Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Engineering

Privacy Preservation in Federated Learning for the IoT

Ruairí Grant

Supervisor: Meriel Huggard

April 15, 2024

A dissertation submitted in partial fulfilment
of the requirements for the degree of
MAI (Computer and Electronic Engineering)

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Signed: _____ Date: _____

Abstract

The landscape of modern technological advancements is characterized by the pervasive integration of Internet of Things (IoT) devices and the burgeoning expansion of deep learning methodologies. This dissertation presents a comprehensive exploration into the realm of privacy within the context of Federated Learning (FL), with a particular focus on differential privacy - a common method of applying privacy to deep learning models. In addition, this work was motivated by potential applications in the medical domain where datasets are often imbalanced, as positive diagnoses for many diseases are rare, and are therefore, a minority class in most datasets. This research emulates a well-known differential privacy framework - Dopamine, in order to investigate the issue of imbalanced datasets. The findings show that differential privacy(DP) causes a significant decrease in the model's ability to learn minority datasets, which calls for further work on refining the techniques used to add differential privacy to deep learning. Moreover, this investigation is extended to assess the practical implications of deploying DP within constrained environments, exemplified by constrained devices. By deploying the framework onto such devices, we systematically examine the impact of DP on network and CPU utilization. Our analysis exposes a substantial increase in both the time required to complete an epoch and the CPU usage of the device following the integration of DP. These findings underscore the resource-intensive nature of DP, emphasizing the need for optimization strategies, particularly when operating within resource-constrained environments.

Acknowledgements

I would like to thank my parents and surrounding family, who have supported me throughout my academic journey.

To Ailís, and my friends and classmates who have been and continue to be a constant source of support and encouragement throughout the past years.

I would like to extend my gratitude to my supervisor, Meriel Huggard, for her guidance and support throughout this project. Her expertise and advice have been invaluable in shaping this work.

Finally, I would like to thank the School of Engineering for providing me with the opportunity to undertake this project and for the resources and support they have provided throughout my time at Trinity College Dublin.

Contents

1 Introduction **1**
1.1 Background . . . 1
1.2 Motivation and Objectives . . . 2

2 Technical Background and Literature Review **4**
2.1 Technical Background . . . 4
2.1.1 Deep Learning . . . 4
2.1.2 Federated Learning . . . 6
2.2 Literature Review . . . 7
2.2.1 Privacy Concerns in Federated Learning . . . 7
2.2.2 Privacy preservation techniques . . . 9
2.2.3 Privacy Metrics . . . 13
2.3 Constrained devices and fairness . . . 15
2.4 Conclusion . . . 15

3 Methodology **16**
3.1 Framework Selection . . . 16
3.2 Rock Pi Experiements . . . 18
3.2.1 Monitoring CPU and Network Usage . . . 18
3.2.2 Rock Pi setup . . . 19
3.2.3 Mnist Model . . . 20
3.2.4 Differential Privacy Implementation . . . 22
3.2.5 Flower Framework Overview . . . 22
3.2.6 Experiments . . . 22
3.3 Class Imbalance Experiments . . . 24
3.3.1 Diabetic Retinopathy Dataset . . . 25
3.3.2 Data Preprocessing and SqueezeNet . . . 25
3.3.3 Mnist imbalanced dataset experiments . . . 27
3.3.4 Experiments . . . 27
3.4 Summary . . . 28

4 Evaluation **29**
4.1 Rock Pi Experiements 29
4.1.1 Model Evaluation 29
4.1.2 Profiling Results 32
4.2 Class Imbalance Experiements 35
4.3 Diabetic Retinopathy Dataset 35
4.4 Imbalanced Mnist Dataset 37
4.5 Summary 38

5 Conclusion **39**
5.1 Conclusion 39
5.2 Further Work 40

A1 Appendix **46**

List of Figures

2.1 Overiew of FL with multiple nodes and a server [1] 6
2.2 Overiew of MPC [2] 10
2.3 Practical implementation of differential privacy in FL [3] 12

3.1 Samples of the Mnist dataset 21
3.2 Summary of the Mnist model 21
3.3 Flower Architecture Overview 23
3.4 Images of Diabetic Retinopathy of each class 26

4.1 Classification reports from the Rock Pi Experiments 30
4.2 Accuracy plots from the Rock Pi Experiments 31
4.3 Classification report from the Diabetic Retinopathy Experiments 36
4.4 Classification report from the Imbalanced Mnist experiments 38

List of Tables

2.1 Attack described in the 5W taxonomy [4] 8
2.2 MPC complexity [2] 10

3.1 Comparison of the different available frameworks for FL. 18
3.2 Hyperparameters used for the Rock Pi experiments 23
3.3 Class distribution of the dataset 25
3.4 Class distribution of the Mnist dataset after dropping samples 27
3.5 Hyperparameters used for the imbalanced dataset experiments 27

4.1 Results of the Rock Pi Profiling: High Accuracy Model 33
4.2 Results of the Rock Pi Profiling: Lower Training Time Model 33

Nomenclature

CPU	Central Processing Unit
DL	Deep Learning
DP	Differential Privacy
DPSGD	Differentially Private Stochastic Gradient Descent
DR	Diabetic Retinopathy
FL	Federated Learning
GDPR	General Data Protection Regulation
HE	Homomorphic Encryption
IoT	Internet of Things
ML	Machine Learning
MPC	Multi-Party Computation
SBC	Single Board Computer
SGD	Stochastic Gradient Descent

1 Introduction

1.1 Background

Advances in the Internet of Things(IoT) have made more data available at the edges of networks, and it allows for the development of data-hungry machine learning(ML) methods such as deep learning(DL). Medicine is one field where DL technologies provide the ability to help diagnose medical problems and assist doctors in making decisions, helping to progress their understanding of medicine [5]. It is vital some data, such as medical data, is protected and remains private, increasing concerns have been raised about the use of real medical data in ML models [6]. Federated learning (FL) [7], where an ML model is trained on distributed data, without requiring the data to be centralized to one location, provides the foundation to get the best out of both of the above.

As DL gains traction in a large number of fields, more work is being done to evaluate the privacy implications of these models. GDPR is one such example of legislation that has been introduced to protect people's data, and it is important to understand how these models can be used in a way that does not breach this legislation. [8]

Separately, the Internet of Things (IoT) is a rapidly growing field, with a large number of devices collecting data and making decisions on our behalf. Naturally, DL is integrated into these increasingly ubiquitous devices, which raises many questions on how to deal with the data collected by these devices that we may want to use in a DL model.

FL is a new paradigm on top of ML, where the training data exists exclusively on distributed devices, and each device trains its own model. This model is then shared with a server that aggregates each of the local model updates and distributes the updated model to the local devices. FL removes the requirement to move data to a central server, and although it is still possible to infer information about the training data, several studies in recent years have discussed various ways to ensure privacy for this training data [4]. A number of these methods, discussed in section 2.2, are computationally intensive which is an important consideration, especially on small devices typical for the IoT. Therefore it is necessary to understand how these privacy methods perform on constrained devices, where they may be

deployed.

1.2 Motivation and Objectives

The motivation for this dissertation is to investigate the privacy implications of Federated Learning on constrained devices, such as the Radixa Rock Pi. Federated learning has the potential to expand the reach of DL into constrained devices and medical settings, but privacy concerns need to be accounted for in both cases.

IoT is exciting and growing in many different areas. Due to its nature, it can facilitate the collection of lots of private data. While the development of comprehensive privacy frameworks is necessary it is important to balance the model's utility and privacy concerns. It is also important to try to quantify those privacy risks. This thesis hopes to contribute to this area.

The key objectives and goals of this dissertation are:

1. As a base goal, the existing techniques for protecting the privacy of training data will be reviewed. This will include a review of the literature on privacy-preserving methods for ML models, with a focus on FL models.
2. The available federated learning frameworks will be analyzed and reviewed for their suitability for deployment on constrained devices.
3. The methods implemented in Dopamine [9], a well-known privacy-preserving framework, will be implemented on the FLower framework. The performance of the models trained using these methods will be evaluated in the context of medical data.
4. FL models will be deployed to the Radixa Rock Pi SBC and the CPU and memory usage during the training process will be profiled. This will be compared with the computational requirements of the same model trained in a centralized manner.
5. Differential privacy, further discussed in chapter 2, will be implemented during the training process. The training will again be profiled for CPU and memory usage with the additional overhead of the privacy protections on the Rock Pi. The effect of these privacy measures on model accuracy will also be measured and compared.
6. Finally, as there is little clarity in the literature on privacy metrics for FL models, further investigation into specific metrics will be done, in particular about metrics available for centralized ML. These metrics will be evaluated for effectiveness on the trained models with various privacy features implemented.

As mentioned in section 1.1, the key issues identified are the need for large amounts of data for DL, and the importance of ensuring people's data privacy, in particular their medical data

privacy, is not breached. Objectives 1 and 2 provide a comprehensive overview of the current methods for ensuring data privacy is not breached, with a focus on constrained devices. Objectives 3 to 5 provide a practical application of these methods, allowing for a better understanding of how these methods perform with imbalanced datasets and on constrained devices. This can aid further research into the area allowing for more accurate medical models without risking people's privacy. Objective 6 could add to this research by understanding what metrics or tools may be available to help quantify and understand any potential privacy loss incurred by the use of these methods.

The following chapter provides an overview of the literature and technical background information relevant to the planned work. Chapter 3 then describes the implementation and experimental work conducted, while Chapter 4 provides a comprehensive evaluation of the outputs of this work. The dissertation concludes with a discussion of future research directions.

2 Technical Background and Literature Review

The field of machine learning has witnessed a significant surge in popularity in recent years, with deep learning algorithms emerging as a powerful tool for various applications. In particular, deep learning, which utilizes neural networks as its foundation, has demonstrated remarkable success in image-processing tasks, including medical data analysis. However, the availability of large amounts of private medical data poses challenges in training these models. To address this, federated learning has emerged as a promising approach, allowing distributed devices to collaboratively train a global model while preserving data privacy. In this chapter, we delve into the technical background of deep learning and federated learning, exploring their key concepts and methodologies. We also discuss the privacy concerns associated with federated learning and the various attacks that can compromise the confidentiality of training data. By understanding these foundations, we can better appreciate the significance of the research presented in this thesis.

2.1 Technical Background

2.1.1 Deep Learning

Deep Learning(DL) is a class of machine learning(ML) algorithms that have grown enormously in popularity in recent years [5]. These algorithms use neural networks as a backbone and typically require large amounts of data [10]. DL and a subset called Convolutional Neural Networks(CNNs) have been used extensively and successfully in image processing for medical data [6], and have been shown to be very useful in the medical field. As a result, there is a large amount of data available for training these models, but this data is often private and must be protected [6].

Neural networks calculate the output of the model based on a set of parameters, which are optimized during training. Stochastic gradient descent (SGD) is a common optimization algorithm used in training deep learning (DL) models. At its core, SGD aims to minimize a

predefined loss function that quantifies the difference between the model's predictions and the actual ground truth labels. This optimization algorithm is applied iteratively to the neural network to update its weights, with the goal of reducing the value given by the loss function.

During each iteration of SGD, a subset of the data, known as a mini-batch, of training examples is randomly sampled from the dataset. The model output is then computed for each sample in this batch of data, and compared with the ground truth using the loss function. The gradient of the loss function with respect to the model parameters is then computed using this mini-batch. This gradient indicates the direction of the steepest descent in the parameter space that would reduce the loss. This gradient calculation is repeated for batches of data until the entire dataset has been used, and this process as a whole is known as an epoch.

Once the gradient is computed, SGD updates the model parameters by taking a small step in the opposite direction of the gradient. This step is scaled by a parameter known as the learning rate, which controls the size of the updates. The learning rate is crucial in determining the convergence behavior of SGD; a larger learning rate may lead to faster convergence but risks overshooting the optimal solution, while a smaller learning rate may lead to slower convergence but ensures stability.

Typically, DL models are evaluated based on accuracy, which is the proportion of correctly classified examples in the dataset. However, accuracy alone may not provide a complete picture of the model's performance, especially in the presence of class imbalance or misclassification costs. Additional metrics such as precision, recall, and F1 score, are commonly used to evaluate the model's performance on specific classes or tasks. Precision measures the proportion of true positive predictions among all positive predictions, while recall measures the proportion of true positive predictions among all actual positive examples. The F1 score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance. These metrics are particularly useful in scenarios where class imbalance or misclassification costs are present, as they provide a more nuanced evaluation of the model's behavior. These values are calculated as part of the scikit-learn classification report [11], which is a common tool for the evaluation of classification models and is used in this project.

Privacy in the context of training data refers to the protection of sensitive information contained within the data used to train machine learning models. This is particularly important in the medical field, where patient data is often involved. Privacy concerns arise when training data contains personally identifiable information or other sensitive attributes that should remain confidential. The General Data Protection Regulation (GDPR) is a regulation in the European Union that aims to protect the privacy and personal data of

individuals. It imposes strict rules on the collection, storage, and processing of personal data, including data used for training machine learning models. Deep learning algorithms, which rely on large amounts of data, may be affected by GDPR as they require careful handling of training data to ensure compliance with privacy regulations [8]. Therefore, it is crucial to implement privacy-preserving techniques, such as federated learning, to train models while preserving the privacy of the underlying data.

2.1.2 Federated Learning

Federated Learning(FL) [7] is a new paradigm of ML, where the training data exists exclusively on distributed devices, typically referred to as clients. Each client has a local copy of the model, which is typically homogenous across all clients. In the case of horizontal FL, each client has the same features but different samples of data. This is the specific paradigm that this project focuses on. Vertical FL is a separate paradigm where each client may introduce new features, but this introduces different security concerns and is not a focus of the project.

Each client trains its model on the data available to it locally. This model is then shared with a server that aggregates each of the local model updates and distributes the updated model to the local devices. The overall architecture resembles the image in figure 2.1. Much research has been done into various aggregation algorithms for FL, and an overview is given below.

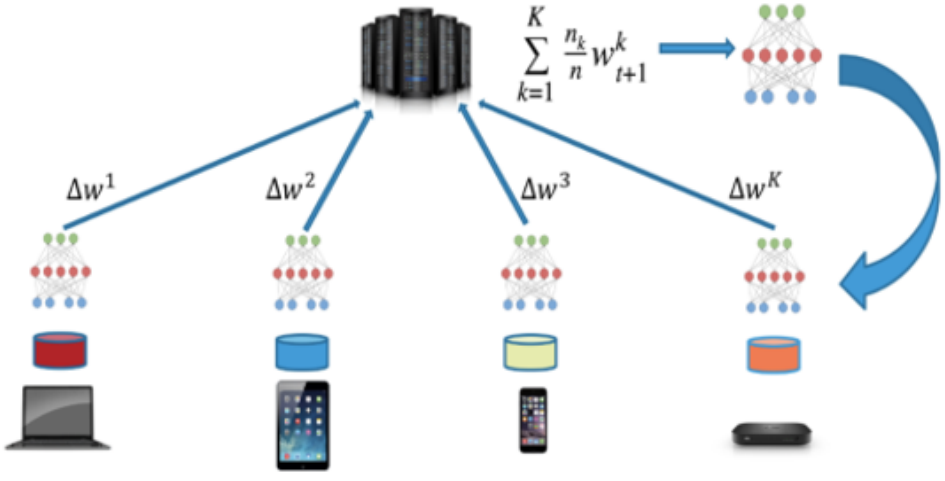


Figure 2.1: Overiew of FL with multiple nodes and a server [1]

There are two main classes of aggregation algorithms, FedSGD and FedAvg. In FedSGD, gradient updates are sent to the server from each node after each epoch, and the server

aggregates the gradient updates and applies SGD on the global model before broadcasting the updated global weights back to the individual clients. FedAvg works similarly, but the SGD algorithm is computed on each model locally, and the updated model weights are sent to the server for aggregation. FedAvg allows the local device to train more than one epoch at a time, decreasing the communication overhead at the cost of increased computation overhead for the client. In both cases, the client sends either the gradient update or the model weights to the server. This transfer of data is where privacy concerns can arise, as the data can be used to infer information about the training data, as discussed further in section 2.2.1 below.

FL can typically approach the accuracy achieved by a centralized model and has the advantage that it allows access to data that would otherwise be inaccessible due to privacy laws such as GDPR as well as being more efficient and cheaper when it comes to the IoT [12].

2.2 Literature Review

2.2.1 Privacy Concerns in Federated Learning

While McMahan et al. [7] showed that although the data remains on the computer on which the model was trained, data privacy is not guaranteed, as the model updates or the model itself can be used to infer information about the training data. However, FL is still more private than a centralized model, and additional protections can be added to the federated training process to further protect the privacy of the training data. The first step in protecting the privacy of the training data is to understand the different types of attacks that can be performed during the training process, and these are now considered in more detail.

Several different attacks have been found on the FL training process and the final trained model, that could be used to infer information about the training data as a whole or specific instances of training data. This violates the privacy that may be required from the model as discussed in section 2.1.1. Yin et al. [4] describe a taxonomy of privacy leakage attacks in FL. There are a large number of different classes of attacks on FL models, both to disrupt training and to gain insight into the training data. The 5W taxonomy divides the types of attacks based on five categories, summarized below:

Who: This concerns where the attack originates, it can be Insiders: Malicous clients and servers involved in the training process, or Oustiders: Model Consumers and Eavesdroppers, who are not part of the training process.

What: This divides the attacks into passive and active attacks, where an active attack aims

to alter or disrupt the federated training process and passive attacks do not.

When: A privacy attack can happen during the federated training process or after training is complete and the model is being used for inference.

Where: Three parts of the algorithm can be attacked, the Weight Update, Gradient Update, and the Trained Model.

Why: Finally the desired outcome of the attack is split into four groups, Inferring Class Representatives, Inferring Membership, Inferring Properties of Private Training data, and Inferring Training Input and Labels.

The attacks this dissertation focuses on are described using the 5W taxonomy in table 2.1.

The concerns of privacy differ depending on whether it is an insider or outsider attack. A malicious server may use the model weights or gradient updates to infer information about the training data, whereas a malicious client has less access to these updates. A malicious server could also use the trained model to infer this information. Outsider attacks are more limited as they have less access to the information, and they cannot perform attacks over and above those that a malicious server can.

Passive attacks are attacks that do not alter the system and therefore can be more difficult to detect. Primarily the goal of these passive attacks is to infer information about the training data. Active attacks are attacks that alter the system and can be easier to detect. The goal of these attacks is to disrupt the training process, and much research has been done on detecting these attacks [13] [14] [15].

This project focuses on passive attacks, as the goal is to understand the privacy implications of FL. As shown in the table 2.1, passive attacks can occur on the weight or gradient updates or on the trained model and are discussed in more detail below. These passive attacks are specifically known as passive white box attacks as they can access the

Who	What	Where
Malicious Servers Outsiders	Passive Attacks	Training Phase Inference Phase
When	**Why**	
Weight Update Gradient Update Trained Model	Inferring Class Representitives Inferring Membership Inferring Properties of Private Training Data Inferring Training Input and Labels	

Table 2.1: Attack described in the 5W taxonomy [4]

intermediate updates during the training process, whereas passive black box attacks can only query the model result [4].

All three parts of the where category can be attacked, and they are all considered in this dissertation.

Many attacks have been performed on both the gradient/weight update and the trained model. These attacks can be used to infer information about the training data. For example, Zhu et al. performed an attack on the gradient update of multiple models, with both datasets consisting of images and natural language [16]. This paper used the model weights and the gradient update to optimize a set of dummy data with respect to the weights as opposed to optimizing the weights with respect to the data as is done in machine learning. This successfully recovered exact text and recognizable images from the training data. Attacks are also performed on the weight update, but typically to less success [16]. Nevertheless, the weights can still leak private information as shown by Geng et al. [17].

Suliman et al. [18] designed an attack on the trained model using two models to infer the contents of the training data of a word prediction model. This is categorized as inferring training input and labels. As can be seen from the variety of attacks mentioned above, the privacy of the training data is a significant concern in FL, and it is necessary to add additional privacy-preserving techniques to the training process as is discussed. These will now be considered in more detail.

2.2.2 Privacy preservation techniques

The 5W taxonomy also describes the different types of privacy preservation schemes as one of four categories: Encryption-based, Perturbation-based, Anonymization-based, and Hybrid. These schemes have had some success in preventing or reducing the effectiveness of privacy attacks. These schemes are all software-based, and while some hardware-based privacy preservation techniques do exist, they have been shown not to work without additional software-based methods [19], and are not the focus of this project.

The primary Encryption-based techniques used in FL are secure multi-party computation(MPC) and homomorphic encryption(HE).

Secure multi-party computation enables distributed participants to collaborate and not reveal their data. Formally, N participants $P_1, P_2, ..., P_n$ compute a global function $f(\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n)$ based on each individual dataset $\mathcal{D}_i, \forall i \in N$. An MPC protocol must compute a global function that is correct and the protocol must not reveal private information of $\mathcal{D}_i$ to other participants [4].

A practical implementation of MPC in FL is the SecAgg(+) protocal [19] shown in figure

2.2. In this implementation, each client generates a public and private key pair. The public key is sent to the server, and shares of the private key are sent to the other clients. Each client encrypts the model weights with a private key and a randomly generated seed and sends the encrypted weights to the server. The server asks each client to contribute secret shares of the private keys to remove the masks of the aggregated masked vector.

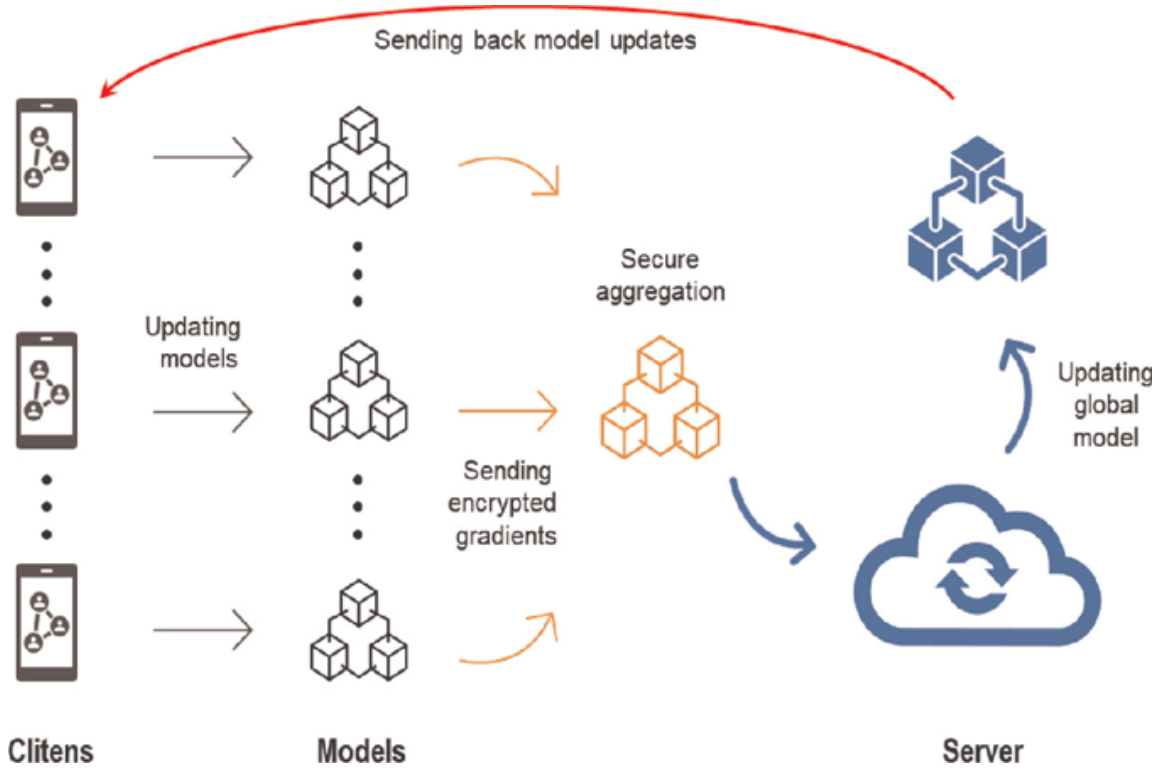


Figure 2.2: Overiew of MPC [2]

MPC in general is a computationally expensive process, and the communication overhead can be high. This is due to the need to communicate with several other clients in the network [2]. Table 2.2 shows an estimate of the complexity of MPC in FL, where n is the number of clients, l is the size of the model, and k is the number of shares of each private key that is created and sent to different clients.

	Computation	Communication
Server	O(nk(k+l))	O(n(k+l))
Client	O(k(k+l))	O(k+l)

Table 2.2: MPC complexity [2]

The security of MPC is based on the assumption that the majority of the clients are honest, and the protocol is secure if the majority of the clients are honest. Some other forms of MPC have the computation occur on the server, this then assumes that the server is honest [20]. Additionally, pasquini et al. created an active attack from the server side that could be used to infer information about the training data, even with the use of MPC [21].

Homomorphic encryption is a class of encryption algorithms, where the encryption algorithm *En* is homomorphic over the operator $\star$ if

$$En(m_1) \star En(m_2) = En(m_1 \star m_2), \forall m_1, m_2 \in \mathbb{M}, \tag{2.1}$$

where $\mathbb{M}$ is a set of plaintext [4]. This allows a local device to encrypt the model weights with a private key and send the encrypted weights to the server. The server can then perform the aggregation operation on the encrypted weights, and send the updated weights back to the local device. The local device can then decrypt the weights with the private key. This allows the server to aggregate the weights without knowing the weights themselves.

Homomorphic encryption requires that the encrypted data is encrypted with the same key, which means that each client must use the same private key. This can be a security concern and is usually avoided by including an aspect of MPC in the protocol to share the key.

While HE ensures that the server does not gain access to model weights during training, it is computationally expensive and can be slow or impractical for small devices. Unlike MPC methods, HE does not assume that either the clients or the server are honest [20], which provides greater privacy guarantees. Some literature has questioned its usefulness in practice due to the high computation and communication costs [22]. HE also typically contains similar communication overhead as MPC, so it provides no advantage over this set of security protocols.

Both of these techniques suffer from high computation and communication costs, and their practicality is questioned in the literature [2] [19]. They do however provide a high level of privacy protection, and directly protect the gradient updates, in particular from outsiders to the training process.

Perturbation-based techniques work by adding noise to the original data, making the information calculated from the perturbed data statistically indistinguishable from the original data [4]. Differential privacy is based on this concept, and it ensures that a change in a single sample will not change the overall statistics of the dataset [23].

Differential Privacy (ϵ, δ)-DP is defined as follows: A randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{O}$ is (ϵ, δ)-differentially private if for two adjacent datasets $D, D' \subseteq \mathcal{D}$ that differ in at most one data sample and any subset of the outputs $\mathcal{S} \subseteq \mathcal{O}$, then:

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

For the perturbation techniques described, DP is applied as Gaussian noise at each local node before the model weights or updates are sent to the server [12]. The concept of DP is summarized by the image in figure 2.3.

In most cases, differential privacy is applied at an example level, this is also known as record level. This means each sample in the dataset is individually protected by the DP guarantee. To practically apply DP to a model, a two-step process is done. First, the gradients for each sample are clipped to a maximum L2 norm value. This ensures that no one sample can have a large effect on the model weights. Second, Gaussian noise is added to the gradients before they are sent to the server. This noise is scaled by a second parameter called the noise multiplier. This process is repeated each epoch. DP can be applied at a number of different locations in the training process, but the most common is at the gradient update step. Dopamine [9] applies DP in this way, and is discussed later in section 3.2.4. Several versions of differential privacy do exist, such as user-level differential privacy, which protects the client as a whole, as opposed to each individual sample [24].

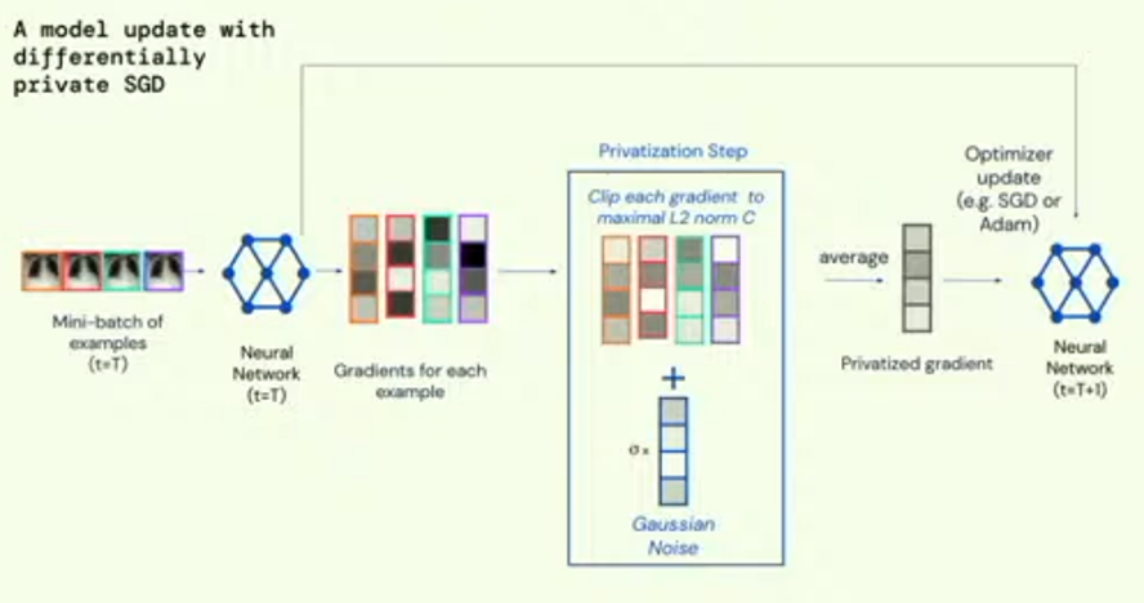


Figure 2.3: Practical implementation of differential privacy in FL [3]

Differential privacy has become a standard method of adding privacy to FL models [4]. It has been shown to be effective at preventing attacks on the training data, but it is not perfect. It has shown to only be effective after certain levels of noise have been added to the data [16], and it is difficult to get an easily interpretable measure of how DP has been added. In tandem with this, larger neural networks can become unstable and not converge predictably, as a small amount of noise at each dimension can add up to a large total power of the noise [3].

Additionally, there is the issue of competing to add more noise, if a client adds more noise than the rest of the clients, they receive similar accuracy without the same level of privacy concern. It has been suggested this would incentivize clients to add more noise than necessary in many real-world applications [25]. Finally, as the gradient is clipped for a set of data that is smaller than the batch size, gradient updates need to be calculated multiple

times per batch, which slows down the training process and increases the computational overhead.

Anonymization-based techniques work by removing or replacing identifying information in the dataset. This can be done by removing the names of the data samples or by replacing the names with a unique identifier. There are three types of attributes that anonymization-based techniques are concerned with: Identifiers, Sensitive attributes and not-sensitive attributes. Sensitive attributes are attributes that can be used to infer information about the training data, such as the birthday of a person or their medical data.

$k - anonymity$ is a common method of anonymization, where each data sample is indistinguishable from at least $k - 1$ other samples in the dataset. It is vulnerable to inference attacks on sensitive attributes. $l - diversity$ is an extension of $k - anonymity$, which adds diversity within a group to the sensitive attributes. This is resistant to attacks on sensitive attributes and eliminates attacks on unique identifiers in the data, but may be vulnerable to attribute linkage attacks. This is improved on with $t - closeness$ which further prevents attacks by maintaining the distribution of attributes, but at the cost of data utility [4].

This group of techniques is not as effective as the other techniques, as it is possible to re-identify the data samples using other information in the dataset. [4]. Many de-anonymization attacks have been done successfully such as the Netflix de-anonymization attack [26]. It is still recommended by GDPR but should not be relied on [27] [28].

Hybrid approaches are also commonly referred to in the literature, such as Stevens et al, [29] and Truex et al. [30] which used both MPC and DP. Sébert et al combined DP and HE [31]. In both cases, the combination of methods further increased the privacy gained without sacrificing model accuracy.

2.2.3 Privacy Metrics

Finally, in the field of ML, there exists a large number of metrics to measure the privacy of an ML model, surveyed by Wagner et al. [32]. Specifically Differential Privacy was mentioned as a metric of privacy without specific information on how it could be measured. Similarly, metrics based on the probability of an attack being successful were mentioned, but no applications of this to FL were found.

Specifically in FI, DP has been mentioned as a metric of privacy [23], but no evaluation of it as a metric has been found. Similarly, Malekzadeh, et al, [9] mention the use of a metric named the moments accountant at each local round to estimate privacy loss, but there is limited information available about these privacy metrics.

Both Hao et al. [23] and Malekzadeh et al. [9] discuss the use of the moments accountant [33] as a method of estimating the privacy loss for differential privacy. The moments accountant is used to estimate the privacy loss at each local round, and the privacy loss is then used to determine the amount of noise to add to the model weights. This is a novel method of tracking privacy loss, and it is not clear how effective it is at tracking privacy loss.

The moments accountant provides an upper bound on the ϵ parameter of the (ϵ,δ) DP guarantee given a certain δ and amount of noise σ. This bound is defined as:

$$\sigma \geq c_2 \frac{q\sqrt{Tlog(1/\delta)}}{\epsilon}$$

where $\epsilon \leq c_1 q^2 T$ and c_1 and c_2 are constants and q is the sampling probability of the SGD algorithm, which is $q = \frac{L}{N}$ where L is the subset of data sampled as part of the algorithm and N is the size of the data. T is the number of epochs that the model was trained for. In practice, this is computed for a set value of δ by finding the values for c_1 and c_2 that minimize the value of ϵ. This method only applies to Differential Privacy. Tensorflow Privacy [34], implement this using the `RdpAccountant` class.

L, which is given the name lot size, is the number of samples used when calculating the gradient update. Most recent literature refers to this concept as a micro-batch, and it is typically a subset of the larger batch used during the forward pass of the training process.

Importantly, from the above equation, the number of iterations and the micro-batch size increase the privacy loss, and the number of samples decreases the privacy loss.

Finally, some very recent research has proposed Epsilon* as a privacy metric [35]. It is based on a generalization of differential privacy but can be applied to any privacy-preserving method. This paper is still a preprint and has not been peer-reviewed, but it is a promising new metric for privacy in FL models.

In summary, the privacy of the training data in FL is a significant concern, and several different methods have been proposed to protect the data. The primary methods are homomorphic encryption, secure multi-party computation, differential privacy, and anonymization. Each of these methods has its advantages and disadvantages, and the choice of method depends on the specific use case. Differential privacy is the most commonly used method in FL, as it is effective at preventing attacks on the training data. However, it is not perfect, and there are still concerns about the privacy of the data. The moments accountant is a novel method of tracking privacy loss, but it is not clear how effective it is at tracking privacy loss. Finally, there is a lack of privacy metrics available for FL models, and further

research is needed to develop effective privacy metrics for FL models.

2.3 Constrained devices and fairness

Some research has been done on the case of constrained devices, in particular about HE. Hu et al. [12] designed DP-periodic-averaging SGD (PASGD) which is similar to FedAvg. The key contribution Hu et al. made is to design an optimization algorithm to balance communication and computation constraints for constrained devices. To reduce the effects of the added computation cost of HE, Feng et al created FLZip [36], which compresses the gradient updates by ignoring the gradients with lower values. FLZip shows significant decreases in both the computation cost as well as the communication cost with the compressed gradient update. No baselines of the additional resource requirements of privacy-preserving methods were found in the literature.

Farrand et al. [37] investigated the effect of DP on the fairness of a DL model trained in a central paradigm. They focused on the fairness of the model on sub-groups within the training data, not related to the expected output classes. They found that the model was less fair when DP was applied and that the fairness of the model was dependent on the amount of noise added to the model.

2.4 Conclusion

In this chapter, an overview of the literature and technical concepts needed to understand the research conducted is provided. The following chapter provides an in-depth look at the research methodology used in this project.

3 Methodology

This chapter will discuss the methodology used in this project. The chapter is divided into three sections, the first section will discuss the framework selection process, and the next two sections will discuss the experiments conducted in this project. The experiments are divided into those relating to the Dopamine paper and imbalanced data, experiments relating to implementing FL on an embedded device and an investigation of the computational costs of differential privacy.

3.1 Framework Selection

Many frameworks are available for implementing federated learning. In this section, we will discuss the selection process for the framework used in this project. To select the framework, we first need to understand the requirements of the project, which are as follows:

The framework must support privacy-preserving techniques such as Differential Privacy and Secure Aggregation, which are described in Section 2.2.2, or make the implementation of these techniques feasible by being well-documented and not overly complex. The framework should facilitate the interaction between multiple clients and a single server. It should facilitate communication across a network, allowing both local and remote clients to communicate with the server. The framework should be lightweight, as it will be deployed on a constrained device, the Rock Pi. Finally, the framework should be open source, allowing for modifications and extensions to be made to the framework.

The frameworks considered for this project are listed below and evaluated on how they fulfill these requirements. This is summarized in Table 3.1.

FATE [39] is a framework that focuses on industrial applications of FL, their primary focus is aimed towards large applications and data centers. It can support multiple machines, and privacy methods, and is open source. However, it does not allow a mixture of local clients running on the server machine and remote clients running on other machines. It is also a complex framework, with a large number of features that may not suit the lightweight nature of the Rock Pi.

TensorFlow-Federated(TFF) [34] is an extension of the popular DL framework TensorFlow, designed for research and simulation of federated learning. It is open source, with support for a large range of privacy methods. However, it does not support multiple machines and is designed solely for simulating FL on a single machine. As with FATE, the complexity of TFF makes installation difficult on the constrained resources of the Rock Pi.

Substra [40] is a comparatively small open-source framework, that claims to provide a privacy-preserving framework for federated learning, but it is not clear which privacy methods it supports. It is also not clear if it supports multiple machines. It also has limited documentation, which makes it difficult to use.

Open FL [41] is a framework that supports multiple machines but does not support privacy methods. It is however open source. It does not have much documentation and works exclusively on Linux Ubuntu. This makes it incompatible with the laptop used as a server, which uses Windows 11.

PySyft [42] is another popular framework that supports machine learning with many privacy methods available and is open source. With PySyft, communication across networks is not supported, and an additional framework, PyGrid, is required to support this. This makes PySyft complex to set up and use. Additionally, PySyft is required to be built from the source on the Rock Pi, which caused issues with the OS of the Rock Pi. This is likely due to the new and relatively untested nature of the Rock Pi OS. The OS is further discussed in section 3.2.2.

Flower [43] is a growing framework that focuses on providing lightweight support for federated learning. It is model-agnostic, meaning it can work with any ML framework, and is open source. Flower supports simulation on multiple machines, simulation on a single machine, and simulation with a mixture of local and remote clients. It also has some basic privacy methods implemented but does not support all privacy methods discussed in Section 2.2.2.

IBM FL [44] is a framework that primarily focuses on large data center applications. This framework supports both local and remote clients, but it is not clear which privacy methods it supports. IBM FL is not open source, which makes it difficult to modify and use for this project. Nvidia Flare [45] is a large industry-focused framework provided by Nvidia. It is not open source and provides little documentation. Similarly to IBM FL it is focused on larger applications and is not suitable for this project. These two frameworks are mentioned for completeness.

Based on the requirements of the project, the Flower framework was selected as the most suitable framework for this project. Flower supports all of the requirements, and is well-documented, making it easy to use. Flower also has a growing community, which means

Framework	Supports multiple machines	Supports Privacy Methods	Open Source
Fate [39]	Yes, with restrictions	Yes	Yes
TFF [34]	No	Yes	Yes
Substra [40]	Unknown	Unknown	Yes
Nvidia Flare [45]	Yes	Unknown	No
PySyft [42]	Yes	Yes	Yes
Flower [43]	Yes	Partially	Yes
IBM FL [44]	Yes	Unknown	No
Open FI [41]	Yes	No	Yes

Table 3.1: Comparison of the different available frameworks for FL.

that it is likely to be supported in the future.

3.2 Rock Pi Experiements

3.2.1 Monitoring CPU and Network Usage

Psutil [46] is a Python library designed for system monitoring and management tasks. It provides an easy-to-use interface for accessing a wide range of system information, including CPU and network usage metrics. By leveraging psutil, real-time data on system performance, such as network usage and CPU usage, can be monitored and tracked.

Psutil for Linux uses the Linux kernel's APIs to access detailed system information like processes, CPU usage, memory allocation, disk I/O, and network activity [46]. It efficiently gathers this data, ensuring consistency across different Linux distributions. Psutil also allows users to interact with processes, offering functionalities such as querying specific process attributes.

CPU and network usage were monitored during the training of federated learning models on the Rock Pi. As discussed in section 2.2, it is expected different privacy methods will impact the CPU and network usage of the client. CPU and network usage also serve as proxies for power consumption due to their close correlation with the energy demands of computing systems. High CPU usage indicates intensive processing tasks, which typically require more power to execute. Similarly, increased network usage signifies data transmission and reception, which also demand energy.

In this project, psutil was used to profile the CPU and network usage of the Rock Pi during the training of federated learning models. By monitoring these metrics, we can gain insights into the energy consumption of the Rock Pi and evaluate the performance of the federated learning models.

The federated learning client was run in a subprocess of the monitoring script. This allowed

tracking of the CPU usage of that specific process. The CPU usage was monitored using `psutil.cpu_percent()` to get the percentage of CPU usage. These metrics were logged at regular intervals during the training process to capture the system's performance over time. The network usage was tracked using `psutil.net_io_counters().packets_sent` and `psutil.net_io_counters().packets_recv` to get the number of packets sent and received, respectively. This was called at both the start and end of the training process to calculate the total network usage. It should be noted that the network usage was tracked for the rock pi as a whole, and therefore the numbers regarding network usage should not be treated in absolute terms, but rather as a relative measure of the network usage. The function used to call the federated learning training process and monitor the CPU usage is shown in Listing 3.1.

```
import multiprocessing as mp
import time
import psutil

def monitor(target):
    '''Monitor CPU usage of a target function.'''
    worker_process = mp.Process(target=target)
    worker_process.start()
    p = psutil.Process(worker_process.pid)

    # log cpu usage of `worker_process` every 10 ms
    cpu_percents = []
    while worker_process.is_alive():
        cpu_percents.append(p.cpu_percent())
        time.sleep(0.1)

    worker_process.join()
    return cpu_percents
```

Listing 3.1: Monitoring CPU Usage with Psutil

3.2.2 Rock Pi setup

The Radixa Rock Pi 5B is a single-board computer (SBC) that features a Quad-core ARM Cortex-A76 MPCore processor and quad-core ARM Cortex-A55 MPCore processor as the main CPU available on the board. [47]. This SBC will be used for testing the deployed model in this project.

The Rock Pi was set up with Ubuntu 20.04 LTS, running in headless mode. As the OS

software was released in September 2023, there were several issues with the available versions of Debian and Tensorflow. Radxa also provided a version of Ubuntu Jammy (20.04), which was used for this project [48]. The image for this OS was flashed to a 32GB microSD card using the Balena Etcher tool.

The Rock Pi was connected to a router via ethernet, on the same network as the server machine, which hosted the federated learning server. The Rock Pi was then accessed remotely using SSH, allowing for the execution of commands and scripts on the Rock Pi from the server machine.

To ensure reproducibility, the Rock Pi was set up with the same software as the server machine, including the same version of Python, and all required libraries, which were managed using a Python virtual environment and pip. The Rock Pi was also set up with the same dataset as the server machine, allowing for the training of the same federated learning models on both machines.

3.2.3 Mnist Model

The MNIST dataset is a popular dataset used for benchmarking machine learning models. It consists of 60,000 training images and 10,000 test images of handwritten digits. The images are grayscale and have a resolution of 28x28 pixels. The dataset is commonly used for training image classification models, making it an ideal choice for this project. The dataset is also sufficiently small to be used on the Rock Pi, which has limited resources compared to the server machine. Samples of the MNIST dataset are shown in Figure 3.1.

To preprocess this data so that it could be used for training, it was normalized. No further transformations were necessary, as the model described below could achieve the required accuracy. The data was also shuffled so that each client would get a random distribution of the data.

A neural network was designed to classify the MNIST dataset. The model was designed using Tensorflow Keras [34], a powerful library that is commonly used for designing neural networks. The model consisted of two convolutional layers, with a pooling layer in between them. The output of the convolutional layers was flattened and passed through a dense layer, followed by a final dense layer with 10 units, one for each digit. The summary of the model is shown in figure 3.2.

The models were trained using the SGD optimizer, and in the case of differential privacy, the Keras VectorizedDPKerasSGDOptimizer, which is discussed in further detail in section 3.2.4, implementation was used.

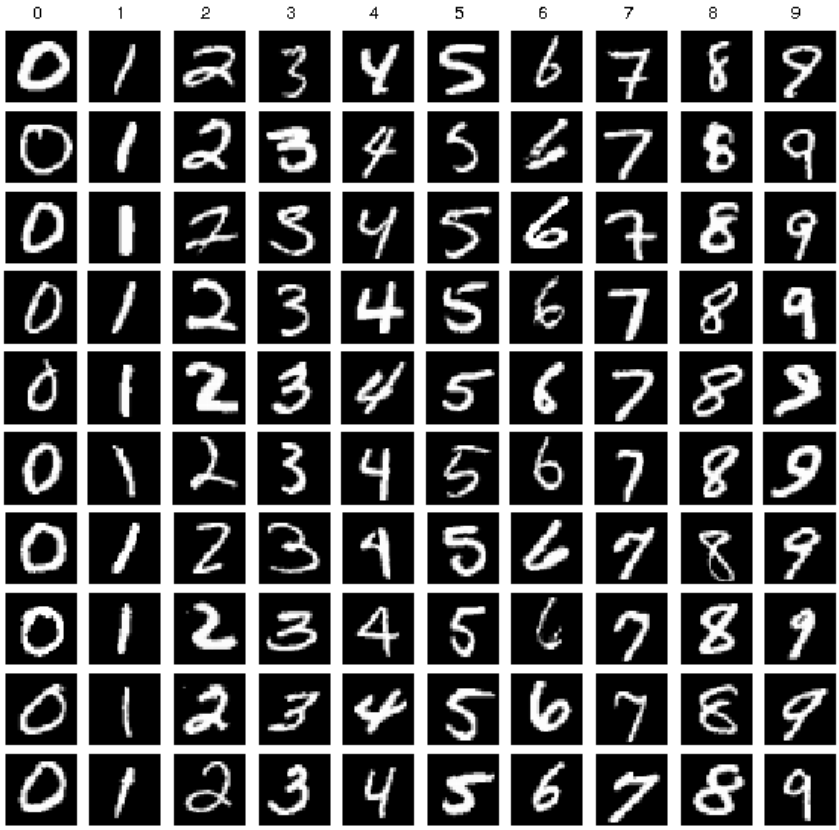


Figure 3.1: Samples of the Mnist dataset

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 14, 14, 16)        1040

 max_pooling2d (MaxPooling2  (None, 13, 13, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 5, 5, 32)          8224

 max_pooling2d_1 (MaxPoolin  (None, 4, 4, 32)          0
 g2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 32)                16416

 dense_1 (Dense)             (None, 10)                330

=================================================================
Total params: 26010 (101.60 KB)
Trainable params: 26010 (101.60 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3.2: Summary of the Mnist model

3.2.4 Differential Privacy Implementation

To apply differential privacy to the model, the `VectorizedDPKerasSGDOptimizer` optimizer was used which is provided by the TensorFlow Privacy module [34]. This optimizer is a variant of the SGD optimizer that implements differential privacy as discussed in section 2.2.2. The DPSGD optimizer requires two parameters: the L2 norm clip and the noise multiplier. These parameters and their meaning are discussed in section 2.2.2.

Additionally, the loss needed to be calculated on a per-example basis, rather than its mean over a minibatch which is typically done. This allows the DP layer to be applied over the microbatch of data used.

The upper bound of the privacy loss was also estimated using the RdpAccountant. The operation of this function is as described in section 2.2.3, where it returns an estimate for ϵ for a given value of δ. The RdpAccountant estimates the privacy loss on a per epoch basis, and this privacy loss is added to the running total of the privacy loss for each training process. In the case of these experiments, the value for δ was set at $1x10^{-5}$.

3.2.5 Flower Framework Overview

For the purpose of the experiments conducted in this project, the Flower Client-Server architecture was used [43]. This allowed the client process to run on different machines, and communicate over the RPC protocol.

The client class is configured with fit and evaluate functions, which dictate the operation of each client during the training process, and the results of these functions are sent to the server. Each client can load its own data before the training process begins.

The server is given a strategy to configure how it samples and configures the client's training process, as well as evaluation functions to evaluate the responses from the clients. The server is also given parameters to decide how many clients are sampled at each round. This architecture is shown in Figure 3.3.

To run this on the server machine, two Windows .bat scripts were created to set up a server and to set up and run a predefined amount of clients. On the RockPi, as only one client was needed, no .sh script was required.

3.2.6 Experiments

Two sets of four different experiments were conducted on the Rock Pi, with each set using the same hyperparameters as shown in table 3.2. Within each set, the same hyperparameters were used, this allowed consistency for comparison between the different training methods.

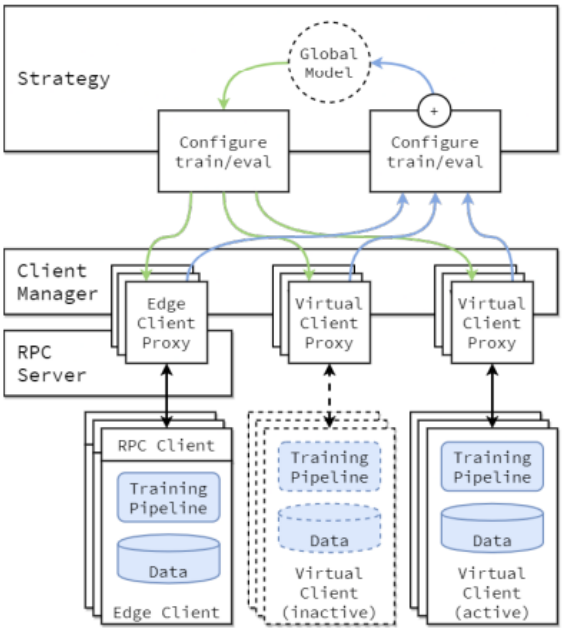


Figure 3.3: Flower Architecture Overview

Of the two sets of hyper-parameters used, one had a high batch size and learning rate, and the other with a low batch size and learning rate. Tuning the model for higher accuracy, especially when differential privacy was applied resulted in a higher batch size and learning rate, this model required fewer rounds to converge and achieved a higher accuracy. As discussed in section 4.1 this model takes a large amount of time to train. Therefore, a second model was trained with a lower batch size, which had a lower accuracy but it took less time to train, making it feasible to train this for a larger amount of epochs. This allowed better analysis of the network usage due to the higher amount of rounds.

Batch Size	Learning Rate	Rounds	Number of Microbatchs	L2 Norm Clip	Noise Multiplier
256	0.25	5	256	1.5	1.3
32	0.15	15	32	1.0	1.1

Table 3.2: Hyperparameters used for the Rock Pi experiments

In each set, two of the experiments were centralized training, where the model was trained on the entire dataset, as is done in conventional machine learning. One of these experiments also added differential privacy to the model. The other two experiments were federated learning experiments, where four clients were each trained on a subset of the dataset, and the model was aggregated on the server. One of these experiments also added differential privacy to the model.

On the rock pi, the client.py script was passed to the monitoring function shown above. This allowed the process related to this script to be monitored. The CPU usage and

Network package volume were tracked as described above. On the server machine, the server.py script was run, which allowed the server to communicate with the Rock Pi. Additionally, three other clients were run on the server machine, which simulated multiple clients communicating with the server.

The privacy bound was also calculated using the rdpAccountant which is described in section 3.2.4.

During the experiments, the Rock Pi was accessed via SSH from the server machine. This will have increased the value of the network usage, but this was not measured. It is expected that this was small and constant over time, and therefore should not have a significant impact on the results.

3.3 Class Imbalance Experiments

Dopamine [9], which was mentioned in section 2.2 is a well-known framework that implements differential privacy. Dopamine claims to be the first system that implements FL-based DPSGD that guarantees record-level DP for a dataset of medical images. The paper also provides code for the implementation of the framework, which was used as a reference for the implementation of Dopamine in the Flower framework.

As mentioned in section 2.2, differential privacy is a technique that adds noise to the gradients of the model to protect the privacy of the data. It also comes with a risk of reducing model utility, especially with regard to minority classes. This is an important consideration, as the class distribution of the dataset can have a significant impact on the performance of the model. In particular, the minority classes are often the most important to protect in a medical setting, as they represent the most vulnerable patients. By analyzing the model utility across the class distribution of the dataset before and after applying differential privacy, we can gain insights into the impact of differential privacy on the model's performance.

Some work has been done in the area of the effects of differential privacy on imbalanced datasets, but this work has been limited to centralized models. This project aims to extend this work to federated learning models. Dopamine was used as an example implementation of DP for this purpose. The dataset used in Dopamine, a dataset of medical images containing different severities of Diabetic Retinopathy(DR), is used as a dataset and is discussed further in the next section. As discussed in section 2.2.2, the power of the noise added to large neural networks can make the models unstable. For this reason, these experiments were also conducted on the MNIST dataset, which is a smaller dataset, to see if the results are consistent across models of different sizes.

3.3.1 Diabetic Retinopathy Dataset

Dopamine uses a dataset of medical images of the eyes, which contains samples with different severity of diabetic retinopathy(DR), a medical condition that causes blindness. The dataset is available on Kaggle[1], a popular machine-learning competition website. The dataset consists of images of the eyes, which are labeled with the severity of DR. The dataset contains five classes, each with increasing levels of severity of DR. Images of each of the five levels are shown below in figures 3.4. As can be seen, the feature that can be used to predict the severity of diabetic retinopathy shows up as specs of whiteness in the center of the eye.

The dataset is imbalanced with the majority of the samples belonging to the class representing no diabetic retinopathy. This is a common occurrence in medical datasets, as the majority of patients are healthy [49]. The class distribution of the dataset is shown in Table 3.3. As can be seen the classes with severity of 1, 3 and 4, are underrepresented in the dataset.

Class	Number of Samples
0	1805
1	370
2	999
3	193
4	295

Table 3.3: Class distribution of the dataset

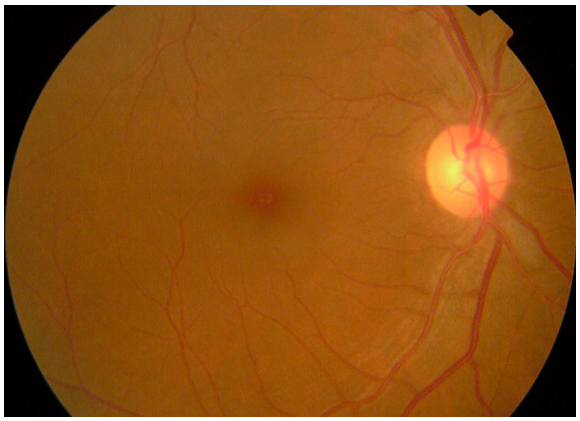
3.3.2 Data Preprocessing and SqueezeNet

The images are in color and have a resolution of 265x265 pixels. The dataset is split into a training set and a test set, with 80% of the images in the training set and 20% in the test set. As the data contains samples of left and right eyes, the input was randomly flipped horizontally to augment the data. The images are preprocessed by resizing them to 224x224 pixels and normalizing the pixel values to be between 0 and 1. The dataset is then split into batches of 32 images, which are used to train the model.

For training with a federated methodology, the dataset was split into four distinct datasets, each containing the same distribution of each class, with no overlap between the datasets. This is to simulate the data being distributed across multiple clients and to satisfy the condition that the data is independent and identically distributed, which is an assumption in many federated learning models.

The Dopamine framework used SqueezeNet at the base model. This is a lightweight model,

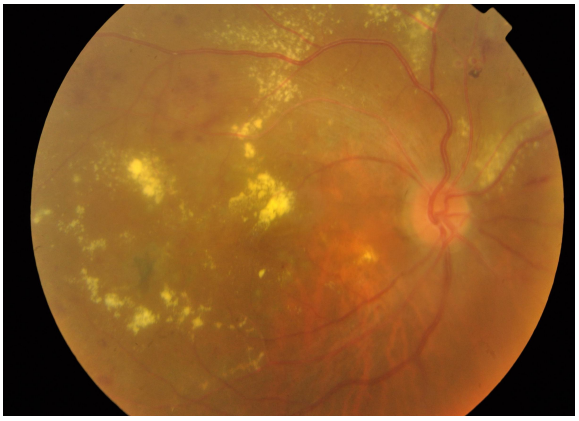
[1]the dataset as available: https://www.kaggle.com/c/aptos2019-blindness-detection/data



(a)Diabetic Retinopathy: Severity 0

(b)Diabetic Retinopathy: Severity 1

(c)Diabetic Retinopathy: Severity 2



(d)Diabetic Retinopathy: Severity 3

(e)Diabetic Retinopathy: Severity 4

Figure 3.4: Images of Diabetic Retinopathy of each class

with pre-trained weights on the ImageNet dataset. The model was used as a base model, with a final convolutional layer and a dense layer added to predict the class. The model was trained using the SGD optimizer, with a learning rate of 0.01, and a batch size of 50. The model was trained for 100 epochs, with the cross-entropy loss function used as the loss function.

The weights from the model are not available as part of the TensorFlow library, so it was necessary to import these from an online GitHub repository [2]. The model weights in this repository are from training the model on ImageNet, a well-known dataset of images typically used for transfer learning.

3.3.3 Mnist imbalanced dataset experiments

The Mnist dataset is discussed above in section 3.2.3. The dataset was modified by dropping 85% of the samples from the class representing the digits 2 and 4. This resulted in a dataset with a class distribution as shown in Table 3.4. The dataset was then normalized before training as was done in the RockPi experiments.

Class	0	1	2	3	4	5	6	7	8	9
original Number of Samples	5923	6742	5958	6131	5842	5421	5918	6265	5851	5949
Imbalanced Number of Samples	5923	6742	894	6131	877	5421	5918	6265	5851	5949

Table 3.4: Class distribution of the Mnist dataset after dropping samples

3.3.4 Experiments

For each of the two datasets, four separate experiments were done. The Mnist model was trained as was done in the Rock Pi experiments in section 3.2, except without using the RockPi as a client. The DR model also had four experiments, with each combination of with and without DP and with and without FL. In both cases, the FL experiments were run with four clients. In the case of the Imbalanced Mnist dataset, the FL training was done with 5 rounds of 3 epochs each. The DR dataset was trained with 10 rounds of 10 epochs each. The hyperparameters used for these experiments are shown in Table 3.5.

	Batch Size	**Learning Rate**	**Epochs**	**Number of Microbatches**	**L2 Norm Clip**	**Noise Multiplier**
Mnist	256	0.25	15	256	1.5	1.3
DR	50	0.002	100	50	2	1.15

Table 3.5: Hyperparameters used for the imbalanced dataset experiments

[2] found at: https://github.com/rcmalli/keras-squeezenet

3.4 Summary

In this chapter, the experiments used in this dissertation are discussed in detail. The Flower framework was selected as the FL framework for use in these experiments. Two sets of experiments were conducted, one focused on investigating the effect of differential privacy on the computational requirements of the client, and the other on the effect of differential privacy on imbalanced datasets. In the next chapter, the results of these experiments are presented and discussed in detail.

4 Evaluation

The evaluation of the proposed methodology is presented in this chapter. The evaluation aims to assess the performance and utility of the models trained using both centralized and federated learning approaches. Additionally, the impact of incorporating differential privacy (DP) into the models is analyzed. The evaluation includes two sets of experiments: class imbalance experiments and Rock Pi experiments. The class imbalance experiments focus on the classification performance of the models, while the Rock Pi experiments investigate the computational and communication aspects. The results of these experiments provide insights into the effectiveness and limitations of the proposed methodology.

4.1 Rock Pi Experiements

In this section, the results of the Rock Pi experiments are presented. The Rock Pi experiments were conducted to evaluate the computational resource requirements of the models trained using differential privacy. An evaluation of the accuracy of the models trained as part of the experiments is also presented.

4.1.1 Model Evaluation

This section focuses on the model tuned for accuracy, with the hyperparameters discussed in section 3.2.6. The classification reports generated by scikit-learn for the Rock Pi experiments are shown below in Fig. 4.1. In all four experiments, the models achieved an accuracy of over 95% and achieved high scores for precision, recall and f1 scores across all of the classes. This indicates that the models can predict the classes well and that the addition of DP and FL has not significantly impacted the model's ability to learn.

Surprisingly, vanilla federated learning achieved the highest accuracy of all the models, with an accuracy of 99%. This contradicts the literature, which suggests that FL should reduce the model's ability to learn. As both accuracies are at the upper bound, this difference is not significant and could be due to the randomness of how the data is split for training and testing.

	precision	recall	f1-score	support
0	0.96	1.00	0.98	980
1	0.99	1.00	0.99	1135
2	1.00	0.97	0.98	1032
3	0.99	0.99	0.99	1010
4	0.96	1.00	0.98	982
5	0.98	0.99	0.99	892
6	0.99	0.98	0.99	958
7	0.98	0.99	0.99	1028
8	0.98	0.99	0.99	974
9	1.00	0.93	0.96	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

(a)vanilla centralized learning

	precision	recall	f1-score	support
0	0.99	1.00	0.99	237
1	1.00	1.00	1.00	302
2	0.99	1.00	0.99	235
3	0.98	0.99	0.98	256
4	0.99	1.00	1.00	220
5	0.98	0.97	0.98	218
6	0.99	1.00	0.99	239
7	0.99	0.98	0.99	273
8	1.00	0.98	0.99	247
9	0.99	0.99	0.99	273
accuracy			0.99	2500
macro avg	0.99	0.99	0.99	2500
weighted avg	0.99	0.99	0.99	2500

(b)vanilla federated learning

	precision	recall	f1-score	support
0	0.96	0.98	0.97	980
1	0.98	0.99	0.98	1135
2	0.95	0.96	0.95	1032
3	0.95	0.93	0.94	1010
4	0.96	0.97	0.96	982
5	0.96	0.92	0.94	892
6	0.96	0.97	0.96	958
7	0.94	0.96	0.95	1028
8	0.92	0.94	0.93	974
9	0.95	0.91	0.93	1009
accuracy			0.95	10000
macro avg	0.95	0.95	0.95	10000
weighted avg	0.95	0.95	0.95	10000

(c)centralized learning with DP

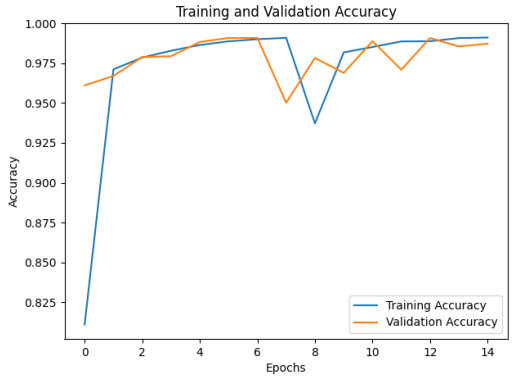
	precision	recall	f1-score	support
0	0.99	0.99	0.99	187
1	1.00	1.00	1.00	229
2	0.98	0.97	0.98	179
3	0.98	0.98	0.98	229
4	0.99	0.97	0.98	200
5	0.96	0.98	0.97	164
6	0.99	0.99	0.99	191
7	1.00	0.99	0.99	217
8	0.96	0.99	0.98	184
9	0.98	0.97	0.97	220
accuracy			0.98	2000
macro avg	0.98	0.98	0.98	2000
weighted avg	0.98	0.98	0.98	2000

(d)federated learning with DP

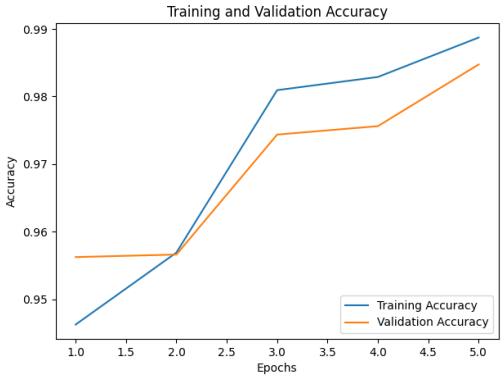
Figure 4.1: Classification reports from the Rock Pi Experiments

The classification reports show that the models trained with DP have a slightly lower accuracy than the models trained without DP. This is expected, as noted in the section 2.2, and is likely caused by the noise added to the gradients. The accuracy of the models trained with DP is 95% and 98% for the centralized and federated models respectively. This is a small decrease in accuracy, and in many cases, this may be acceptable.

The accuracy of the models plotted per epoch is shown in the figure below in 4.2. Note that the federated models are trained for 15 epochs, but the plots show 5 different points. This is due to internal constraints in the flower library, which limits what can be passed from clients to the server. Each of these points is the accuracy of the final epoch of that training round. Note that the axis in figure 4.2(a) is different from the other plots, which gives the impression that the accuracy dropped significantly on the eighth epoch. This is misleading as the drop was less than 5%.

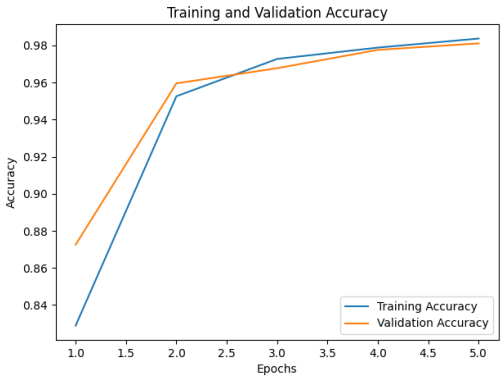


(a)vanilla centralized learning



(b)vanilla federated learning

(c)centralized learning with DP



(d)federated learning with DP

Figure 4.2: Accuracy plots from the Rock Pi Experiments

Both the centralized learning models appear to have flattened off before the end of training, which suggests that the models have converged before the end of training. Conversely, the

federated models have not flattened off to the same degree. This illustrates that federated learning tends to require more epochs to converge than centralized learning.

The privacy loss of the models trained with DP was also calculated, using the `rdp_accountant` discussed in section 2.2.3. The upper bound for the value of ϵ is 0.81 in the case of the centralized model, and 4.21 in the case of the federated model. This increase in the value of ϵ is due to the reduced amount of data at each client in the federated model, which means that the noise added to the gradients is less effective at preserving privacy.

In order to reduce the value of ϵ, either of the two DP parameters could be changed, i.e. by either increasing the noise or decreasing the L2 gradient clipping. Alternatively, the batch size could be decreased. Either of these choices comes with a trade-off, of reduced model utility in the case of stronger DP parameters or increased training time in the case of decreased batch size. Alternatively, the training could be stopped earlier, as the model may have converged before the end of the training, as seen in figure 4.2.

4.1.2 Profiling Results

The profiling results of the Rock Pi experiments are shown in the table below in tables 4.1 and 4.2. The profiling results are shown for both sets of experiments, the high-accuracy model and the model with a lower training time, as described in section 3.2.6. The time taken per epoch is reported as the average amount of time that each epoch took. In the case of FL, aggregation is also done on the server, so the time taken to aggregate the model weights is also shown. Note that the aggregation time is simply waiting time for the client, as aggregation is done on the server. This means the CPU usage is also lower during this time. As discussed in section 3.2, the average CPU usage and the network packet volume are also reported. The total time taken to complete the training process was also recorded by the profiling script and is shown for the profiling of the high-accuracy model in table 4.1.

The results show a significant increase in the time taken to complete an epoch when DP is added across both sets of experiments. This is expected, as noted in the section 2.2, and is caused by the additional computation required to add noise to the gradients on a per-example basis. The time taken to complete an epoch is decreased when using FL, as the model is trained on less data.

The comparison of the total time taken for each of the models may not be a completely fair comparison, as the number of epochs was constant for each of the two sets of experiments. As shown in figure 4.2, the centralized models appear to have converged before this point for the high-accuracy experiments. In a real-world scenario, the training length would be adjusted to ensure that the model has converged, which would likely result in the FL models taking a larger amount of epochs.

Training Method	Epoch Duration	Total Time (h:mm:ss)	CPU Usage	Packets Volume	Adjusted Packets Volume
Central without DP	11s	0:03:49	202%	2.2k sent 2.1k received	9.7 pkts/s sent 9.3 pkts/s received
Central with DP	1052s	4:24:25	710%	4.8k sent 14.2k received	0.3 pkts/s sent 0.9 pkts/s received
FL without DP	3s	0:01:35	109%	1.3k sent 1.3k received	14.0 pkts/s sent 13.8 pkts/s received
FL with DP	257s	1:05:31	662%	1.9k sent 4.2k received	0.49 pkts/s sent 1.1 pkts/s received

Table 4.1: Results of the Rock Pi Profiling: High Accuracy Model

Training Method	Epoch Duration	CPU Usage	Packets Volume	Adjusted Packets Volume
Central without DP	25s	271%	9.3k sent 8.4k received	12.4 pkts/s sent 11.2 pkts/s received
Central with DP	134s	524%	56.7k sent 56.8k received	14.1 pkts/s sent 14.1 pkts/s received
FL without DP	12s (10s aggregation)	258%	5.9k sent 5.7k received	12.8 pkts/s sent 12.4 pkts/s received
FL with DP	64s (10s aggregation)	535%	30k sent 29.2k received	14.9 pkts/s sent 14.5 pkts/s received

Table 4.2: Results of the Rock Pi Profiling: Lower Training Time Model

The results show a clear increase in CPU usage when DP is added to the model. Here the percentage number shown is related to the number of available cores on the Rock Pi, with 100% representing the usage of one core. The RockPi has eight cores available, so the maximum CPU usage is 800%. The CPU usage is slightly lower when using the FL model, which makes sense as less data is being processed on the Rock Pi. In all cases when DP is added, the CPU usage is increased by a factor between 2 and 4. This is expected, as noted in the section 2.2, and is likely caused by the additional computation required to add noise to the gradients.

The experiments with the lower training time show an increase in the number of packets sent and received when DP is added to the model. This is unexpected, as the data sent by the model should not change significantly when DP is added. Further investigation is needed into why this is the case, ideally using a more complete package sniffer to understand the contents of the packets, and their origin.

In the case of the experiments with hyperparameters tuned for high accuracy, the network packet volume is significantly higher when DP is added to the model. The difference here may be because the CPU usage is significantly higher when DP is added in comparison to the CPU usage when DP is added to the models with a lower training time. This reduced network packet volume could be due to the OS on the RockPi not being able to do peripheral tasks while the CPU is under heavy load.

For the experiments with lower training time, the network packet volume is primarily correlated with the time spent running, with the FL model sending and receiving more packets than the centralized model. These numbers can be adjusted to packets per second, which shows that the FL model is sending and receiving more packets per second than the centralized model. This behavior is matched in the models with hyperparameters tuned for high accuracy, where the network volume is higher when training using FL. This is expected, as noted in the section 2.2, and is likely caused by the additional communication required to train the model in a federated manner.

This section has shown that the models trained using the proposed methodology can achieve high accuracy and that the addition of DP and FL does not significantly impact the model's ability to learn. The profiling results show that the addition of DP significantly increases the time taken to train the model and the CPU usage of the model. The addition of FL also increases the network packet volume but does not significantly impact the time taken to train the model. This indicates that the inclusion of DP must be carefully considered when training a model, in particular in the context of the computational resources available on the training device. This result indicates that the use of DP in a federated learning setting may not be suitable for typical IoT devices.

4.2 Class Imbalance Experiements

The results of the Class Imbalance experiments are presented in this section. The experiments were conducted using imbalanced datasets, with particular attention to the context of medical datasets. The results of the experiments are presented in the form of classification reports, which show the precision, recall and f1 score of the model in each class, and provide an insight into the utility of the trained model.

4.3 Diabetic Retinopathy Dataset

Due to the size of the CNN model used in the Dopamine experiments, training took a significantly increased time. This limited debugging opportunities and the DR experiments appear to be less stable, with the models not always converging. This has been seen in the literature as well, it is referred to as the curse of dimensionality, and it occurs when a model is very large, the power of the overall noise in the model can be very high, which may cause issues with the model converging. This is likely the cause of the issues seen in the Dopamine experiments.

Nevertheless, training the model on the DR dataset was successful on occasion, and the results of these experiments are presented below.

As mentioned in section 3.3, the Mnist dataset was also made artificially imbalanced by removing 85% of the samples from classes 2 and 4. The models trained on this dataset provided a more stable model, which was used to compare the effects of DP and FL on the model's performance.

The results from the DR experiments are shown below in 4.3. The classification report from the scikit-learn python package was used to evaluate the model performance on each class.

Both the models trained using central and federated methodologies observed decreased utility when trained with DP added. This is expected, as noted in the section 2.2, and is caused directly by the DP protocol of clipping the gradients and adding noise. The accuracy was reduced from 80% to 76% and 79% to 73% respectively, which is significant, but in many cases, this alone may be an acceptable model utility. These results are slightly lower than the values achieved in the Dopamine paper but are still within the expected range.

The classification report for each of the classes shows a much more extreme drop in model utility. In all cases where either DP or FL were added to the model, the values for precision, recall and f1 score are shown as 0, which means that the model did not predict the minority class at all. This is in effect a decrease in the complexity the model can learn, which would

	precision	recall	f1-score	support
0	0.97	0.98	0.97	350
1	0.67	0.49	0.57	77
2	0.70	0.83	0.76	205
3	0.48	0.29	0.36	49
4	0.41	0.35	0.38	51
accuracy			0.80	732
macro avg	0.64	0.59	0.61	732
weighted avg	0.79	0.80	0.79	732

(a)Vanilla Centralized learning

	precision	recall	f1-score	support
0	0.97	0.98	0.98	181
1	0.63	0.51	0.57	37
2	0.62	0.90	0.73	100
3	0.44	0.20	0.28	20
4	0.00	0.00	0.00	30
accuracy			0.79	368
macro avg	0.53	0.52	0.51	368
weighted avg	0.73	0.79	0.75	368

(b)Vanilla Federated learning

	precision	recall	f1-score	support
0	0.93	0.98	0.95	354
1	0.41	0.31	0.35	71
2	0.61	0.85	0.71	217
3	0.00	0.00	0.00	30
4	0.00	0.00	0.00	60
accuracy			0.76	732
macro avg	0.39	0.43	0.40	732
weighted avg	0.67	0.76	0.71	732

(c)Centralized learning with DP

	precision	recall	f1-score	support
0	0.90	0.96	0.93	181
1	0.36	0.32	0.34	37
2	0.59	0.82	0.68	100
3	0.00	0.00	0.00	20
4	0.00	0.00	0.00	30
accuracy			0.73	368
macro avg	0.37	0.42	0.39	368
weighted avg	0.64	0.73	0.68	368

(d)Federated Learning with DP

Figure 4.3: Classification report from the Diabetic Retinopathy Experiments

be expected from excessive regularization. This is an extreme case of the result mentioned in section 2.2, where a decrease in the model's ability to predict the minority classes is expected when adding differential privacy.

The vanilla centralized model poorly predicts the minority classes, with classes 3 and 4 both having an f1 score of less than 40%. This indicates that despite the overall accuracy of 80%, the model would not be suitable for a medical setting.

When this model is trained in a federated manner, the overall model accuracy did not reduce significantly, but the f1 score of the minority classes did. Class 4, the most severe occurrence of Diabetic Retinopathy, had an f1 score of 0. This means this class was not predicted once by the model which is already a serious degradation of model usefulness in a real-world medical setting.

The addition of DP to both training processes further reduced the model's ability to predict the minority classes. The recall, precision and f1 score of classes 3 and 4 are all 0, which means the model likely did not have the ability to predict these classes at all. Furthermore, the metrics of class 1 were also significantly reduced, from an f1 score of 57% in both the vanilla centralized and federated models, to an f1 score of 35% and 34% respectively. This decrease in the performance of the three minority classes is in contrast to the performance of the model on both of the majority classes. The f1 score of class 0 only dropped from 97%

to 93% across all four cases, and although the f1 score of class 2 dropped from 76% to 71% and 68%, this is less significant than the drop in performance observed for the minority classes.

To resolve this issue, training on a larger dataset would reduce the issues with the imbalance, and make the model more robust to the noise added by the differential privacy. Further, as mentioned in the literature review, there is some research suggesting that the decrease in model utility can be reduced if a model is pre-trained on images more similar to the dataset, which would also not reduce the privacy of the model.

The results of this experiment highlight the importance of measuring the model performance in multiple different ways, and being aware of the effects of the overall imbalance in the dataset, especially in areas where this imbalance is common such as medical datasets.

4.4 Imbalanced Mnist Dataset

The results of the imbalanced Mnist dataset are shown below in figure 4.4. The classification report shows a similar pattern to what is seen in the case of the DR dataset. The models trained with DP have a lower accuracy than the models trained without DP, and the models trained with FL also have a lower accuracy. In this case, training with vanilla FL did not seem to reduce the model's accuracy. Both the models trained without DP learned all classes similarly well, and there seem to be no negative effects of the imbalance in the dataset.

The models trained with DP have lower accuracy than the models trained without DP, and the model trained with both DP and FL has a significantly reduced accuracy. The model accuracy on the minority classes was affected by the addition of DP, with the recall and f1 score of class 2 and 4 dropping from approximately 95% to a recall of 72% and 76% respectively, and an f1 score of 82% and 85% respectively.

The addition of DP to the model trained with FL caused a further drop in the model's performance on the minority classes, with class 2 having a recall of 25% and an f1 score of 40%, and class 4 having a recall of 2% and an f1 score of 4%. This is a significant drop in the accuracy of the model with respect to the minority classes and may render a similar model useless in real-world applications.

In contrast to the DR dataset, the imbalance in the Mnist dataset did not seem to affect the precision of the model with respect to these classes. This means that when the model does predict a minority class, it is likely to be correct. It is possible the models trained on the DR dataset would have shown similar behavior if the effects of the noise added were not so severe as to prevent the model from predicting the minority classes at all.

	precision	recall	f1-score	support
0	0.98	0.99	0.98	980
1	0.95	1.00	0.98	1135
2	1.00	0.93	0.96	1032
3	0.98	0.99	0.99	1010
4	1.00	0.96	0.98	982
5	0.98	0.98	0.98	892
6	0.98	0.99	0.99	958
7	0.96	0.99	0.98	1028
8	0.99	0.98	0.98	974
9	0.97	0.98	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

(a)Vanilla Centralized learning

	precision	recall	f1-score	support
0	0.96	0.99	0.98	188
1	0.99	0.99	0.99	242
2	0.99	0.94	0.97	191
3	0.96	0.99	0.98	220
4	1.00	0.92	0.96	197
5	0.99	0.98	0.98	167
6	0.99	0.98	0.99	199
7	1.00	0.99	0.99	222
8	0.94	0.98	0.96	173
9	0.93	0.98	0.96	201
accuracy			0.98	2000
macro avg	0.98	0.98	0.98	2000
weighted avg	0.98	0.98	0.98	2000

(b)Vanilla Federated learning

	precision	recall	f1-score	support
0	0.95	0.98	0.97	980
1	0.98	0.99	0.98	1135
2	0.99	0.72	0.83	1032
3	0.91	0.96	0.93	1010
4	0.97	0.76	0.85	982
5	0.96	0.95	0.96	892
6	0.90	0.97	0.93	958
7	0.92	0.95	0.94	1028
8	0.84	0.94	0.89	974
9	0.79	0.95	0.87	1009
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

(c)Centralized learning with DP

	precision	recall	f1-score	support
0	0.96	0.97	0.97	202
1	0.97	0.99	0.98	226
2	0.98	0.25	0.40	224
3	0.77	0.96	0.85	202
4	0.67	0.02	0.04	214
5	0.93	0.96	0.95	161
6	0.79	0.96	0.87	187
7	0.87	0.89	0.88	204
8	0.64	0.91	0.75	199
9	0.46	0.92	0.61	181
accuracy			0.77	2000
macro avg	0.80	0.78	0.73	2000
weighted avg	0.81	0.77	0.72	2000

(d)Federated Learning with DP

Figure 4.4: Classification report from the Imbalanced Mnist experiments

These results suggest a link between the size of the model, the degree of model imbalance and the effects of DP on the model. The upper bound for the value of ϵ is not dependent on the size of the model, but the effects of the noise added to the model are. This is potentially a significant issue with regard to the use of DP in real-world applications, where minority classes may exist and large DL models may be necessary.

4.5 Summary

In this chapter, the results of the experiments conducted as part of this dissertation are presented. These results are then discussed in the context of what they indicate for the application of DP to FL models on constrained devices, and in the context of imbalanced data typical of medical datasets. In the next chapter, the results of the work are discussed in the context of the objectives set out in the introduction, and the limitations of the work are discussed.

5 Conclusion

5.1 Conclusion

As part of the initial contribution of this project, a comparison of different frameworks for federated learning was conducted, where many frameworks were evaluated for their suitability to provide privacy methods on a constrained device such as the Rock Pi. The Flower framework was chosen as the most suitable for this project, as it provided the most comprehensive privacy methods and was the most well-documented.

The flower framework was deployed with a client on the Rock Pi, a single-board computer, to investigate the effects of differential privacy and federated learning on network and CPU usage. The experiments validated the literature that the use of federated learning in a training process can approach the same accuracy as a centralized model. The experiments also showed that the addition of differential privacy did not necessarily reduce the accuracy of the model excessively.

The results of this testing showed that the addition of differential privacy significantly increased the time taken to complete an epoch and the CPU usage of the Rock Pi. The network packet volume on the RockPi had contradicting observations between the two sets of experiments done. It was suggested that this unexpected measurement was caused by the very high CPU usage, leading to different behavior from the OS. This requires further work to understand exactly the effects of the implementation of DP used in this dissertation on network usage.

The effects of federated learning and differential privacy on imbalanced datasets were investigated, using two datasets, Diabetic Retinopathy and a sytheticly imbalanced Mnist dataset. Dopamine, a well-known implementation of differential privacy was used as a baseline for the implementation of DP in this context. Significantly, it was observed that while the model accuracy was not reduced overly by the addition of differential privacy, the model failed to predict minority classes, in both central and federated settings. This is a significant issue, as the minority classes are often the most important to protect in a medical setting. This result was observed in both the Diabetic Retinopathy and Mnist datasets.

The imbalanced dataset experiments highlighted the potential issues with the implementation of differential privacy in a medical setting. They also begin a discussion regarding the effectiveness of DP on larger datasets, due to the increased power of the noise added to the model, for the same upper bound on privacy loss.

5.2 Further Work

This work illustrates the need for more work in the area of privacy in federated learning, in particular with regard to understanding the computational requirements of different privacy-preserving methods. An immediate next step would be to implement secure aggregation and Homomorphic Encryption on the flower framework. This would allow for more comprehensive testing of the implications of privacy on the IoT.

Further work could be done by investigating hybrid approaches and quantifying the privacy loss caused by the models, along with any potential benefits with regard to computational and communication overhead.

This dissertation is an important example that model fairness is decreased by differential privacy and that it can be a significant issue in a medical setting.

The investigation into the effects of differential privacy and federated learning would benefit from conducting similar tests on other models and datasets, in particular with a range of dimensions. This would allow for a more comprehensive understanding of the implications of privacy and fairness on federated learning models.

Finally, it is prudent to investigate methods to quantify the privacy loss caused by different privacy methods, so that more complete comparisons can be made between different privacy methods. It would be advantageous to have a metric that was generally understandable, as opposed to comparisons only being valid for a single dataset or model.

Bibliography

[1] A. Sinha. (2021) Introduction to FLOWER. [Online]. Available: https://medium.com/nerd-for-tech/build-your-own-federated-learning-model-2c882ea8cfde

[2] K. H. Li, P. P. B. de Gusmão, D. J. Beutel, and N. D. Lane, "Secure aggregation for federated learning in flower." [Online]. Available: http://arxiv.org/abs/2205.06117

[3] B. Balle, "Towards practival differentially private training," 2024, Flower Summit. [Online]. Available: https://youtu.be/sHfVgseQG_g?list=PLNG4feLHqCWm2CRMKefoo1-t3e47ahT46&t=14021

[4] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–36, 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3460427

[5] Y. Kumar, A. Koul, R. Singla, and M. F. Ijaz, "Artificial intelligence in disease diagnosis: a systematic literature review, synthesizing framework and future research agenda," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 7, pp. 8459–8486, 2023. [Online]. Available: https://link.springer.com/10.1007/s12652-021-03612-z

[6] M. M. Ahsan, S. A. Luna, and Z. Siddique, "Machine-learning-based disease diagnosis: A comprehensive review," *Healthcare*, vol. 10, no. 3, p. 541, 2022. [Online]. Available: https://www.mdpi.com/2227-9032/10/3/541

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[8] B. v. d. S. Chris Jay Hoofnagle and F. Z. Borgesius, "The european union general data protection regulation: what it is and what it means*," *Information & Communications*

Technology Law, vol. 28, no. 1, pp. 65–98, 2019. [Online]. Available: https://doi.org/10.1080/13600834.2019.1573501

[9] M. Malekzadeh, B. Hasircioglu, N. Mital, K. Katarya, M. E. Ozfatura, and D. Gündüz, "Dopamine: Differentially private federated learning on medical data." [Online]. Available: http://arxiv.org/abs/2101.11693

[10] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8694781/

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] R. Hu, Y. Guo, E. P. Ratazzi, and Y. Gong, "Differentially private federated learning for resource-constrained internet of things." [Online]. Available: http://arxiv.org/abs/2003.12705

[13] C. Zhou, Y. Sun, D. Wang, Q. Gao, and Z. Tan, "Fed-fi: Federated learning malicious model detection method based on feature importance," *Sec. and Commun. Netw.*, vol. 2022, jan 2022. [Online]. Available: https://doi.org/10.1155/2022/7268347

[14] T. Ongun, S. Boboila, A. Oprea, T. Eliassi-Rad, J. Hiser, and J. Davidson, "CELEST: Federated learning for globally coordinated threat detection." [Online]. Available: http://arxiv.org/abs/2205.11459

[15] T. T. Thein, Y. Shiraishi, and M. Morii, "Personalized federated learning-based intrusion detection system: Poisoning attack and defense," *Future Generation Computer Systems*, vol. 153, pp. 182–192, 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X23003783

[16] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients." [Online]. Available: http://arxiv.org/abs/1906.08935

[17] J. Geng, Y. Mou, F. Li, Q. Li, O. Beyan, S. Decker, and C. Rong, "Towards general deep leakage in federated learning." [Online]. Available: http://arxiv.org/abs/2110.09074

[18] M. Suliman and D. Leith, "Two models are better than one: Federated learning is not private for google GBoard next word prediction." [Online]. Available: http://arxiv.org/abs/2210.16947

[19] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly)logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2020, pp. 1253–1269. [Online]. Available: https://dl.acm.org/doi/10.1145/3372297.3417885

[20] J. R. Gilbert, "Secure aggregation is not all you need: Mitigating privacy attacks with noise tolerance in federated learning." [Online]. Available: http://arxiv.org/abs/2211.06324

[21] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2022, pp. 2429–2443. [Online]. Available: https://dl.acm.org/doi/10.1145/3548606.3560557

[22] K. Burlachenko, A. Alrowithi, F. A. Albalawi, and P. Richtarik, "Federated learning is better with non-homomorphic encryption," in *Proceedings of the 4th International Workshop on Distributed Machine Learning*, 2023, pp. 49–84. [Online]. Available: http://arxiv.org/abs/2312.02074

[23] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8859260/

[24] P. Lai, N. Phan, T. Sun, R. Jain, F. Dernoncourt, J. Gu, and N. Barmpalios, "User-entity differential privacy in learning natural language models." [Online]. Available: http://arxiv.org/abs/2211.01141

[25] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses." [Online]. Available: http://arxiv.org/abs/2012.06337

[26] A. Narayanan and V. Shmatikov, "How to break anonymity of the netflix prize dataset." [Online]. Available: http://arxiv.org/abs/cs/0610105

[27] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Anonymizing data for privacy-preserving federated learning." [Online]. Available: http://arxiv.org/abs/2002.09096

[28] Q. Xu, T. Cohn, and O. Ohrimenko, "Fingerprint attack: Client de-anonymization in federated learning." [Online]. Available: http://arxiv.org/abs/2310.05960

[29] T. Stevens, C. Skalka, C. Vincent, J. Ring, S. Clark, and J. Near, "Efficient differentially private secure aggregation for federated learning via hardness of learning with errors." [Online]. Available: http://arxiv.org/abs/2112.06872

[30] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. ACM, 2019, pp. 1–11. [Online]. Available: https://dl.acm.org/doi/10.1145/3338501.3357370

[31] A. G. Sébert, R. Sirdey, O. Stan, and C. Gouy-Pailler, "Protecting data from all parties: Combining FHE and DP in federated learning." [Online]. Available: http://arxiv.org/abs/2205.04330

[32] I. Wagner and D. Eckhoff, "Technical privacy metrics: A systematic survey," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 1–38, 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3168389

[33] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 308–318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[34] T. T. F. Authors, "TensorFlow Federated," Dec. 2018. [Online]. Available: https://github.com/tensorflow/federated

[35] D. M. Negoescu, H. Gonzalez, S. E. A. Orjany, J. Yang, Y. Lut, R. Tandra, X. Zhang, X. Zheng, Z. Douglas, V. Nolkha, P. Ahammad, and G. Samorodnitsky, "Epsilon*: Privacy metric for machine learning models," 2024. [Online]. Available: http://arxiv.org/abs/2307.11280

[36] X. Feng and H. Du, "FLZip: An efficient and privacy-preserving framework for cross-silo federated learning," in *2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. IEEE, 2021, pp. 209–216. [Online]. Available: https://ieeexplore.ieee.org/document/9694138/

[37] T. Farrand, F. Mireshghallah, S. Singh, and A. Trask, "Neither private nor fair: Impact of data imbalance on utility and fairness in differential privacy." [Online]. Available: http://arxiv.org/abs/2009.06389

[38] J.-P. A. Yaacoub, H. N. Noura, and O. Salman, "Security of federated learning with IoT systems: Issues, limitations, challenges, and solutions," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 155–179, 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2667345223000226

[39] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang, "Fate: An industrial grade platform for collaborative learning with data protection," *Journal of Machine Learning Research*, vol. 22, no. 226, pp. 1–6, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-815.html

[40] M. N. Galtier and C. Marini, "Substra: a framework for privacy-preserving, traceable and collaborative machine learning," 2019.

[41] P. Foley, M. J. Sheller, B. Edwards, S. Pati, W. Riviera, M. Sharma, P. Narayana Moorthy, S.-h. Wang, J. Martin, P. Mirhaji, P. Shah, and S. Bakas, "Openfl: the open federated learning library," *Physics in Medicine & Biology*, vol. 67, no. 21, p. 214001, Oct. 2022. [Online]. Available: http://dx.doi.org/10.1088/1361-6560/ac97d9

[42] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose, T. Ryffel, Z. N. Reza, and G. Kaissis, "Pysyft: A library for easy federated learning," 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236690571

[43] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," 2022.

[44] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn *et al.*, "Ibm federated learning: an enterprise framework white paper v0. 1," *arXiv preprint arXiv:2007.10987*, 2020.

[45] H. R. Roth, Y. Cheng, Y. Wen, I. Yang, Z. Xu, Y.-T. Hsieh, K. Kersten, A. Harouni, C. Zhao, K. Lu, Z. Zhang, W. Li, A. Myronenko, D. Yang, S. Yang, N. Rieke, A. Quraini, C. Chen, D. Xu, N. Ma, P. Dogra, M. Flores, and A. Feng, "NVIDIA FLARE: Federated Learning from Simulation to Real-World," *IEEE Data Eng. Bull., Vol. 46, No. 1*, Mar. 2023.

[46] G. Rodola. Psutil. [Online]. Available: https://psutil.readthedocs.io/en/latest/

[47] Radixa, "ROCK 5b hardware details." [Online]. Available: https://wiki.radxa.com/Rock5/hardware/5b

[48] Radxa, "Radxa rock-5b ubuntu jammy cli b39," https://github.com/radxa-build/rock-5b/releases/tag/b39, 2023.

[49] L. Gao, L. Zhang, C. Liu, and S. Wu, "Handling imbalanced medical image data: A deep-learning-based one-class classification approach," *Artificial Intelligence in Medicine*, vol. 108, p. 101935, 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S093336572030261X

A1 Appendix

The source code for this project is available at
`https://github.com/Ruairi-Grant/PrivateFedLearn/tree/submission`