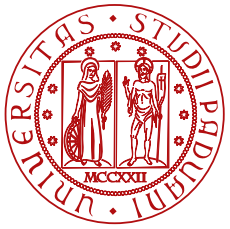


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in ICT for Internet and Multimedia

A Privacy-preserving and Communication-Efficient Federated Learning solution for Industrial Applications

Master Candidate

Mohammadreza Mohammadi

Student ID 2005535

Supervisors

Prof. Mauro Conti

University of Padova, Italy

Dott. Sima Sinaei

RISE Research Institute of Sweden AB, Sweden

Co-supervisor

Dott. Ehsan Nowroozi

Bahçeehir University, Turkey

Academic Year
2022/2023

To my Wife
and my Family

Abstract

There has been a lot of interest in privacy-preserving federated learning because of its potential to allow collaborative model training without compromising participants' privacy. When it comes to federated learning that respects users' privacy, this thesis examines a wide range of possible protection and attack tactics. First, I introduce the idea of privacy-protecting federated learning and discuss its structure, benefits, and drawbacks. Differential privacy, secure aggregation, and homomorphic encryption are only some of the defensive mechanisms I cover next to keep participants' information private. In addition, I look at the attack methods, such as membership inference and model inversion, that potentially jeopardize participants' privacy in privacy-preserving federated learning. I examine the result of model inversion attack and the measures taken to counter them. In this thesis, I consider three distinct industrial use cases from the DAIS project which will be used in real-world applications in a near future and implement a federated learning system for them while keeping in mind the need for privacy in federated learning environments. As a further step, I suggest a new client selection method based on each client's amount of data to improve the federated learning framework's accuracy and the efficacy of its communications. Also, I propose an innovative method, *Parameter Randomization*, to enhance the privacy and communication efficiency of federated learning systems. By introducing these two approaches, this thesis gives a thorough explanation of the field of privacy-preserving and communication-efficient federated learning and emphasizes the need for robust defense and mitigation mechanisms to protect participant privacy against attacks while keeping the accuracy of the models as high as possible.

Sommario

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

HE Homomorphic Encryption

SGD Stochastic Gradient Descent

SMC Secure multiparty computation

HFL Horizontal Federated Learning

VFL Vertical Federated Learning

FTL Federated Transfer Learning

DML Distributed Machine Learning

CSS Client Selection Strategy

FedAVg Federated Averaging

MI Model Inversion

DP Differential Privacy

DAIS Distributed Artificial Intelligent System

SER Speech Emotion Recognition

1 Introduction

As a result of the development of big data, the quantity of data is no longer the primary focus of our analysis. Protecting users' personal information is a pressing concern that must be addressed. Data leaks are never a minor issue, and recently, data security has received more public attention [1]. The defense of data security and privacy is being strengthened not just by individuals but also by society and organizations. As of May 25, 2018, the General Data Protection Rules (GDPR) [2] of the European Union came into force, with the stated goal of protecting the privacy and security of users' personal information. Operators must be upfront and honest with users about what they're agreeing to in terms of privacy, and they must never trick or pressure customers into giving up their rights. As new laws and regulations of varying degrees are introduced, they provide new challenges to the traditional data-processing approaches of artificial intelligence. In order to train models, data are essential because they are the foundation of AI. Yet, data is frequently organized in data islands, and a straightforward solution to this problem is to process the data centrally. In conventional machine learning, the performance and precision of models rely on the resources of a single, centralized server, including its processing speed and access to training data. In conclusion, traditional machine learning involves collecting user input, storing it in a centralized location, and then using it in training and testing to develop final machine learning models. Commonly used machine learning techniques that rely on centralized data storage and processing face several challenges, including issues related to computational power and time. However, the most significant concern is the lack of security and privacy of

users' data, which has often been overlooked. Federated Learning, as advocated by, has lately arisen as a technical answer to similar problems [3]. When it comes to sensitive data and heterogeneity, Federated Learning's ability to decentralize information from the server to end-devices while protecting user privacy is a major benefit. Federated learning has emerged due to two main reasons: firstly, the insufficient amount of data that can be stored on centralized servers due to direct access restrictions, which is in contrast to traditional machine learning approaches. Secondly, data privacy must be preserved; this is accomplished by keeping private information off the server and instead using it locally, at the edge, or at the client, where asynchronous network communication takes place. This approach enables the use of the benefits of artificial intelligence that are offered by machine learning models in various domains while preserving data privacy. Also, instead of relying on a central server, processing resources are dispersed among interested parties via iterative local model training on end-devices. One of the most rapidly growing areas of machine learning recently, federated learning takes a decentralized data method that promises to be in line with new regulations protecting user information. In addition to privacy, federated learning extends machine learning advantages to smaller domains when there is insufficient training data to develop a solo machine learning model [3]. To add, federated learning can let customers acquire a well-trained machine learning model without requiring them to provide any data that could be considered sensitive to a centralized server. In intelligent IoT networks, for instance, a large number of IoT devices may function as workers to connect with a server to carry out neural network training. To be more precise, the aggregator initiates a global model with initial learning parameters. The aggregator provides the current model, and each worker updates the model based on its own dataset locally using an appropriate method such as stochastic gradient descent (SGD), before sending the result back to the aggregator. After collecting all of the updates to individual models, the aggregator may construct a revised and enhanced global one. By using the computational resources of distributed workers, the aggregator has the potential to enhance training quality while reducing privacy breaches for individual users. In the end, the local employees get the global update from the aggregator and figure out their next local update until the global training is finished [4]. Google introduced the concept of Federated Learning in 2016 [5], when they first used Google Keyboard to cooperatively learn from several Android phones [24]. As federated learning can be implemented on any

edge device, it might revolutionize crucial industries such as healthcare, smart homes, and more. The most prominent instance of this is the worldwide effort by researchers and doctors to develop a computer-based pandemic engine for the identification of COVID-19 using chest x-rays. In transportation networks, the automobiles might be programmed for autonomous driving and city route planning, which is a fascinating prospect. Similarly, edge devices in different houses may work together to generate context-aware rules for smart-home apps leveraging a federated learning architecture [6].

1.1 Challenges

As federated learning is still in its infancy as a field of study, many academics from many different organisations are actively working to improve existing frameworks and ensure the privacy and security of user data inside federated learning. The introduction of a new technology and ecosystem often brings about various technological ripple effects over time. Despite its promising features, the implementation of federated learning requires extensive investigation to ensure its security and privacy, as potential challenges may arise. So, we may wonder what current and future security and privacy issues may arise as a consequence of this technology's widespread usage [3]. Some obstacles in federated learning must be overcome in order to properly secure the privacy of organizations and consumers. Along with security and privacy, communication overhead is a key barrier for the reasons listed below. Firstly, as federated learning is taught iteratively, the amount of time it takes to learn, or the convergence time, is influenced by both the optimization technique, such as the number of training steps, and the federated learning parameter transmission delay per training step. Second, millions of edge devices may perform federated learning training, and each of those devices will need to repeatedly share the federated learning parameters for its big dataset with a centralised server. As a consequence, the time it takes for devices to train their local machine learning models may be much longer than the time it takes for federated learning to transmit parameters across a realistic network with limited computation and communication capabilities. Hence, to train massive machine learning models across millions of edge devices, we need a communication-efficient federated learning framework that can significantly improve both convergence time and model accuracy. [7].

In summary, main challenges of federated learning can be categorized as follows [1] [7]:

- **Privacy protection**: Given that federated learning is advocated as a way of protecting user privacy when it comes to machine learning, it is necessary to guarantee that the federated learning training model does not leak sensitive user data.
- **Inadequate data**: Nevertheless, in a dispersed setting, the data available on each mobile device may not be enough to train a model with good performance, contrary to the requirements of classical machine learning. However it might be pricey to gather all data in one place. As a result, for federated learning to work, each device must collect data locally and train its own model, which must then be uploaded to the server and merged with the models of other devices to produce a single, global model.
- **Statistical heterogeneity**: There are numerous edge devices in a federated setting, and the information they store may not be personally identifiable (Non-Independent and Identically Distributed). This suggests that there may be variations in the data distributions or features between devices. Non-IID datasets might be difficult to use for model training because of issues with data format homogeneity, such as those that arise when electronic medical records for various diseases are stored in separate places.
- **Communication overhead**: Both the huge number of parameters in complicated models and the frequency with which clients and servers share these parameters contribute to the communication cost in federated learning. Due to the increased quantity of data that must be sent between the clients and the server, this may lead to high communication costs and extended training durations.
- **Device hardware heterogeneity**: the fact that devices in a federated learning system may have different hardware configurations and processing capabilities. This can lead to performance variations and compatibility issues, as some devices may not be able to execute certain tasks or may take longer to perform them. This heterogeneity must be taken into account when designing federated learning algorithms to ensure that the system is robust and can operate effectively across a diverse range of devices.

1.2 Problem Definition

Federated learning is commonly categorized as either vertical federated learning (VFL) or horizontal federated learning (HFL). In HFL, all training data remains on each user's local device, and the server and clients work together to develop a globally integrated model. In a standard supervised learning scenario, HFL aims to identify the global model parameter that minimizes the loss

function $L(\theta)$, while each connected local client k seeks to identify the local model parameter that minimizes the loss function $L_k(\theta)$. However, in HFL, the training data (x_k, y_k) across multiple users k is non-independent and identically distributed (Non-IID), which makes optimizing HFL more challenging. In federated learning, the training data is not independent and identically distributed (IID), which is in contrast to conventional machine learning and distributed learning. Another need of federated learning is a constant two-way flow of data between the server and each client, including the global model and local updated models for each client k. This indicates that the model size is proportional to the federated learning communication costs, making it a primary research topic to find ways to shrink models without sacrificing performance.
The training process for VFL is different from that of HFL, and it often requires two kinds of clients: a "guest" client who has access to all training data labels, and a "host" client who does not. Typically, a VFL system will consist of a single guest client and many host clients. Each host client k generates an intermediate result, z_k, which is sent to the guest client so that the global loss function may be constructed. In HFL, the guest client serves as the server and the local model parameters k are transferred between the guest and host clients in place of the intermediate output z_k. To complicate matters further, in practice, training data labels are often shared between a number of clients rather than a single one. It calls for a rethinking and rebuilding of VFL learning algorithms and their accompanying safety precautions.

Our focus in this project was on the HFL scenario, where we hoped to make a difference in federated learning's security and efficacy via better communication. This work utilized differential privacy and homomorphic encryption techniques to create a federated learning system that protects users' privacy, and a novel approach called *Parameter Randomization* has been created to improve the system's communication efficiency. In addition, this thesis investigated a client selection strategy to diminish the side effects of differential privacy approach, e.g. having a private system with high accuracy.

1.3 Contributions

My whole Master's thesis project's goal was to build a communication-efficient and privacy-preserving federated learning system for some real-world applications, thus it primarily consists of two parts: one is cutting communica-

tion costs, and the other is improving the federated learning system's degree of privacy. In the following, you can find the contributions of this work.

- I have worked on three different Industrial use-cases of DAIS project [8], including Speech Emotion Recognition (SER), Driver's Fatigue Detection (DFD) and Smart Grid System Anomaly Detection (SGSAD). DAIS (Distributed Artificial Intelligent System) is a pan-European initiative that develops edge AI software and hardware components to provide quicker, more secure, and energy-efficient data processing solutions. DAIS' aim is to enable trustworthy connectivity and interoperability by combining the Internet of Things with Artificial Intelligence in a distributed edge system for industrial applications. The project will include a wide range of industry-driven use cases integrated into the application areas of Digital Life, Digital Industry, and Smart Mobility.

- For federated learning to be successful, it must first and foremost safeguard personal information, homomorphic encryption (HE) and differential privacy are two of the most important tools for providing privacy in the federated learning systems. In this research, I sought to apply one of these methods along with my proposed ideas to the above-mentioned use-cases and analyze how changing parameters and methods affected the federated learning system's convergence, accuracy, and confidentiality. For each use case, I implemented a specific strategy to fit its specific conditions and characteristics.

- Depending on how much noise is introduced, an accuracy drop is usual when we apply differential privacy to the client's model parameters. In order to prevent accuracy loss, I have examined the impact of client selection strategy (CSS) based on the volume of data provided by each client in each training round to increase the federated learning system's accuracy and decrease the standard deviation of additive noise brought on by the aggregation process.

- Federated learning requires users to exchange their local training model parameters rather than real data in order to protect user privacy. In spite of this, new research suggests that there are still privacy concerns with federated learning. This is because the parameters submitted by each participant in the initial training dataset might allow attackers to partly reveal each participant's training data [3]. These serious dangers in federated learning may be categorized into a variety of inference-based attacks. In this thesis, I considered Model Inversion Attack (MI-Attack) [9] to measure the privacy of federated learning systems before and after applying privacy-preserving approaches (differential privacy and homomorphic encryption). Then, I tried to diminish the side effects of those approaches by proposing the client selection strategy (CSS) and parameter randomization methods.

- In a basic federated learning implementation, every client must provide a complete model back to the server after every round. Due to a number of variables, this stage is probably the bottleneck of federated learning for big

models [10]. In order to address this problem, an innovative method called *Parameter randomization* has been proposed. Depending on its various settings, this method may significantly decrease the size of the shared model parameters and also it may be effective for increasing the degree of privacy of federated learning participants by allowing them to not share their whole parameter space with a central server.

1.4 Thesis Organization

In Chapter 2, the preliminaries concept of federated learning, the aggregation algorithms, attacks categories, and defenses strategies will be discussed. Then, in Chapter 3, the proposed methods, *client selection strategy* and *parameter randomization*, will be explained and different federated learning models in addition to their experimental results will be reported. Finally, in Chapter 4, you can find some content about possible future works and conclusions of this thesis.

2 Background

2.1 Chapter Overview

In this chapter, some brief descriptions of the preliminaries and background of Federated Learning will be discussed. At first, Distributed Machine Learning (DML) which is a root of federated learning will be described. Then, Horizontal federated learning (HFL), Vertical federated learning (VFL) and Federated Transfer Learning (TFL) will be introduced which are the most used types of federated learning. Additionally, different parameters aggregation approaches of federated learning server are explained. Then, some famous privacy and security attacks on federated learning will be introduced. Consequently, privacy-preserving defense algorithms and security-enhancement strategies for federated learning will be described. Afterwards, you can find some information about improving communication-efficiency of federated learning which is a bottleneck for large models and plays a key role in industrial IoT applications. At the end, a set applications of federated learning will be represented to give a better insight to us about the conditions that federated learning is able to solve the problems effectively.

2.2 Distributed Machine Learning (DML)

During the last several years, machine learning techniques have seen widespread deployment in more sophisticated uses. Despite the emergence of several dif-

ferent strategies and technologies, the nature of the data models utilized is quite similar. Basic operations on vectors, matrices, or tensors comprise the majority of processing in machine learning tasks. In the huge computational sector, the desire to improve such procedures has been a very active topic of study for decades. Similar to other huge computational problems, there are two general, distinct, and complementary techniques to speed up workloads: Vertical scaling (scaling up), in which more resources are added to an existing node, and horizontal scaling (scaling out), in which more nodes are added to the system, are the two forms of scaling [11].

2.2.1 Vertical Scaling

The present surge of attention to scaling up machine learning models may be ascribed in part to the emergence of devices and programming frameworks that make it simple to harness the many forms of parallelism possible in many learning algorithms. Concurrent computation of data samples or their features is made simple by a variety of systems. Many learning methods that see data as an arbitrary batch of instances and aggregate separate calculations over each of them may now be easily parallelized. The proliferation of extremely big datasets in many current applications has also heightened interest in large-scale machine learning. Such datasets are frequently amassed on distributed database systems, which motivates the introduction of learning algorithms that can be dispersed effectively. Finally, the expansion of various sensors capable of performing real-time prediction based on high-dimensional, complex feature representations increases the requirement for parallel computing in machine learning/deep learning applications. Speech recognition and optical object identification, for example, are increasingly widespread in autonomous robotics and portable devices. The profusion of distributed system solutions gives a variety of possibilities for applying machine learning/deep learning models in order to increase performance or the capacity to analyze extremely big datasets. Configurable integrated circuits (e.g. FPGAs), GPUs, High-Performance Computing (HPC) clusters linked by Ethernet/Internet, and virtual clusters rented from cloud computing vendors are among these options [12].

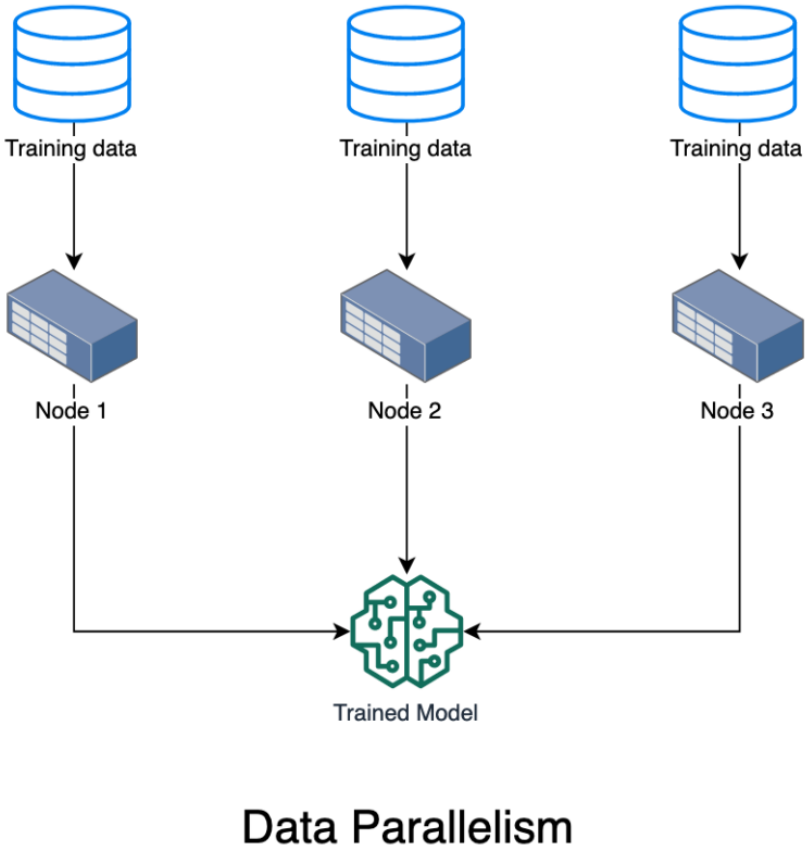


Figure 2.1: Data Parallelism (Vertical Scaling) [13]

2.2.2 Horizontal Scaling

Although there are numerous ways to enhance the processing capacity of a single device for large-scale machine learning and deep learning tasks, there are reasons to consider a scale-out architecture or a hybrid approach. One argument in favor of using equipment is that it is more cost-effective, both in terms of the initial investment and ongoing maintenance. Another advantage is that equipment is more resilient to failure, as a single failure within a high-performance computing (HPC) program can be mitigated through partial recovery. Additionally, using multiple devices results in increased cumulative I/O capacity compared to a single device.[14]. Training machine learning algorithms requires a lot of data, which can become a bottleneck for system performance.

One way to reduce the cost of input/output (I/O) on system performance is by using horizontal scaling, which involves parallelizing reads and writes across multiple servers, each with its own specialized I/O subsystem. Nevertheless, not all machine learning algorithms work well in a distributed environment, which might make horizontal scaling difficult. Methods like this work well for highly parallel algorithms [11].

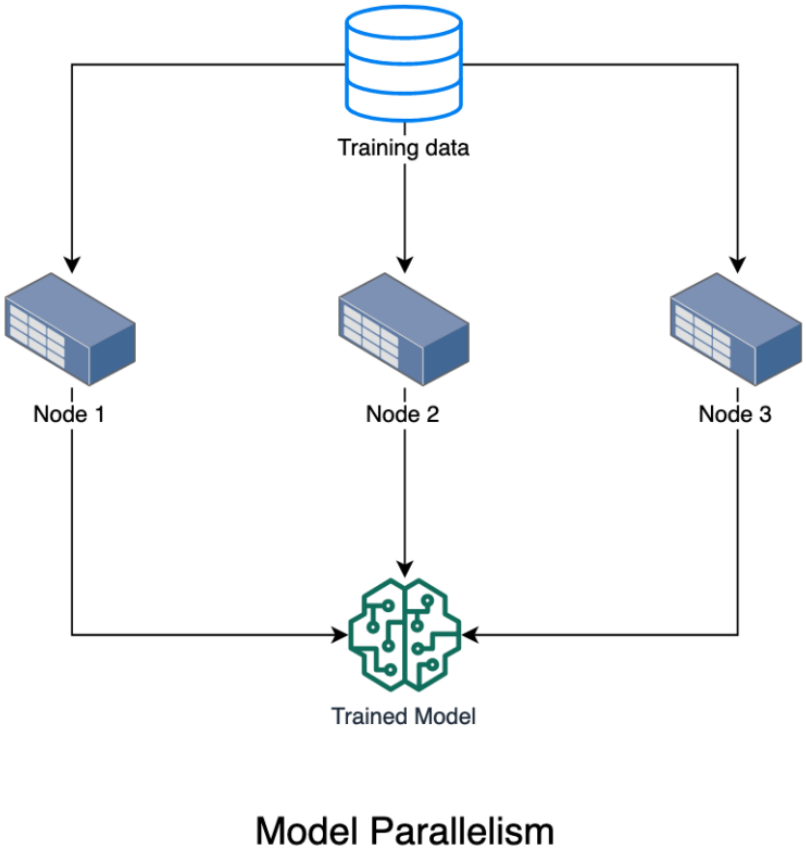


Figure 2.2: Model Parallelism (Horizontal Scaling) [13]

2.3 Federated Learning

Significant advances have been made in areas like natural language processing as a result of using deep learning to extract information from digital data.

Due in part to the availability of incredibly huge data sources for learning, recent remarkable achievements in deep learning have become possible. So, there is great promise in leveraging sensor data to train and fine-tune deep learning models. Concurrently, many customers worry about their personal information being shared without their consent. A lot of incidents of privacy leakage and exploitation have proved that centralized data processing puts people's privacy at risk. Because Sensor nodes regularly gather data in private spaces, frequently without the owners' express knowledge, these issues are notably valid. As a result, sharing this information with a centralized node that might develop a deep learning model is often not a choice. In other cases, local information processing may be advantageous for purposes such as enhanced privacy of the local entity. Concerns have been raised about our ability to train machine learning and deep learning networks using the data collected by millions of Sensor nodes if that data cannot be transported to a centralized repository [15]?
By decentralising data processing from a central node to end-devices and protecting privacy at the same time, federated learning allows artificial intelligence to thrive in scenarios involving sensitive information and heterogeneous data. Two key considerations led to the creation of this framework: (1) the lack of data that can be transferred to a central server, as in traditional machine learning, owing to restrictions like GDPR that ban direct access to user data, and (2) the use of local data from edge devices (clients) to preserve data privacy, rather than transmitting sensitive information to the central server, where network asynchronous communication is leveraged. Ensuring data privacy provides several benefits for the successful implementation of AI using machine learning models across various industries. Additionally, by conducting multiple local model training on local nodes, computing capacity is shared among all involved parties, rather than relying solely on a centralized server. Moreover, federated learning safeguards privacy while extending the reach of machine learning into new areas where there is insufficient training data to create a machine learning model. [3].
Figure 2.3 depicts a simplified four-step approach for delivering this kind of privacy-preserving collaborative learning. The initial stage is for all clients to get the initial global model instance W from the central server. The clients then use stochastic gradient descent (SGD) to enhance the received model according to their own local training data. Finally, all collaborating clients transfer their locally enhanced models W_i back to the server, where they are aggregated to

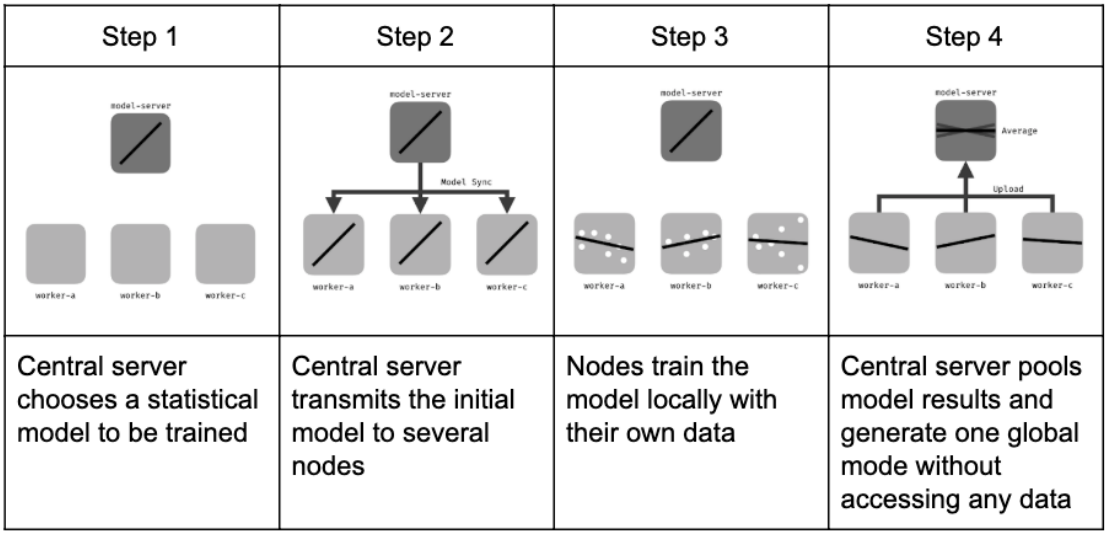


Figure 2.3: Federated Learning Steps [16]

produce a new global model (in reality, weight updates $W = W_{new} - W_{old}$ may be sent instead of whole models W, which is identical as long as all clients stay synchronized). This process is continued until a specified convergence requirement is met. It is important to note that while using this approach, training data never leaves the local devices since only model changes are exchanged [15].

Based on how data characteristics and samples are shared across participants, federated learning may be classified as either horizontally federated learning (HFL), vertically federated learning (VFL), or federated transfer learning (FTL) [17]. Based on how data is distributed among numerous clients in the feature and sample ID space, which may not be identical, we classify federated learning as HFL, VFL, or FTL. The several federated learning frameworks for a two-client setting are shown in 2.4.
In the following, you can find some variables that will be used to formulate different types of federated learning.

- D_i: Dataset of each participant i
- X: Feature space
- Y: Label Space
- I: The sample ID space
- (X, Y, I): A complete training dataset

Algorithm 1 Basic Federated Learning Algorithm

Input: Number of iterations: T, Total Number of clients: N, Local minibatch size: B, Initial global model: W_0^G
Output: federated learning trained model
Start:
The server broadcasts W_0^G and t = 1
for $t \leq T$ **do**
Client Side:
for all clients $i \in N$ **do**
for each batch $b \in B_i$ **do**
Compute gradient g (b) $\leftarrow \nabla_w L^i(w_t^i; b)$
Local model update $w_t^{i+1} \leftarrow w_t^i + \tilde{g}$
end for
end for
Server Side:
Update the global model parameters
$W_{t+1}^G \leftarrow \sum_{i=1}^{K} \frac{|D_i|}{|D|} W_{t+1}^i$
Broadcast updated global model to clients
end for

2.3.1 Horizontal Federated Learning

Horizontal federated learning(HFL), also referred to as sample-based federated learning, is utilized when datasets have a common feature space but differ in their sample space. (Figure 2.4 (a)). For instance, various hospitals may maintain comparable sorts of data on different patients (e.g., epidemiological, clinical, and genetic). If these entities decide to work together utilising federated learning to create a machine learning model, we call this kind of arrangement HFL. From a safety standpoint, a horizontal federated learning system needs authentic users and protection against a trustworthy but nosy server. This means that the only entity that may compromise users' anonymity is the server itself. Another security model that considers hostile users [19] was recently presented, creating further privacy problems. Users in the federated learning system are given access to the global model parameters after the convergence conditions have been satisfied [18].
In summary, HFL can be formulated as:

$$X_i = X_j, Y_i = Y_j, I_i \neq I_j, \forall D_i, D_j, i \neq j \tag{2.1}$$

Figure 2.5 depicts a conventional layout for a federated horizontal learning system. In this approach, k users with the same data type work together to create a machine learning/deep learning model using a centralized server. The training procedure is finished after all of the aforementioned four stages in Figure 2.5 have been repeated until the loss function converges. Under this paradigm, which is not tied to any specific machine-learning technique (SVM, CNN, etc.), the final set of trained model parameters will be shared across all users. In HFL, we consider all of the local clients are honest, however, we may have a honest-but-curious global server. With this assumption, no privacy leakage from any participant is expected [20].

2.3.2 Vertical Federated Learning

HFL application settings are restricted owing to practical considerations such as secrecy between organizations with conflicting interests [21]. On the other hand, VFL or feature-based federated learning is suitable (Figure 2.4 (b)) when there are a lot of commonalities between the entities involved in the sample space but significant differences between them in the feature space, i.e. when different people have varied preferences for the characteristics of the same entries. An instance of VFL can be a partnership between a bank and an e-commerce site to discover individuals' purchasing habits [21]. To create a model using data from both entities collectively, VFL (Vertical Federated Learning) is employed to combine diverse features from various entities and calculate the training loss and gradients in a manner that preserves privacy. This kind of federated method is known as federated learning because it allows all participants to build a "common wealth" approach while maintaining their identical identities and statuses. An honest but curious participant user is often considered in a vertical federated learning system. For instance, in a two-party situation, only one of the two parties is penetrated by an attacker, and the two parties do not have a collision. According to the security definition, the attacker can learn just about the compromised user and can learn nothing more about the other client than what is already revealed via input and output. After the training phase, each participant discards all other model parameters except those that are directly related to their personal characteristics. Hence, cooperation between the two is required to provide outcomes at the time of inference.

In summary, VFL can be formulated as:

$$X_i \neq X_j, Y_i \neq Y_j, I_i = I_j, \forall D_i, D_j, i \neq j \tag{2.2}$$

The architecture of a typical VFL paradigm illustrated in Figure 2.6. Users are assumed to not be able to directly share information between themselves. In order to protect the privacy of VFL system users, it is possible to include a third-party organization. However, it is possible to design a VFL system without the mentioned third-party entity.
VFL training has two main steps: 1-Entity Alignment and 2-Model Training.
Entity Alignment: The initial phase in a VFL system's collaborative training process is entity alignment. This is the process of mapping the entities (e.g., individuals or objects) in different organizations' datasets to a common set of identifiers. This is necessary for collaborative analysis because each organization may use different identifiers or naming conventions for the same entities. It requires careful consideration of data privacy and security issues, as well as robust algorithms for entity matching and record linkage. Once entity alignment is performed successfully, the organizations can proceed with joint analysis and machine learning tasks while maintaining the privacy of their respective datasets.
Model Training: After entity alignment in vertical federated learning, the model training phase involves training a machine learning model on the joint dataset while preserving the privacy of the individual datasets. This is typically done using cryptographic techniques such as secure multi-party computation or homomorphic encryption. During model training, the data is kept decentralized, and only model parameters are exchanged between the organizations. The goal is to jointly learn a model that can make accurate predictions without revealing sensitive information about the individual datasets. Once the model is trained, it can be used for the prediction of new data while preserving the privacy of the individual datasets. A generic VFL training method based on neural networks employing stochastic gradient descent [22] is described in Algorithm 2.

2.3.3 Federated Transfer Learning

Training data generated by numerous organisations sharing the same feature area is a limitation of traditional federated learning. It's not always the case in the

Algorithm 2 A typical VFL procedure

Input: Learning rates $\eta_1 and \eta_2$
Output: Model Parameters $\theta_1, \theta_2, ..., \theta_k, \psi_k$
Start:
Users $1, 2, ..., K$ initialize $\theta_1, \theta_2, ..., \theta_k, \psi_k$
for $t \leq T$ **do**
Randomly sample a mini-batch of size x
for clients $k \in K$ in parallel **do**
User k computes $H_k = g_k(x_k, \theta_k)$; {/*Computing local model outputs*/}
User k send $[H_k]$ to Active user K
end for
Active user K updates $\psi_K^{t+1} = \psi_K^t - \eta_1 \frac{\partial l}{\partial_{\psi K}}$ {/*Computing gradients of global module*/}
Active user K computes and sends $\frac{\partial l}{\partial H_k}$ to all other users {/*Computing gradients for each user*/}
for clients $k \in K$ in parallel **do**
User k computes $\nabla_{\theta_k} l = \frac{\partial l}{\partial_{\theta_k}}$ {/*Computing gradients of local module*/}
User k updates $\theta_k^{t+1} = \theta_k^t - \eta_2 \nabla_{\theta_k} l$ {/*Updating local model parameters*/}
end for
end for

actual world, especially in fields like finance and medicine. Federated Transfer Learning (FTL) was suggested to address this flaw [24], [25]. It's possible that educational psychology was where the idea of transfer learning first originated. C. H. Judd, a psychologist, proposed the generalization hypothesis of transfer, which states that gaining more life experience leads to learning abilities that may be used in other contexts. By extrapolating from one situation to another, one might get insight in how to handle similar situations. The precondition for transfer, according to this idea, is that there must be a relationship between two learning processes. Since both the bicycle and the motorcycle are transportation tools and may share some similar knowledge, in practice, someone who has learned the bicycle may learn the motorcycle quicker than others. Keep in mind that prior learning is not always helpful when applying it to new situations. If there are few shared characteristics across the domains, information sharing may not be effective. When it comes to language acquisition, for instance, practising cycling won't help us much [26]. When two datasets have dissimilar sample sizes and feature spaces, federated transfer learning may be used to bridge the gap between them [18]. As an example, think about two firms, one in Iran and the other in Italy, both of which operate in the healthcare and insurance industries.

Geographical constraints mean there is little overlap between the two groups' target audiences. Despite this, there is only a little overlap between the two companies in terms of feature space. Because of this, FTL is a great option for a federated system that can provide results across the full sample and feature space (Figure 2.4 (c)) [18].
In summary, FTL can be formulated as:

$$X_i \neq X_j, Y_i \neq Y_j, I_i \neq I_j, \forall D_i, D_j, i \neq j \tag{2.3}$$

The VFL architecture is restricted to datasets that overlap. However, we will now explore federated transfer learning, which extends the coverage to the entire sample space. Although the general architecture shown in Figure 2.6 remains unchanged, the manner in which users share intermediate computed gradients is modified, as depicted in Figure 2.7. The goal of federated transfer learning is to increase the precision with which predictions are made for labels in the target domain by developing a common representation of the distinctive features of numerous participants. Reducing the number of incorrect predictions is made possible by using labels provided by the source-domain party. In federated transfer learning, each participant gradient computations are handled independently, unlike in VFL. Yet, during the inference phase, it is necessary for both parties to independently calculate the prediction results [18].
After building the model, its performance can be assessed in real-world applications, and a long-term data recording method, like Blockchain, can be utilized to maintain its efficacy. The model's success is contingent on the contributions made by data providers, and those organizations that offer more data will experience greater benefits. Through federated processes, the effectiveness of these models is disseminated, thus incentivizing new organizations to join the data federation.

2.4 Federated Learning aggregation methods

The priorities and architecture of federated learning determine which algorithms are used to compute global model parameters from transferred local parameters updates of users to the global server in each round of training. Setting up this logic is essential since it handles users' heterogeneity, differences in updated sent parameters from each user, and communication problems. The two

fundamental aims of the suggested aggregation strategies, which we shall explore in this section, are optimum client selection and communication-efficiency. These mechanisms are utilized in various federated learning techniques for integrating, improving, optimizing, aggregating, and collaborating. In describing aggregation algorithms, our focus is on HFL setting, i.e. in each round the global server receives local models parameters and combines them to compute an aggregated set of parameters and send back them to clients.
There is a baseline aggregation algorithm, FedAvg, introduced by Google [27] on 2017 and there are various other algorithms proposed after FedAvg tried to optimize a factor (s) of it. FedProx [28], FedKD [29], FedGEMS [30], FedMD [31] and Scaffold [32] are some the famous aggregation algorithms which, in this work, FedAvg, FedProx and FedKD will be explained to provide a better insight of different optimizations of aggregation algorithms.

2.4.1 FedAvg

Google's federated learning implementation presented the Federated Averaging [27] technique (commonly known as FedAvg), which is based on the SGD optimizer [3]. In studies of federated learning, FedAvg serves as a reference point against which new approaches are evaluated. FedAvg controls training with a centralised server that calculates the shared global model weights w_t, where t is the communication round. However, local client training of the models is how the real optimization is carried out [33]. Three important parameters determine how much computing is done in FedAvg algorithm: 1- (C), the amount of customers that take part in each training epoch; 2- (E), number of training iterations performed by each client on their local dataset; and 3- (B), the local minibatch size that is utilized for client updates. To indicate that the whole local dataset is handled as a single minibatch, we write $B = \infty$. As a result, we may have $B = \infty$ and $E = 1$ as the special setting of this algorithm family, which is referred to FedSGD. The number of local updates each round for a client with n_k local data samples is given by $u_k = E\frac{n_k}{B}$; complete pseudo-code is provided in Algorithm 3.

2.4.2 FedProx

In the same way as FedAvg averages updates from a set of devices over time, FedProx conducts local updates on certain devices at each round. To guarantee

Algorithm 3 FedAvg Algorithm

Input: K: Number of users; B: the local minibatch size, E: the number of local epochs, and η: the learning rate
Output: Updated weights w
START:
Initialize w_0
for each $t = 1, 2, ...$ **do**
$m \leftarrow max(C.K, 1)$
$S_t \leftarrow$ random set of m users
for each user $k \in S_t$ in parallel **do**
$w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
end for
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for
END

ClientUpdate (k, w_t)**:**
START:
$B \leftarrow$ split local dataset D_k into batches of size B
for each local epoch e from 1 to E **do**
for batch $b \in B$ **do**
$w \leftarrow w - \eta \nabla l(w; b)$
end for
end for
return w to global server
END

convergence, however, FedProx includes the following seemingly minor adjustments that have led to significant experimental successes [28]. The first change is **Tolerating partial work**. End-nodes in federated systems may have varying degrees of processing power, network access, and battery life. Thus, it is not acceptable to require all nodes in FedAvg to do the same amount of work (i.e., local training for the same number of local epochs (E)). FedProx generalizes FedAvg by enabling various proportions of work to be completed locally across devices depending on their accessible resources and then aggregating the incomplete solutions delivered by the stragglers. Another important change in FedProx approach is **Proximal Term**. The negative effects of systems heterogeneity may be mitigated by permitting different local epochs to be done across users, however excessive local updates may (possibly) lead to method divergence as a result of the heterogeneous data. A proximal term was proposed by FedProx to be

added to the local subproblem in order to mitigate the effect of mutable local updates. In addition to the local function $F_k(.)$, each user also tries to minimise the following function h_k.:

$$\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2 \tag{2.4}$$

The proximal term has two positive aspects: In order to reduce statistical heterogeneity, (1) local updates are constrained to be more in line with the initial (global) model without the requirement for a user-specified number of local epochs. The system's heterogeneity is accommodated for, since it permits the safe absorption of varied amounts of local epochs [28].

2.4.3 FedKD (Federated Knowledge Distillation)

To describe this algorithm, Knowledge Distillation (KD) technique should be explained first. KD is a strategy for transferring knowledge from a big teacher model to a small student model [34], Model compression for better network efficiency is KD's primary focus. Considering N users who each keep their sensitive data on their own local machine. The dataset on the i-th user is referred to as D_i. According to FedKD, each user maintains a local copy of a shared student model S with a parameter set Θ^S and a huge local teacher model T_i with a parameter set Θ_i^T. The training of a machine learning model in a federated learning environment is carried out by a group of users working together, with the help of a central server. In the end, we want to have a model that is reliable and productive, one that can protect users' personal information while keeping down their communication costs. Under the guidance of the labelled local data and the knowledge distilled from one another, users in FedKD simultaneously calculate the update of the local teacher model and the student model for each global epoch of the FedKD algorithm. While various users exchange and collaborate to learn the student model, the teacher models are updated individually. The beneficial knowledge contained by the teacher model may be used to instruct the student model since the local teacher models have more complex structures than the student model. Also, the teacher model may also profit from the knowledge distilled from the student model as the student model can view the data on all users while the teacher model can only learn from local data [34]. FedKD utilizes an adaptive mutual distillation

framework to facilitate the reciprocal learning of a student and a teacher model on each client. This approach avoids the direct communication of large models between users and server. Instead, only the student model is shared by different users and updated collaboratively to reduce communication costs. Through this framework, each user's teacher and student models learn from local data and from each other, with the degree of knowledge distillation being determined by the accuracy of the predictions made. [29]. Different steps of FedKD algorithm illustrated in Figure 2.8.

2.5 Privacy Attacks on Federated Learning

The concept of federated learning has caught the attention of many organisations that want to conduct substantial study on confidential data sets. Data leakage from the deep learning model's parameters is still possible, however federated learning makes collaborative machine learning possible without sharing datasets with a third party. Attacks against machine learning models often target trained models, with the goal of analysing their parameters and learning more about the underlying training data. Training data stored in the model parameters, therefore, has been shown to be vulnerable to a wide range of privacy assaults [35]. generative adversarial network (GAN) reconstruction [36], model inversion [37], and membership inference [38] are the three main categories of privacy attacks on federated learning.

2.5.1 Threat Model

I initially provide a description of the threat models before going into the attacks on federated learning. The threat models in federated learning may generally be divided into the following types: 1- Internal vs external attacks are classified depending on the nature of the adversary; 2- Semihonest against malicious attacks are classified based on if the adversary will follow protocol or not, and 3- Depending on the attack's occurring phase, training phase and inference phase should be contrasted.

1. ***Internal attacks vs. External attacks:***

- **Internal Attacks**: The federated learning server or the federated learning system's users both have the potential to execute internal attacks. As the server has access to each participant's transferred model parameters and each participant can view the global aggregated parameters of federated learning system [39].
- **External Attacks**: When the federated learning model is implemented as a service, attackers may also conduct external attacks by eavesdropping in on the participants' interactions with the federated learning server [40].

2. ***Semi-honest vs. Malicious attacks:***

   - **Semi-honest Scenario**: Adversaries are seen to being passive or honest-but-curious. Without departing from the federated learning protocol, they attempt to find out the private details of other users. The attackers can only examine the information they have already been given, or the global model's parameters [39].
   - **Malicious Scenario**: An active or malicious attacker alters communications to intentionally depart from the federated learning protocol while trying to discover the secret information of the other honest parties. The attacker is able to launch extremely destructive attacks in this situation [39].

3. ***Training Phase vs. Test Phase***

   - **Training Phase Attacks:** Attacks made during the training process make an effort to understand, control, or damage the federated learning model itself. The validity of the learning process may be compromised during the training step by the adversary using model poisoning attacks or data poisoning attacks to tamper with the gathering of training datasets. Regarding the risk to privacy throughout the training process, the adversary may also carry out a variety of inference attacks, e.g. membership inference attacks, on the parameters that each participant has supplied or the parameters that have been aggregated [39].
   - **Test Phase Attacks:** Attackers won't change the targeted model at this phase; instead, they will query it to reveal some confidential data or deceive it to reduce resilience by making false predictions. The attacker's knowledge of the model is a key factor in determining how successful such attacks will be. In inference phase attacks, while the attacker cannot see the model parameters themselves, they may see the input and output pairs, allowing them to execute model theft attacks and reconstruct model parameters. In addition, recently, it has been proved that federated learning models similar to centralize machine learning models are vulnerable to adversarial attacks during the test phase [39].

2.5.2 Attacks Classification

The analysis of gradients can lead to the revelation of sensitive personal information, such as class representatives, membership, and attributes of a specific portion of the training data. The situation becomes even more alarming as malicious actors can infer the actual training images by deducing labels from the shared gradients, even in the absence of prior knowledge of the training data. In the following, the possible privacy leakages of federated learning are described, taking into account the specific kind of sensitive data the adversary is aiming to access [39], [41].

1. **Inferring Class Representatives:** In [19], a new active attack called GAN attack against deep federated learning models were presented. A malicious user in this attack has the ability to deliberately compromise any other party. The adversarial user may train a GAN to produce prototype instances of the intended private training data using the GAN attack, which takes use of the federated learning learning process' real-time nature. The training data's distribution seems to be mirrored by the produced samples. The GAN attack, it should be emphasized, requires that the whole dataset for training for a particular class originates from a single user, which implies that the GAN-constructed samples are comparable to the training samples only when all members of the class are similar. Model Inversion [37] attack has similar assumptions to be effective in centralized machine learning [39].

2. **Membership Inference Attack:** The goal of an inference attack is to extrapolate information from the training set. The goal of the Membership Inference attack [38] is to get data by first verifying that it is present in a training set. The adversary takes use of the global model to learn about the users' respective training sets. When this occurs, we train the predictive model to make predictions based on the original training examples and use those predictions to extract information about the training data. The neural network's tendency to remember its training data leaves it open to passive and active inference attack [3].

3. **Property Inference Attack:** A malicious party may infer specific qualities of other users' training data using either passive or active property inference attacks. It is assumed in property inference attacks that the attacker has access to additional training data that has been appropriately labeled with the target attribute [39]. For instance, inferring some information about the genders of federated learning users by analyzing the gradients is a kind of property inference attack.

4. **Inferring Training Inputs and Labels:** A method known as DLG (Deep Leakage from Gradient) [41] is a novel study that has developed an optimization approach capable of extracting both the training data and labels, making it a particularly severe attack when compared to other methods.

This approach has the ability to recover the original data used for training a deep learning model with exceptional accuracy, making it a serious threat to data privacy. In a subsequent paper [42], the authors developed an analytical method dubbed improved DLG (iDLG) to extract labels based on the shared gradients and an investigation into the relationship between the labels and the signs of the gradients. Differentiable models trained with cross-entropy loss and one-hot labels are a common setup for classification problems, and iDLG may be used to attack them.

To sum up, inference attacks in federated learning assume that the attackers possess advanced technical skills and unlimited processing capabilities. Therefore, most of these attacks rely on the fact that the adversarial players can be selected during several iterations of the federated learning training phase to update the global model. The need for gradient protection in federated learning is emphasized by these inference attacks, which can be prevented by using various privacy-preserving techniques discussed in the following section.

2.6 Defenses Strategies against Privacy Attacks

Although federated learning allows users to maintain their data local, current research has proven it is ineffective in preventing known inference attacks from being used to compromise the privacy of the underlying training data. The model parameters that were transferred throughout the training phase and the model's outputs are nonetheless vulnerable to privacy leaks [43]. So, in the following, I will discuss methods for preserving users' privacy while they are being trained. These methods enable the cooperative training of a model amongst several users (i.e., participants) while maintaining the confidentiality of each user's dataset.

2.6.1 Differential Privacy

In federated learning, differential privacy [44], [45] can be used to protect the privacy of the client's data during model training. The basic approach is to add noise to the model updates before sending them to the central server. The server aggregates the noisy updates and returns an updated model to the clients. The noise added to the updates ensures that no individual's data can be inferred from the model or the updates.
Differential privacy has been applied to a variety of federated learning scenarios, including image classification, natural language processing, and health-

care applications. Some recent works have proposed novel differential privacy mechanisms specifically designed for federated learning, such as Federated Differential Privacy (FDP) [5] and Differentially Private Stochastic Gradient Descent (DP-SGD) [46]. Applications for DP may be separated into two groups:

- **Central differential privacy (CDP)**, as described in [29], needs users to trust the database administrator (i.e., the data curator) to protect their confidentiality. After gathering data from people, CDP involves of injecting random noise. The random noise is introduced to the dataset or to the outputs of queries run against it [47].

- When people do not trust the data keeper, **local differential privacy (LDP)** addresses CDP's weaknesses and protects privacy. Users tamper and/or encrypt their replies during data collecting before sending them to the main server. Since each person uniquely perturbs his answer, LDP techniques should be properly designed [48]. This is because the predicted frequencies on the dataset may not be true [47].

In distributed data processing systems, (ϵ, δ)-DP provides a decisive criterion for privacy preservation. As $\epsilon > 0$ implies that all outputs in a database have distinct bounds on adjacent datasets D_i and D_i', respectively, and δ implies that the ratio of the probabilities for two adjacent datasets D_i, D_i' cannot be bounded by e^ϵ after adding a privacy-preserving mechanism. Larger ϵ provides a clearer distinction between neighbouring datasets, thus increasing the risk of privacy violations when a δ is indeterminately calculated. Here is a formal definition of DP.

Definition 1 ((ϵ, δ)-DP [49]): A randomized mechanism $M : X \rightarrow R$ with domain X and range R satisfies (ϵ, δ)-DP, if for all measurable sets $S \subseteq R$ and for any two adjacent databases D_i, D_i' $i \in X$,

$$Pr[M(D_i) \in S] \leq e^\epsilon Pr[M(D_i') \in S] + \delta. \tag{2.5}$$

A Gaussian mechanism defined in [49] can be used for numerical data to guarantee (ϵ, δ)-DP. According to [49], we present the following DP mechanism by adding artificial Gaussian noise. In order to ensure that the given noise distribution $n \sim N(0, \sigma^2)$ preserves (ϵ, δ)-DP, where N represents the Gaussian distribution, we choose noise scale $\sigma \geq c\Delta s/\epsilon$ and the constant $c \geq \sqrt{2ln(1.25/\delta)}$ for $\in (0, 1)$. In this result, n is the value of an additive noise sample for data in the dataset, s is the sensitivity function given by $s = max_{D_i, D_i'} \|s(D_i) - s(D_i')\|$.

2.6.2 Homomorphic Encryption

Homomorphic encryption is a powerful cryptographic technique that allows computation to be performed on encrypted data without the need for decryption. This makes it an attractive tool for privacy-preserving federated learning, where multiple parties collaboratively train a machine learning model without sharing their sensitive data with each other. Federated learning entails a central server overseeing the training process and collecting model updates from participating clients. Homomorphic encryption offers a means of safeguarding model updates against unauthorized access by encrypting them before transmission to the server. By doing so, the server is unable to glean any insight into the underlying data utilized by the clients, thus reinforcing data privacy and security. There are two main types of homomorphic encryption: fully homomorphic encryption (FHE) and partially homomorphic encryption (PHE). FHE allows for arbitrary computations to be performed on encrypted data, but it is currently too computationally expensive to be practical for most real-world applications. PHE, on the other hand, supports only a limited set of operations (e.g., addition and multiplication), but it is much faster and more efficient.

PHE, or partially homomorphic encryption, is a common technique employed in federated learning to protect the privacy of the client's data. Specifically, PHE is applied to encrypt the model updates generated by the clients during the training process. Once encrypted, the server can aggregate and manipulate the updates without accessing the underlying raw data. This approach ensures the confidentiality of the data and enables secure collaboration among the parties involved in the federated learning process. The updated model can be encrypted again and sent back to the clients for further training, without the server ever seeing the unencrypted data.

One potential drawback of homomorphic encryption in federated learning is that it can increase the communication and computation overhead. The encryption and decryption operations are computationally expensive, and the encrypted data is typically much larger than the original data. This can result in slower training and higher resource usage. However, recent advances in homomorphic encryption algorithms and hardware acceleration have made it more feasible for practical use cases.

In conclusion, homomorphic encryption is a promising technique for privacy-preserving federated learning. It allows multiple parties to train a machine

learning model without revealing their sensitive data, and it can be used in conjunction with other privacy-preserving techniques such as differential privacy and secure multi-party computation. However, it is important to carefully consider the trade-offs between privacy and efficiency when choosing a specific homomorphic encryption scheme for a given application.

2.6.3 Secure Multiparty Computation (SMC)

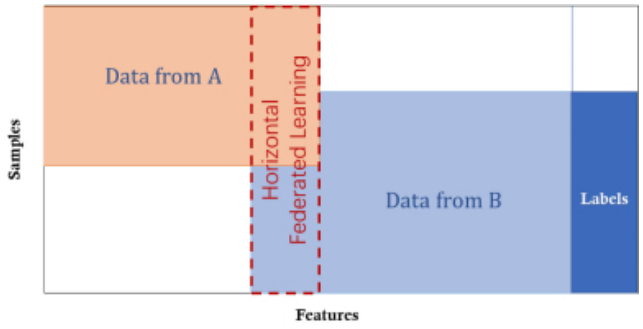
Secure multiparty computation (SMC) [50] is another cryptographic technique that can be used for privacy-preserving federated learning. It enables multiple parties to jointly compute a function on their private inputs without revealing their inputs to each other. This makes it a useful tool for federated learning, where the clients may be unwilling or unable to share their data with a central server or with each other. In SMC protocol, each party encrypts their input using a secret key and shares the encrypted data with the other parties. The parties then collaboratively compute the desired function on the encrypted data, without decrypting it. The output of the computation is also encrypted and can be revealed to each party without revealing the underlying inputs [51]. SMC can be used for a variety of machine learning tasks, including model training, prediction, and evaluation. For example, in federated learning, the clients can use SMC to collaboratively train a machine learning model without revealing their private data to each other or to the central server. One of the advantages of SMC is that it can support a wide range of computation types, including arithmetic, logic, and comparison operations. This makes it a powerful tool for privacy-preserving computation in federated learning. Additionally, SMC is highly customizable, and different SMC protocols can be designed to meet the specific security and efficiency requirements of different applications. However, SMC also has some limitations. One of the primary challenges is that it can be computationally expensive, especially for large datasets or complex computations. The communication overhead can also be significant, as the parties need to exchange many messages during the computation. These limitations can make SMC impractical for some real-world federated learning scenarios.

In conclusion, secure multiparty computation is a promising technique for privacy-preserving federated learning. It allows multiple parties to jointly compute a function on their private data without revealing it to each other, and it

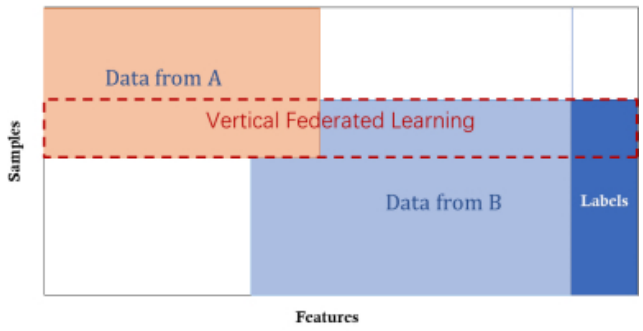
can be used in conjunction with other privacy-preserving techniques such as homomorphic encryption and differential privacy. However, it is important to carefully consider the trade-offs between privacy and efficiency when choosing a specific SMC protocol for a given application.

2.7 Communication Efficient Federated Learning

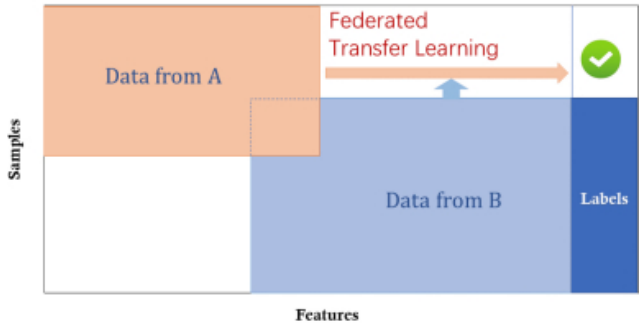
Communication efficiency is a critical factor in the performance of federated learning systems. In federated learning, the clients exchange their local model updates with a central server or with each other. The communication cost can be high, especially when the number of clients is large or the models are complex. Thus, improving the communication efficiency of federated learning is an active area of research. One approach to improving the communication efficiency of federated learning is to reduce the amount of data that needs to be communicated between the clients and the central server. This can be achieved by compressing the model updates [52] or by using sparse updates [53], where only the non-zero components of the update are transmitted. Another approach is to use quantization [54] to reduce the number of bits used to represent the model updates. To decrease the communication cost in federated learning, an alternative method is to adopt local model training. This technique involves the clients conducting multiple rounds of training before exchanging their model updates with the central server. While it can help decrease communication overhead, it can also increase the training time and the number of communication rounds required to reach convergence.
To summarize, minimizing communication cost is a crucial aspect of federated learning systems. Several methods have been suggested to achieve this goal, such as compressing models, using sparse updates, quantizing data, and adopting local model training. These techniques aim to enhance scalability and efficiency in federated learning systems. In the future, further research is expected to improve the performance of federated learning algorithms and to address scalability and efficiency issues.



(a) Horizontal Federated Learning



(b) Vertical Federated Learning



(c) Federated Transfer Learning

Figure 2.4: Different federated learning types [18]

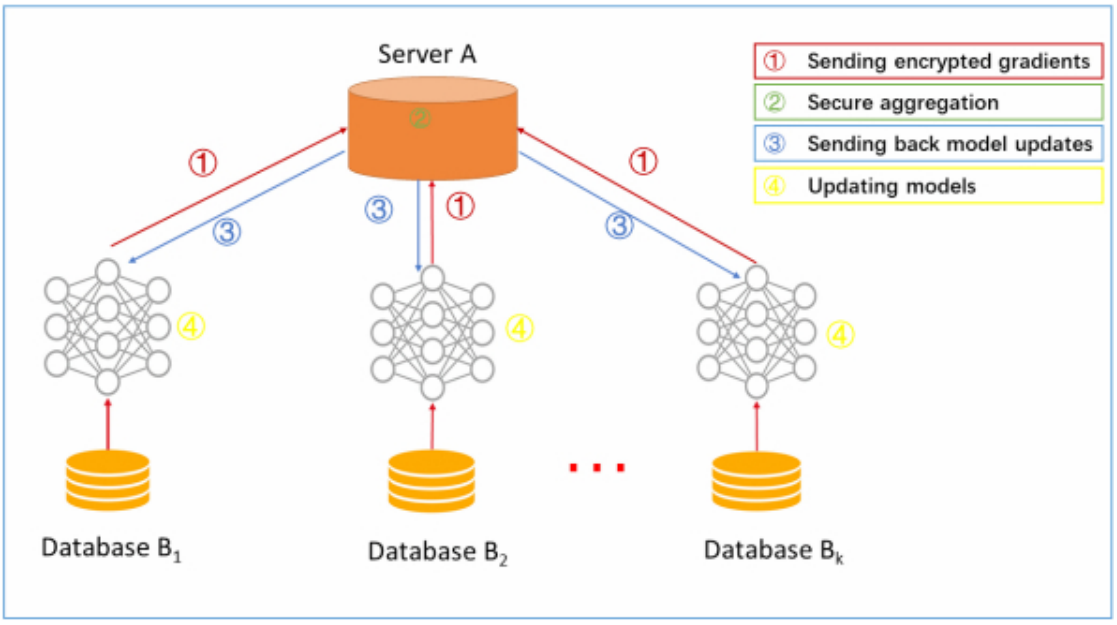


Figure 2.5: Horizontal federated learning steps and architecture [18]

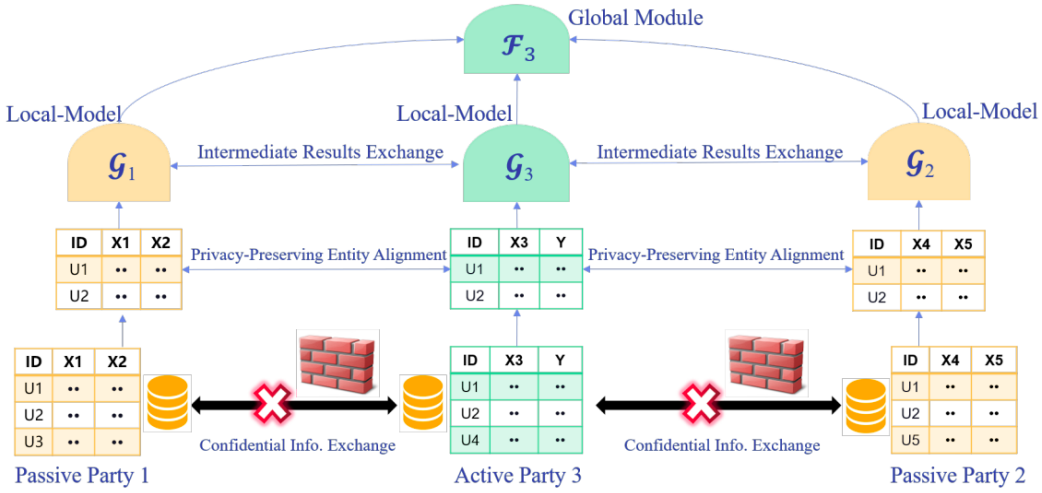
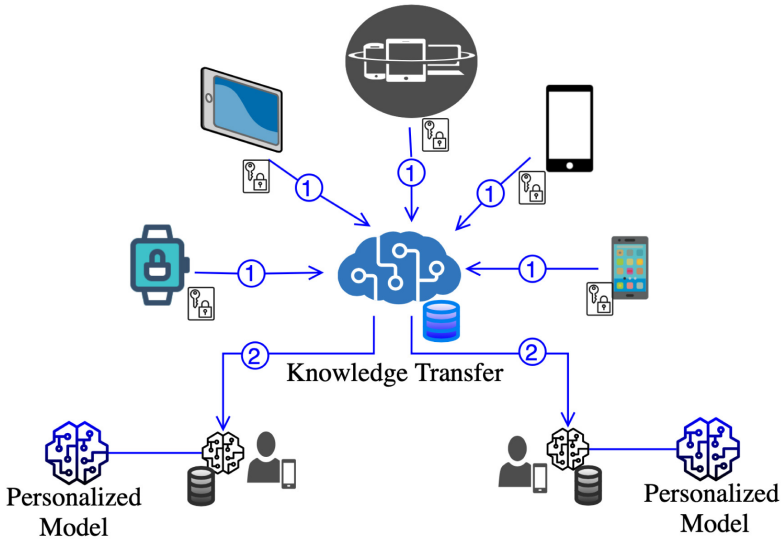


Figure 2.6: Vertical federated learning architecture [23]



Step 1: Heterogeneous clients train a global model on cloud with encrypted techniques, similar to VFL
Step 2: From the pre-trained cloud ML model transfer learning is applied to get personalized individual ML models

Figure 2.7: Federated Transfer Learning architecture [3]

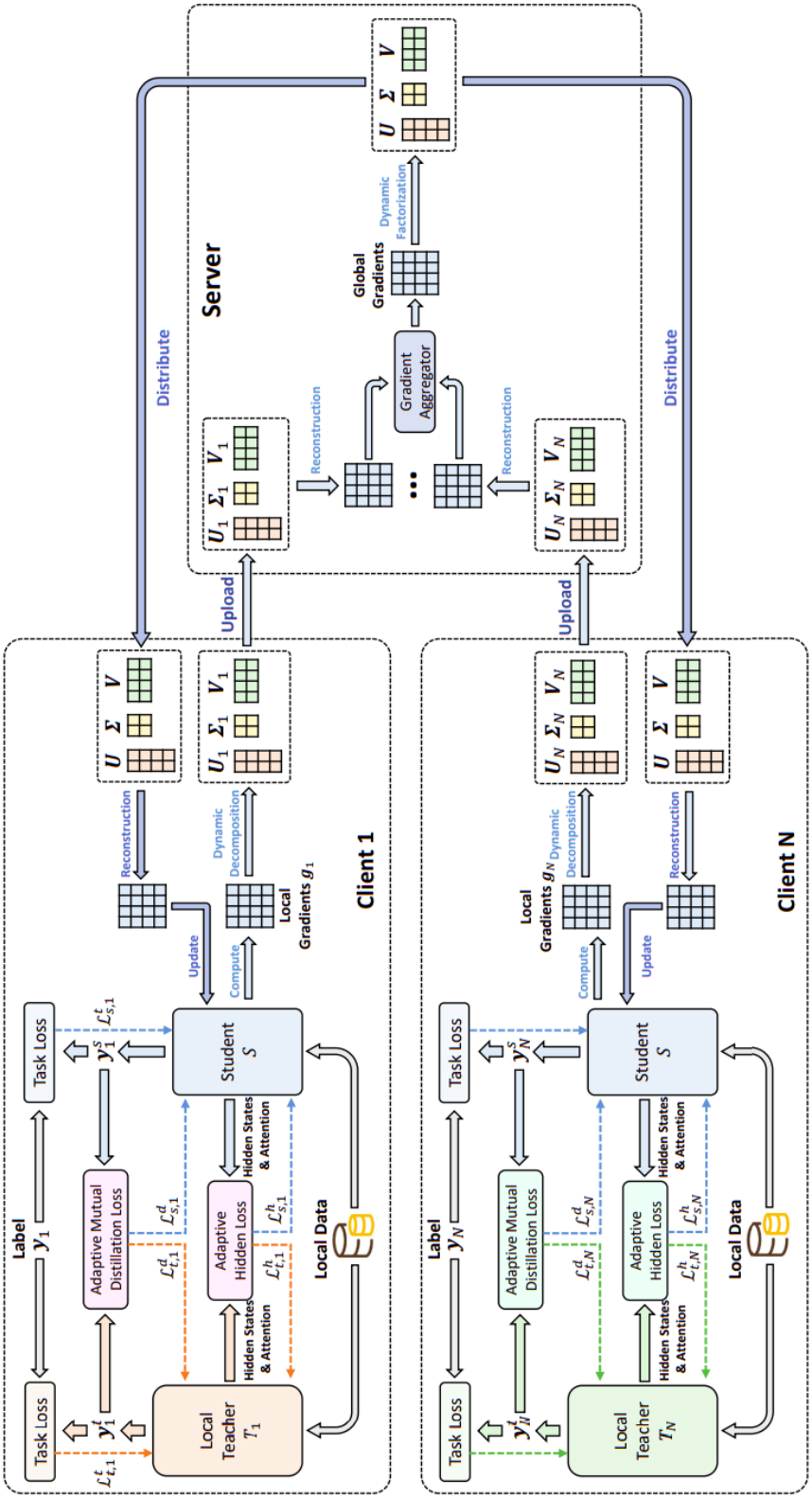


Figure 2.8: Framework of FedKD Algorithm [29]

Figure 2.9: An Instance of Model Inversion Attack [37]

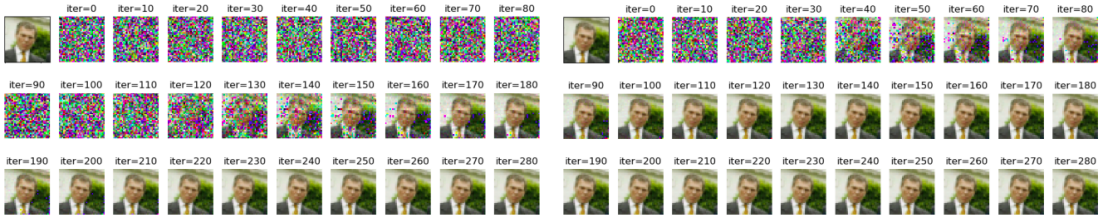
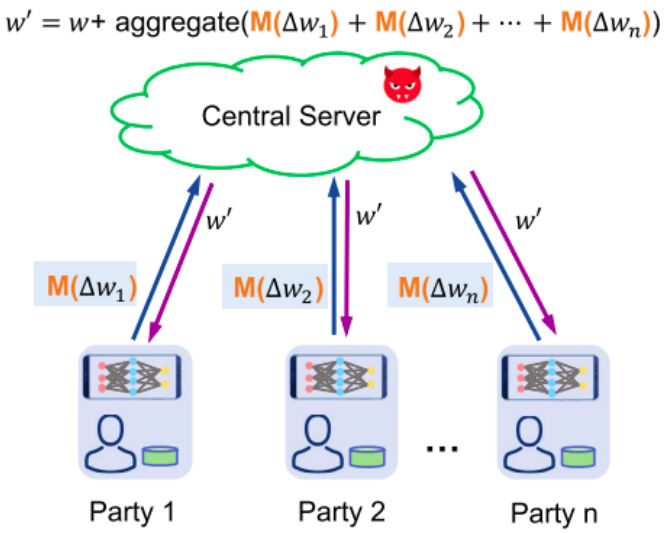
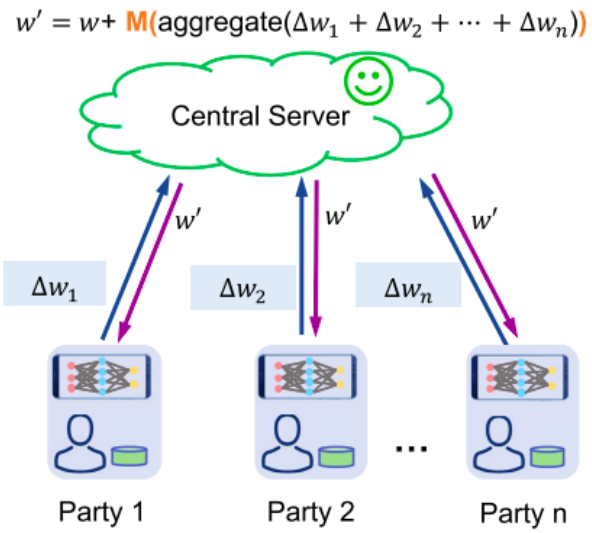


Figure 2.10: Comparison of DLG (left) and iDLG (right) attacks on the LFW face dataset [42]



(a)



(b)

Figure 2.11: LDP (left) vs. CDP (right) [39]

3

Analysis of Industrial Use-cases

In this section, the main contributions of this thesis will be discussed. First, we have the methodologies wherein two different approaches named: 1- Client Selection Strategy (CSS) and 2- Parameter Randomization are explained. Then, in separate sections, three different use cases of the DAIS project will be introduced. According to the requirements of the project, A privacy-preserving, communication-efficient federated learning system should be designed for each of them.

3.1 Methodologies

3.1.1 Client Selection Strategy

Local model updates are collected iteratively from clients in federated learning. At each iteration, clients are usually selected by random selection (RS). It has some drawbacks, such as long training or convergence times with heterogeneous clients. However, studies are being conducted to determine how to select clients more effectively, which can improve system performance. In order to increase the accuracy and training efficiency of the federated learning system, we propose in this work a client selection technique called "CSS" that is dependent on the sample size of each client. The bigger global datasets it can supply for training can increase the model's performance and lower the standard deviation of additive noise brought on by the aggregation procedure because this proposed technique chooses half of the candidates from a wider

pool of customers.

Algorithm 4 Clients Selection Strategy (CSS)

Input: Number of iterations: T, Clients list: L, Number of selected Clients: K
Output: Selected clients
  for $t \leq T$ **do**
    M = Sorted L in descending order by sample size
    Select K/2 randomly from M_0 to M_k
    Select K/2 randomly from L
  end for
  Return K selected clients

Algorithm 4 show the CSS method used during each training round. At each training round, clients with the biggest datasets are more probable to attend. To have an unbiased client selection, we use half of the selections based on our proposed method (line 3) and half based on random selection (line 4). Both halves should not overlap.

3.1.2 Parameter Randomization

The effectiveness of federated learning systems depends heavily on communication efficiency. In federated learning, clients communicate with a central server to share local model changes. The cost of communication may be considerable, particularly if there are many users or complicated models. Hence, enhancing the communication efficiency of federated learning is an important field of study. One way that federated learning enhances communication efficiency is by decreasing the quantity of data transferred between clients and the global server.

I proposed a solution to increase the communication efficiency of federated learning systems for the SER and DFD use cases that its name is *Parameter Randomization* method.

The parameter randomization algorithm is shown in Algorithm 5. Using this strategy, the amount of communication parameters may be decreased according to the number of clusters. For the sake of explanation, let's say the server has decided to group the clients into C clusters, and X represents the amount of the model parameters. In the conventional approach, each client needs to transmit all of the X model parameters to the server. However, in the parameter randomization method, each client only needs to send a random subset of size X/C of

its model parameters to the server. Hence, the communication efficiency of the federated learning system is improved since less data needed to be sent between the client and the server.

Algorithm 5 Parameter Randomization in federated learning

Input: Number of iterations: T, Total Number of clients: N, Selected Clients for each round of training: K, Initial global model: W_0^G, Number of clusters: C
Output: Communication Efficient federated learning trained model
Start:

1. The server broadcasts W_0^G and $t = 1$
2. The server selects $K \subset N$ clients, Clusters them randomly into C clusters, and assigns a set of random non-repetitive indices of parameters to each cluster.

for $t \leq T$ **do**
Client Side:
for all Selected Clients $i \in K$ **do**
client i trains its local model
end for

Server Side:

1. The server gets the randomly selected parameters from the clients of each cluster, aggregates them, and concatenates the aggregated parameters of clusters together to create the whole parameter space.
2. The server broadcast the aggregated and concatenated global model to clients.
3. The server selects $K \subset N$ clients, Clusters them randomly into C clusters, and assigns a set of random non-repetitive indices of parameters to each cluster.

end for

3.2 Usecase 1: Speech Emotion Recognition

3.2.1 Use-case Explanation

In this use case, our goal is to develop a distributed media recommendation engine, which is to be leveraged by AI-based novel technologies. The recom-

mendation system will have to work in a distributed and privacy-protecting manner. Users devices will act as nodes that belong to a big cluster and run pieces of the recommendation engine process so that personal data is processed in the users device, not in the cloud, a sample architecture of the overall system is illustrated below.
The use case has three main objectives:

- Developing a privacy-protecting mood-centric distributed media recommendation engine leveraging novel distributed machine learning/deep learning (federated learning) techniques running on edge clusters.
- Current mood of the user is recognized through a voice-based assistant interacting with the user.
- Showing that the recommendation system works in a distributed and privacy-protecting manner on DAIS components.

The first and foremost benefit is to make the recommendation engine distributed through federated learning techniques instead of a centralized media recommendation. Secondly, media recommendations currently do not take into account the users feelings. With this technology, media recommendations will incorporate the current mood of the user. Furthermore, while making a media recommendation, the system protects the privacy of the users, and does not share the data, but instead processes the data where the data originated.

3.2.2 System Description

The purpose of this section is to describe the system design of federated learning in SER applications 3.2.3, the threat model 3.2.4, the proposed PFL-SER method 3.2.5, including algorithms design and the proposed client selection strategy.

3.2.3 System Design

A federated learning approach involves training machine learning models collaboratively across several clients over several iterations. In Fig. 3.1, the federated learning system designed for SER application is comprised of clients, such as mobile phones, laptops, and TV, denoted as $\{C_1, ..., C_n\}$. These clients are responsible for performing speech emotion machine learning models without sharing their speech data, denoted as $\{C_1, ..., C_n\}$, with the central server. As

illustrated in Fig. 3.1, the federated learning training process typically involves three steps.

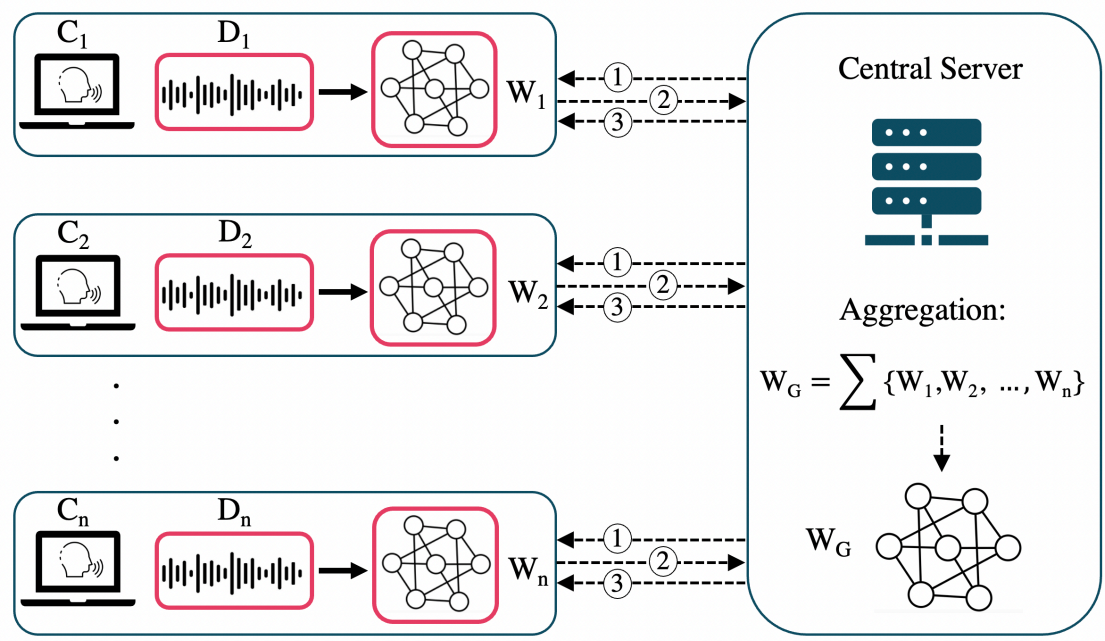


Figure 3.1: A schematic diagram of federated learning in SER applications.

- **Step 1:** Model initialization and distribution.
- **Step 2:** Local training, sharing the trained model with the server and then updating the server.
- **Step 3:** Clients parameters aggregation.

Repeat steps 2-3 until the global loss function converges or an acceptable degree of accuracy is reached.

3.2.4 Threat Model

The server is trusted to be trustworthy throughout this use-case. There are, however, honest-but-curious clients and external adversaries who target clients' private information. Federated learning requires the server and clients to share the intermediate parameter W_i and the global parameter W_G, despite the fact that in this setup the client's copy of the individual dataset D_i remains on the client's local machine. As demonstrated by model inversion attacks [37], which require only "black-box" access to a trained model in order to reconstruct private information. The adversary with black-box access is believed to have

prior knowledge of a label created by the model parameters W_G or W_i, such as an emotion label or a unique identification, and to seek to "reconstruct" characteristics or voice data on the client corresponding with that label. Using this strategy, the attacker has a chance of recreating the voice data of every person in the training set. Those whose voice recordings are used for training purposes may feel violated by this.

3.2.5 Proposed Method: PFL-SER

According to the previous section, sharing the model update parameters between clients and server can leak private training data. The honest-but-curious client j can reconstruct training private data of clients i by conducting model inversion attacks with limited knowledge. In order to prevent model inversion attacks, we provide a new method in this study called "Private Federated Learning in Speech Emotion Recognition" (PFL-SER) that combines Local Differential Privacy (LDP) with Stochastic Gradient Descent (SGD). In order to maintain anonymity, we include Gaussian noise into the parameters of the local training model. By retaining the model's convergence and performance, as well as offering extremely appealing privacy safeguards, this method may minimize the amount of noise introduced to the gradients. Thereafter, we'll go more into differential privacy, the algorithmic architecture of PFL-SER, and the client-selection method.

Algorithms Design

To generate noise for the client before delivering local model parameters, PFL-SER makes use of LDP based on SGD. In order to optimize the federated learning algorithm's privacy and convergence, we intend to limit the impact of individual clients' training data during the training phase, particularly in the SGD computation inspired by [46]. Basically, gradients are computed for each batch size, but before averaging them out, each gradient is clipped at a predetermined threshold C. When the average is calculated, the clipping threshold is used to add calibrated noise. The approach safeguards privacy while preserving convergence and speed in gradients.

Algorithm 1 shows the PFL-SER with the LDP mechanism. Initially, the server broadcasts the initial global model w_0^G to k selected clients. Then during that time slot t, each client $i \in k$ trains his/her own local dataset D_i by mini-

mizing the loss function ∇L^i. The gradient $g(b)$ for each client is calculated by each $b \in B_i$. To bound the gradient contribution of each b, we clip each gradient in the $\|L\|_2$; i.e., the gradient vector $g(b)$ is replaced by $g(b)/Max(1, \frac{\|g(b)\|_2}{C})$ for a clipping threshold C. This clipping ensures that if $\|g\|_2 \leq C$, then g is preserved, whereas if $\|g\|_2 \geq C$, it gets scaled down to be of norm C [46]. After clipping the gradient, we take the average of all gradients in B and add a scaled Gaussian noise $N_t^i \sim N(0, \sigma_i^2 C^2)$ to the client side to achieve LDP.

After receiving the noisy local model parameter w_{t+1}^i from selected clients, the server aggregates these parameters using the FedAvg algorithm to obtain the federated model w_{t+1}^G. The federated model w_{t+1}^G is then updated and sent back to all local clients for further training.

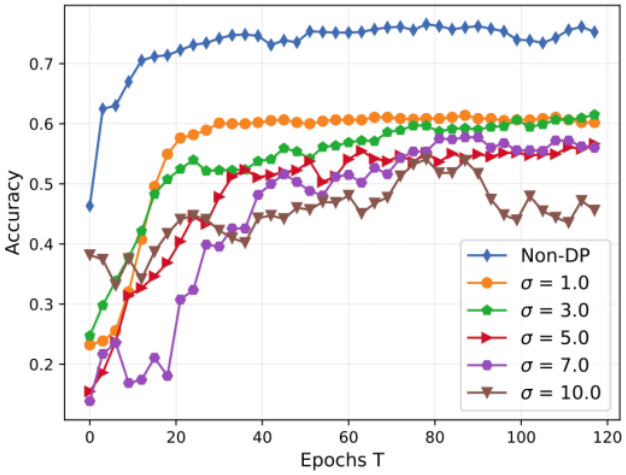
3.2.6 Experimental Results

We discuss the industrial use case, the simulation environment, and the publicly available dataset that were all part of our evaluation here. In the next subsection, we'll discuss how to construct experiments to assess PFL-efficacy, SER's how CSS affects PFL-SER efficiency, how to test PFL-SER robustness against attacks, and how to strike a good balance between privacy and precision.

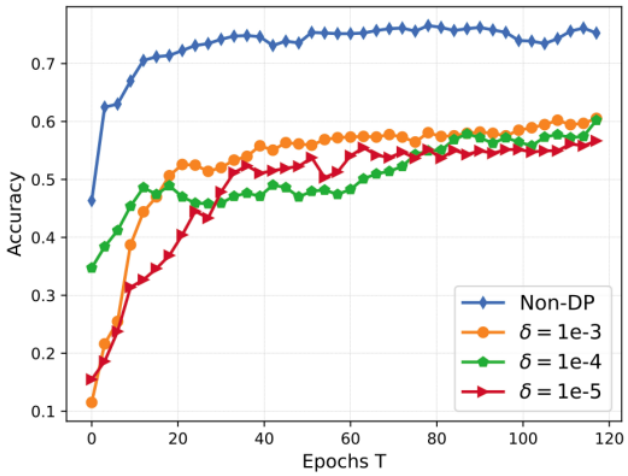
3.2.7 Evaluation Results

We illustrate the effect of different DP settings on accuracy and study the system's convergence to draw conclusions about PFL-SER's performance. In the following setting for Fig. 3.2, the number of selected clients is $K = 51$, and the number of training epochs is $T = 120$. More specifically, we demonstrate the effect of noise scale σ on accuracy in Fig. 3.2 (a), the impact of failure probability δ variation on accuracy in Fig. 3.2 (b), and clipping threshold C and its effect on accuracy in Fig. 3.2 (c).

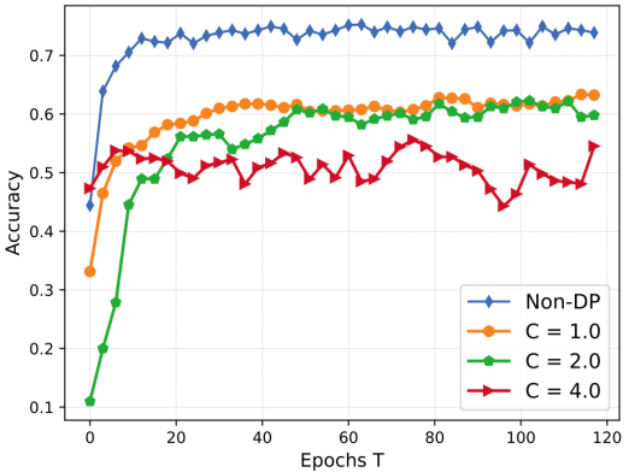
Based on the results presented in Fig. 3.2 (a), When T, the total number of training epochs, is increased, the rate of improvement in accuracy slows down until it stabilizes. This pattern suggests that the PFL-SER technique tends towards convergence over time. Additionally, an increase in the noise scale (σ) leads to slower convergence since a higher noise scale implies stronger privacy protection with a larger amount of injected noise during the training process. A significant increase in the noise scale, such as $\sigma = 10$, can result in an unstable



(a)



(b)



(c)

Figure 3.2: Performance evaluation on the convergence of PFL-SER mechanism, (a) accuracy versus noise scale σ, (b) accuracy versus failure probability δ, and (c) accuracy versus clipping threshold C.

system without convergence. In Figure 3.2 (b), the impact of δ on accuracy is evaluated. Unlike σ, a higher value of δ results in faster convergence, less privacy protection, and higher accuracy. For instance, when $\delta = 10^{-3}$, it achieves the highest accuracy compared to other values because it sacrifices more privacy to maintain utility.

In Fig. 3.2 (c), we demonstrate the effect of various clipping thresholds C on the accuracy of the PFL-SER method. Thus, because of the impact on sensitivity, we are compelled to increase the amount of noise in the parameters as we increase C. This means that for our technique, C should be between 1.0 and 2.0 for fast convergence and good accuracy.

System performance may be affected by a number of variables, including but not limited to those unrelated to the DP itself, such as the choice of clients. Consequently, we proposed the CSS method, which selects the client according to the explanations provided in Section 3.1.1. A further comparison was made between the CSS method and random selection (RS) in federated learning systems on both PFL-SER and its non-LDP counterparts (federated learning-SER).

In Fig. 3.3, when $\sigma = 1.0$, $C = 2$, $\delta = 10^{-5}$ and $K = 51$, we applied the CSS and RS method on the PFL-SER and federated learning-SER methods. According to Fig. 3.3, when the PFL-SER system utilized the CSS method for selecting clients, the result showed a significantly improved accuracy from 0.60 to 0.70 in comparison with the RS method. With CSS, half of the clients are selected from a larger pool, which allows a larger global dataset to be used for training, increasing the model's accuracy and reducing additive noise standard deviations. When these two methods of client selection are applied to the federated learning-SER system, the accuracy of the system is not significantly affected. Thus, an efficient client selection strategy will improve the system's performance and mitigate the negative effects that differential privacy may have on the system's accuracy.

Moreover, we evaluate the PFL-SER method against privacy-related attacks by implementing model inversion attacks [37], which are capable of reconstructing the private dataset from the model parameters and labels. Moreover, the system takes into account active and passive attack configurations with black-box access. During an attack, the adversary may see the victim's local model parameters and its output. Yet, in a passive attack, the attacker has access to the model's output and just the global parameters.

By comparing the reconstructed data to the original private data, we can quantify the attack's success in the federated learning system and provide an

Table 3.1: Attack effectiveness versus noise scale σ and clipping threshold C

Attack Type	Clipping Threshold (C)	Mean Squared Error (MSE)					
		PFL-SER					FL-SER
		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 7$	$\sigma = 10$	Non-DP
Passive Attack	C=1	1.053	5.196	96.049	568.619	4350.461	1.38
	C=2	1.445	10.269	286.263	6125.767	43359.953	1.38
	C=4	23.012	10085.062	159371.700	2296731.092	17429575.658	1.38
Active Attack	C=1	0.971	1.189	19.830	45.132	157.640	1.02
	C=2	1.028	9.189	78.910	1139.474	5807.308	1.02
	C=4	1.886	344.000	8508.620	59164.757	572765.191	1.02

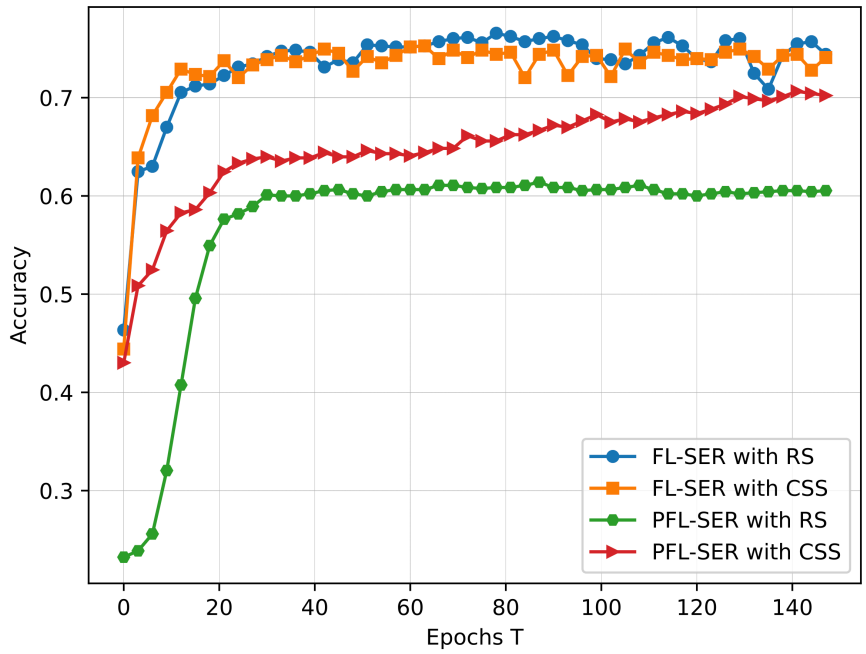


Figure 3.3: Evaluation of different client selection methods based on accuracy.

estimate in terms of Mean Squared Error (MSE). Table 3.1 compares the effectiveness of the attack when implemented into PFL-SER and federated learning-SER systems. The adopted federated learning setting is $K = 7$, $C = [1, 2, 4]$, and $\delta = 10^{-5}$. According to Table 3.1, when the noise scale σ is equal to 1.0, the amount of MSE is close together in PFL-SER and federated learning-SER. The MSE increases significantly with an increase in noise scale σ and clipping threshold C, which implies that attack effectiveness is declining, which proves that our proposed PFL-SER method works as expected. It has been shown that active attacks are more powerful than passive attacks in model inversion attack settings for reconstructing the private data of specific clients, and the value of MSE for active attacks is lower than that for passive attacks on the same noise scale.

The PFL-SER method relies on the LDP mechanism to maintain privacy, while the noise scale has a negative impact on learning performance metrics such as accuracy. To preserve privacy in PFL-SER systems without sacrificing accuracy, it may be necessary to consider a trade-off between privacy and accuracy. Fig. 3.4 illustrates the trade-off between the noise scale σ and the accuracy of PFL-SER. In this setup, $K = 7$, $C = 1$, and $\delta = 10^{-5}$. When the noise scale σ is between $[3.0 - 5.0]$, and the accuracy of the PFL-SER is near 62%, there is a possibility of reaching the trade-off. This noise scale allows the PFL-SER system to preserve the privacy of the SER application against model inversion

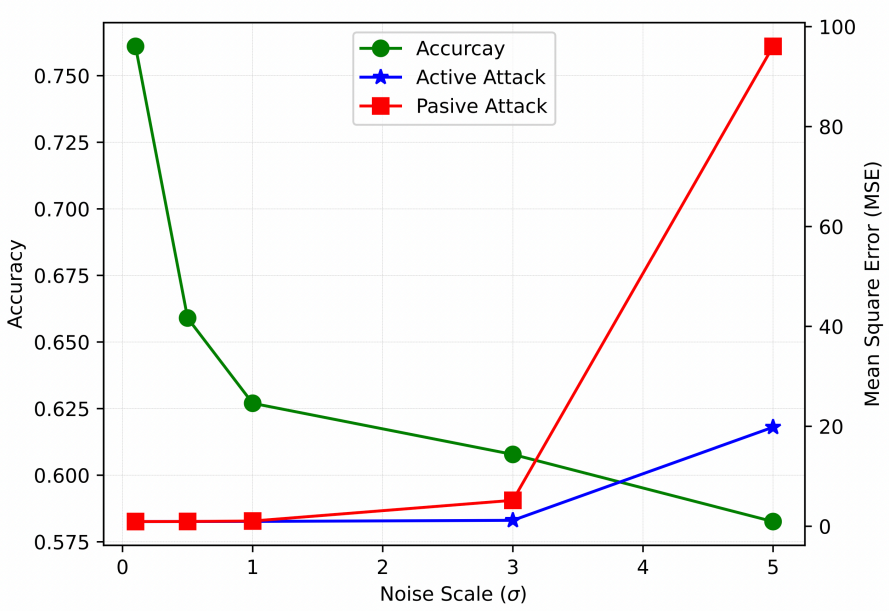


Figure 3.4: Privacy and accuracy trade-offs using noise scale σ, attack effectiveness by MSE, and PFL-SER model accuracy.

attacks without compromising its accuracy. In order to preserve privacy against a passive attack, the noise scale can be near $\sigma = 3.0$, while to preserve privacy against an active attack, the noise scale can be higher than $\sigma = 3.0$ and lower than $\sigma = 5.0$ to reach a reasonable approach and trade-offs.

3.2.8 Parameter Randomization on Speech Emotion Recognition

By applying Parameter Randomization to SER, we can cut down on the number of training-phase parameter swaps. In federated learning, the number of clusters might vary from one to as many as the total number of users (full clustering-each client is a cluster). Figure 3.5 shows that when the number of clusters is increased, training time decreases. Parameter randomization is intriguing since it has no negative effect on the SER federated learning system's accuracy. In addition, the server and any possible snoops cannot use the segmented parameter space to undertake reconstruction attacks since clients do not exchange their whole parameter spaces. Furthermore, As you can see in Figure 3.6 (a) and Figure 3.6 (b), the parameter randomization approach has the same behavior as the normal SER federated learning system.

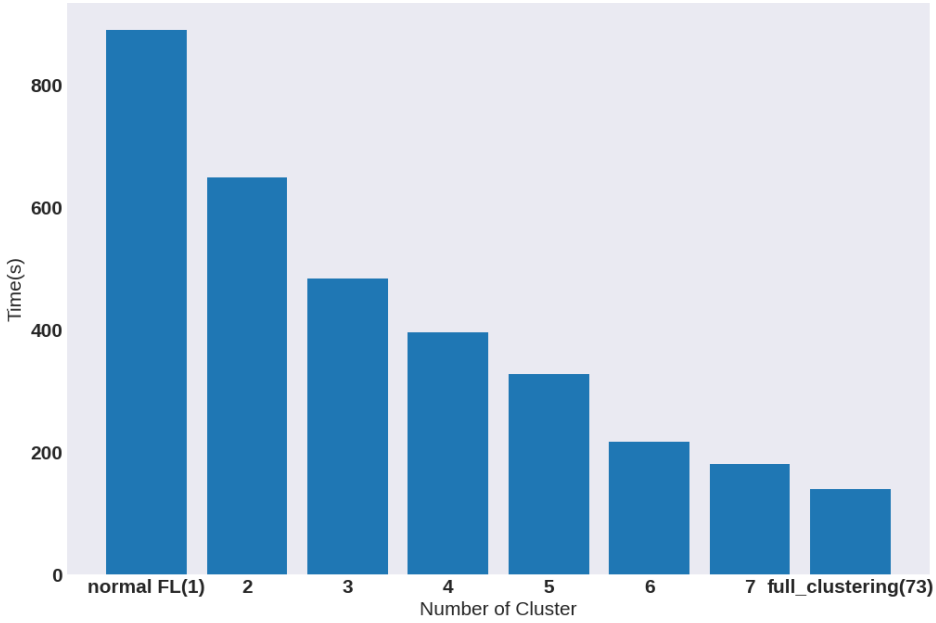


Figure 3.5: Parameter Randomization on SER: Effect of different number of clusters on the training time of federated learning system.

3.3 Usecase 2: Fatigue Detection

3.3.1 Use-case Explanation

A frequent cause of fatalities on the road is driver fatigue and sleepiness. Drivers in the transportation and logistics sectors are often required to make lengthy trips, which contributes to fatigue. This use case and demonstration seek to address this issue by detecting driver weariness in a distributed way. Machine Learning-based systems might identify fatigue or sleepiness based on various information sources such as video/images, physiological or vehicle data. These solutions may be effective in reality, but they depend largely on personal data from drivers, which presents the second difficulty, data privacy. The aim of this use case is to provide a machine learning solution that addresses both challenges while adhering to current regulations such as the General Data Protection Regulation (GDPR). For privacy reasons, the suggested solution must not send user information to a remote server or out of the car where it may be used by a centralized machine learning system. Models for machine learning

are trained in a dispersed fashion across several devices. Personal information about drivers is kept in the vehicle, while only models are uploaded or shared in the cloud or across devices. The American Automobile Association projects that in 2021, drowsy driving will account for 9.5% of all incidents. The purpose of this use case is to test a prototype that might reduce fatigue-related traffic deaths. This use case's overarching goal is to find a way to forecast when a driver needs a break while keeping the driver's data private. As a result, when it comes to distributing the training of machine learning models among vehicles while maintaining individual privacy, the Federated Learning solution is the clear winner.

3.3.2 Dataset

We have made use of the UTA-RLDD developed by scientists at the University of Texas in Arlington [55, 56]. It is one of the more complete datasets for fatigue detection through facial features. The collection comprises of about 30 hours of RGB video of 60 healthy volunteers and each video is tagged by the subjects themselves as one of three states: alert, low vigilance and sleepy. The three states are based on the Karolinska Sleepiness Scale (KSS) [57] which is a scale from 1 to 10 of drowsiness levels ranging from Extremely Alert to Extremely Sleepy. There is one video per state and per subject which makes a total of 180 videos, each about 10 minutes long. All participants were instructed to record three separate videos of themselves sleeping, awake, and awake but drowsy using a smartphone or camera (of any make or model). The three classes were explained to the participants as follows:

- Alert (label 0 in UTA-RLDD): the range 1-3 on the KSS. To explain this state, subjects "were informed that being alert meant they were entirely conscious so they could easily drive for long time" [56].
- Low Vigilant (label 5 in UTA-RLDD): level 6 or 7 on the KSS. This state "corresponds to subtle cases when some signals of sleepiness is seen, or sleepiness is present however no effort to keep alert is required. While subjects could possibly drive in this state, driving would be discouraged" [56].
- Drowsy (label 10 in UTA-RLDD): level 8 or 9 on the KSS. This state "means that the subject needs to actively try to not fall asleep" [56].

3.3.3 Tools

To simulate a federated learning scenario, we used Pytorch [58] together using the free and open-source AIJack framework, which can simulate attacks on a simulated machine learning system in order to ensure the system's security and privacy [59]. The system architecture for this use case consists of a server and 18 clients. Communication between the server and clients is facilitated through a Client interface. The server selects a client to teach and communicates training instructions to it via the network. When receiving these directives, the client runs the code included inside one of the Client methods in order to train the neural network. The real setup for federated learning includes the server defining the aggregation approach, the total number of epochs, and the number of clients that take part in the training and assessment process. Instead, while training and assessing the model, each client uses data from his own environment.

3.3.4 Data pre-processing

We have used a preprocessed version of the dataset [56] constructed by hackenjoe and available at [60]. In this section we describe in detail these pre-processing steps. Only a certain number of frames from each video is extracted. More precisely, 240 frames per video per fatigue condition from 18 individuals were extracted. For each movie, beginning at the three-minute mark and continuing to the finish, a single frame is taken at a time using the OpenCV package. So, each person gets access to a dataset that contains 720 total frames: 240 from the awake condition, 240 from the intermediate state, and 240 from the sleepy state. Dlib [61] is used to extract 68 facial landmarks from each frame. In this work, the facial landmarks associated with the eyes and mouth are the most significant. The following characteristics are employed in the neural network's training and assessment:

- EAR (Eye Aspect Ratio): This is a measure of eye proportions, which is smaller when the eyes are closed than when they are open. The rationale for including EAR as feature is than drowsiness often induces frequent blinking and half shut eyes.
- MAR (Mouth Aspect Ratio): The ratio between the height and breadth of the mouth is a metric that has been used in sleepiness detection systems. Drowsiness, in theory, may be identified by monitoring for changes in the MAR over time, since this is how facial expressions like yawning alter when one is tired.

- PUC (Pupil Circularity): This metric evaluates how perfectly round one's pupils are. The hypothesis was that, like the EAR, drowsiness would lead to a lessening of the circularity of the pupils.
- MOE (Mouth aspect ratio Over Eye aspect ratio): To calculate MOE, divide MAR by EAR.

It seems sense to equalize characteristics across subjects given that there may be persistent variances in the sizes of people's eyes and mouths, for example. To this end, we've utilised the features from the first three frames of the alert state to calculate the mean and standard deviation of each feature for each participant, and then used these values to normalise each feature on an individual basis. When this is done, the feature set will have eight features: the core features and their normalized counterparts. At this point the dataset is composed of 12240 training points (240 entries for each of the three states, for each participant). The data is further split in local datasets, one for each subject.

3.3.5 System model

For the system model, we use a model called Multi-Layer Perceptron (MLP) classifier, which is implemented using Pytorch.The aggregation method selected by the servers is the default method, i.e., FedAvg.

In this application, the model architecture has four completely linked layers (sometimes called dense layers) with respective sizes of 256, 128, 64, and 3. Apart from the final output layer, which utilizes a sigmoid activation function to generate a probability distribution across the three classes (alert, drowsy, and sleeping), all preceding layers use the ReLU activation function. Extracted features from the mouth area of the face picture serve as input to the model, while a projected class label serves as the output.

3.3.6 Attack Model

To prove the viability of our approach, we used a model inversion attack (MIA) as an example of an attack. To violate someone's privacy, one may use a model inversion attack, which entails recreating the secret dataset that was used to train a trained neural network. The attacker makes a prediction for the target variable by feeding the training model into the MIA model. That is to say, the MIA can provide a representative and varied sample that adequately characterises each class in the confidential information, providing a clear example of

an attack on the confidentiality of the characteristics. The MIA can identify the person even the face of the target person is blurred [37]. We examined the MIA attack in our fatigue detection settings in the federated learning setup.

3.3.7 Privacy-preserving federated fatigue detection

To incorporate privacy into the federated learning process, we introduce Gaussian noise to the model updates and apply gradient clipping. This approach guarantees a certain degree of Differential Privacy, which is determined by the level of noise added. [46]. In more detail:

- To limit the impact of individual training points in a minibatch on gradient computations and the resultant updates to model parameters, one must set a maximum sensitivity for each gradient. For this purpose, it is possible to clip the gradients at each training point.
- The clipped gradients are sampled and mixed with random Gaussian noise to provide some degree of differential privacy.

There are three privacy-specific hyperparameters in this setting:

- Norm clipping: each gradient's highest Euclidean (L2) norm, which is used to update model parameters. This hyperparameter is used to limit how sensitive the optimizer is to specific training points.
- Noise multiplier: how much noise is taken as a sample during training and multiplied by gradients. To be more precise, it is the clipping norm as a fraction of the standard deviation.
- Micro-batches: each batch of data is split into smaller units called micro-batches. The default setting is that each micro-batch contains a single training example. With this, gradients can be clipped per-example instead of being averaged over the minibatch. In turn, this minimizes utility and lessens the (negative) impact of clipping on the gradient's signal. The size of micro-batches can be increased to contain more than one training example, which will reduce computational overhead.

Moreover, there is the learning rate parameter, which is already present in the scenario without differential privacy.

3.3.8 Experimental Setup

This subsection describes the simulated environment in which the proposed federated learning based fatigue detection system is tested. For the simulation, we utilised the 12-hour long UTA-RLDD dataset. We design a fully connected

network for federated learning system in such a way that there will be 18 clients. The clients in the federated learning setup correspond to the participating subjects in the dataset and the local data on a client are precisely the datapoints associated with a given subject. This will result in a local datasets of size 720. The detailed network parameters are given in Table 3.2. The model inversion attack was carried out using the following settings: lambda = 0.1 and iteration = 1000. As the gradients of the trained model's output serve as the input for the MIA, an attacker who manipulates the input may predict the model's result. The final product is a set of samples that can be accurately categorised as the specified target label.

Table 3.2: Simulation setup parameters

Network architecture	**Fully connected network**
Total no. of datasets	12240
No. of all clients	18
No. of clients for training	14
No. of clients for testing	4
No. of samples for each client	720

3.3.9 Evaluation

Here, we take a look at how well the suggested federated learning model for fatigue detection can protect users' privacy. Specifically, the accuracy of the system is what we use to evaluate its performance in our case. The model's accuracy statistic reveals how reliably it makes predictions. With 14 clients utilized for training and 4 for testing, the total number of samples is 720.

Each federated learning client's accuracy is calculated after each communication cycle, and the results are then averaged. Using a batch size of 64 and a learning rate of 0.0001, the model was trained using 2160 training points. This federated learning configuration proposes using a global epoch of 100 and a local epoch of 1.

3.3.10 Results

We used federated learning and the differential privacy method to perform fatigue detection while still protecting user privacy. When discussing Gaussian mechanisms and differential privacy, a certain degree of noise is associated

with a specific value for the privacy parameters (ϵ, δ). The number of epochs, noise multiplier, batch size, and the number of training data points all affect the privacy parameter ϵ. We held the delta constant at 10^{-6} and manipulated the standard deviation of the extra noise, σ, between 0.5 and 7.

As was predicted, model performance diminishes as the variance of the noise injected in the gradient updates increases. If you look at 3.7, you can see how well the federated learning process for fatigue detection performs under varying levels of noise (σ). Adding too much noise can quickly degrade the average test accuracy showing the aforementioned trade-off between privacy and accuracy in the context of differential privacy.

We analyzed how well a federated learning mechanism for fatigue detection fared under different noise conditions (σ). This is seen in Fig. 3.8. This demonstrates that the level of privacy achieved may be compromised in exchange for greater levels of accuracy when dealing with varying noise levels.

Equally, we use a graph to illustrate the trade-off between privacy and accuracy when varying the noise level, the efficacy of an attack when using the Mean Square Error (MSE) approach, and the accuracy of a federated learning model based on fatigue detection. In Fig. 3.9, we can see that the accuracy decreases as we increase the noise levels (i.e., σ) while the MSE increase as we increase the noise levels. We use MSE as reconstruction error because it is one of the popular reconstruction loss function that indicates whether the specific attack works well or not in the proposed scenario. The best value of $sigma$ is shown in Fig. 3.9, where the accuracy and MSE overlap. By extrapolating the result from Fig. 3.9, we may determine the best value for $sigma$. With an MSE of 6.85 and an accuracy of 64%, the sweet spot for $sigma$ is about 1.

3.3.11 Parameter Randomization on Fatigue Detection

To lessen the burden of exchanging so many parameters during training, we use Parameter Randomization on the fatigue detection use-case. In federated learning, the number of clusters might vary from one to as many as the total number of users (full clustering-each client is a cluster). Figure 3.10 shows that the training time decreases as the number of clusters rises. Fascinatingly, randomizing parameters does not reduce the accuracy of the federated learning system for detecting fatigue, but it does increase the variability across individual instances. The server and snoopers are also unable to use reconstruction attacks

on the fragmented parameter space since clients do not communicate their whole parameter space. Furthermore, As you can see in Figure 3.11 (a) and Figure 3.11 (b), The parameter randomization strategy behaves similarly to a conventional federated learning system for fatigue detection.

3.4 Usecase 3: Smart Grid System Anomaly Detection

3.4.1 Use-case Explanation

Many sensors and IoT devices are used by the smart electric grid system to gather information on the electricity flowing through the grid's substations. Traditionally, substations' energy data would be sent to a cloud or edge device, where it would undergo analysis and interpretation. There is a risk that this method might result in serious data abuse, data manipulation, or privacy breach. In this article, we introduced a methodology for spotting outliers in the industrial data collected by remote terminal units stationed in the smart electric grid's substations. The system for spotting outliers relied on LSTM autoencoders, and it used both Mean Standard Deviation (MSD) and Median Absolute Deviation (MAD) methods. In order to keep the substations' data secure, we use a technique called federated learning (FL). To avoid providing sensitive information to a central server, FL allows energy providers to jointly train the shared AI model. Unfortunately, the suggested architecture still has flaws in terms of security and privacy because of the subpar quality of the localised models that are exchanged. To ensure that user information remains secure inside our proposed framework, we have used Paillier-based homomorphic encryption.

3.4.2 Dataset

In this part, we evaluate the effectiveness of the proposed method by applying it to a synthetic industrial dataset created with a general behavior of an electric grid node in mind. We filtered the dataset by deleting duplicates to make sure the information was distributed fairly. Our method involves just training the model on typical data and then labeling it as anomalous everything that doesn't conform to the norm. This contributes to the model's enhanced ability to correctly identify and recognize outliers. This allows us to identify previously

unseen anomalies or threat patterns. As such, it was a labeled dataset. There are a total of 1378 data points in the simulated scenario. We did this by separating the data into a "train" set and a "test" set. The majority of the data (1240 samples) is found in the "train" dataset, whereas just 138 samples appear in the "test" dataset (or 10% of the total). In addition, during the testing step, we created and added 100 artificial anomaly samples.

3.4.3 Performance Evaluation Metric

In the proposed ADA-FL framework, for performance evaluation, we use metrics such as Accuracy, Precision, Recall as well as F1-score that is similar to other machine learning research. For expressing these metrics, we need to consider some common parameters such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) for the anomaly detection process. Based on the above parameters, we present the confusion matrix with four different combinations of predicted and actual values that is used to determine the performance of a model. The confusion matrix is given in Table 3.3.

Table 3.3: Confusion Matrix

		Predicted Values	
		normal	anomaly
True Values	normal	TN	FP
	anomaly	FN	TP

According to [62], the ROC curve is a visual depiction of the balance between a machine learning model's true-positive rate (TPR) and false-positive rate (FPR) across various thresholds, with TPR on the y-axis and FPR on the x-axis. The area under the ROC curve (AUC) is a statistical metric used to evaluate the efficacy of an ML model. It indicates the model's ability to correctly categorise observations into positive and negative groups. Improved performance may be shown in an AUC-ROC that is larger. AUC-ROC is a helpful statistic for assessing model correctness since it offers a credible visual depiction of the performance of the proposed model. The AUC is computed as shown in the following equation:

$$AUC_{ROC} = \int_0^1 \frac{TP}{TP + FN} d \frac{FP}{TN + FP} \tag{3.1}$$

3.4.4 Experimental Setup

Our simulation is executed on a machine that has an Intel Core i7-11800H processor, an Nvidia GeForce RTX 3050 graphics card with 4 GB GDDR6 of video memory, and a total of 16.0 GB of RAM. To construct the models computationally, we use Python 3.8 and Pytorch 1.12.1.
Our method for federated learning includes a global server hosted in the Cloud and three separate clients. For locally trained models, each client has access to 413 training data samples. During the training phase of the neural network-based approaches, we use ADAM optimizer with a learning rate of 0.001 to provide a level playing field. A batch size of 8 is used alongside a local epoch of 5, and a global epoch of 70. For federated learning, we employ FedAvg as an aggregate technique. We compare the outcomes of the proposed framework using two distinct methods of assessment. The MSD technique is the first method of assessment, while the MAD score is the second. We use homomorphic encryption with 128-bit key and 256-bit key to protect the privacy of the smart grid system.

Table 3.4: Simulation setup parameters

FL Network	1-Server and 3- clients
Evaluation Approach	MAD-score, MSD approach
Model Type	LSTM-AutoEncoder
Aggregation Method	FedAvg
Loss Function	MSE loss
Epochs	Local epochs: 5 Global Epochs: 70

For this simulation, we use the aforementioned data collection. Due to the small size of the dataset, we restrict the number of federated learning clients to three. During the federated learning process, FL clients utilise their locally-available dataset to train a centralised global model by exchanging model parameters with the global server. Each client has a local dataset of the same size with which to train the model, and the training set is dispersed evenly and at random among the three clients. We use the whole testing set as a benchmark against which to evaluate the global model created. In addition, our federated learning scenario has each client do five local gradient update epochs before sending the learned model to the cloud server for aggregation. The cloud server then performs seventy global epochs.

3.4.5 Privacy-preserving ADA-FL with HE

To protect the privacy of the federated learning system, this section explores how Homomorphic Encryption may be put into practice. We deployed HE with a pair of key sizes, 128 bits, and 256 bits. For this, we use Paillier homomorphic encryption for additive cases in our simulation that was obtained from the PHE (Partially Homomorphic Encryption) library in Python. The additive homomorphic encryption computation is much faster than multiplicative computation, which is suitable for the devices in the Fog layer. Each FL clients encrypt the data received from the RTUs before sending it to the Cloud to perform cipher text calculation. The Cloud computes the additive operation in its ciphertext without knowing the contents of the data. We discuss the results obtained from our simulation in anomaly detection based on AE using a federated learning framework. Simulated results from the proposed framework are then analyzed using LSTM autoencoder methods to determine its efficacy. The experimental setup details how we utilize three FL clients and a single server to test out homomorphic encryption, which we employ to keep the FL clients' data secure. We mainly used the MSD and MAD-score techniques to evaluate the performance of our proposed LSTM-AE-based architecture.

$$Threshold_{MSD} = MEAN(MSE - Errors) + (3 * STD(MSE - Errors)) \quad (3.2)$$

We show the results of several simulations we ran, both with and without encryption (using 128-bit and 256-bit homomorphic keys).

3.4.6 Results and discussion

We present the distribution of normal and anomaly test data losses in terms of MSE loss for MSD approach. Fig. 3.12 shows the test data loss distribution of normal and anomaly data before implementing the homomorphic encryption technique. The results show that the reconstruction error or the MSE losses are exceptionally low. In the figure, the blue line represents the threshold (0.00243) which is common to both normal and anomaly results. The test data samples which are greater than the threshold value are detected as an anomaly (i.e., the right side of the threshold are anomalies).

Fig. 3.13 shows the test data loss distribution of normal and anomaly data after implementing the homomorphic encryption i.e., HE-128bit key. The MSE

loss slightly increases for normal and anomaly data due to encryption and decryption mechanism. Similarly, Fig. 3.14 illustrates loss distribution of normal and anomaly data while implementing HE-256-bit key with threshold value of 0.01126. It is obvious that the MSE loss increase as compared with HE-128bit and non-HE mechanism. The performance metrics such as Recall, Precision, Accuracy, and F1-score also decreases with the implementation of homomorphic encryption as given in Table 3.5.

Our findings demonstrate that our system performs better without the use of homomorphic encryption. Reconstruction errors are less and the threshold value is lower for the HE-128 bit key compared to the HE-256 bit key. Moreover, the HE-128-bit key has a higher Recall value i.e., 79% whereas the HE-256bit key has a lower recall value of only 70% as can be seen in Table IV. As Recall is one of the important metrics, this shows that the HE-128-bit key provides better performance than the HE-256-bit key. If we compare the HE scenario with the non-HE scenario, it is obvious that the HE scenario has an overall worse performance. This is because of extra computation overhead operations on the encrypted data. This additional overhead leads to increased computation time resulting in decreased performance metrics. In addition, homomorphic encryption can also introduce errors or noise into the data, which can further degrade the performance of anomaly detection algorithms. This is because the algorithms rely on accurate and precise data to identify anomalies, and any errors or noise in the data can lead to false positives or false negatives. So, if we implement homomorphic encryption in the suggested framework, there is a trade-off between the performance and privacy of the system. Fig. 3.15 shows the model performance using a confusion matrix. The confusion matrix indicates that there are 138 normal test data samples, which is 10% of the training dataset. And in addition to them 100 data samples are abnormal samples. The proposed model was able to correctly identify a total of 91 data samples as anomalies among 100 abnormal data samples i.e., it can correctly identify 91% of anomalies. The model correctly identified 125 data samples out of 138 total normal test data samples, i.e., it can correctly identify 90.58% of normal data. However, the proposed model incorrectly identified 13 data samples as anomaly i.e., FP while it incorrectly identified 9 anomalous samples as normal i.e., FN.

We showed the performance of the proposed model by using AUC-ROC curve in Fig. 3.16 This curve displays the trade-off between the TPR and FPR of the proposed model. The figure displays the ROC curves for the LSTM-AE model

implementing non-HE, 128-bit HE, and 256-bit HE. The LSTM-AE model with different homomorphic encryption strategies is represented by respective colors, where the anomalies are detected based on the test datasets. The results show that model with non-HE performs better with a high value of TPR and a low value of FPR and on the flipside, the model implementing HE-256-bit key has degraded performance. The ROC curve shows that the model performance implementing HE-128bit lies in between non-HE and HE-256. Fig. 3.17 evaluates the training loss performance on the server side. The server training loss takes some iteration to converge the loss. Similarly, Fig. 3.18 shows the training loss performance on the clients' side. The training loss performance on the client side converges significantly as compared to the server. The training loss of each of the three clients converges at around 2 to 3 epochs with a learning rate of 0.001 showing better performance of the proposed framework.

Table 3.5: Comparison of the proposed framework based on MSD Approach with threshold value

Model	**HE**	**Recall**	**Precision**	**Accuracy**	**F1-Score**	**Threshold**
LSTM-AE	Non-HE	93%	94%	93%	93%	0.00243
	HE-128	79%	86%	82%	80%	0.00693
	HE-256	70%	80%	74%	70%	0.01126

The detailed comparison of LSTM-AE based on various performance metrics with MSD evaluation approach is given in the table below. Table 3.5 compares the proposed framework for the MSD approach with HE using various encryption keys as a privacy-preserving technique. The standard measurements for measuring performance, such as the threshold values, are used to make the comparison. While assessing the efficacy of the suggested ADA-FL framework's AE model, the Recall performance parameter is crucial. The Recall metric minimizes the number of false negative predictions, in other words, we want to make sure that all the data samples that are anomalous (TP) are correctly detected because missing an anomaly (FN) can have a significant consequence for the proposed system. From the table, it is obvious that the LSTM-AE models perform better without the HE strategy. Even if we select the HE-128 bit key for preserving the privacy of the system, LSTM-AE has a higher Recall percentage of 79% as well as less reconstruction error threshold than HE-256 bit. F1-score is one of the most commonly used metrics for evaluating the model. The F1-score balances precision and recall, offering a single value that shows the model's

overall performance. The HE-128 bit has F1-score of 80% while HE-256bit has 70% as can be seen in Table 3.5. Similarly, Table 3.6 compares the proposed

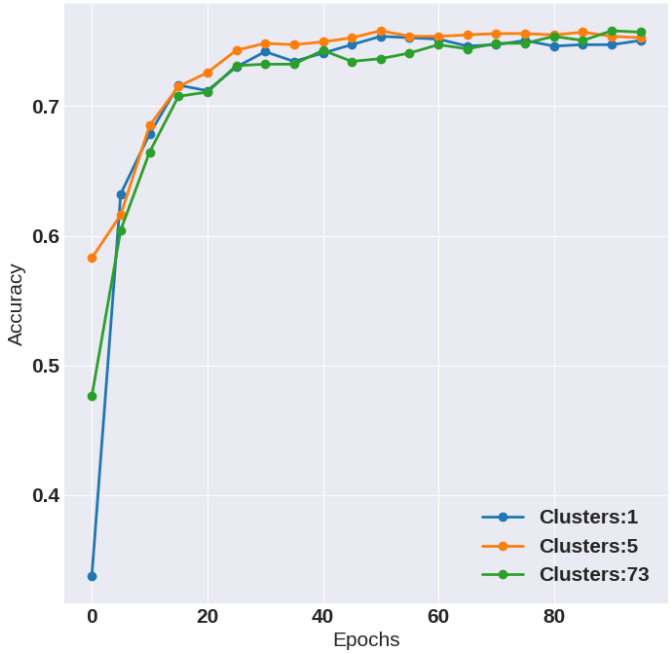
Table 3.6: Comparison of the proposed framework based on the MAD-Score Approach

Model	HE	Recall	Precision	Accuracy	F1-Score
LSTM-AE	Non-HE	81%	89%	84%	83%
	HE-128	71%	85%	76%	72%
	HE-256	68%	82%	73%	67%

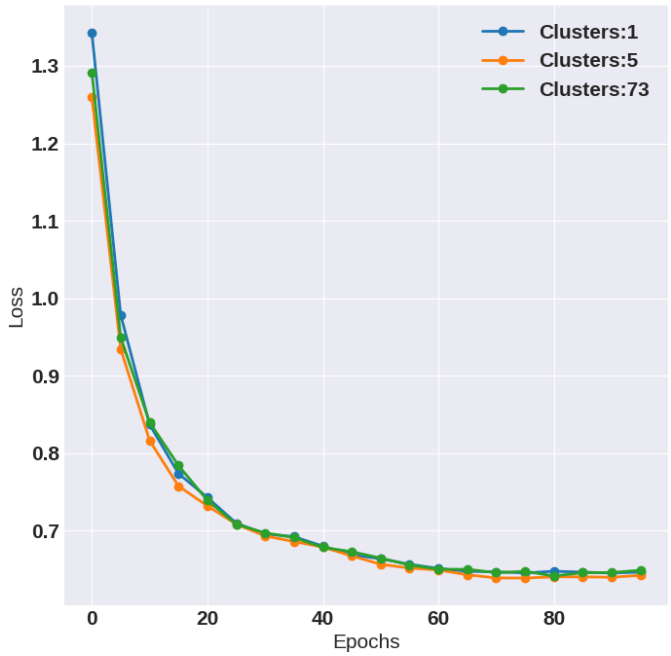
framework based on the MAD-score approach with the LSTM-AE model. The comparison is based on standard performance evaluation metrics considering different homomorphic encryption strategies. From the table, it can be seen that LSTM-AE performs better without homomorphic encryption. However, the performance of LSTM-AE slightly degrades when the HE-128-bit key and HE-256-bit key are implemented in the proposed framework. The HE-128bit key has a recall value of 71% and an F1-score of 72% which is slightly higher than that of the HE-256bit, which has a Recall value of 68% and an F1-score of 67%. Similarly, the HE-128 bit performs better than HE-256bit in terms of Precision and Accuracy with values 85% and 76% respectively. This is due to the fact that LSTM-AE consists of a higher number of parameters space and these parameters can change a bit during the encryption and decryption process.
Moreover, from Table 3.5 and Table 3.6, we can see that the MSD approach can detect anomalies effectively by considering Recall, Precision, Accuracy, and F1-score compared to the MAD score approach. Even while implementing the homomorphic encryption, the anomaly detection performance of MSD is better than that of the MAD score approach.

We analyzed the computation cost of implementing HE in the proposed framework using LSTM-AE. We compare the results by using 128-bit key, 256-bit keys, and without homomorphic encryption in the proposed framework. The result of computation time taken by the LSTM-AE methods with varying HE encryption keys is given in Fig. 3.19 The execution time taken to perform LSTM-AE without implementing HE is 29.61s. The execution time taken to perform LSTM-AE with implementing HE-128bit is 1387s and HE-256bit is 4662s. In practice, there is a trade-off between performance and computation time. A higher key length provides a superior level of protection against various attacks. On the other hand, using a higher key length incurs a computational overhead, causing slower en-

cryption and decryption times, as well as higher memory requirements. In general, key length should be chosen depending on a trade-off between security and performance needs. A higher key length, such as 256 bits, may be more appropriate if a high degree of privacy and security is desired. If performance is required, a shorter key length, such as 128 bits, may suffice.



(a)



(b)

Figure 3.6: Performance evaluation on the convergence of parameter randomization method comparing to the normal SER federated learning system, (a) Accuracy, (b) Loss.

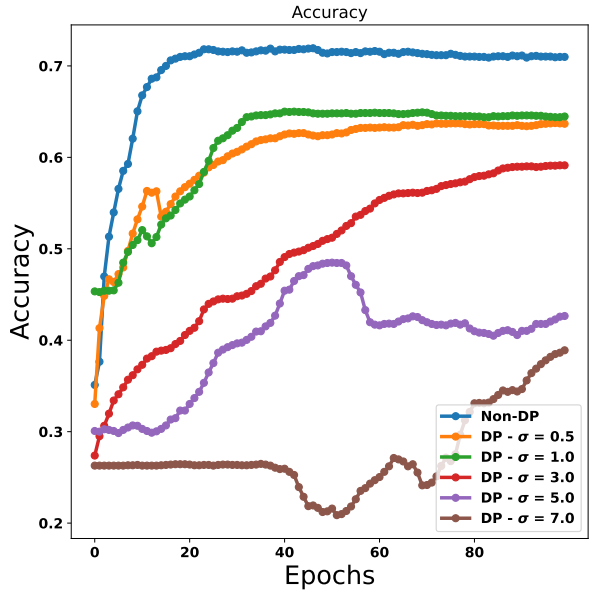


Figure 3.7: Performance evaluation on the Accuracy of Fatigue Detection-Federated Learning mechanism with different noise scales (σ)

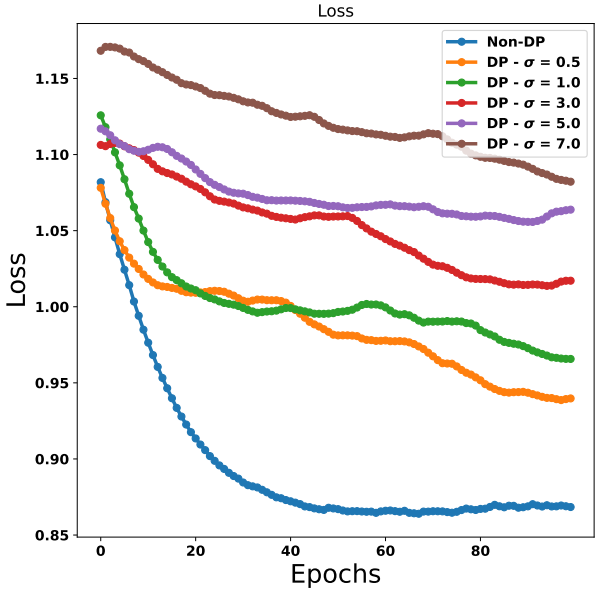


Figure 3.8: Performance evaluation on the Loss of Fatigue Detection-Federated Learning mechanism with different noise scales (σ)

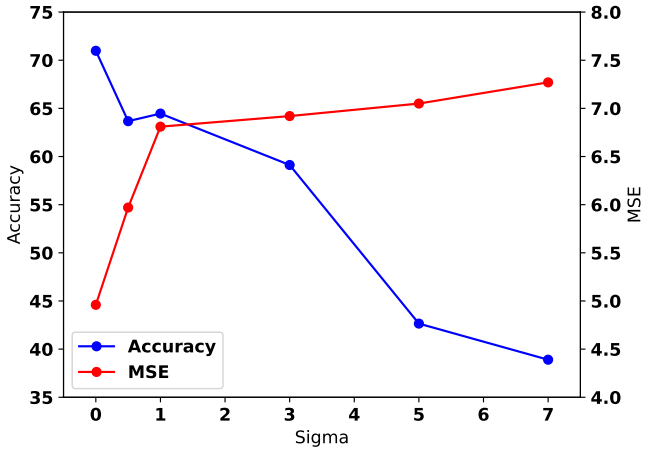


Figure 3.9: Privacy and accuracy trade-offs using noise scale σ, attack effectiveness by MSE, and Fatigue Detection-Federated Learning model accuracy.

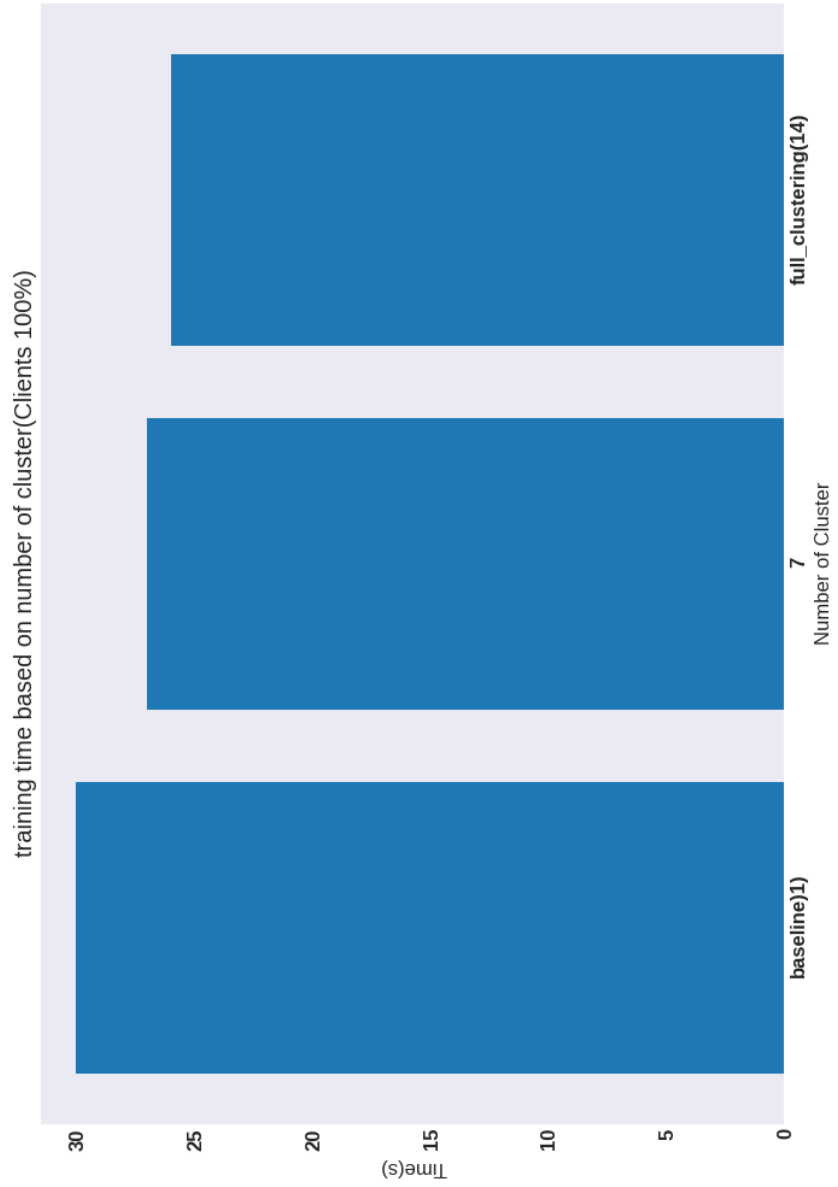
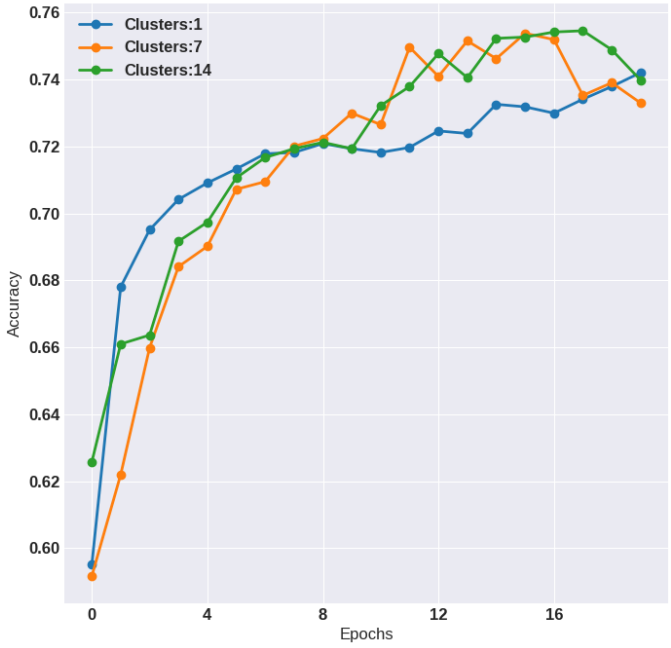
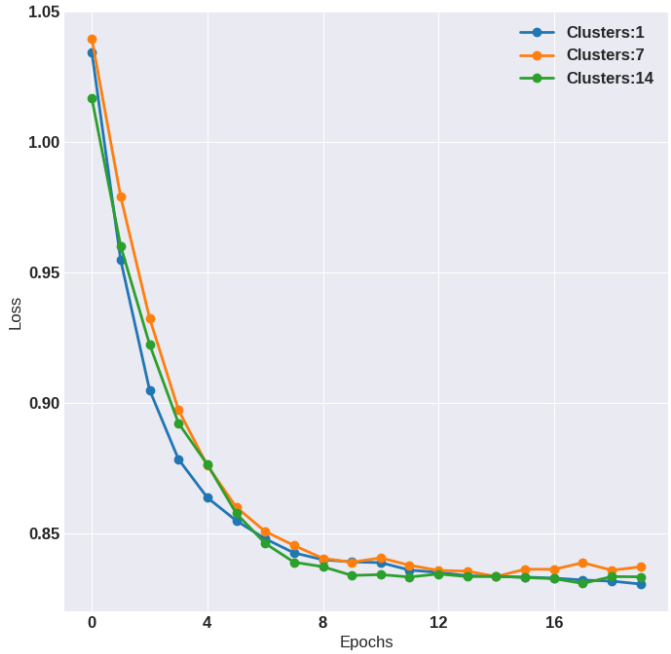


Figure 3.10: Parameter Randomization on fatigue Detection: Effect of different number of clusters on the training time of federated learning system.

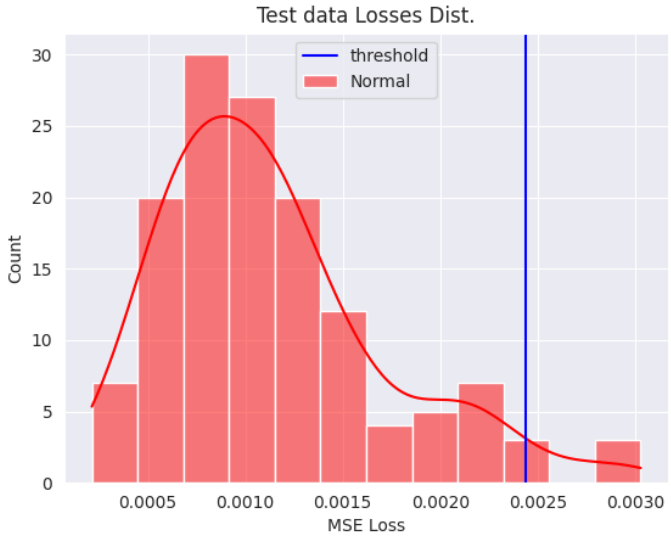


(a)



(b)

Figure 3.11: Performance evaluation on the convergence of parameter randomization method comparing to the normal fatigue detection federated learning system, (a) Accuracy, (b) Loss.



(a)

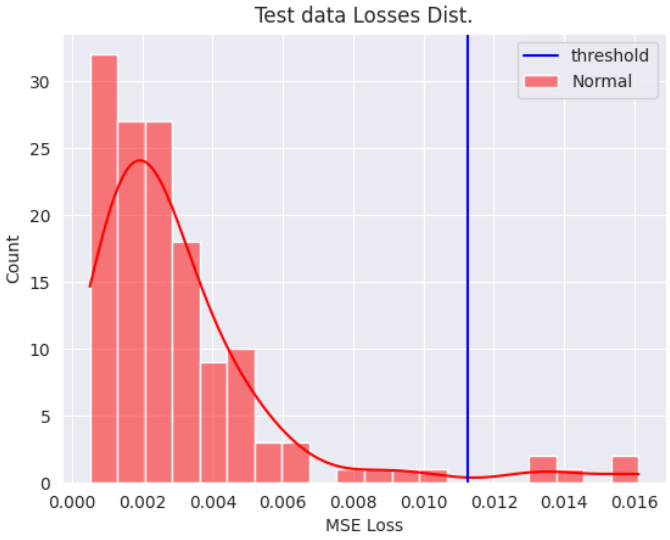
(b)

Figure 3.12: Test data loss distribution for normal and anomaly data without HE and threshold of 0.00243 .

(a)

(b)

Figure 3.13: Test data loss distribution for normal and anomaly data HE-128bit and threshold of 0.0069 .



(a)

(b)

Figure 3.14: Test data loss distribution for normal and anomaly data HE-256 bit and threshold of 0.01126 .

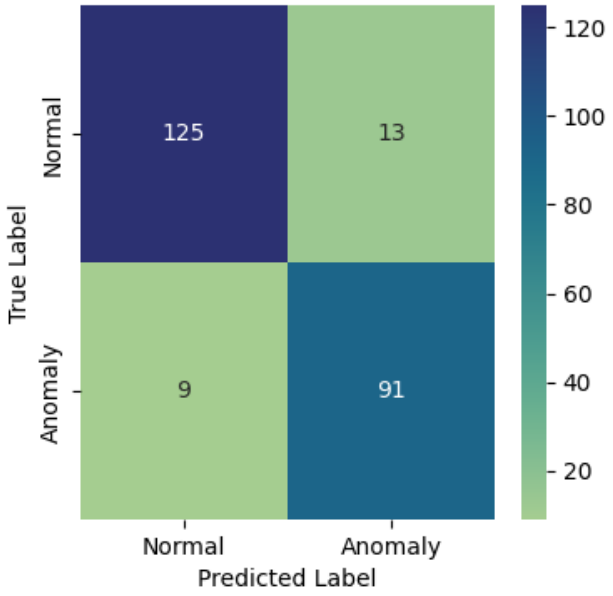


Figure 3.15: Confusion matrix showing model performance.

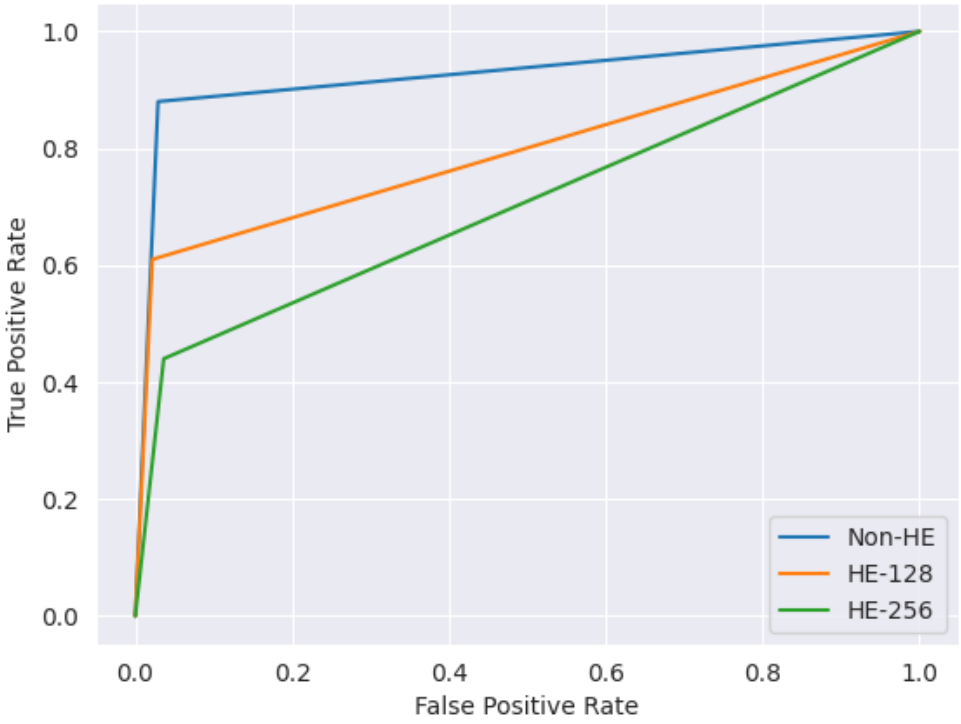


Figure 3.16: AUC-ROC of the model

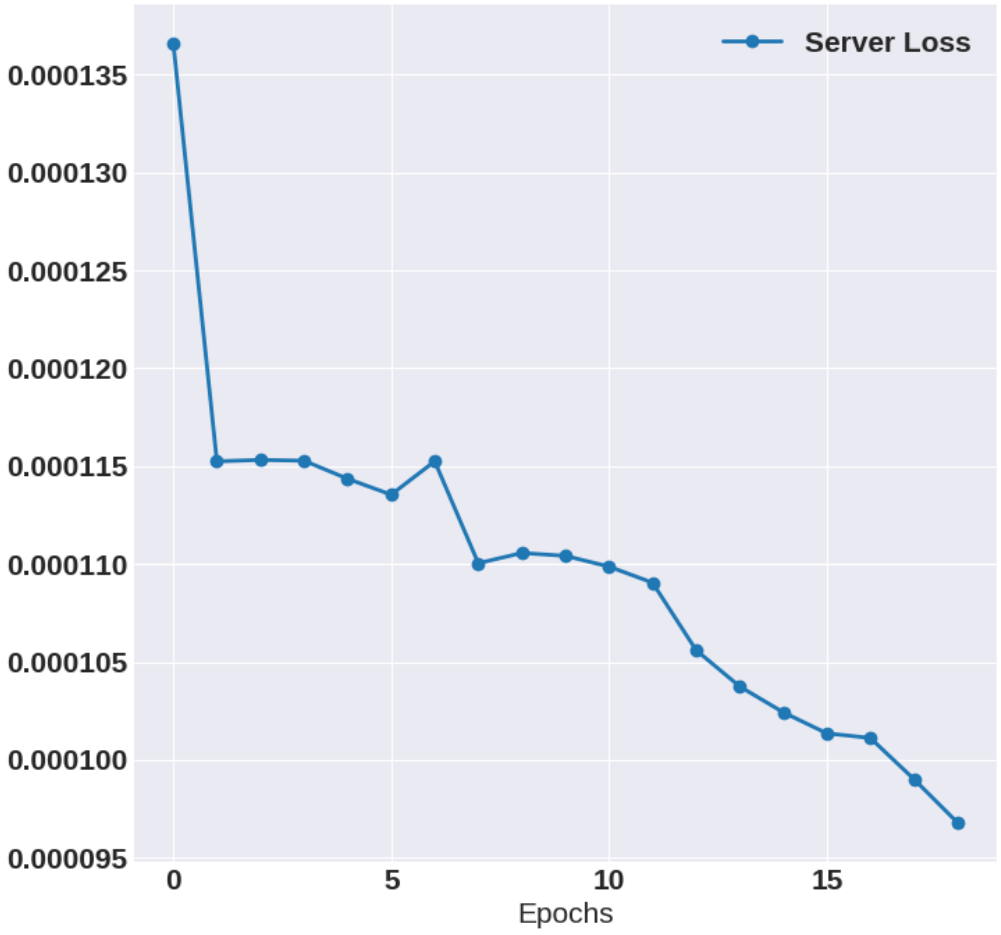


Figure 3.17: Traning loss performance at server side Fig.11 Training loss at client side

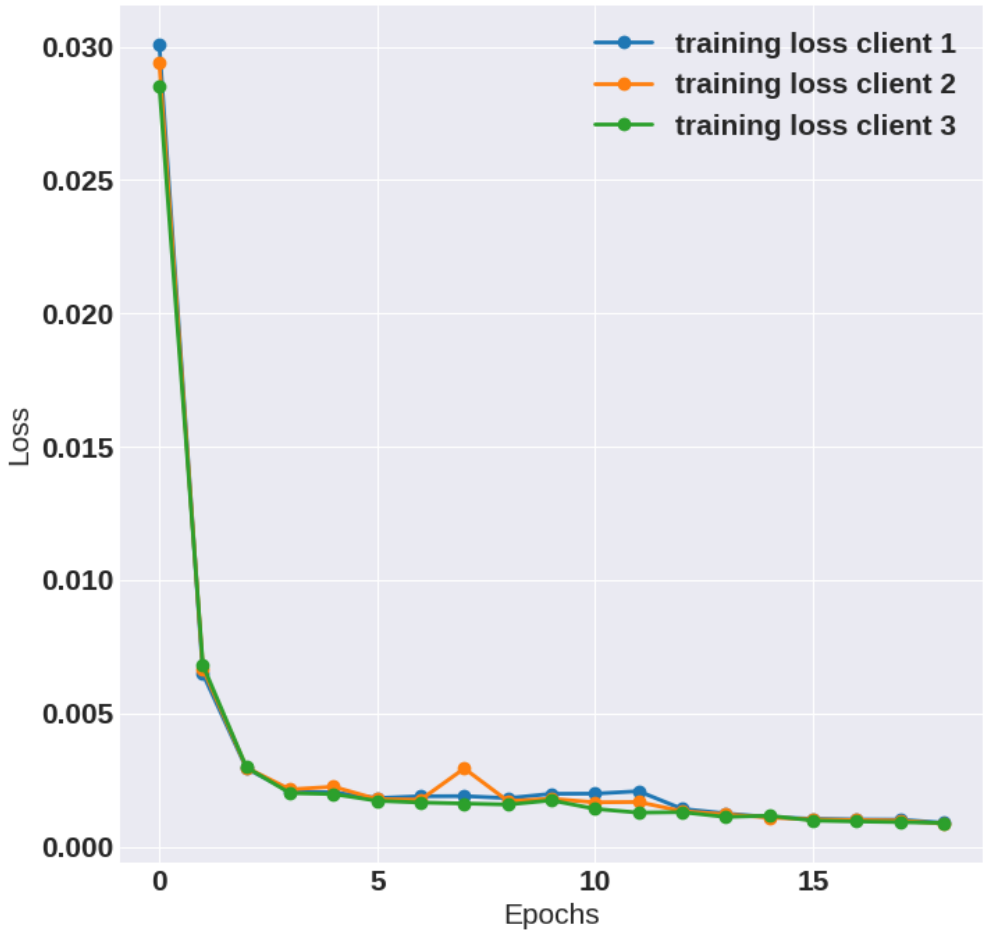


Figure 3.18: Training loss at client side

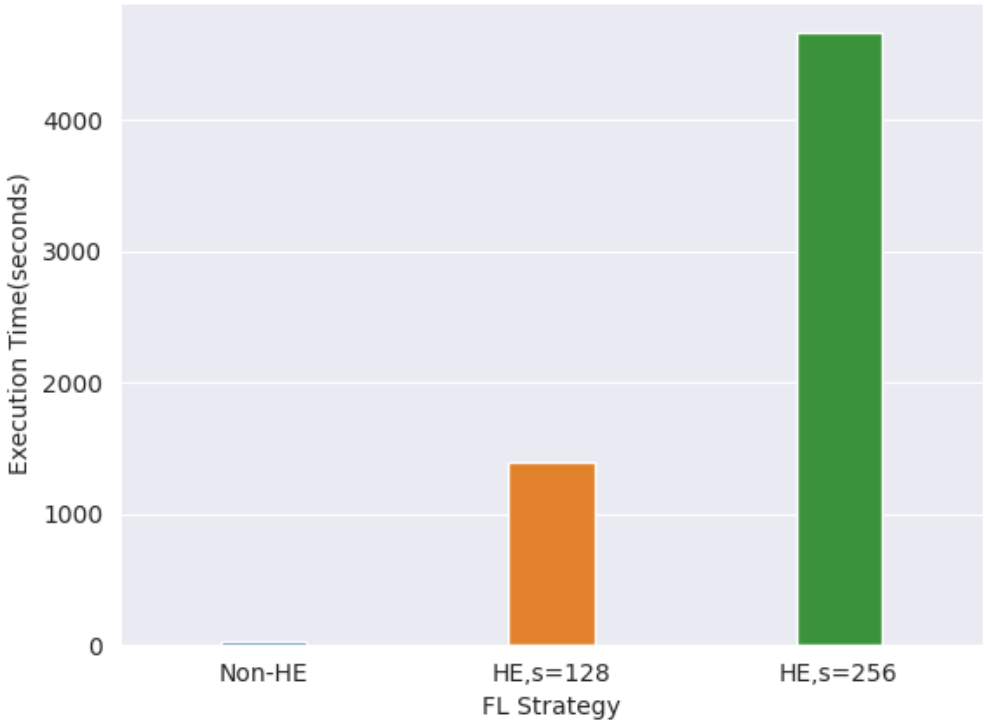


Figure 3.19: Execution time taken by LSTM-AE with varying HE encryption keys in the proposed framework.

4

Conclusions and Future Works

There is a lot of room for growth in the study of federated learning, which is still a relatively new academic discipline. Federated learning has been dissected in this thesis, from its structure to its pros and cons. Some future lines of inquiry that might build upon the findings in this thesis are discussed below.

Explore the effects of various communication protocols on federated learning as a potential field for future study. For the time being, most federated learning approaches depend on master servers to coordinate the learning. Recent studies, nevertheless, have looked at decentralized methods that can make federated learning more scalable and reliable. Consequently, in the future, we may look at other communication protocols for federated learning, such as blockchain-based methods.

Improving the efficiency and effectiveness of federated learning algorithms is another crucial area for future study. Given the dispersion and variety of data sources, federated learning may be more difficult than conventional machine learning. Consequently, it is crucial for the success of federated learning to build more robust and adaptable algorithms that can manage these issues. Creating novel methods for model compression, regularisation, and transfer learning are all possible outcomes.

The legal and ethical ramifications of federated learning might be investigated in further studies. Data privacy and security can be improved by federated learning since it allows for cooperation between numerous parties without the need to share data. On the other hand, it prompts concerns about data privacy, security, and responsibility. Consequently, in the future, researchers may look

at the moral and legal consequences of federated learning and provide solutions to these problems.

The term "adversarial attacks" is used to describe an adversary's deliberate attempt to undermine a federated learning system's reliability by tampering with its data or model parameters. Federated learning is a distributed learning system in which several clients or devices work together to train a single shared model without transmitting or storing any underlying raw data. An attacker in such a setup has the option of either attacking the clients or the server. It is possible to use the defense ideas to prevent adversarial attacks and attack's transferability from [63], [64] and [65]. These ideas has satisfactory results on decentralized deep learning and there is a possibility to have also good results in federated learning learning to build secure models against adversarial attacks.

Lastly, federated learning has numerous possible uses in many different industries, including medicine, finance, and the Internet of Things. Hence, further study might examine the unique needs and difficulties of adopting federated learning in these areas, and then design and execute appropriate responses.

To sum up, federated learning is a fascinating and fruitful area with several open research questions. There are several potentials for future studies to expand and improve the methodologies given in this thesis, and the research presented here provides the groundwork for further exploration of the problems and opportunities of federated learning.

Because of its promise to provide privacy-preserving and decentralized training of models, federated learning, a novel and promising method of machine learning, has gained favor in recent years. The scope, structure, benefits, and drawbacks of federated learning have all been examined throughout this thesis.

Findings from this thesis's study point to the promise of federated learning as a solution to some of the most pressing issues plaguing conventional machine learning today, including data privacy, data dissemination, and data heterogeneity. Using federated learning, many parties may work together on training models without compromising their privacy or security. This allows them to pool their expertise while still protecting their personal information.

The communication overhead and the requirement for strong communication protocols are two of the key obstacles of federated learning. Furthermore, the quality of the model may be compromised by the participants' varying degrees of access to resources.

In spite of these obstacles, this thesis shows that federated learning may be as

effective as, and even more so than, centralized methods. In addition, federated learning may pave the way for previously unimaginable uses, such as in tailored medicine and at the network's periphery with edge computing.

Finally, the thesis demonstrated that federated learning is a viable alternative to standard machine learning that may address some of the most pressing problems with the latter. Notwithstanding the need for greater study to overcome the existing obstacles, federated learning has the potential to offer a more decentralised and privacy-preserving approach to data analysis, which might significantly impact the way machine learning models are trained.

References

[1] Chen Zhang et al. "A survey on federated learning". In: *Knowledge-Based Systems* 216 (2021), p. 106775. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.106775. URL: https://www.sciencedirect.com/science/article/pii/S0950705121000381.

[2] Jan Philipp Albrecht. "How the GDPR Will Change the World". In: *European Data Protection Law Review* 2 (2016), pp. 287–289.

[3] Viraaji Mothukuri et al. "A survey on security and privacy of federated learning". In: *Future Generation Computer Systems* 115 (2021), pp. 619–640. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.10.007. URL: https://www.sciencedirect.com/science/article/pii/S0167739X20329848.

[4] Dinh C. Nguyen et al. "Federated Learning for Internet of Things: A Comprehensive Survey". In: *IEEE Communications Surveys Tutorials* 23.3 (2021), pp. 1622–1658. DOI: 10.1109/COMST.2021.3075439.

[5] Brendan McMahan and Daniel Ramage. *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[6] Priyanka Mary Mammen. *Federated Learning: Opportunities and Challenges*. 2021. DOI: 10.48550/ARXIV.2101.05428. URL: https://arxiv.org/abs/2101.05428.

[7] Mingzhe Chen et al. "Communication-efficient federated learning". In: *Proceedings of the National Academy of Sciences* 118.17 (2021), e2024789118. DOI: 10.1073/pnas.2024789118. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2024789118. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2024789118.

[8] *DAIS (Distributed Artificial Intelligent System)*. https://dais-project.eu/.

[9] 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. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1322–1333. ISBN: 9781450338325. DOI: 10.1145/2810103.2813677. URL: https://doi.org/10.1145/2810103.2813677.

[10] Jakub Konený et al. *Federated Learning: Strategies for Improving Communication Efficiency*. 2016. DOI: 10.48550/ARXIV.1610.05492. URL: https://arxiv.org/abs/1610.05492.

[11] Joost Verbraeken et al. "A Survey on Distributed Machine Learning". In: *ACM Comput. Surv.* 53.2 (Mar. 2020). ISSN: 0360-0300. DOI: 10.1145/3377454. URL: https://doi.org/10.1145/3377454.

[12] Ron Bekkerman, Mikhail Bilenko, and John Langford. "Scaling up Machine Learning: Parallel and Distributed Approaches". In: *Proceedings of the 17th ACM SIGKDD International Conference Tutorials*. KDD '11 Tutorials. San Diego, California: Association for Computing Machinery, 2011. ISBN: 9781450312011. DOI: 10.1145/2107736.2107740. URL: https://doi.org/10.1145/2107736.2107740.

[13] *Distributed Machine Learning Part 2 Architecture*. https://www.studytrails.com/2021/02/10/distributed-machine-learning-2-architecture/. Accessed: 2023-01-24.

[14] Michael Ferdman et al. "Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware". In: *SIGPLAN Not.* 47.4 (Mar. 2012), pp. 37–48. ISSN: 0362-1340. DOI: 10.1145/2248487.2150982. URL: https://doi.org/10.1145/2248487.2150982.

[15] Felix Sattler et al. "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2020), pp. 3400–3413. DOI: 10.1109/TNNLS.2019.2944481.

[16] *Federated learning*. https://en.wikipedia.org/wiki/Federated_learning. Accessed: 2023-01-25.

[17] Shuyue Wei et al. "Efficient and Fair Data Valuation for Horizontal Federated Learning". In: *Federated Learning: Privacy and Incentive*. Ed. by Qiang Yang, Lixin Fan, and Han Yu. Cham: Springer International Publishing, 2020, pp. 139–152. ISBN: 978-3-030-63076-8. DOI: 10.1007/978-3-030-63076-8_10. URL: https://doi.org/10.1007/978-3-030-63076-8_10.

[18] Qiang Yang et al. "Federated Machine Learning: Concept and Applications". In: *ACM Trans. Intell. Syst. Technol.* 10.2 (Jan. 2019). ISSN: 2157-6904. DOI: 10.1145/3298981. URL: https://doi.org/10.1145/3298981.

[19] 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. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 603–618. ISBN: 9781450349468. DOI: 10.1145/3133956.3134012. URL: https://doi.org/10.1145/3133956.3134012.

[20] Le Trieu Phong et al. "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption". In: *IEEE Transactions on Information Forensics and Security* 13.5 (2018), pp. 1333–1345. DOI: 10.1109/TIFS.2017.2787987.

[21] Yong Cheng et al. "Federated Learning for Privacy-Preserving AI". In: *Commun. ACM* 63.12 (Nov. 2020), pp. 33–36. ISSN: 0001-0782. DOI: 10.1145/3387107. URL: https://doi.org/10.1145/3387107.

[22] Li Wan et al. "Privacy-Preservation for Gradient Descent Methods". In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '07. San Jose, California, USA: Association for Computing Machinery, 2007, pp. 775–783. ISBN: 9781595936097. DOI: 10.1145/1281192.1281275. URL: https://doi.org/10.1145/1281192.1281275.

[23] Yang Liu et al. *Vertical Federated Learning*. 2022. DOI: 10.48550/ARXIV.2211.12814. URL: https://arxiv.org/abs/2211.12814.

[24] Sudipan Saha and Tahir Ahmad. *Federated Transfer Learning: concept and applications*. 2020. DOI: 10.48550/ARXIV.2010.15561. URL: https://arxiv.org/abs/2010.15561.

[25] Yang Liu et al. "A Secure Federated Transfer Learning Framework". In: *IEEE Intelligent Systems* 35.4 (2020), pp. 70–82. DOI: 10.1109/MIS.2020.2988525.

[26] Fuzhen Zhuang et al. "A Comprehensive Survey on Transfer Learning". In: *Proceedings of the IEEE* 109.1 (2021), pp. 43–76. DOI: 10.1109/JPROC.2020.3004555.

[27] Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 20–22 Apr 2017, pp. 1273–1282. URL: https://proceedings.mlr.press/v54/mcmahan17a.html.

[28] Tian Li et al. *Federated Optimization in Heterogeneous Networks*. 2018. DOI: 10.48550/ARXIV.1812.06127. URL: https://arxiv.org/abs/1812.06127.

[29] Chuhan Wu et al. "Communication-efficient federated learning via knowledge distillation". In: *Nature Communications* 13.1 (Apr. 2022). DOI: 10.1038/s41467-022-29763-x. URL: https://doi.org/10.1038%2Fs41467-022-29763-x.

[30] Sijie Cheng et al. *FedGEMS: Federated Learning of Larger Server Models via Selective Knowledge Fusion*. 2021. DOI: 10.48550/ARXIV.2110.11027. URL: https://arxiv.org/abs/2110.11027.

[31] Daliang Li and Junpu Wang. *FedMD: Heterogenous Federated Learning via Model Distillation*. 2019. DOI: 10.48550/ARXIV.1910.03581. URL: https://arxiv.org/abs/1910.03581.

[32] Viraaji Mothukuri et al. "A survey on security and privacy of federated learning". In: *Future Generation Computer Systems* 115 (2021), pp. 619–640. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.10.007. URL: https://www.sciencedirect.com/science/article/pii/S0167739X20329848.

[33] Adrian Nilsson et al. "A Performance Evaluation of Federated Learning Algorithms". In: *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*. DIDL '18. Rennes, France: Association for Comput-

ing Machinery, 2018, pp. 1–8. ISBN: 9781450361194. DOI: 10.1145/3286490.3286559. URL: https://doi.org/10.1145/3286490.3286559.

[34] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. DOI: 10.48550/ARXIV.1503.02531. URL: https://arxiv.org/abs/1503.02531.

[35] Malhar S. Jere, Tyler Farnan, and Farinaz Koushanfar. "A Taxonomy of Attacks on Federated Learning". In: *IEEE Security Privacy* 19.2 (2021), pp. 20–28. DOI: 10.1109/MSEC.2020.3039941.

[36] 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. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 603–618. ISBN: 9781450349468. DOI: 10.1145/3133956.3134012. URL: https://doi.org/10.1145/3133956.3134012.

[37] 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. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1322–1333. ISBN: 9781450338325. DOI: 10.1145/2810103.2813677. URL: https://doi.org/10.1145/2810103.2813677.

[38] Reza Shokri et al. *Membership Inference Attacks against Machine Learning Models*. 2016. DOI: 10.48550/ARXIV.1610.05820. URL: https://arxiv.org/abs/1610.05820.

[39] Lingjuan Lyu et al. "Privacy and Robustness in Federated Learning: Attacks and Defenses". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022), pp. 1–21. DOI: 10.1109/TNNLS.2022.3216981.

[40] Lingjuan Lyu, Han Yu, and Qiang Yang. *Threats to Federated Learning: A Survey*. 2020. DOI: 10.48550/ARXIV.2003.02133. URL: https://arxiv.org/abs/2003.02133.

[41] Ligeng Zhu, Zhijian Liu, and Song Han. "Deep Leakage from Gradients". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf.

[42] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. *iDLG: Improved Deep Leakage from Gradients*. 2020. DOI: 10.48550/ARXIV.2001.02610. URL: https://arxiv.org/abs/2001.02610.

[43] Stacey Truex et al. "LDP-Fed: Federated Learning with Local Differential Privacy". In: *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. EdgeSys '20. Heraklion, Greece: Association for Computing Machinery, 2020, pp. 61–66. ISBN: 9781450371322. DOI: 10.1145/3378679.3394533. URL: https://doi.org/10.1145/3378679.3394533.

[44] Cynthia Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Proceedings of the Third Conference on Theory of Cryptography*. TCC'06. New York, NY: Springer-Verlag, 2006, pp. 265–284. ISBN: 3540327312. DOI: 10.1007/11681878_14. URL: https://doi.org/10.1007/11681878_14.

[45] Cynthia Dwork. "Differential Privacy: A Survey of Results". In: *Theory and Applications of Models of ComputationTAMC*. Vol. 4978. Lecture Notes in Computer Science. Springer Verlag, Apr. 2008, pp. 1–19. URL: https://www.microsoft.com/en-us/research/publication/differential-privacy-a-survey-of-results/.

[46] Martin Abadi et al. "Deep Learning with Differential Privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2016. DOI: 10.1145/2976749.2978318. URL: https://doi.org/10.1145%2F2976749.2978318.

[47] Ahmed El Ouadrhiri and Ahmed Abdelhadi. "Differential Privacy for Deep and Federated Learning: A Survey". In: *IEEE Access* 10 (2022), pp. 22359–22380. DOI: 10.1109/ACCESS.2022.3151670.

[48] Raef Bassily et al. "Practical Locally Private Heavy Hitters". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran

Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3d779cae2d46cf6a8a99a35ba4167977-Paper.pdf.

[49] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy". In: *Foundations and Trendső in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.

[50] Fattaneh Bayatbabolghani and Marina Blanton. "Secure Multi-Party Computation". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 2157–2159. ISBN: 9781450356930. DOI: 10.1145/3243734.3264419. URL: https://doi.org/10.1145/3243734.3264419.

[51] Keith Bonawitz et al. *Practical Secure Aggregation for Federated Learning on User-Held Data*. 2016. DOI: 10.48550/ARXIV.1611.04482. URL: https://arxiv.org/abs/1611.04482.

[52] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. "FedBoost: A Communication-Efficient Algorithm for Federated Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 3973–3983. URL: https://proceedings.mlr.press/v119/hamer20a.html.

[53] Pengchao Han, Shiqiang Wang, and Kin K. Leung. "Adaptive Gradient Sparsification for Efficient Federated Learning: An Online Learning Approach". In: *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. 2020, pp. 300–310. DOI: 10.1109/ICDCS47774.2020.00026.

[54] Mohammad Mohammadi Amiri et al. *Federated Learning With Quantized Global Model Updates*. 2020. DOI: 10.48550/ARXIV.2006.10672. URL: https://arxiv.org/abs/2006.10672.

[55] Reza Ghoddoosian, Marnim Galib, and Vassilis Athitsos. "A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019), pp. 178–187.

[56] *UTA-RLDD*. https://sites.google.com/view/utarldd/home. Real Life Drowsines Dataset, accessed: 2022-10-20.

[57] Torbjörn Åkerstedt and Mats Gillberg. "Subjective and Objective Sleepiness in the Active Individual". In: *International Journal of Neuroscience* 52.1-2 (1990). PMID: 2265922, pp. 29–37. DOI: 10.3109/00207459008994241. eprint: https://doi.org/10.3109/00207459008994241. URL: https://doi.org/10.3109/00207459008994241.

[58] Adam Paszke et al. "Automatic differentiation in PyTorch". In: *NIPS-W*. 2017.

[59] Koukyosyumei. *AIJack: Security and Privacy Risk Simulator for Machine Learning*. 2022. URL: https://github.com/Koukyosyumei/AIJack.

[60] hackenjoe. *Advanced Drowsiness Detection*. Last commit October 19 2021. 2021. URL: https://github.com/hackenjoe/Advanced_Drowsiness_Detection.

[61] Davis E. King. "Dlib-ml: A Machine Learning Toolkit". In: *Journal of Machine Learning Research* 10 (2009), pp. 1755–1758.

[62] Rakesh Shrestha et al. "Machine-Learning-Enabled Intrusion Detection System for Cellular Connected UAV Networks". In: *Electronics* 10.13 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10131549. URL: https://www.mdpi.com/2079-9292/10/13/1549.

[63] Mauro Barni et al. *On the Transferability of Adversarial Examples Against CNN-Based Image Forensics*. 2018. arXiv: 1811.01629 [cs.CR].

[64] Ehsan Nowroozi et al. *Demystifying the Transferability of Adversarial Attacks in Computer Networks*. 2022. arXiv: 2110.04488 [cs.CR].

[65] Ehsan Nowroozi et al. *SPRITZ-1.5C: Employing Deep Ensemble Learning for Improving the Security of Computer Networks against Adversarial Attacks*. 2022. arXiv: 2209.12195 [cs.CR].

Acknowledgments

At this time, I'd want to thank everyone who has helped me out with this study.

Before anything else, I want to express my gratitude to my wife and my family for their unending love and encouragement while I pursued my academic goals. The presence of my family and friends has been the driving force behind my accomplishments, and I will be eternally thankful to them.

My supervisors, Dr. Sima Sinaei, Dr. Ehsan Nowroozi, and notably Professor Mauro Conti, deserve my sincere gratitude for their excellent advice, assistance, and knowledge. Their guidance has been crucial to my development as a researcher and scholar. Their understanding, encouragement, and insightful criticism have been invaluable to me as I've worked to hone my craft.

Furthermore, I am grateful to the University of Padua and the SPRITZ Security and Privacy Research Group for allowing me to pursue my academic interests. The academic environment, with its resources and facilities, has been essential to my success.

To the RISE Research Institute of Sweden AB, I am thankful for the chance to contribute to the DAIS project. Having worked with the firm and the great people, I have obtained great experience and insight into research.

Lastly, I'd want to express my gratitude to everyone who helped with this study effort. Without their help, this study would not have been feasible. Their involvement and input were crucial.