n\}$
 1: $\mathcal{P} \leftarrow \emptyset$
 2: for $i \leftarrow 1$ to $n$ do
 3:     $(\_, shape_i, s_i) \leftarrow \mathcal{I}[i]$
 4:     if $s_i =$ True then
 5:         $\tilde{p}_i \leftarrow$ CKKS_Decrypt$(c_i, Sk)$
 6:         $p_i \leftarrow$ Reshape$(\tilde{p}_i, shape_i)$                ▷ Restore shape
 7:     else
 8:         $p_i \leftarrow c_i$                                        ▷ Already in plaintext
 9:     end if
10:     Append $p_i$ to $\mathcal{P}$
11: end for
12: return $\mathcal{P}$
```

Algorithm 10 Selective_Encrypted_FedAvg

```
Require:
    Client submissions: $\{(\mathcal{C}^{(j)}, n_j)\}_{j=1}^{m}$
    Parameter info list $\mathcal{I} = \{(name_i, shape_i, sensitive_i)\}$
Ensure:
    Aggregated model $\mathcal{C}^{\text{agg}}$
 1: $N \leftarrow \sum_{j=1}^{m} n_j$                                  ▷ Total data points
 2: $\mathcal{C}^{\text{agg}} \leftarrow \emptyset$
 3: for $i \leftarrow 1$ to number of parameters do
 4:     $(\_, \_, s_i) \leftarrow \mathcal{I}[i]$                              ▷ Sensitivity flag
 5:     if $s_i =$ True then                                ▷ Encrypted aggregation
 6:         $c_i^{\text{agg}} \leftarrow 0$
 7:         for $j \leftarrow 1$ to $m$ do
 8:             $w_j \leftarrow \frac{n_j}{N}$
 9:             $val \leftarrow \mathcal{C}^{(j)}[i]$                          ▷ Encrypted value
10:             $c_i^{\text{agg}} \leftarrow c_i^{\text{agg}} + (val \cdot w_j)$
11:         end for
12:     else                                                ▷ Plaintext aggregation
13:         $c_i^{\text{agg}} \leftarrow 0$
14:         for $j \leftarrow 1$ to $m$ do
15:             $w_j \leftarrow \frac{n_j}{N}$
16:             $val \leftarrow \mathcal{C}^{(j)}[i]$                          ▷ Plaintext value
17:             $c_i^{\text{agg}} \leftarrow c_i^{\text{agg}} + (val \cdot w_j)$
18:         end for
19:     end if
20:     Append $c_i^{\text{agg}}$ to $\mathcal{C}^{\text{agg}}$
21: end for
22: return $\mathcal{C}^{\text{agg}}$
```

3.6 Conclusion

Responding to the privacy challenges in Federated Learning , this chapter has laid out the architectural blueprint for a privacy-enhancing Federated Learning system, carefully designed to address the critical need for secure collaboration in sensitive domains like healthcare. By strategically integrating the CKKS homomorphic encryption scheme, we have detailed methodologies for both comprehensive end-to-end encryption and a novel selective encryption strategy. This latter approach, guided by gradient norm sensitivity analysis, particularly seeks a pragmatic balance, aiming to provide robust cryptographic protection for the most vital model components while significantly mitigating the computational overhead typically associated with full encryption, thereby paving the way for more practical and efficient privacy-preserving FL.

Chapter 4

Experiments, Results, and Discussion

4.1 Introduction

This chapter presents the empirical evaluation of the proposed privacy-preserving federated learning framework. The primary objective is to assess the performance and practical impact of integrating Homomorphic Encryption into collaborative model training. The chapter begins by detailing the experimental environment and evaluation metrics employed. The results are then systematically presented and analyzed across three learning configurations: centralized learning, standard federated learning, and HE-enhanced federated learning using both full and selective encryption strategies. This comparative analysis focuses on model accuracy, computational cost, and the level of privacy protection achieved.To support the analysis, performance trends and comparisons are illustrated using informative plots and graphs. The chapter concludes with a discussion of the main findings and the trade-offs between privacy, performance, and system efficiency.

4.2 Experimental Environment

4.2.1 Hardware and Execution Platform

The experimental setup was carried out using Google Colab [101], a cloud-based development environment that offers high-performance computing resources for machine learning research. The experiments were executed on a system equipped with an NVIDIA L4 GPU (22.5 GB VRAM), an Intel Xeon processor, and approximately 53 GB of RAM, collectively forming a robust environment for deep learning and homomorphic encryption tasks. This cloud infrastructure allowed for scalable, consistent, and reproducible experimentation without reliance on specialized local hardware.

4.2.2 Programming Language

The entire framework was implemented in Python [102], a high-level, interpreted programming language renowned for its simplicity and readability. Python has become a standard in the machine learning and data science communities due to its extensive ecosystem of libraries and frameworks, which facilitate rapid prototyping and deployment of complex systems.

4.2.2.1 Libraries

- **PyTorch:** A deep learning framework [103] used for defining and training the neural network models. PyTorch provides tensor computation capabilities and GPU acceleration, making it suitable for training both centralized and federated models efficiently.
- **NumPy (Numerical Python) :** it is a fundamental library for numerical computing in Python [104]. It provides high-performance, multidimensional array structures and a wide range of mathematical functions for array manipulation, linear algebra, and statistical operations. Its array-processing capabilities form the basis for many scientific and data-centric applications, making it an essential component in machine learning and data science.

- **Matplotlib and Seaborn :** These are data visualization libraries commonly used for generating statistical and scientific plots. Matplotlib [105] offers comprehensive low-level control over figure creation, while Seaborn [106] builds on top of it to provide high-level functions for creating aesthetically pleasing and informative statistical graphics.
- **Flower (FLWR) :** Flower is an open-source Python library for federated learning that enables decentralized model training across multiple devices or organizations while ensuring data remains confidential. It supports popular frameworks such as PyTorch, TensorFlow, and includes built-in strategies such as FedAvg for model aggregation [107]. Flower offers tools for both simulation and real-world deployment, with features such as secure aggregation and cross-platform communication via gRPC or REST. Its flexible and modular design makes it ideal for building scalable, privacy-preserving machine learning systems.
- **TenSEAL :** TenSEAL[108] is an open-source Python library that enables homomorphic encryption operations on tensors, allowing computations to be performed directly on encrypted data without decrypting it first. It is built on top of Microsoft SEAL , a widely used HE library, and provides a simple, PyTorch-like interface for working with encrypted tensors. TenSEAL is particularly useful in privacy-preserving machine learning scenarios, such as secure inference or federated learning, where sensitive data must remain confidential during computation. The library supports operations like addition, multiplication, and dot products on encrypted vectors and matrices.

4.3 The Experimental Dataset

In the context of our study on privacy-preserving federated learning, we selected the healthcare domain due to the critical importance of data privacy and the growing need for secure machine learning solutions. To support empirical evaluation within this domain, we adopted a publicly available benchmark that enables both reproducibility and scalability in medical imaging tasks.

We specifically used MedMNIST v2, a standardized large-scale benchmark collection tailored for biomedical image classification tasks. This dataset collection was designed to facilitate fair evaluation of machine learning algorithms across diverse biomedical image modalities, scales, and tasks while remaining lightweight and educational. MedMNIST v2 includes 12 datasets for 2D and 6 datasets for 3D classification, with all images preprocessed into standardized resolutions—28×28 for 2D and 28×28×28 for 3D. The datasets span a variety of modalities such as X-ray, CT, histopathology, and fundus photography, and support different task types including binary classification, multi-class classification, multi-label prediction, and ordinal regression.[109]

4.3.1 PathMnist Dataset

Among the available datasets in MedMNIST v2, we selected **PathMNIST**, which is derived from the NCT-CRC-HE-100K and CRC-VAL-HE-7K datasets - two collections of histopathological images used in colorectal cancer research. These images have been pre-processed and standardized by the MedMNIST authors to ensure consistency and suitability for lightweight benchmarking tasks.[109]

The PathMNIST dataset comprises 107,180 color image patches, each resized to 28×28 pixels in RGB format. These are divided into 89,996 training samples, 10,004 validation samples, and 7,180 test samples. The classification task is a 9-class multi-class problem , corresponding to nine different tissue types such as tumor epithelium, lymphocytes, mucus, and stroma (Figure 4.1).

Table 4.1: Description of the PathMNIST Dataset

Aspect	**Description**
Data Modality	Colon Histopathology
Total Samples	107,180
Training Set	89,996 samples (83.96%)
Validation Set	10,004 samples (9.33%)
Test Set	7,180 samples (6.69%)
Image Size	28 × 28 pixels, RGB
Classification Type	Multi-class
Number of Classes	9
Source Datasets	NCT-CRC-HE-100K and CRC-VAL-HE-7K
Use Case	Colorectal tissue classification in histopathology

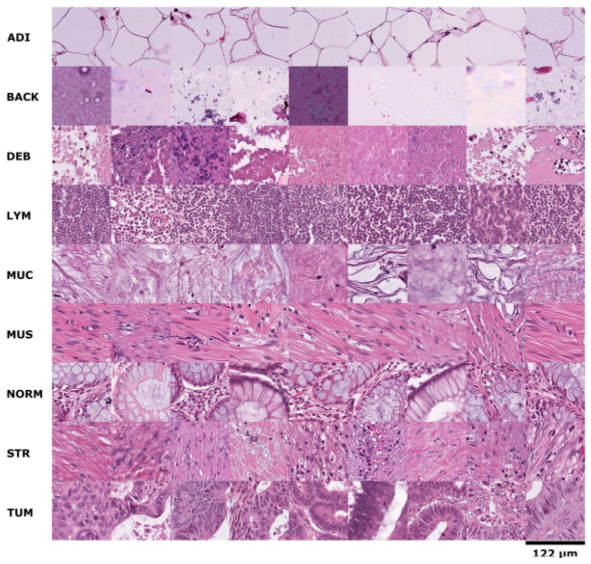


Figure 4.1: Example images for each of the nine tissue classes represented in the Pathmnist data set. ADI, adipose tissue; BACK, background; CRC, colorectal cancer; DEB, debris; HE, hematoxylin–eosin; LYM, lymphocytes; MUC, mucus; MUS, smooth muscle; NCT, National Center for Tumor Diseases; NORM, normal colon mucosa; STR, cancer-associated stroma; TUM, colorectal adenocarcinoma epithelium.

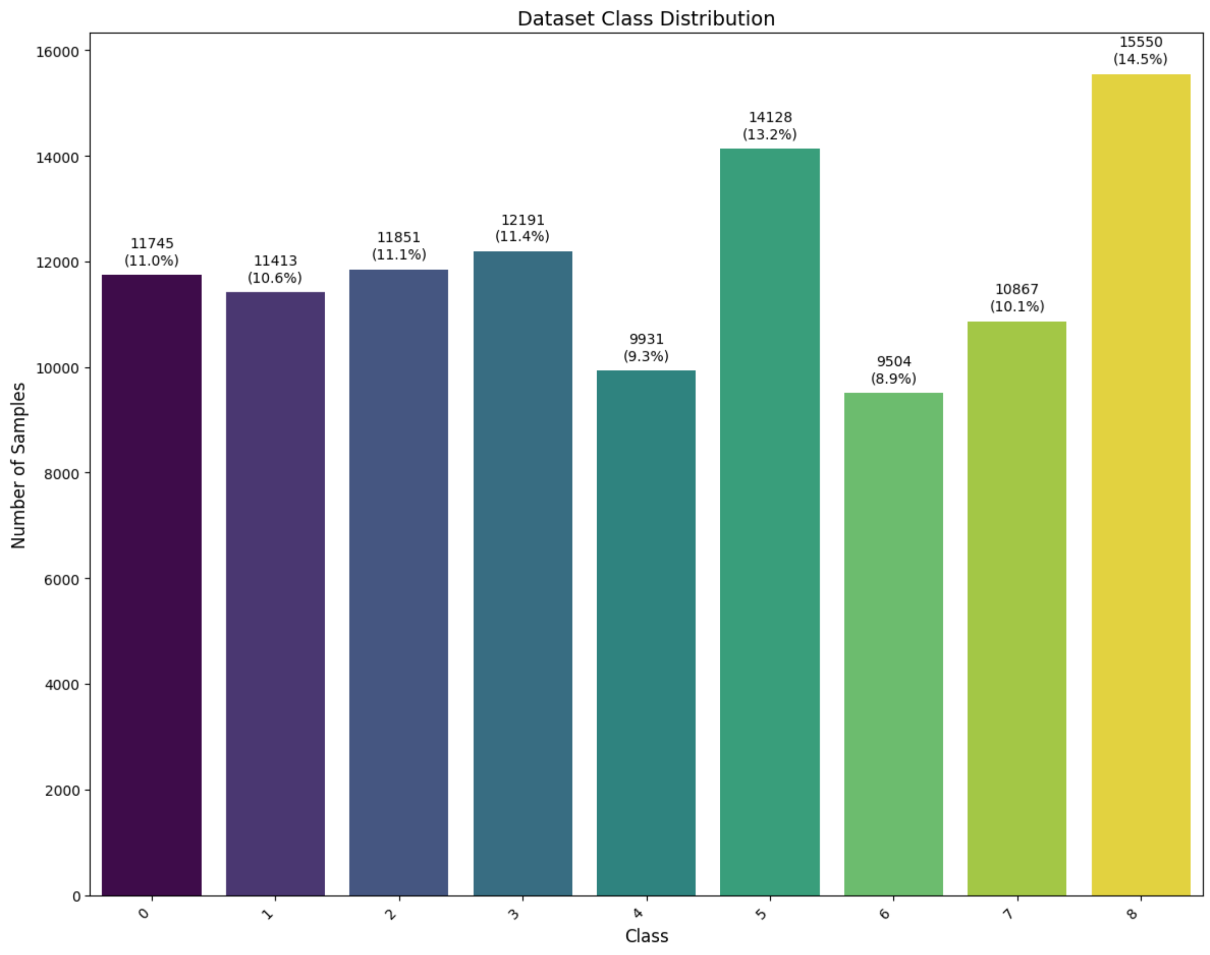
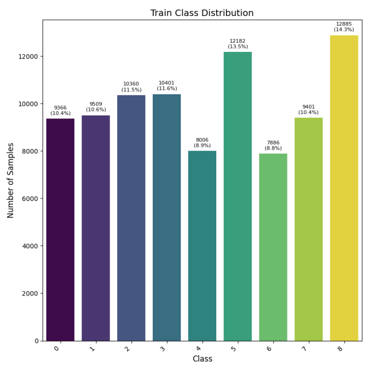
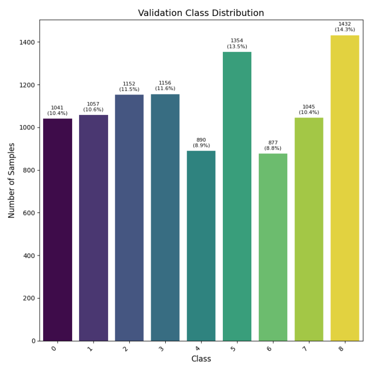


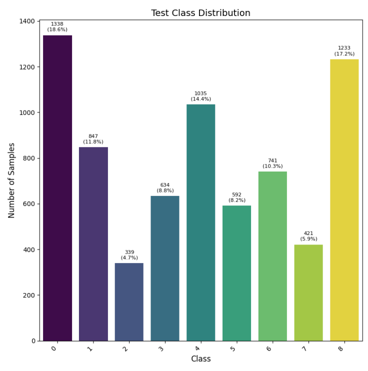
Figure 4.2: Pathmnist Dataset Class Distribution



(a) Train Dataset



(b) Validation Dataset



(c) Test Dataset

Figure 4.3: Class Distribution for subset (Train, Validation, Test)

4.3.2 Dataset Preprocessing

To prepare the PathMNIST dataset for training with convolutional neural networks , several preprocessing steps were applied. These steps help improve model performance, enhance generalization, and ensure compatibility with deep learning workflows:

1. **Tensor Conversion:**The input images were first converted from standard image formats (PIL images) into tensors. This conversion is essential for feeding the data into a CNN model and enabling batch-wise computation during training.

2. **Normalisation:**Pixel values were normalized, typically by scaling them to a range between 0 and 1 or by applying dataset-specific mean and standard deviation normalization. Normalization improves training stability and speeds up convergence by ensuring that all input features are on a similar scale.

3. **Data Augmentation:** To improve generalization and reduce overfitting, data augmentation techniques were applied during training. These included random horizontal flips, small rotations, and cropping. Such transformations increase the diversity of the training dataset without requiring additional labeled data, helping the model learn to recognize patterns under varying conditions.

4.4 Performance Evaluation Metric

To assess the effectiveness of the models under different configurations (centralized, federated, encrypted), multiple evaluation metrics were utilized. These metrics reflect both predictive performance and the computational and privacy-related trade-offs introduced by encryption.

4.4.1 Confusion Matrix

The confusion matrix is a core evaluation tool used in classification tasks to quantify the performance of a predictive model. It compares the predicted class labels to the actual ground truth and organizes the results in a structured table format. This matrix allows for detailed analysis of classification accuracy by displaying how many instances were correctly or incorrectly classified per class.
In binary classification tasks, the confusion matrix takes the following form:

Table 4.2: Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	**True Positive (TP)**	False Negative (FN)
Actual Negative	False positive (FP)	**True Negative (TN)**

Each term in the matrix has a specific interpretation:

- **True Positive :** The model correctly predicted the positive class.
- **False Negative :** The model incorrectly predicts a negative class for a positive instance.
- **False Positive :** The model incorrectly predicts a positive class for a negative instance.
- **True Negative :** The model correctly predicts a negative class.

In multi-class classification tasks, this concept is extended to compare predictions across all class labels.
The confusion matrix allows one to assess not only overall accuracy but also which specific classes are prone to misclassification.

4.4.2 Accuracy

Accuracy is the most fundamental metric for classification tasks, measuring the proportion of correctly predicted labels out of the total number of predictions. It provides a direct indication of the model's overall performance.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

4.4.3 Classification Report Metrics

Derived from the confusion matrix, these metrics are computed per class and are crucial for detailed performance assessment:

4.4.3.1 Precision

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

This reflects how many of the predicted positives are actually correct.

4.4.3.2 Recall

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

This measures the ability of the model to correctly identify actual positives.

4.4.3.3 F1-Score

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

The F1-score balances precision and recall and is particularly useful when dealing with class imbalance.

4.4.4 Computational Cost

Computational cost reflects the overall efficiency of the federated learning process, especially under encryption constraints. It helps measure the practical viability of deploying privacy-preserving FL systems. The components considered include::

- **Training Time:** The duration taken for local model updates on the client side.
- **Encryption/Decryption Time:** Time consumed in securing data using homomorphic encryption.

- **Aggregation Time:** The time taken by the server to secure the aggregation of the encrypted parameters.
- **Serialization/Deserialization** Time: Time needed to convert encrypted vectors into a transmittable format and reconstruct them.
- **Total Round Time:** The overall time required to complete one round of FL, including all of the above steps.

4.4.5 PSNR and SSIM

To evaluate the extent of privacy preservation against gradient leakage attacks, we employ two image-based similarity metrics:

4.4.5.1 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) Measures the similarity between the original image and the reconstructed image in decibels (dB). Higher PSNR values indicate better reconstruction, which implies greater privacy leakage .It is calculated according to the following equation:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{\text{MSE}} \right) \tag{4.5}$$

where :

- MAX_I is the maximum possible pixel value of the image (typically 255),
- MSE is the Mean Squared Error between the original and reconstructed images.

4.4.5.2 Structural Similarity Index Measure (SSIM)

Structural Similarity Index (SSIM) Evaluates perceptual similarity based on luminance, contrast, and structure. A value closer to 1 implies strong similarity and potential leakage while the lower values suggest stronger privacy.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.6}$$

Where:

- μ_x, μ_y are means of the images.
- σ_x^2, σ_y^2 are variances.
- σ_{xy} is the covariance.
- C_1, C_2 are constants to stabilize the division.

4.5 Training Configuration

The training configuration of a deep learning model involves setting up the model with the essential components required for training, such as loss function, optimizer, and learning rate.

4.5.1 Categorical Cross-Entropy Loss

The Categorical Cross-Entropy Loss is widely used for multi-class classification problems, especially when the model's outputs represent probability distributions over multiple classes.
Given a dataset of N samples, the loss is computed as:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) \tag{4.7}$$

where:

- N: number of samples
- C: number of classes
- $y_{i,c}$:true label (1 if sample i belongs to class c, 0 otherwise)
- $\hat{y}_{i,c}$:predicted probability that sample i belongs to class c

This function penalizes incorrect predictions by increasing the loss when the predicted probability for the true class is low. It is especially suitable for our use case, where the task involves classifying image samples into nine histopathological categories.

4.5.2 AdamW optimizer

AdamW [110] is an improved version of the Adam optimizer that introduces a more effective way of handling weight decay during training. While Adam adaptively adjusts the learning rate for each parameter using first and second moment estimates of gradients, AdamW decouples the weight decay (used for regularization) from the gradient-based update. The update rule becomes:

$$\theta_t = \theta_{t-1} - \eta \cdot \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \cdot \theta_{t-1} \right) \tag{4.8}$$

where:

- m_t is the bias-corrected first moment estimate (mean of gradients).
- v_t is the bias-corrected second moment estimate (variance).
- η:learning rate.
- ϵ: small constant to avoid division by zero.
- $\lambda \cdot \theta_{t-1}$ is the explicit weight decay applied after the Adam step.

4.5.3 Learning Rate Scheduling: Cosine Annealing

To further enhance training efficiency, a Cosine Annealing Learning Rate Scheduler [111] was applied. This schedule gradually reduces the learning rate following a cosine curve, which helps the model converge smoothly without abrupt changes in the update step. The learning rate η_t at epoch t is adjusted using the following formula

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right) \tag{4.9}$$

where :

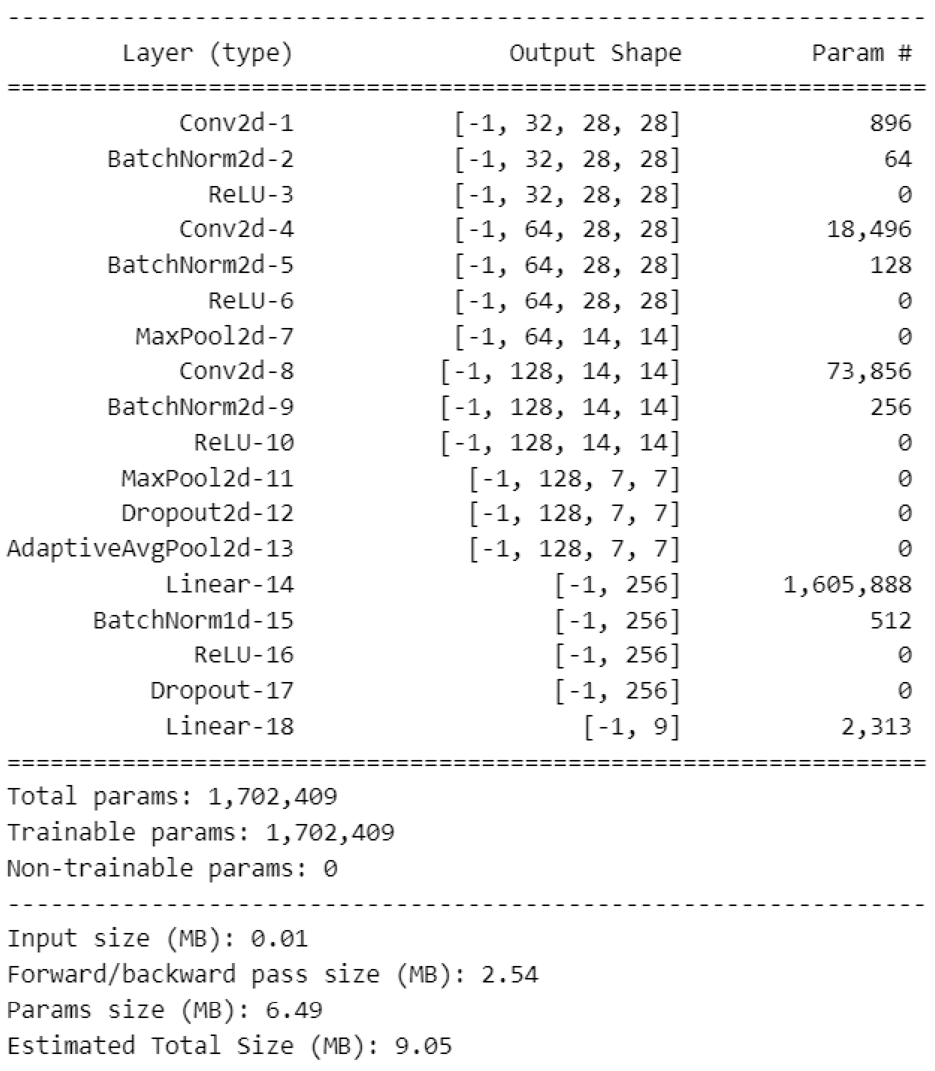
- η_{min} and η_{max} are the minimum and initial learning rates.
- T_{cur} is the current epoch.
- T_{max} is the total number of epochs.

4.6 Centralized Deep Learning Models

Prior to implementing the federated learning framework, several deep learning models were evaluated using a centralized approach. This preliminary step was carried out to identify the most performant architecture to serve as the baseline for subsequent experiments in the federated setting. To this end, we compared a range of architectures, including custom-designed CNNs (CNNmed and CNNmed augmented with self-attention), as well as popular pre-trained models such as ResNet-18, MobileNetV3 Small, and EfficientNet-B0.

4.6.1 CNNMed

The CNNMed network is a custom-designed convolutional neural architecture consisting of three convolutional layers, each followed by batch normalization and ReLU activation. The feature extractor is structured into two downsampling stages: the first includes two convolutional layers and a max pooling operation, while the second adds another convolutional layer and a second max pooling step. A dropout layer is applied after the final pooling operation to reduce overfitting. To ensure consistent feature dimensions regardless of input size, an adaptive average pooling layer produces a fixed output of feature maps. These are flattened and passed through a classifier composed of two fully connected layers, with batch normalization, ReLU activation, and an additional dropout layer between them. The final output layer maps to 9 target classes, producing the model's class predictions.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 28, 28]             896
       BatchNorm2d-2           [-1, 32, 28, 28]              64
              ReLU-3           [-1, 32, 28, 28]               0
            Conv2d-4           [-1, 64, 28, 28]          18,496
       BatchNorm2d-5           [-1, 64, 28, 28]             128
              ReLU-6           [-1, 64, 28, 28]               0
         MaxPool2d-7           [-1, 64, 14, 14]               0
            Conv2d-8          [-1, 128, 14, 14]          73,856
       BatchNorm2d-9          [-1, 128, 14, 14]             256
             ReLU-10          [-1, 128, 14, 14]               0
        MaxPool2d-11            [-1, 128, 7, 7]               0
        Dropout2d-12            [-1, 128, 7, 7]               0
AdaptiveAvgPool2d-13            [-1, 128, 7, 7]               0
           Linear-14                  [-1, 256]       1,605,888
      BatchNorm1d-15                  [-1, 256]             512
             ReLU-16                  [-1, 256]               0
          Dropout-17                  [-1, 256]               0
           Linear-18                    [-1, 9]           2,313
================================================================
Total params: 1,702,409
Trainable params: 1,702,409
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 2.54
Params size (MB): 6.49
Estimated Total Size (MB): 9.05
```

Figure 4.4: Summary of CNNMed Architecture

4.6.2 CNNMed with Self-attention

In this variant, A self-attention mechanism was integrated into CNNMed to enhance spatial feature learning by focusing on relevant regions within the image. This addition aims to improve performance without a significant increase in computational complexity.

4.6.3 Pre-trained models

A Pre-trained models are deep neural networks initially trained on large datasets such as ImageNet and can be reused or fine-tuned for a different but related task. These models provide a well-initialized set of weights that capture general patterns such as edges, textures, and object shapes, which can be effectively transferred to domain-specific problems.
The use of pre-trained models offers several advantages:

- It significantly reduces training time and computational cost.
- It improves model performance, especially when task-specific data is limited.

- It allows developers to leverage existing, well-optimized architectures trained on vast datasets.

In medical imaging tasks, where labeled data is often scarce, pre-trained models such as ResNet, MobileNet, and EfficientNet are frequently adapted for classification and segmentation tasks through fine-tuning, improving both accuracy and efficiency.

4.6.3.1 ResNet-18

ResNet-18 [112] is a convolutional neural network that introduced the concept of residual learning to facilitate the training of deeper networks. By incorporating identity shortcut connections, it mitigates the vanishing gradient problem, enabling the construction of substantially deeper architectures without degradation in performance. The 18-layer variant offers a balanced trade-off between depth and computational efficiency, making it suitable for various image classification tasks.

4.6.3.2 MobileNetV3 Small

MobileNetV3 [113] Small is a lightweight convolutional neural network designed for mobile and embedded vision applications. It combines automated neural architecture search with advanced design strategies, including squeeze-and-excitation modules and hard-swish activations, to achieve a balance between latency and accuracy. This model is optimized for resource-constrained environments, offering efficient performance for tasks like image classification.

4.6.3.3 EfficientNet-B0

EfficientNet-B0 [114] introduces a novel compound scaling method that uniformly scales network depth, width, and resolution using a set of fixed scaling coefficients. This approach enables the model to achieve superior accuracy and efficiency compared to previous convolutional networks. EfficientNet-B0 serves as the baseline model in the EfficientNet family, demonstrating strong performance across various image classification benchmarks.

After training and evaluating the selected models within the centralized learning setup, their performance was measured using standard classification metrics. Each model was trained with a batch size of 64 and an initial learning rate set to 0.001 to ensure consistent optimization conditions across experiments. The results obtained are summarized in the table below.

Table 4.3: Model Performance Comparison

Models	Epochs	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)
CNNMed	30	93.57	93.55	93.33	93.55
CNNMed & self attention	30	94.00	94.11	93.94	94.11
MobilenetV3 Small	10	94.83	94.85	94.82	94.85
Resnet-18	10	95.15	95.04	94.91	95.04
EfficientNet B_0	10	95.38	95.35	95.23	95.35

Figure 4.5 illustrates the accuracy and evaluation metrics of the tested models, highlighting the superior performance of EfficientNet-B0.

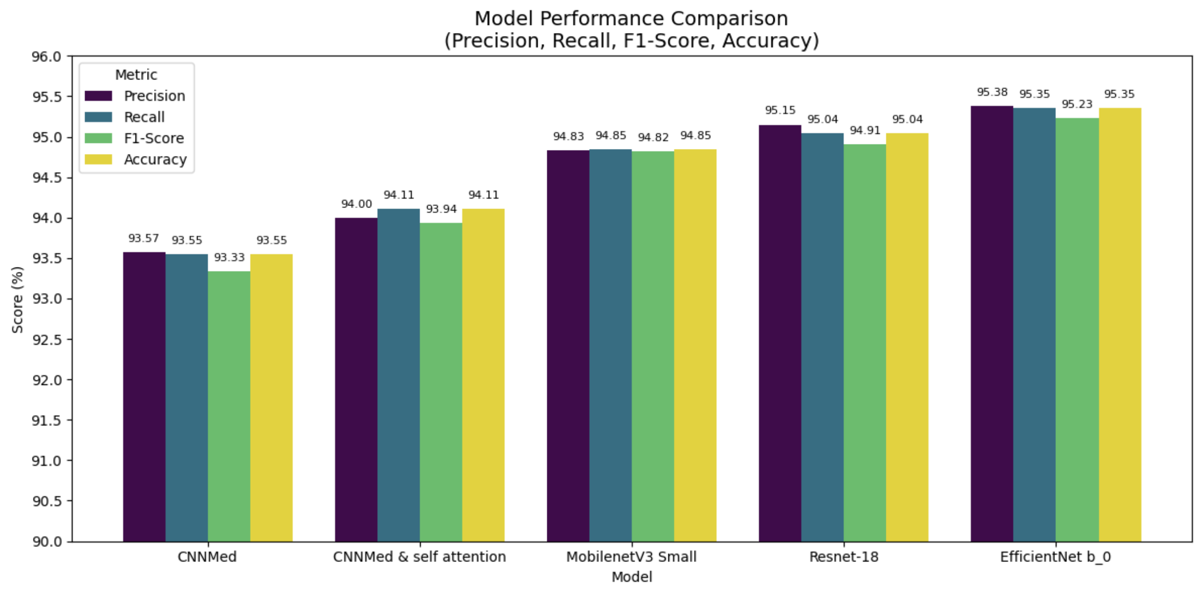


Figure 4.5: Model performance Comparison

Since the EfficientNet-B0 model achieved the highest performance in all evaluation metrics, it was selected for subsequent federated learning experiments. This choice reflects a balance between computational efficiency and model accuracy, as EfficientNet-B0 offers a middle ground, being lighter than ResNet-18 but more expressive than MobileNetV3 Small.

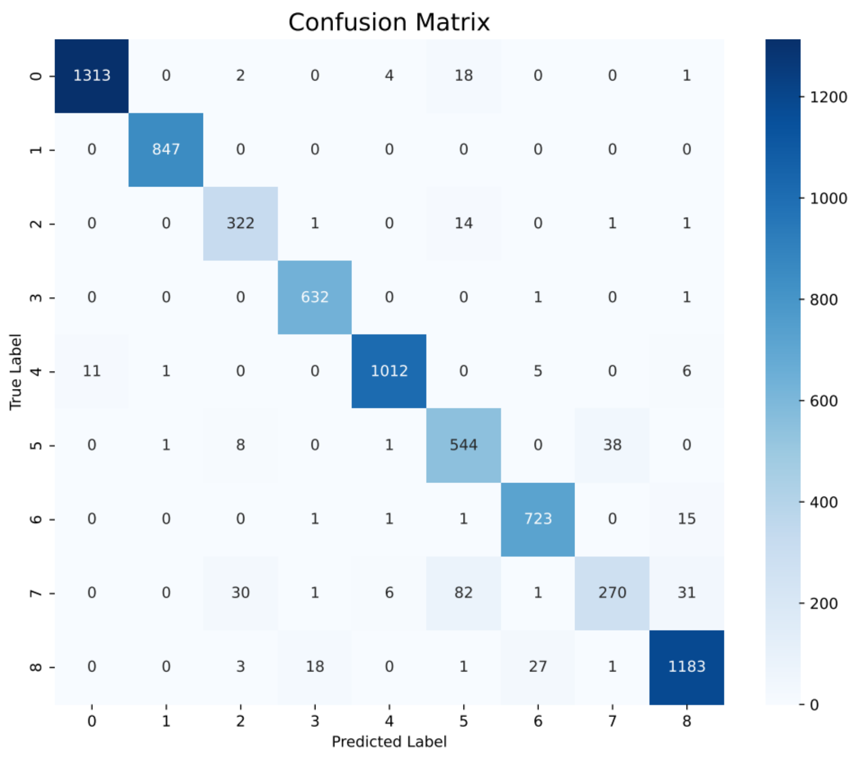


Figure 4.6: Confusion matrix of EfficientNet-B0 model

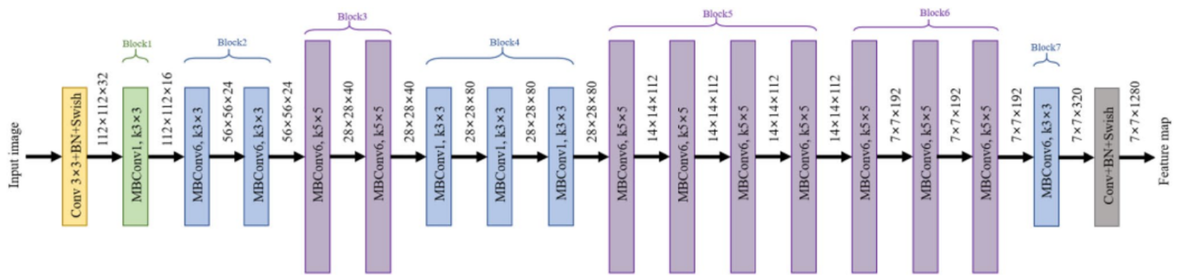


Figure 4.7: The detailed architecture of EfficientNet-B0. consists of seven blocks which are shown in different colours. The basic building block of EfficientNet-B0 is a mobile inverted bottleneck convolution (MBConv), with each MBConv block annotated with its respective kernel size.

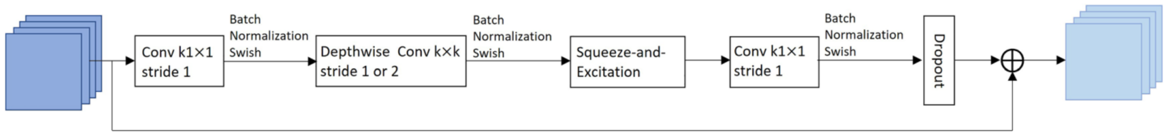


Figure 4.8: The structure of the MBConv block

4.7 Federated Learning Approach

To evaluate the robustness and performance of the proposed learning models in decentralized environments, The Federated Learning approach was evaluated under two distinct data distribution scenarios: *independent and identically distributed* and *non-independent and identically distributed* . The experiments were carried out in two configurations, one with 10 clients and another with 5 clients, to observe the impact of the client participation scale on performance.

Table 4.4: Hyperparameters Settings for federated learning approach

Hyperparameter Settings	
Client Model	EfficientNet-B0
Number of Clients	5 and 10 (tested separately)
Data Distribution Scenarios	IID and Non-IID (Dirichlet distribution)
Number of Rounds	10
Local Epochs per Round	1
Batch Size	64
Learning Rate	0.001
Aggregation Strategy	Federated Averaging (FedAvg)

4.7.1 IID Data Distribution

To assess the behavior of the proposed model in a well-balanced distributed scenario, we conducted experiments under the IID setting, where each client's local dataset follows a similar distribution. Figure 4.9 demonstrates the uniform distribution of data samples across 5 clients, highlighting the balanced nature of the IID setup.

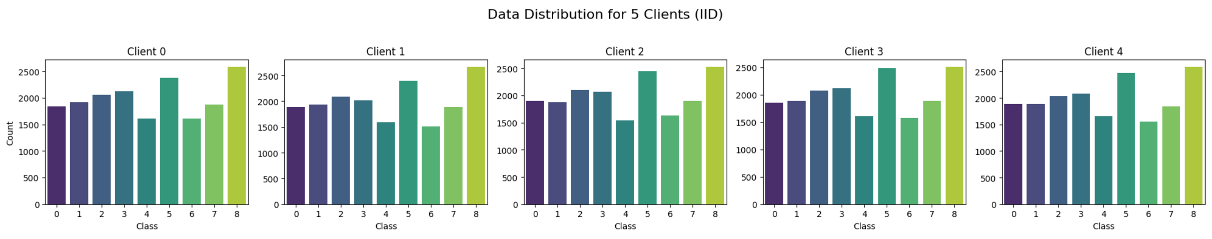


Figure 4.9: IID Data Distribution for 5 client

Following this, we trained the federated model with both 5 clients and 10 clients to evaluate how the number of participants affects convergence and performance. The learning progress is depicted in Figure 4.10, which combines the accuracy and loss curves across rounds for both settings. As shown, The training process exhibits stable convergence under both configurations, though minor fluctuations appear as the number of clients increases.

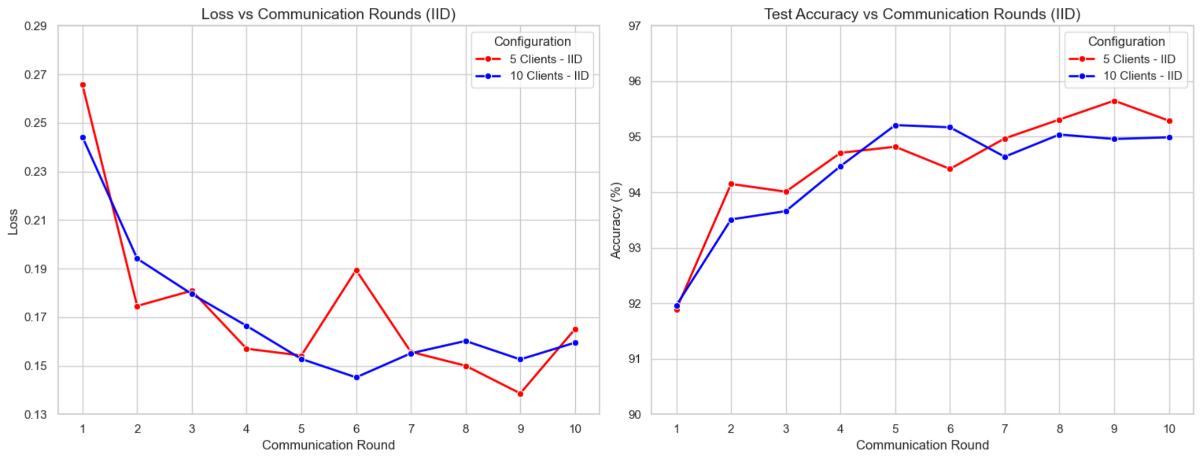


Figure 4.10: Performance Curves of Federated Learning Under IID Data Distribution

The final accuracy and loss results are summarized in Table4.5, which also includes the centralized learning baseline for comparison. As the table indicates, the federated approach under IID conditions achieves competitive results, with minimal performance degradation compared to the centralized model. This confirms the effectiveness of our federated setup in preserving learning performance while distributing the data.

Table 4.5: Performance Comparison: Centralized vs Federated Learning (IID)

Setting	**Accuracy (%)**	**Loss**
Centralized	95.35	0.1701
Federated (5 Clients)	**95.29**	0.1651
Federated (10 Clients)	**94.99**	0.1595

4.7.2 Non-IID Data Distribution

In the Non-IID configuration, client datasets were generated using a Dirichlet distribution with a concentration parameter $\alpha = 0.5$, a commonly adopted method for simulating heterogeneous data partitions in federated learning. This setup introduces variability in class representation across clients, closely reflecting real-world decentralization scenarios. As illustrated in Figure 4.11, the resulting class distributions exhibit significant imbalance, demonstrating the statistical heterogeneity introduced by the non-IID partitioning strategy.

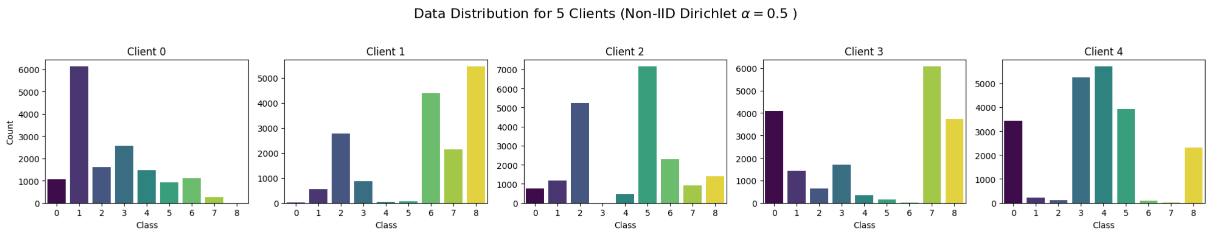


Figure 4.11: Non-IID Data Distribution for 5 client

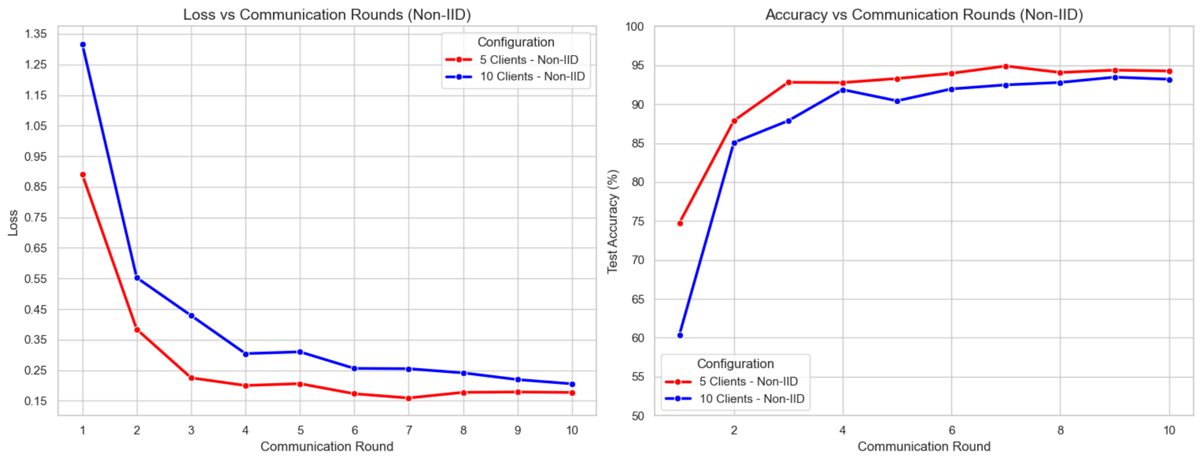


Figure 4.12: Performance Curves of Federated Learning Under Non-IID Data Distribution

Table 4.6: Comparison of Test Accuracy and Loss for IID and Non-IID Settings

Setting Type	**Clients**	**Final Accuracy (%)**	**Final Loss**
IID	5	95.29	0.1651
	10	94.99	0.1595
Non-IID	5	94.26	0.1774
	10	93.20	0.2055

To assess the impact of data heterogeneity,Experiments were conducted under both IID and non-IID configurations. The accuracy and loss curves under non-IID conditions reveal less stable convergence compared to the IID setting (as illustrated in Figure 4.12) , reflecting the challenges posed by statistical heterogeneity. While performance under non-IID settings remains competitive, it exhibits a slight drop in accuracy and convergence speed. This highlights the potential benefit of tailored aggregation strategies (e.g., FedProx) that are specifically designed to address client drift in non-IID scenarios. Table 4.6 presents a concise comparison of final test accuracy and loss across both configurations.

4.8 Evaluation of the Federated Learning-Based Homomorphic Encryption Framework

This section presents a comprehensive evaluation of the proposed Federated Learning framework enhanced with Homomorphic Encryption. The primary goal is to investigate the extent to which encryption affects model performance and computational cost, while validating that privacy is preserved effectively. The evaluation is organized into three main components: Model Performance, Computational Overhead, and defence against privacy attacks.

To isolate and clearly observe the impact of encryption strategies, the experiments in this section were conducted under a standardized IID data distribution using 5 clients. This setup allows us to minimize variability due to data heterogeneity and instead focus directly on the implications of applying homomorphic encryption.

In the selective encryption setting, only a subset of model parameters was encrypted. Specifically, a selective encryption ratio of $\gamma = 0.15$ was applied, which means that the top 15% of the layers, based on gradient sensitivity and predefined structural criteria, was protected using CKKS encryption. This design choice aims to achieve a practical trade-off between privacy and computational efficiency.

The CKKS encryption scheme was employed for all encrypted computations. The following parameters were adopted to balance security, efficiency, and numerical precision:

- **poly modulus degree:** Defines the degree of the polynomial modulus. A value of 16,384 was chosen, offering 128-bit security and allowing encoding of up to 8,192 real values in a single ciphertext. This high-dimensional packing is especially useful for encoding flattened model parameters in deep networks.
- **coeff modulus bit sizes:** Specifies the modulus chain used for ciphertext operations. The configuration [60, 40, 40, 60] supports multi-step encrypted computations while maintaining control over noise growth.
- **global scale:** A scaling factor that determines the precision of the real numbers encoded in ciphertexts. A scale of 2^{40} was selected to ensure high numerical fidelity during operations such as encrypted aggregation and model updates.

4.8.1 Model Performance

With the CKKS scheme of homomorphic encryption applying to federated learning frameworks, assessing its impact on model performance has become critical. As CKKS pertains to approximate arithmetic, there is the possibility of affecting the accuracy of deep learning models. Therefore, three training settings were compared: baseline federated learning without encryption, federated learning with full homomorphic encryption, and federated learning with selective encryption. This evaluation used the metrics of test accuracy and loss recorded through various communication rounds (as shown in Figure 4.13).

Table 4.7: Final Model Performance (Accuracy and Loss)

Method	**Final Test Accuracy (%)**	**Final Loss**
Baseline Federated Learning	95.29	0.165
Full Homomorphic Encryption	95.13	0.161
Selective Homomorphic Encryption	94.79	0.167

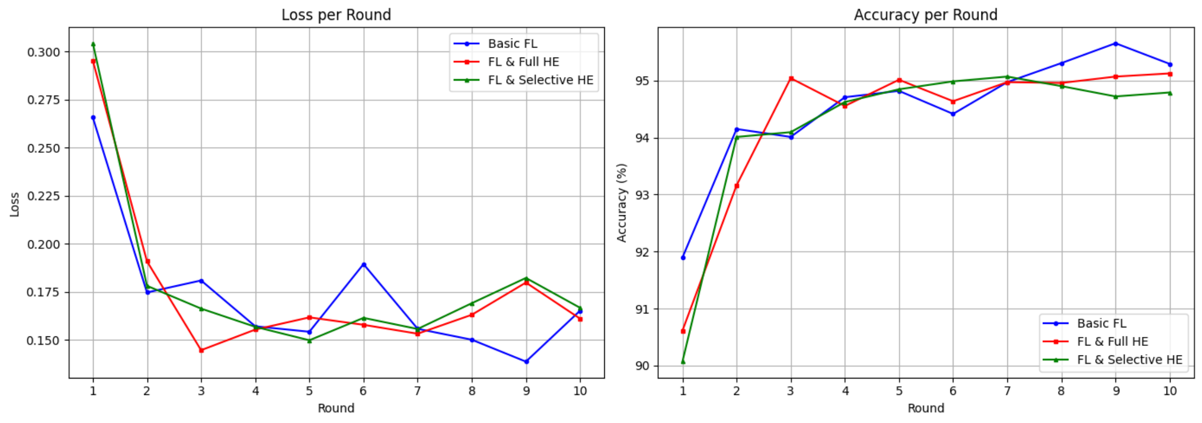


Figure 4.13: Test loss and accuracy over communication rounds for baseline FL, full homomorphic encryption, and selective homomorphic encryption

4.8.2 Computational Overhead Analysis

The objective of this evaluation is to assess the computational impact of integrating HE into the federated learning process. Thus, we examine computational costs incurred under three configurations: basic conditions under which just FL, the full encryption FL, and the FL with selective encryption. With such an analysis, the analysis covers two key dimensions: cumulative round duration and a breakdown of client-side execution times, including those for encryption and decryption overhead.
Figure 4.14 presents the total time consumed per communication round in the 10 training rounds for all configurations. As expected, both full and selective HE introduce additional computational time compared to baseline. However, selective encryption demonstrates a clear reduction in overhead relative to full encryption.

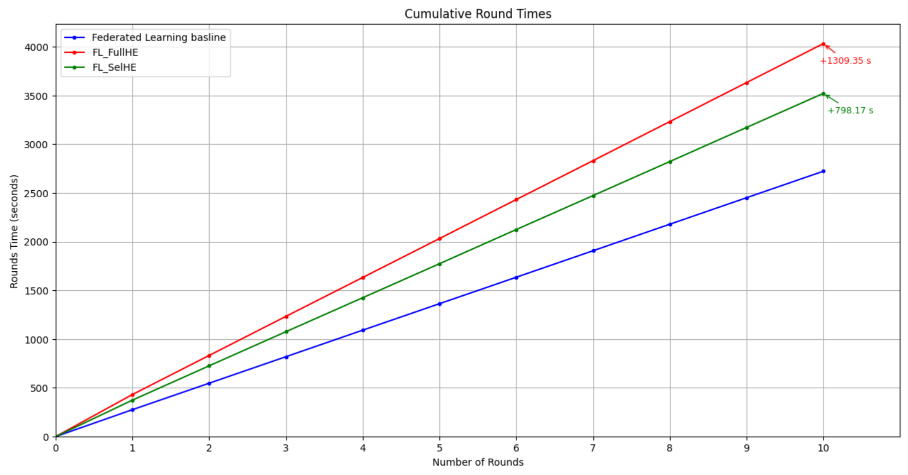


Figure 4.14: Comparison of Cumulative Round Times for Baseline FL, FL with Full HE, and FL with Selective HE

Table 4.8 summarizes the detailed timing metrics for each configuration.This includes component-wise breakdowns such as client training, encryption, decryption, (de)serialization, aggregation, and overall round duration. The table also includes percentage comparisons to highlight the relative cost differences among the three approaches, offering a comprehensive view of the computational burden introduced by privacy-preserving mechanisms.

Table 4.8: Comparison of metric times for different FL approaches.

Metric Time (s)	FL	FL_fullHE	FL_SelHE	FL vs FL_fullHE (%)	FL vs FL_SelHE (%)	FL_fullHE vs FL_SelHE (%)
Training	53.98	54.04	54.02	~0%	~0%	~0%
Encryption	(N/A)	11.75	6.98	(N/A)	(N/A)	-40.56
Decryption	(N/A)	1.33	0.86	(N/A)	(N/A)	-35.38
Serialization	(N/A)	5.57	3.16	(N/A)	(N/A)	-43.25
Deserialization	(N/A)	1.16	0.90	(N/A)	(N/A)	-22.30
Client Execution	53.98	73.86	65.93	+36.79	+22.11	-10.74
Total Round Duration	272.21	403.14	352.02	+48.08	+29.30	-12.67
Total Time for 10 Rounds	2722.07	4031.42	3520.24	+48.10	+29.32	-12.67
Average Aggregation Time	0.18	22.06	13.02	+12155.5	+7133.3	-40.98

Figure 4.15 shows the comparison of the average execution time on the client side between the three configurations. The figure clearly illustrates the time-intensive operations and emphasizes the benefit of encrypting only selected model layers.

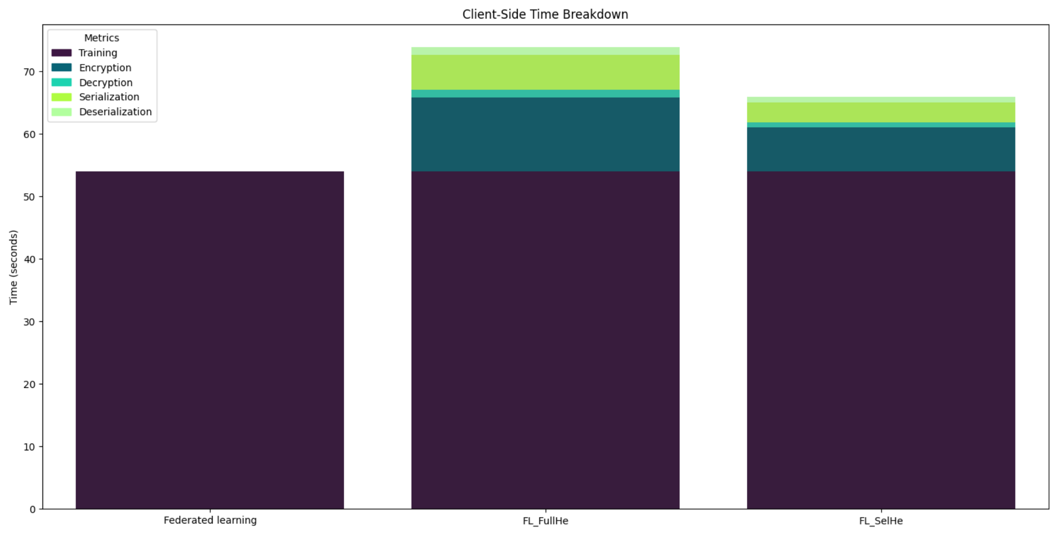


Figure 4.15: Client-side Time Breakdown

4.8.3 Evaluation of Privacy Protection in the Selective Encryption Strategy

Since selective encryption does not encrypt all model parameters, some layers remain in plaintext to preserve computational efficiency. Consequently, it becomes essential to evaluat of this partially protected configuration against known privacy attacks. In this section, we simulate the Deep Leakage from Gradients attack[72], which reconstructs input data by exploiting exposed gradients during federated learning. We compare the effectiveness of this attack under two conditions: a baseline federated learning setting without encryption and a federated learning setting with selective homomorphic encryption applied to sensitive layers.

The DLG attack assumes access to shared model gradients, allowing an adversary to reconstruct input data through optimization. The attacker initializes a random input and label, then iteratively updates them to minimize the distance between the computed gradients from the dummy input and the actual shared gradients(as illustrated in Algorithm 11).

Algorithm 11 Deep Leakage from Gradients

```
Require: $\mathcal{F}(x; W)$: Differentiable model with parameters $W$; $\nabla W$: Shared gradients from training data
Ensure: Reconstructed training data $x$, $y$
1: procedure DLG($\mathcal{F}, W, \nabla W$)
2:     $x'_1 \leftarrow \mathcal{N}(0,1)$ , $y'_1 \leftarrow \mathcal{N}(0,1)$          ▷ Initialize dummy input and label
3: for $i \leftarrow 1$ to $n$ do
4:     $\nabla W'_i \leftarrow \frac{\partial \ell(\mathcal{F}(x'_i, W_t), y'i)}{\partial W_t}$          ▷ Compute dummy gradients
5:     $D_i \leftarrow |\nabla W'i - \nabla W|^2$          ▷ Compute distance between dummy and real gradients
6:     $x'i+1 \leftarrow x'_i - \eta \nabla x'_i D_i, \quad y'_{i+1} \leftarrow y'_i - \eta \nabla y'_i D_i$          ▷ Gradient update step
7: end for
8: return $x'_{n+1}, y'_{n+1}$
9: end procedure
```

4.8.3.1 DLG Attack on Baseline Federated Learning

Figure 4.16 shows the reconstructed image over multiple iterations of optimization. As the number of iterations increases, the dummy input gradually converges to the true image.

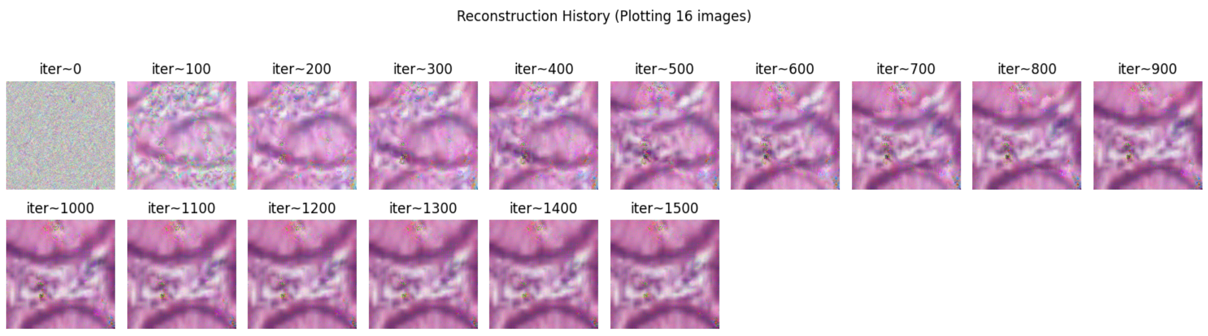


Figure 4.16: The image reconstruction process over iterations (FL)

Figure 4.17 presents the evolution of PSNR and SSIM metrics, both of which increase steadily, confirming the quality and similarity of the reconstructed image.

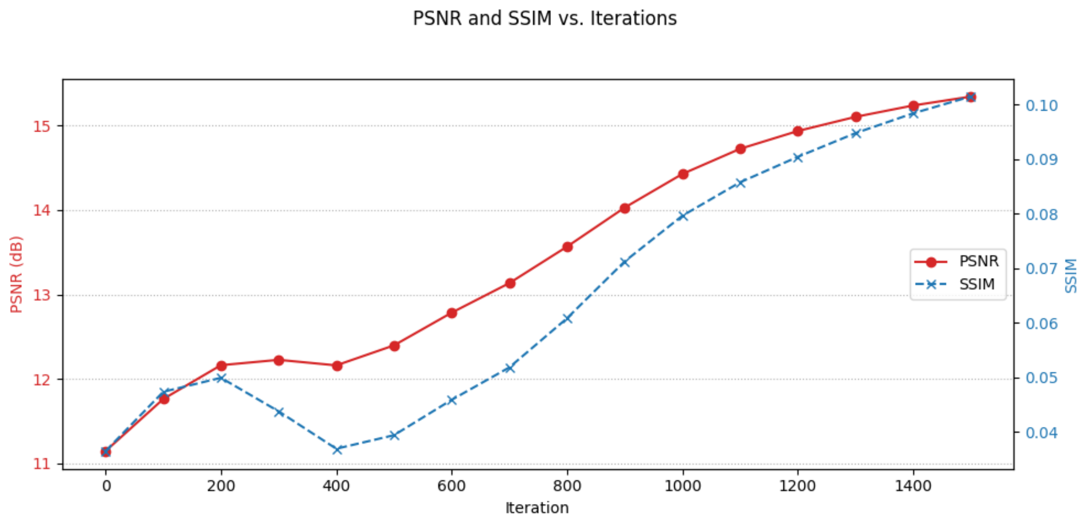


Figure 4.17: PSNR and SSIM (FL)

Figure 4.18 compares the original ground-truth image with the final reconstructed image. The high visual similarity illustrates the success of the DLG attack.

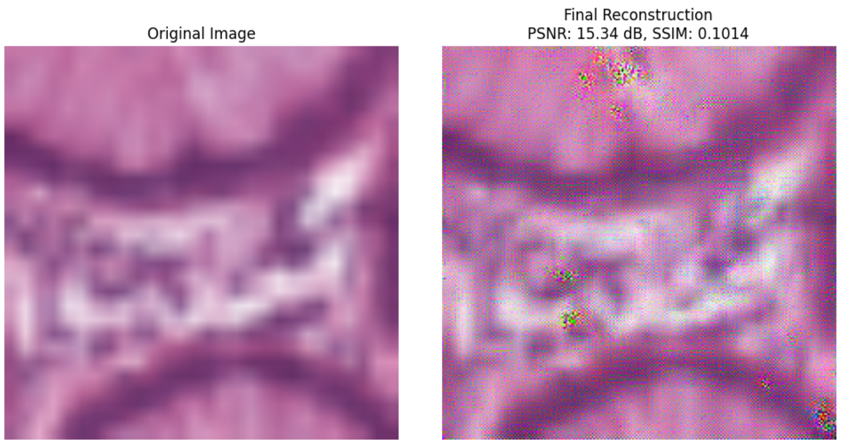


Figure 4.18: Original Image vs Final Reconstructed Image (FL)

4.8.3.2 DLG Attack on Selective Homomorphic Encryption Strategy

Figure 4.19 depicts the progression of the reconstructed image over multiple optimization iterations when applying DLG under the selective encryption strategy. Unlike the baseline scenario, the dummy input fails to approximate any meaningful visual content, showing no convergence toward the original image.

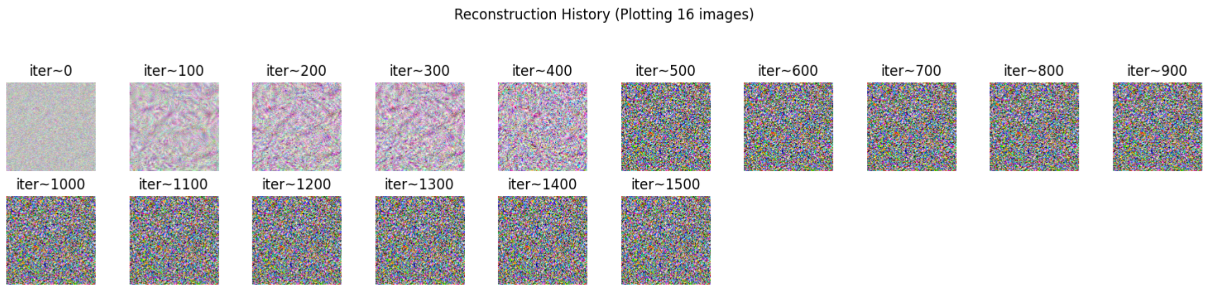


Figure 4.19: The image reconstruction process over iterations (FL_SelHE)

Figure 4.20 shows the PSNR and SSIM values during the DLG attack under the selective encryption setting. Both metrics drop quickly and stabilize at very low levels, with PSNR below 6 dB and SSIM near 0.004. This indicates that the reconstructed image bears no meaningful similarity to the original, confirming the effectiveness of the selective encryption strategy in resisting the attack.

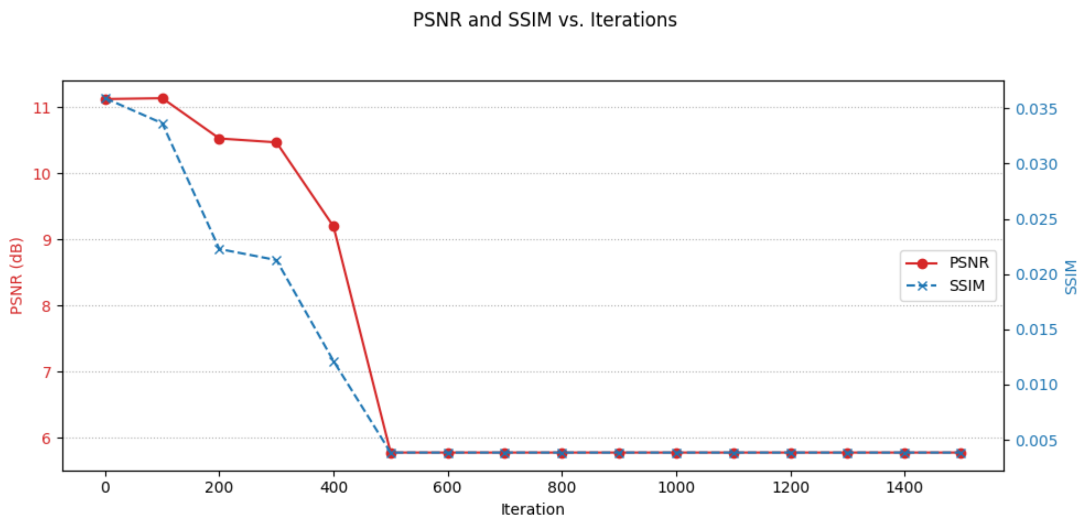


Figure 4.20: PSNR and SSIM (FL_SelHe)

Figure 4.21 compares the original sample image with the final reconstructed result. The reconstructed output appears as noise, bearing no visual similarity to the ground truth, demonstrates the ineffectiveness of the attack under the selective encryption strategy.

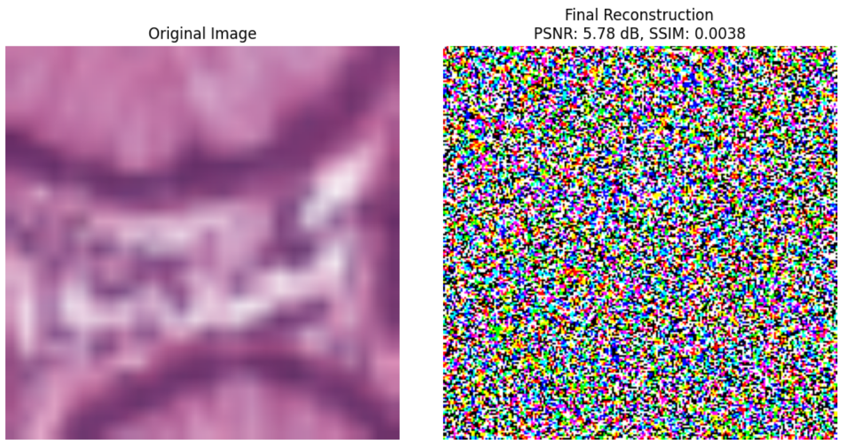


Figure 4.21: Original Image vs Final Reconstructed Image (FL_SelHE)

4.9 Discussion

This section provides an integrated discussion of the experimental results presented in Chapter 4. It brings together key findings from the centralized and federated learning experiments, the evaluation of privacy-preserving mechanisms, and the effectiveness of the proposed selective encryption strategy.

The evaluation began with training and comparing several deep learning models under a centralized setting using the PathMNIST dataset. Among the models assessed, EfficientNet-B0 demonstrated the most favorable trade-off between classification accuracy and architectural efficiency, making it a suitable choice for subsequent federated learning experiments. The results also highlighted the importance of data preprocessing steps, such as normalization and augmentation, which contributed to improving model generalization and stability during training.

Federated learning was then experimented under two data distribution conditions: independent and identically distributed , and non-IID. In the IID scenario, the federated model achieved performance nearly equal to the centralized baseline, confirming that the distributed training process did not significantly impact model accuracy. Under the non-IID condition, which introduced statistical variability between clients using a Dirichlet distribution, the model experienced a moderate drop in accuracy and slower convergence. Despite this, the learning process remained stable and competitive, supporting the applicability of federated learning in more realistic, heterogeneous environments.

The main contribution of this work was the integration of homomorphic encryption (HE) into the federated learning process to enhance privacy protection. Two encryption strategies were evaluated: full encryption and selective encryption, both based on the CKKS scheme. While full encryption secured all model updates, it introduced a computational burden. To address this, the selective encryption strategy applied a selective encryption ratio of γ=0.15, meaning that only the most privacy-sensitive layers were encrypted. This approach resulted in a clear improvement in computational efficiency compared to full encryption, while maintaining strong protection against privacy leakage and preserving model performance.

An important aspect of this study was the manual integration of encrypted computation into the Flower framework. Since Flower does not provide native support for encrypted tensors, custom mechanisms were developed for encryption, serialization, deserialization, and aggregation of model updates using the TenSEAL library. This experience emphasizes the need for federated learning frameworks to support encrypted computation as a built-in feature, which would greatly improve usability and efficiency for privacy-focused applications.

To validate the framework's resistance to privacy attacks, a Deep Leakage from Gradients attack was simulated. In the unencrypted federated learning configuration, the attack successfully reconstructed private training images, with high visual similarity and measurable similarity scores such as PSNR and SSIM. However, under the selective encryption setting, the reconstructed outputs failed to resemble the original data, and similarity metrics dropped to low values. This result confirms that encrypting only a targeted portion of the model is sufficient to disrupt gradient inversion and protect user data.

Overall, the experimental results demonstrate that the proposed privacy-preserving federated learning framework effectively maintains learning performance while enhancing user privacy. The selective encryption approach offers a promising balance between computational cost and security, and its successful integration into an existing federated learning system highlights the practical potential of encryption-aware learning architectures in sensitive domains such as healthcare.

4.10 Conclusion

The chapter examined the proposed privacy-preserving federated learning framework within a series of controlled experiments. Different learning configurations were evaluated to study model performance, adaptability, and privacy guarantees. The results supported the effectiveness of federated learning in context of ideal and heterogeneous data. Encryption techniques provided very meaningful dimension of protection against privacy threats. Selective encryption yielded a balanced solution between security and efficiency. All experimental results thus confirmed the practical possibility of the approach proposed.

General Conclusion

The primary objective of this study was to develop a privacy-preserving federated learning framework that allows collaborative model training without exposing sensitive data. Our motivation stems from the increasing need to apply artificial intelligence to domains such as healthcare, where data privacy and regulatory compliance are critical. A detailed investigation into the foundations of artificial intelligence and federated learning provided the necessary background to understand how data can remain decentralized while still contributing to global model improvement. In parallel, we explored the major privacy threats inherent in federated learning, with a particular focus on gradient-based attacks that can reconstruct client data. We reviewed several privacy-preserving techniques, including differential privacy, secure multiparty computation, and homomorphic encryption, the last of which emerged as a strong candidate for securing model updates without disrupting learning dynamics.

Building on this theoretical foundation, we introduced a federated learning framework that integrates homomorphic encryption into the training process. Our design incorporates two levels of encryption: full encryption of all model parameters, and a selective encryption strategy where only the most sensitive parts of the model are protected. The latter was guided by an analysis of model structure and gradient sensitivity, enabling the framework to focus encryption resources where privacy risks are highest. This selective approach reduced computational load while maintaining effective protection against potential data leakage. The model design was tailored to operate efficiently within federated environments while supporting encrypted training flows between clients and the server.

To evaluate our proposed framework, we conducted a series of experiments using both centralized and federated learning configurations under different data distributions. The results confirmed that federated learning can perform reliably in both uniform and non-uniform data settings. The encrypted configurations, particularly those using selective encryption, maintained high accuracy while significantly strengthening resistance to privacy attacks. A simulated gradient leakage attack demonstrated the ability of the framework to prevent reconstruction of client data, validating its privacy-preserving capabilities in practical conditions.

This thesis confirms that it is possible to design and implement a federated learning framework that effectively balances data privacy and model performance. The proposed system demonstrates a viable path toward deploying secure machine learning solutions in sensitive domains. While the encryption logic was integrated manually into the federated pipeline, this effort highlights the importance of developing native support for encryption within federated learning infrastructures.

Future research may explore the following directions:

- **Advancing the selective encryption strategy** by incorporating more dynamic and intelligent selection mechanisms that consider contextual factors such as training progress, model convergence behavior, or data heterogeneity across clients. This could improve both security and computational efficiency in diverse deployment scenarios.

- **Integrating hybrid privacy-preserving approaches** that combine homomorphic encryption for critical model components with differential privacy to add formal noise-based guarantees, offering layered protection against various attack types.
- **Developing adaptive encryption schemes** that adjust encryption scope or intensity in real-time, depending on risk indicators, sensitivity scoring, or system resource constraints.
- **Validating the framework on real-world datasets** from healthcare, finance, or other privacy-sensitive sectors to assess generalizability, robustness, and practical deployment feasibility.

These directions represent promising extensions to strengthen the effectiveness, adaptability, and scalability of privacy-preserving federated learning systems.

Bibliography

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

[2] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, United States, 4th edition, 2021.

[3] Dimple Patil, Nitin Liladhar Rane, Pravin Desai, and Jayesh Rane. Machine learning and deep learning: Methods, techniques, applications, challenges, and future research opportunities. In *Trustworthy Artificial Intelligence in Industry and Society*, pages 28–81. Deep Science Publishing, 2024.

[4] Batta Mahesh et al. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020.

[5] Arpana Chaturvedi and Nitish Pathak. Introduction and types of machine learning. In *Artificial Intelligence and their Applications*, volume 3 of *IIP Series*, pages 151–186. NBN Uurch, April 2024.

[6] Mohd Noor Mohd Halim and Olalekan Ige Ayokunle. A survey on state-of-the-art deep learning applications and challenges. *arXiv preprint arXiv:2403.17561*, 2025.

[7] Sawsan Abdulrahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, 2021.

[8] Samuel Acheme Glory Nosawaru Edegbe. A systematic review of centralized and decentralized machine learning models: Security concerns, defenses and future directions. *NIPES - Journal of Science and Technology Research*, 6(4), January 2025.

[9] Peter Kairouz and McMahan et al. Advances and open problems in federated learning, 2019.

[10] Hao Guan, Pew-Thian Yap, Andrea Bozoki, and Mingxia Liu. Federated learning for medical image analysis: A survey, 2024.

[11] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

[12] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization, 2021.

[13] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications, 2019.

[15] Dashan Gao, Ce Ju, Xiguang Wei, Yang Liu, Tianjian Chen, and Qiang Yang. Hhhfl: Hierarchical heterogeneous horizontal federated learning for electroencephalography, 2020.

[16] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. Privacy preserving vertical federated learning for tree-based models, 2020. [CrossRef].

[17] Sudipan Saha and Tahir Ahmad. Federated transfer learning: concept and applications, 2021.

[18] Qiongxiu Li, Wenrui Yu, Yufei Xia, and Jun Pang. From centralized to decentralized federated learning: Theoretical insights, privacy preservation, and robustness challenges, 2025.

[19] Hao Ye, Le Liang, and Geoffrey Ye Li. Decentralized federated learning with unreliable communications. *IEEE Journal of Selected Topics in Signal Processing*, 16(3):487–500, 2022.

[20] István Hegedundefineds, Gábor Danner, and Márk Jelasity. Gossip learning as a decentralized alternative to federated learning. In *Distributed Applications and Interoperable Systems: 19th IFIP WG 6.1 International Conference, DAIS 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings*, page 74–90, Berlin, Heidelberg, 2019. Springer-Verlag.

[21] Youyang Qu, Longxiang Gao, Tom Hao Luan, Yong Xiang, Shui Yu, Bai Li, and Gavin Zheng. Decentralized privacy using blockchain-enabled federated learning in fog computing. *IEEE Internet of Things Journal*, PP:1–1, 03 2020.

[22] Daoyuan Chen, Dawei Gao, Yuexiang Xie, Xuchen Pan, Zitao Li, Yaliang Li, Bolin Ding, and Jingren Zhou. Fs-real: Towards real-world cross-device federated learning, 2023.

[23] Chao Huang, Jianwei Huang, and Xin Liu. Cross-silo federated learning: Challenges and opportunities, 2022.

[24] Mohamad Arafeh, Ahmad Hammoud, Hadi Otrok, Azzam Mourad, Chamseddine Talhi, and Zbigniew Dziong. Independent and identically distributed (iid) data assessment in federated learning. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 293–298, 2022.

[25] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020.

[26] Sawsan Abdulrahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, 2021.

[27] Wenyuan Xu, Weiwei Fang, Yi Ding, Meixia Zou, and Naixue Xiong. Accelerating federated learning for iot in big data analytics with pruning, quantization and selective updating. *IEEE Access*, 9:38457–38466, 2021.

[28] Yujia Wang, Lu Lin, and Jinghui Chen. Communication-efficient adaptive federated learning, 2023.

[29] Yaqian Qi, Yuan Feng, Xiangxiang Wang, Hanzhe Li, and Jingxiao Tian. Leveraging federated learning and edge computing for recommendation systems within cloud computing networks, 2024.

[30] Shanfeng Huang, Zezhong Zhang, Shuai Wang, Rui Wang, and Kaibin Huang. Accelerating federated edge learning via topology optimization, 2022.

[31] Marzyeh Ghassemi, Tristan Naumann, Peter Schulam, Andrew L Beam, Irene Y Chen, and Rajesh Ranganath. A review of challenges and opportunities in machine learning for health. *AMIA Joint Summits on Translational Science Proceedings*, 2020:191–200, May 30 2020.

[32] Md Shahin Ali, Md Manjurul Ahsan, Lamia Tasnim, Sadia Afrin, Koushik Biswas, Md Maruf Hossain, Md Mahfuz Ahmed, Ronok Hashan, Md Khairul Islam, and Shivakumar Raman. Federated learning in healthcare: Model misconducts, security, challenges, applications, and future research directions – a systematic review, 2024.

[33] Rodolfo Stoffel Antunes, Cristiano André da Costa, Arne Küderle, Imrana Abdullahi Yari, and Björn Eskofier. Federated learning for healthcare: Systematic review and architecture proposal. *ACM Trans. Intell. Syst. Technol.*, 13(4), May 2022.

[34] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

[35] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14:513–535, 11 2022.

[36] Zhihua Cui, Xianghua Xu, Fei XUE, Xingjuan Cai, Yang Cao, Wensheng Zhang, and Jinjun Chen. Personalized recommendation system based on collaborative filtering for iot scenarios. *IEEE Transactions on Services Computing*, 13(4):685–695, 2020.

[37] Guanyu Lin, Feng Liang, Weike Pan, and Zhong Ming. Fedrec: Federated recommendation with explicit feedback. *IEEE Intelligent Systems*, 36(5):21–30, 2021.

[38] Zhiyong Jie, Shuhong Chen, Junqiu Lai, Muhammad Arif, and Zongyuan He. Personalized federated recommendation system with historical parameter clustering. *Journal of Ambient Intelligence and Humanized Computing*, 14, 02 2022.

[39] Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor. Fedfast: Going beyond average for faster training of federated recommender systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1234–1242, New York, NY, USA, 2020. Association for Computing Machinery.

[40] Chong Zhang, Xiao Liu, Xi Zheng, Rui Li, and Huai Liu. Fenghuolun: A federated learning based edge computing platform for cyber-physical systems. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–4, 2020.

[41] Qiong Wu, Kaiwen He, and Xu Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Computer Graphics and Applications*, PP:1–1, 05 2020.

[42] Younas Khan, David Sánchez, and Josep Domingo-Ferrer. Federated learning-based natural language processing: a systematic literature review. *Artificial Intelligence Review*, 57:1–39, 10 2024.

[43] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions, 2018.

[44] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction, 2019.

[45] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. Fedbert : When federated learning meets pre-training. *ACM Transactions on Intelligent Systems and Technology*, 13, 02 2022.

[46] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Comput. Surv.*, 54(6), July 2021.

[47] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706, 2019.

[48] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning, 2020.

[49] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 14(6):2073–2089, 2021.

[50] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Eavesdrop the composition proportion of training labels in federated learning, 2019.

[51] Ruihan Wu, Xiangyu Chen, Chuan Guo, and Kilian Q. Weinberger. Learning to invert: Simple adaptive attacks for gradient inversion in federated learning, 2023.

[52] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 739–753, 2019.

[53] Mengkai Song, Zhibo Wang, Zhifei Zhang, Yang Song, Qian Wang, Ju Ren, and Hairong Qi. Analyzing user-level privacy attack against federated learning. *IEEE Journal on Selected Areas in Communications*, 38(10):2430–2444, 2020.

[54] Xiaoyun Xu, Jingzheng Wu, Mutian Yang, Tianyue Luo, Xu Duan, Weiheng Li, Yanjun Wu, and Bin Wu. Information leakage by model weights on federated learning. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, PPMLP'20, page 31–36, New York, NY, USA, 2020. Association for Computing Machinery.

[55] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[56] Dimitar Iliev Dimitrov, Mislav Balunovic, Nikola Konstantinov, and Martin Vechev. Data leakage in federated averaging. *Transactions on Machine Learning Research*, 2022.

[57] Zhibo Wang, Zhiwei Chang, Jiahui Hu, Xiaoyi Pang, Jiacheng Du, Yongle Chen, and Kui Ren. Breaking secure aggregation: Label leakage from aggregated gradients in federated learning, 2024.

[58] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[59] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 14(6):2073–2089, 2021.

[60] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2017.

[61] David Enthoven and Zaid Al-Ars. An overview of federated deep learning privacy attacks and defensive strategies, 2020.

[62] Junpeng Zhang, Hui Zhu, Fengwei Wang, Jiaqi Zhao, Qi Xu, and Hui Li. Security and privacy threats to federated learning: Issues, methods, and challenges. *Security and Communication Networks*, 2022:2886795, 2022.

[63] Li Bai, Haibo Hu, Qingqing Ye, Haoyang Li, Leixia Wang, and Jianliang Xu. Membership inference attacks and defenses in federated learning: A survey. *ACM Comput. Surv.*, 57(4), December 2024.

[64] Huiqiang Chen, Tianqing Zhu, Tao Zhang, Wanlei Zhou, and Philip S. Yu. Privacy and fairness in federated learning: On the perspective of tradeoff. *ACM Comput. Surv.*, 56(2), September 2023.

[65] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 619–633, New York, NY, USA, 2018. Association for Computing Machinery.

[66] Raouf Kerkouche, Gergely Ács, and Mario Fritz. Client-specific property inference against secure aggregation in federated learning. In *Proceedings of the 22nd Workshop on Privacy in the Electronic Society*, WPES '23, page 45–60, New York, NY, USA, 2023. Association for Computing Machinery.

[67] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[68] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 603–618, New York, NY, USA, 2017. Association for Computing Machinery.

[69] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 06 2014.

[70] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 2512–2520. IEEE, 2019.

[71] Pengrui Liu, Xiangrui Xu, and Wei Wang. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecurity*, 5(1), December 2022.

[72] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[73] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients, 2020.

[74] Zihao Zhao, Mengen Luo, and Wenbo Ding. Deep leakage from model in federated learning, 2022.

[75] Joshua C. Zhao, Ahmed Roushdy Elkordy, Atul Sharma, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. The resource problem of using linear layer leakage attack in federated learning, 2023.

[76] Wang Xin, Li Jiaqian, Ding Xueshuang, Zhang Haoji, and Sun Lianshan. A survey of differential privacy techniques for federated learning. *IEEE Access*, 2024.

[77] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[78] Yehuda Lindell. Secure multiparty computation. *Commun. ACM*, 64(1):86–96, December 2020.

[79] Fengxia Liu, Zhiming Zheng, Yexuan Shi, Yongxin Tong, and Yi Zhang. A survey on federated learning: a perspective from multi-party computation. *Front. Comput. Sci.*, 18(1), February 2024.

[80] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), July 2018.

[81] Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP J. Multimed. Inf. Secur.*, 2007:1–10, 2007.

[82] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[83] Haokun Fang and Quan Qian. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, 13(4), 2021.

[84] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[85] Zhiqiang Wang, Xinyue Yu, Qianli Huang, and Yongguang Gong. An adaptive differential privacy method based on federated learning, 2024.

[86] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy, 2020.

[87] Wu Fan and Gao Maoting. Federated learning algorithm based on gaussi-an local differential noise. In Zeng Nianyin and Ram Bilas Pachori, editors, *Proceedings of 2024 International Conference on Machine Learning and Intelligent Computing*, volume 245 of *Proceedings of Machine Learning Research*, pages 325–339. PMLR, 26–28 Apr 2024.

[88] Huafei Zhu, Zengxiang Li, Mervyn Cheah, and Rick Siow Mong Goh. Privacy-preserving weighted federated learning within oracle-aided mpc framework, 2020.

[89] Yong Li, Yipeng Zhou, Alireza Jolfaei, Dongjin Yu, Gaochao Xu, and Xi Zheng. Privacy-preserving federated learning framework based on chained secure multiparty computing. *IEEE Internet of Things Journal*, 8(8):6178–6186, 2021.

[90] Renuga Kanagavelu, Qingsong Wei, Zengxiang Li, Haibin Zhang, Juniarto Samsudin, Yechao Yang, Rick Siow Mong Goh, and Shangguang Wang. Ce-fed: Communication efficient multi-party computation enabled federated learning. *Array*, 15:100207, 2022.

[91] Lvjun Chen, Di Xiao, Xiangli Xiao, and Yushu Zhang. Secure and efficient federated learning via novel authenticable multi-party computation and compressed sensing. *IEEE Transactions on Information Forensics and Security*, 19:10141–10156, 2024.

[92] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: efficient homomorphic encryption for cross-silo federated learning. In *Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC'20, USA, 2020. USENIX Association.

[93] K. R. Jayaram, Archit Verma, Ashish Verma, Gegi Thomas, and Colin Sutcher-Shepard. Mystiko : Cloud-mediated, private, federated gradient descent, 2020.

[94] Jing Ma, Si-Ahmed Naas, Stephan Sigg, and Xixiang Lyu. Privacy-preserving federated learning based on multi-key homomorphic encryption. *International Journal of Intelligent Systems*, 37(9):5880–5901, January 2022.

[95] Weizhao Jin, Yuhang Yao, Shanshan Han, Jiajun Gu, Carlee Joe-Wong, Srivatsan Ravi, Salman Avestimehr, and Chaoyang He. Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system, 2024.

[96] Yao Pan, Zheng Chao, Wang He, Yang Jing, Li Hongjia, and Wang Liming. FedSHE: privacy preserving and efficient federated learning with adaptive segmented CKKS homomorphic encryption. *Cybersecurity*, 7(1), July 2024.

[97] Abdulkadir Korkmaz and Praveen Rao. A selective homomorphic encryption approach for faster privacy-preserving federated learning, 2025.

[98] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights Imaging*, 9(4):611–629, August 2018.

[99] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[100] Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank H. P. Fitzek, and Najwa Aaraj. Survey on fully homomorphic encryption, theory, and applications. *Proceedings of the IEEE*, 110(10):1572–1609, 2022.

[101] Google Research. Colaboratory. https://colab.research.google.com/. Accessed: 10-05-2025.

[102] Python Software Foundation. Welcome to python.org. https://www.python.org/. Accessed: 10-05-2025.

[103] PyTorch Contributors. Pytorch. https://pytorch.org/. Accessed: 10-05-2025.

[104] The NumPy Community. Numpy — a fundamental package for scientific computing with python. https://numpy.org/, 2025. Accessed: 10-05-2025.

[105] The Matplotlib Development Team. Matplotlib: Visualization with python. https://matplotlib.org/, 2025. Accessed: 10-05-2025.

[106] The Seaborn Developers. Seaborn: Statistical data visualization in python. https://seaborn.pydata.org/, 2025. Accessed: 10-05-2025.

[107] The Flower Authors. Flower: A friendly federated learning framework. https://flower.ai/, 2025. Accessed: 2025-04-05.

[108] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. Tenseal: A library for encrypted tensor operations using homomorphic encryption, 2021.

[109] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. MedMNIST v2 - a large-scale lightweight benchmark for 2D and 3D biomedical image classification. *Sci. Data*, 10(1):41, January 2023.

[110] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[111] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[112] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[113] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

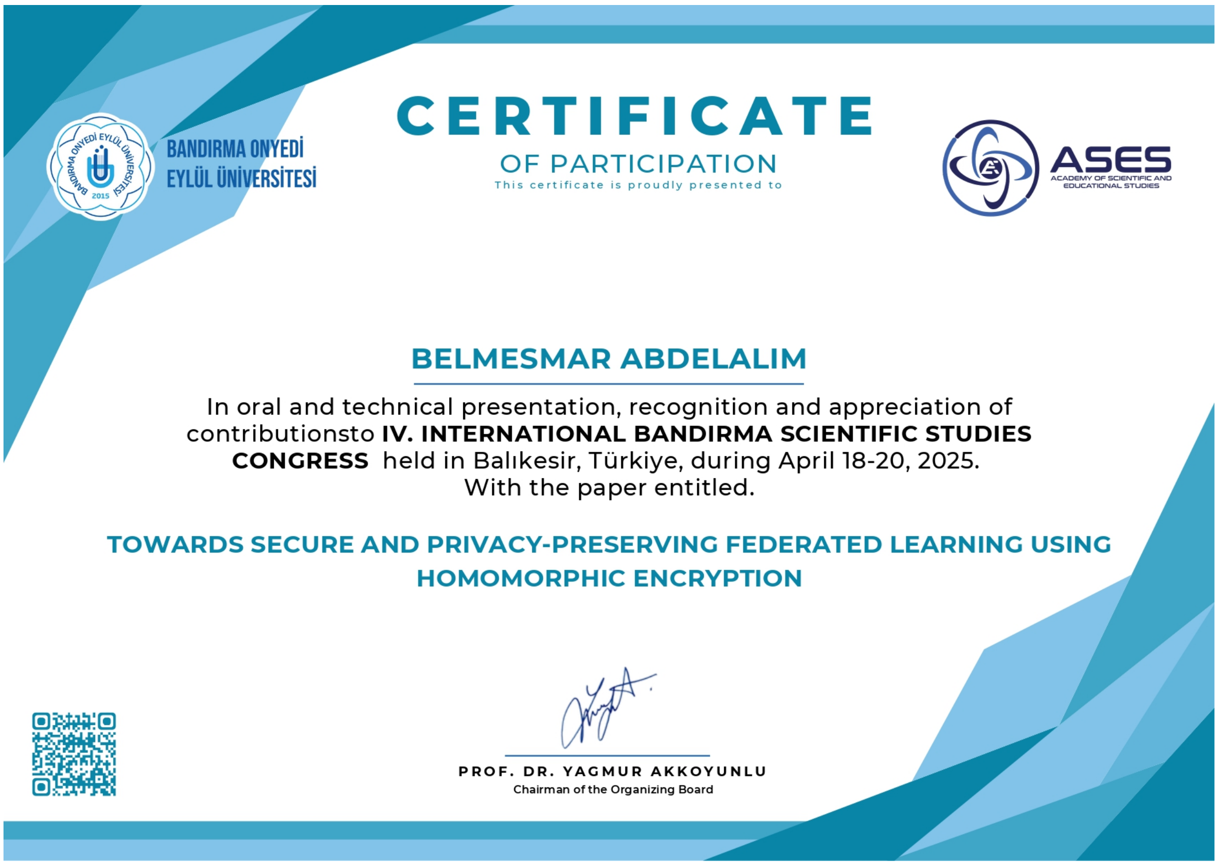
[114] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

International Conference

Acceptance certificates in this work

Our research, titled ”Towards Secure and Privacy-Preserving Federated Learning Using Homomorphic Encryption,” has been accepted for presentation at the IV. INTERNATIONAL BANDIRMA SCIENTIFIC STUDIES CONGRESS April 18-20, 2025, BALIKESIR, TURKIYE.

We present the certificates of participation in this conference:



BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ

CERTIFICATE

OF PARTICIPATION

This certificate is proudly presented to

ASES ACADEMY OF SCIENTIFIC AND EDUCATIONAL STUDIES

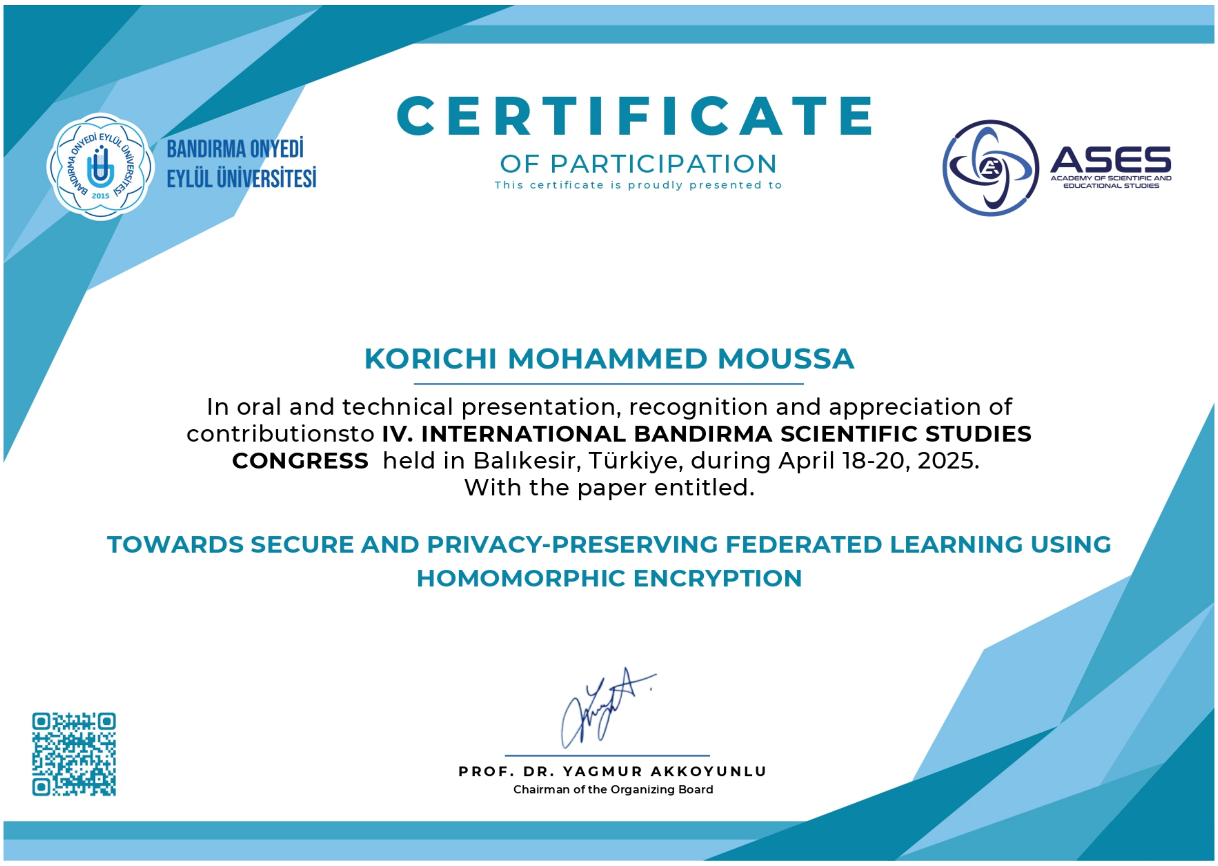
BELMESMAR ABDELALIM

In oral and technical presentation, recognition and appreciation of contributionsto **IV. INTERNATIONAL BANDIRMA SCIENTIFIC STUDIES CONGRESS** held in Balıkesir, Türkiye, during April 18-20, 2025.
With the paper entitled.

TOWARDS SECURE AND PRIVACY-PRESERVING FEDERATED LEARNING USING HOMOMORPHIC ENCRYPTION

[illegible]

PROF. DR. YAGMUR AKKOYUNLU
Chairman of the Organizing Board


BANDIRMA ONYEDİ EYLÜL ÜNİVERSİTESİ

CERTIFICATE

OF PARTICIPATION

This certificate is proudly presented to

ASES ACADEMY OF SCIENTIFIC AND EDUCATIONAL STUDIES

KORICHI MOHAMMED MOUSSA

In oral and technical presentation, recognition and appreciation of contributionsto **IV. INTERNATIONAL BANDIRMA SCIENTIFIC STUDIES CONGRESS** held in Balıkesir, Türkiye, during April 18-20, 2025. With the paper entitled.

TOWARDS SECURE AND PRIVACY-PRESERVING FEDERATED LEARNING USING HOMOMORPHIC ENCRYPTION

[illegible]

PROF. DR. YAGMUR AKKOYUNLU

Chairman of the Organizing Board