university of groningen

faculty of science and engineering

Privacy-Preserving Mechanisms in Federated Learning

Lisa Overeijnder

(s3241459)

December 22, 2021

Thesis BSc Computing Science

Thesis committee:

Prof. dr. B. Koldehofe - First supervisor

Dr. M. Lotfian Delouee - Second supervisor

Abstract

In many Machine Learning applications, the datasets contain sensitive information about individuals such as location, contacts, and medical information. By exploiting the output of these Machine Learning models, an attacker may identify some of the individuals in the dataset, thus presenting severe privacy concerns. This fact has led to broad research in the literature on privacy-preserving methods in Machine Learning.

In this thesis, it is investigated whether using Differential Privacy mechanisms in a Federated Learning process affects the model's accuracy and, if so, to what extend privacy can be preserved while retaining the model's utility. It is also compared whether the choice of a Differential Privacy mechanism will affect the accuracy of the model. It will be proven that the choice of a Differential Privacy mechanism over another in a Differentially Private Federated Learning setting does not significantly affect its model's accuracy.

Keywords: Federated Learning, Machine Learning, Distributed Machine Learning, Differential Privacy, Differentially Private Federated Learning, Differential Privacy Mechanisms, Privacy-Preserving Federated Learning, Trade-off Accuracy and Privacy.

Contents

1 Introduction

Traditionally, Machine Learning (ML) has been used to generate smart models that interact with (mobile) sensors to train a learning model. One must store the data required for the training of the ML models in a centralized database. These centralized databases with personal data/information however have raised a lot of concerns over the past years for privacy. Federated Learning (FL) and Differential Privacy (DP) can be applied to address some of these problems.

FL is a ML approach in which the data used to train the models stays on the (mobile) device itself. The training is performed on local models on the data stored on the devices and exclusively send unlabeled parameters to the server to improve the global model. The single centralized model aggregates the trained local models, by for example, averaging the parameters. The improved centralized model is then returned to the devices to be trained again. This process is performed iteratively until a required level of the global model accuracy by the application is achieved.

The applications and opportunities of FL are promising in terms of data privacy. However, several challenges have to be overcome before one can apply FL on a larger scale. One of these challenges it the trade-off between privacy preservation and the accuracy of the model. How can different design approaches of FL meet the trade-off regarding users' privacy and the accuracy of the training model? DP is one of the most common techniques used to preserve privacy in a FL environment. However, one can implement DP with different types of mechanisms. This research will analyze the trade-off between privacy preservation and trained model accuracy with the Laplace, Gaussian, and Staircase mechanisms. Specifically, this research aims to answer the question:

To what extent can Federated Learning meet the privacy concerns to share sensitive data and what is the trade-off between privacy and accuracy.

The rest of the thesis is organized as follows: In Section 2, some background information on Machine Learning, Federated Learning, and Differential Privacy is provided. Section 3 describes the experimental setup, with Section 4 describing the results and a discussion therein. Finally, a conclusion and directions for future work follows.

2 Background Information

This chapter provides the background information that is needed to understand the initial research. It consists of machine learning, distributed machine learning and differential privacy with three different mechanisms.

2.1 Machine Learning

According to Michie et al. [15], Machine Learning (ML) is generally taken to embrace automatic computing procedures based on logical or binary operations that learn a task from a series of examples. ML is the science of programming to think and act like humans without being specifically programmed to do so. It can be described as the building of a computer systems that automatically improves with experience and implements a learning process. ML can also be described as building good probabilistic models by a process of automatic learning and adapting from analyzing data without following explicit instructions. To train the ML model, try to fit the model in such a manner that with an input x, it is possible to predict an output y. In Figure 1, it is possible to see from existing data that Bob likes fast, joyful music better than slow sad music. From this existing data, we can predict if Bob likes a new song X or not.

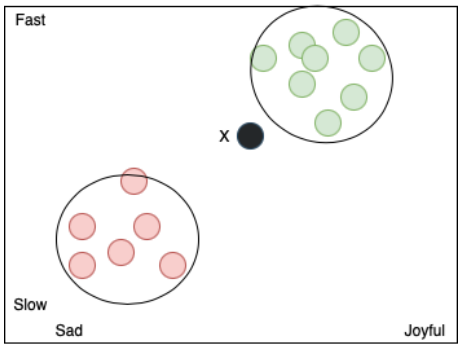

Figure 1: From existing data, it can be predicted through a ML model in which cluster a new song X will fit.

The distance between our prediction and the truth can be calculated by a loss-function. The aim is to minimize the loss as much as possible. When the loss value is minimal, the model maximized the prediction. This calculated loss-function shows the accuracy of the model. There are multiple ways to calculate such a loss. A common loss-function is the mean squared error given in Equation 1.

$$loss(x, y) = \frac{\sum_{i=1}^{n}(x - y)^2}{n} \tag{1}$$

The application of ML is used intensely around us today. A considerable amount of ML research focuses on automatically producing models that explain the working of a system, i.e. its rules. This includes the discovery of new algorithms, the limitations of certain ML methods, the issue of coping with imperfect training data and the way that the data is stored.

2.2 Federated Learning

Federated Learning (FL) is a distributed ML approach that enables training on a large collection of decentralized data sitting on devices like mobile phones [13]. FL is an illustration of the more

general approach of "bringing the code to the data, instead of the data to the code" [1]. In general, ML models have fundamental problems, such as privacy, ownership, and location of data. FL is a ML approach that addresses these problems. McMahan and Ramage [12] give a general description of FL, and the theory has been explored by Konečný et al. [8], McMahan et al. [14], and many more.

In a FL setting, multiple entities (clients) solve a ML dilemma under the structure of a central server or service provider. The raw data of each client is stored locally and is not exchanged or transferred to the server. Instead, focused updates for prompt aggregation are used to achieve the learning objective. Focused updates are updates that contain the minimum amount of information needed for the specific learning task. The implementation of early aggregation from the data can reduce many of the privacy risks and costs that occur in traditional, centralized ML. Thus, FL is a possible solution for many constraints and challenges that often occur in ML models on decentralized data where privacy is dominant.

Traditional ML assumes that the data on the clients are sampled as independent and identically distributed (i.i.d). Whereas, FL assumes non-i.i.d. data, as different clients contain different types of data [11]. Secondly, traditional ML assumes that the total number of clients is smaller than the available local training examples per client. For FL this assumption cannot be made as it is designed for large-scale scenarios where the number of clients can be larger in number. Finally, traditional ML assumes that the data is evenly distributed among all clients. This assumption is technically unimaginable for FL, as the expected number and the actual number of clients are different in real-time scenarios. Therefore, FL divides the number of fragments among the participated clients so that each client can receive an equal amount of data.

A FL model is created for a specific application and is typically steered by a model engineer. The workflow of a FL model involves the following.

1. Identify the problem which will be solved with FL.
2. Instruct the clients to store the necessary training data locally (with limits on time and quantity).
3. Optional, prototyping the model architecture and test the learning hyper-parameters in a FL simulation using a proxy dataset.
4. Multiple federated tasks start to train different variations of the model or the use of different optimization hyper-parameters.
5. When the federated tasks are completed, trained sufficiently, the models are analyzed and 'good' candidates are selected.
6. The models are pushed to the local devices for evaluation on local client data.
7. Once a good model is selected, the launch process will start. This includes manual quality assurance, live A/B testing and a staged roll-out.

The final step is independent on how the model is trained. Hence, this step would be the same for a model that is trained with a traditional ML model. FL is an continuous workflow, therefore each of those steps will be performed iteratively.

A typical FL training process includes the Federated Averaging algorithm, FedAvg [8]. The following workflow would be part of the described workflow of a FL model aforementioned.

1. The server samples from a set of clients that meet the requirements such as, network connection, sufficient battery power, etc.
2. The selected clients download the current model weights and a training program.
3. The selected clients locally compute an update to the model by executing the training program on their device.
4. The server collects an aggregate of the device updates.
5. The server updates the shared model based on the aggregated update computed from the clients that participated in the current iteration.

The described workflow is repeated until the training process is stopped. A key infrastructure challenge is to not impact the device of the client and therefore impact the users' experience. For example, heavyweight computation will only be executed when the device is idle, charging and connected to a WiFi network. These requirements are taken into consideration in the aforementioned Step 1 and will avoid impacting the users' experience.

Large technology companies such as Google have already deployed FL, and several startups were founded aiming to use FL to address privacy and data collection challenges in various industries. The scope of papers explored in this work suggests that FL is gaining traction in a wide range of interdisciplinary fields. This includes ML, optimization, information theory and statistics, cryptography, fairness and privacy. Up until now, FL has primarily considered supervised learning tasks where labels are naturally available on each client. Extending FL to other ML models such as semi-supervised and unsupervised learning are interesting and open challenges.

2.3 Differential Privacy

Differential Privacy (DP) is a privacy-preserving framework for determining to what extent individual privacy in a statistical dataset is guaranteed when releasing aggregate statistics about the dataset. Introduced by Dwork [2], it addresses many limitations of previous approaches like k-anonymity. The basic idea is to randomize part of the mechanism's behaviour to provide privacy. More generally, according to the most popular definition by Dwork and Roth [4], DP describes a promise made by a data holder, or curator, to a data subject (owner). The promise is like this: "You will not be affected adversely or otherwise by allowing your data to be used in any study or analysis, no matter what other studies, datasets, or information sources are available." Therefore, DP algorithms probably resist identification and re-identification attacks.

DP operates by adding statistical noise to the data (either inputs or outputs). The main idea behind DP is how much the output of the model change, if the data of a single individual in the dataset changes. The outcome of any analysis is approximately equal likely independent of whether the individual is part or not from the dataset. According to Dwork and Roth [4], the definition of DP is the following (Definition 1).

Definition 1. (Differential Privacy). A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|x|}$ is (ϵ, δ)-DP if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all $x.y \in \mathbb{N}^{|x|}$ such that $\|x - y\|_1 \leq 1$:

$$\mathsf{Pr}[\mathcal{M}(x) \in \mathcal{S}] \leq e^{(\epsilon)}\mathsf{Pr}[\mathcal{M}(y) \in \mathcal{S}] + \delta$$

where:

$\mathcal{M}$ = Randomized algorithm i.e. query(DB) + noise or query(DB + noise).
$\mathcal{S}$ = All potential outputs of $\mathcal{M}$ that could be predicted.
x = Entries in the database (i.e., N).
y = Entries in the parallel database (i.e., N-1).
ϵ = The maximum distance between a query or database(x) and the same query on database(y).
δ = Probability of information accidentally being leaked.

The measure by which these two probabilities of random distributions of the entire database x and the parallel database y can differ is given by epsilon (ϵ) and delta (δ). Epsilon is the maximum distance between a query on database x and the same query on database y. That is, it is a metric of privacy loss at a differential change in data (i.e., adding or removing one entry). Also known as the privacy parameter or the privacy budget. The nature of the privacy guarantees with differing but small ϵ's, is the same as someone is not able to distinguish the database by observing the output. Small values of ϵ require to provide very similar outputs when given similar inputs, and therefore provide higher levels of privacy; larger values of ϵ allow less similarity in the outputs and therefore provide less privacy.

Delta is the probability of information accidentally being leaked. If $\delta = 0$, we say that the output of mechanism $\mathcal{M}$ is ϵ-DP. Typically, we are interested in values of δ that are less than the inverse of any polynomial in the size of the database. Epsilon is independent of the size of the database, whereas, in the case of δ, the chances of privacy leaks might increase with the size of the database. Hence, ideally, we would want to get the δ value to be less than the inverse of the size of the database. According to Geng and Viswanath [6], ϵ-DP is defined as the following (Definition 2).

Definition 2. (ϵ-differential privacy). A randomized mechanism $\mathcal{K}$ gives ϵ-differential privacy if for all data sets D_1 and D_2 differing on at most one element and all $\mathcal{S} \subset$ Range($\mathcal{K}$),

$$\mathsf{Pr}[\mathcal{K}(D_1) \in S] \geq e^{(\epsilon)}\mathsf{Pr}[\mathcal{K}(D_2) \in S]$$

where $\mathcal{K}(D)$ is the random output of the mechanism $\mathcal{K}$ when the query function q is applied to dataset D.

$(\epsilon, 0)$-DP ensures that, for every run of the mechanism $\mathcal{M}(x)$, the output observed is (almost) equally likely to be observed in one very neighboring database simultaneously. In contrast, (ϵ, δ)-DP says that for every pair of neighbouring databases x, y, it is unlikely that, the observed value $\mathcal{M}(x)$ will be much more or much less likely to be generated when the database is x, then when the database is y. It ensures that for all adjacent x, y, the absolute value of the privacy loss will be bounded by ϵ with a probability of at least $1 - \delta$.

In case of $(\epsilon, 0)$-DP or ϵ-DP, where $\delta = 0$, i.e., probability of data leak δ is to be zero. Thus, a DP dataset with a distance of one or less will have a very similar output. Whereas, in the case of (ϵ, δ)-DP, the chances of data leaks are possible, maybe because of higher sensitivity values or large database size with large rows, etc. Hence, if the value of ϵ is not small, then the outputs are less likely to be the same. We usually go for ϵ-DP-based mechanisms – like the Laplace mechanism – not only because of its simplicity, but also because in real life, chances of the δ being high, i.e., data leak, is pretty low. (ϵ, δ)-DP usually works well with large datasets where chances of privacy leak are high and might not also have low sensitivity.

One way to achieve ϵ-DP and (ϵ, δ)-DP is to add noise sampled from Laplace and Gaussian distributions respectively, where the noise is equivalent to the sensitivity of the mechanism $\mathcal{M}$. Geng and Viswanath [6] defines the sensitivity as the following (Definition 3).

Definition 3. (Query Sensitivity). For a real valued query function $q : \mathcal{D}^n \to \mathbb{R}$, the sensitivity of q is defined as

$$\Delta = \max_{D_1, D_2 \in \mathcal{D}^n} |q(D_1) - q(D_2)|$$

for all D_1, D_2 differing in at most one element.

The sensitivity of a query can be stated both in the global and local context. Global sensitivity refers to the consideration of all possible datasets differing in at most one element. In contrast, local sensitivity is the change in one dataset with differing at most one element. Global sensitivity is the maximum difference of the output a query function can result in, when one change is made to any dataset [4]. Global sensitivity determines the magnitude of the noise needed to meet the ϵ-DP requirements. The formal definition of global sensitivity by Dwork et al. [3] shown in Definition 4.

Definition 4. (Global Sensitivity). For $f : \mathcal{D}^n \to \mathbb{R}^k$, and use the l_1-norm on $\mathbb{R}^k$ (denoted by $\|.\|_1$ or simply $\|.\|$) as a distance metric on outcomes of f. Then, the global sensitivity of f is

$$GS(f) = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|$$

Where:

$\|.\|_1 =$ the l_1-norm distance between datasets differing at most one element.
$\mathsf{max} =$ the maximum results of $f(D_1) - f(D_2)$ for all datasets D_1, D_2.

The global sensitivity is the minimum sensitivity needed for a query to cover all possible datasets. The definition of global sensitivity says that for any two neighboring datasets D_1 and D_2, the difference between $f(D_1)$ and $f(D_2)$ is at most $GS(f)$. This measure of sensitivity is called global, because it is independent of the actual dataset being queried: it holds for any choice of neighboring datasets D_1 and D_2.

The formal definition of local sensitivity by Nissim, Raskhodnikova, and Smith [18] is shown in Definition 5.

Definition 5. (Local Sensitivity). For $f : \mathcal{D}^n \to \mathbb{R}^k$, and $x \in \mathcal{D}^n$, the local sensitivity of f

$$LS(f) = \max_{D_2} \|f(D_1) - f(D_2)\|_1$$

Where:

$D_1 =$ the known dataset.
$D_2 =$ the other dataset with one different element relative to D_1.

Local sensitivity is a function of both the query f and the actual dataset D [4]. Unlike in the case of global sensitivity, we cannot talk about the local sensitivity without also considering the dataset at which that local sensitivity occurs.

The l_1-norm of a vector V of length k is defined as, i.e., the sum of the vector's elements. The l_1-sensitivity of a vector-valued function is equal to the sum of the element-wise sensitivities. For example, if we define a vector-valued function f that returns a length-k vector of l_1-sensitive results, then the l-sensitivity of f is k [4]. The l_1-sensitivity of a function gives an upper bound on how much we must perturb its output to preserve privacy, i.e., the l_1-sensitivity of a function f captures the magnitude by which a single individual's data can change the function f in the worst

case, and therefore, intuitively, the uncertainty in the response that we must introduce to hide the participation of a single individual [4].

The l_2-norm of a vector V of length k is defined as, i.e., the square root of the sum of the squares. The l_2-sensitivity of a vector-valued function is a vector-valued function f returning a length-k vector of l-sensitive results has l_2-sensitivity of $\sqrt{k}$. For long vectors, the l_2-sensitivity will be much lower than the l_1-sensitivity. For some applications like ML algorithms (which sometimes return vectors with thousands of elements), l_2-sensitivity is significantly lower than l_1-sensitivity [4].

DP satisfies a simple composition property: when two mechanism with privacy budgets ϵ_1 and ϵ_2 are performed on the same data, together they consume a privacy budget of $\epsilon_1 + \epsilon_2$. Thus, composing multiple DP mechanisms lead to a linear increase in the privacy budget (or corresponding increases in noise to maintain a fixed total privacy budget).

2.3.1 The Laplace Mechanism

The Laplace mechanism is one of the available solutions to add noise to the data to satisfy the DP. One way to achieve DP for a query function f is to add noise to its answer. The challenge is to find how much noise is enough to satisfy the definition of DP. If too much noise is added, the answer will get useless. In this case, the Laplace mechanism will compute F shown in Equation 2 and is agitated by the noise from the Laplace distribution.

$$F(x) = f(x) + \mathsf{Lap}(\frac{s}{\epsilon}) \tag{2}$$

where:

s $= $ the (global) sensitivity of f
$\mathsf{Lap}(S) =$ sampling from the Laplace distribution with $\mu = 0$ and scale S

In Figure 2, different cases of the Laplace distribution are shown. The parameter μ provides the x-coordinate of the centre of the peak. The parameter b provides the steepness of the curve. In the case of the Laplace mechanism, the sensitivity s will be equal to one and δ is always equal to zero [4]. The noise is scaled to $1/\epsilon$, hence adding noise drawn from $\mathsf{Lap}(1/\epsilon)$. The expected error is independent of the size of the database.

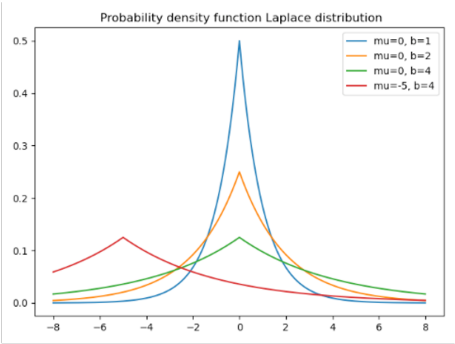

Figure 2: Probability Density Function of the Laplace distribution.

The Laplace mechanism preserves $(\epsilon, 0)$-DP, also written as ϵ-DP. This corresponds to the hypothesis that the more sensitive the query function f and the stronger the desired guarantee, the more "noise" is needed to achieve guarantee of ϵ-DP.

The Laplace mechanism has several drawbacks:

- It cannot be applied to non-numeric valued queries.
- Running multiple queries, a large ϵ value is needed to achieve the privacy guarantee, but it will produce a less accurate result.
- It can only be used for low-sensitivity queries (usually with l_1-sensitivity).

2.3.2 The Gaussian Mechanism

The Gaussian mechanism is an alternative to the Laplace mechanism, which adds Gaussian noise instead of Laplace noise. The Gaussian mechanism does not satisfy ϵ-DP but does satisfy (ϵ, δ)-DP. According to the Gaussian mechanism [4], for a query function $f(x)$ which returns a number, Definition 3 of $F(x)$ satisfies (ϵ, δ)-DP:

$$F(x) = f(x) + \mathcal{N}(\sigma^2) \tag{3}$$

where:

$$\sigma^2 = \frac{2s^2 log(1.25/\delta)}{\epsilon^2}$$

where

s = the sensitivity of f,
$\mathcal{N}(\sigma^2)$ = sampling from the Gaussian (normal) distribution with center $\mu = 0$ and variance σ^2.

In Figure 3, different cases of the Gaussian (normal) distribution with the parameters μ and b are shown. The parameter μ provides the x-coordinate of the centre of the peak and the parameter b provides the steepness of the curve.

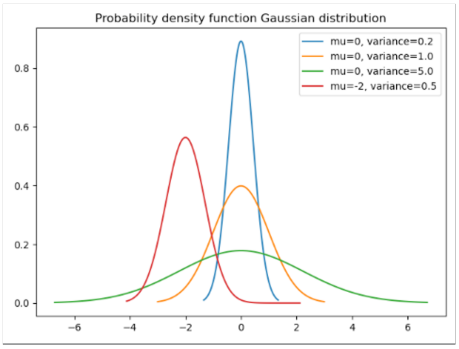

Figure 3: Probability Density Function of the Gaussian distribution.

For real-valued functions,

$$f : D \to \mathbb{R} \tag{4}$$

we can use the Gaussian mechanism in precisely the same way as we do the Laplace mechanism.

The Gaussian mechanism has two major drawbacks.

- It requires the use of the relaxed (ϵ, δ)-DP definition.
- It is less accurate than the Laplace mechanism.

It is however possible to extend both the Laplace and Gaussian mechanisms to vector-valued functions. Nevertheless, there is a critical difference between these two extensions shown in Definition 6 and 7 [17].

Definition 6. The vector-valued Laplace mechanism releases $f(x) + (Y_1, ..., Y_k)$, where Y_i are drawn $i.i.d$ from the Laplace distribution with scale $\frac{s}{\epsilon}$ and s is the l_1-sensitivity of f.

Definition 7. The vector-valued Gaussian mechanism releases $f(x) + (Y_1, ..., Y_k)$, where Y_i are drawn $i.i.d$ from the Gaussian distribution with $\sigma^2 = \frac{2s^2 log(1.25/\delta)}{\epsilon^2}$ and s is the l_2-sensitivity of f.

The vector-valued Laplace mechanism requires l_1-sensitivity, while the vector-valued Gaussian mechanism allows the use of either l_1 or l_2-sensitivity. This is a significant strength of the Gaussian mechanism. For applications in which the l_2-sensitivity is much lower than l_1-sensitivity, the Gaussian mechanism adds much less noise.

2.3.3 The Staircase Mechanism

The Staircase mechanism is introduced by Geng and Viswanath [6] for the one-dimensional case. Thereafter, Geng et al. [5] proves that given an ϵ-DP constraint under a general cost-minimization model adding a query-output independent noise is indeed optimal. In this case, the optimal noise distribution is not a Laplace distribution but has a staircase-shaped probability density function, which is shown in Figure 4. The following two settings must be true. First, the domain of the query output is the entire real line or the set of integers. Second, nothing more about the query function is known beyond the global sensitivity, and that either local sensitivity is unknown or it is the same as the global sensitivity as in the case of count queries.

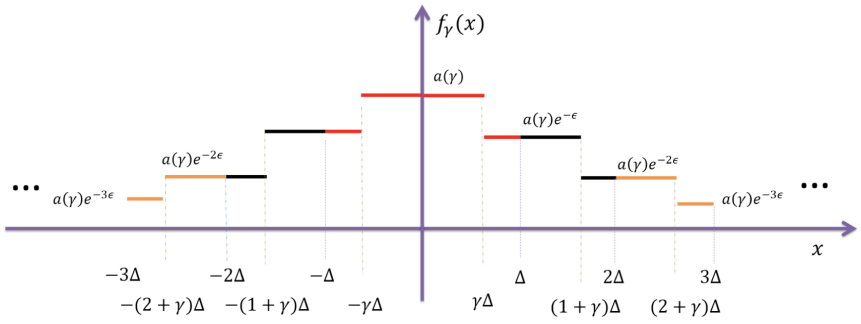

Figure 4: One-dimensional Staircase-Shaped Probability Density Function [5].

If any of the conditions mentioned above are violated, then the optimal privacy mechanism does not have to be data, or query output dependent. Furthermore, Geng et al. [5] generalizes the results of [6], from the single-dimensional setting to a multidimensional environment. According to Geng et al. [5], the Staircase mechanism is formalized as the following (Definition 8).

Definition 8. (Staircase Mechanism). Consider a class of multidimensional probability distributions with symmetric and staircase-shaped probability density function defined as follows. Given $\gamma \in [0,1]$, define $\mathcal{P}_\gamma$ as the probability distribution with probability density function $f_\gamma(.)$ defined as

$$f_\gamma(x) = \begin{cases} e^{-k\epsilon}\alpha(\gamma) & \|x\|_1 \in [k\Delta, (k+\gamma)\Delta) \\ e^{-(k+1)\epsilon}\alpha(\gamma) & \|x\|_1 \in [(k+\gamma)\Delta, (k+1)\Delta) \end{cases}$$

for $k \in \mathbb{N}$, where $\alpha(\gamma)$ is the normalization factor to make

$$\int\int ... \int f_\gamma(x) dx_1 dx_2 ... dx_d = 1$$

Define $b \triangleq e^{-\epsilon}$, and define

$$c_k \triangleq \sum_{i=0}^{+\infty} i^k b^i, \forall k \in \mathbb{N}$$

where by convention 0^0 is defined as 1. Then the closed-form expression for $\alpha(\gamma)$ is

$$\alpha(\gamma) \triangleq \frac{d!}{2^d \Delta^d \sum_{k=1}^{d} \left(\begin{array}{c} d \\ k \end{array} \right) c_{d-k}(b + (1-b)\gamma^k)}$$

The Staircase mechanism can be viewed as a geometric mixture of uniform random variables. Examples of Staircase mechanisms include binary and randomized response mechanisms [7].

To create a clear overview of how the Staircase mechanism would be implemented, Algorithm 1 [5] will give a simple implementation where the dimension is set to one.

Algorithm 1 Generation of Random Variable with Staircase Distribution

1: **input:** ϵ, Δ, and $\gamma \in [0,1]$.
2: $S \leftarrow \mathsf{Pr}[S=1] = \mathsf{Pr}[S=-1] = 1/2$ {Generate a r.v.}
3: $G \leftarrow \mathsf{Pr}[G=i] = (1-b)b^i$ for $i \geq 0, b = e^{-\epsilon}$ {Generate a geometric r.v.}
4: $U \leftarrow U(0,1)$ {Generate uniform on [0,1]}
5: $B \leftarrow \mathsf{Pr}[B=0] = \gamma/(\gamma + (1-\gamma)^b)$ and $\mathsf{Pr}[b=1] = (1-\gamma)^b/(\gamma + (1-\gamma)^b)$
6: $X \leftarrow S((1-B)((G+\gamma U)\Delta) + B((G+\gamma+(1-\gamma)U)\Delta))$
7: **output:** X {a random variable (r.v.) with staircase distribution specified by the input}.

3 Initial Research

A simulator is needed to analyze the performance of the accuracy and privacy preservation of the three aforementioned differential privacy mechanisms. In this section the simulator and the environment will be explained.

3.1 The Simulator

For the simulator, PrivacyFL is used. It is an extensible, easily configurable, and scalable simulator for FL environments [16]. The key features of the simulator include latency simulation, robustness to client departure/failure and support for both centralized (with one or more servers) and decentralized (serverless) learning. On top of that, it also has configurable privacy and security mechanisms based on DP and Secure Multiparty Computation (MPC).

The simulator supports the use of two DP mechanisms, namely Laplace and Gamma. For the experiments in this research two mechanisms are added, namely the Gaussian and Staircase mechanisms. The explanation of the two mechanisms can be found in the Sections 2.3.2 and 2.3.3.

The PrivacyFL simulator provides an efficient Python framework. It enables the users of the simulator to simulate a privacy-preserving secure FL. The following parameters will be changed independently for each of the three DP mechanism.

- The number of clients (C) that will participate in the learning process,
- the amount of data divided per client per round (S), and
- the number of iterations (N) for the learning process.

By doing this, the relationship between the privacy loss and the accuracy of the FL model can be examined. For the accuracy of the trained FL model, the DP mechanisms uses logistic regression [16]. Logistic regression is a supervised learning classification algorithm which calculates the probability of a discrete outcome given an input variable. The accuracy of the FL model is calculated by the number of correct predictions divided by the total number of predictions. The number of predictions is equal to the size of the test set of the chosen dataset.

By default, the simulator allows the experiments to be run on the MNIST dataset. For the experiments in this research, the MNIST and CIFAR-10 datasets are used to compare and evaluate the different mechanisms and settings. The MNIST dataset is a large database of images with handwritten digits zero through nine. The CIFAR-10 dataset is a large database with images containing 10 different classes. Both datasets are a multi-class classification problem, containing 10 classes. Logistic regression supports the use of multi-class classification. The code of the simulator is adapted to have the dataset of choice configurable, instead of only the MNIST dataset.

3.1.1 Parameters

The PrivacyFL simulator has multiple configuration parameters, shown in Table 1, that are set to True or False by the system designer. During the experiments in this research, these parameters will remain fixed. In table 2, the parameters that are interchangeable during the experiments are shown. The setting column shows the changeable parameters used during the experiments in this research. They can be set to different values if preferred by the system designer. In Section 4, the results of each different scenarios of the parameters are explained and displayed.

Table 1: Fixed configuration parameters of the PrivacyFL simulator [16].

Parameter	Setting	Explanation
USE_SECURITY	False	If True, the clients perform a Diffie-Hellman key exchange in the offline portion of the simulation to establish common keys for encryption. Has no effect on federated accuracy.
USE_DP_PRIVACY	True	If True, clients will add DP noise with parameters specified in the `config.py` file.
SUBTRACT_DP_NOISE	False	If True, clients will subtract the noise they added to the federated model upon receiving it from the server. If False, clients use the federated model computed by the server.
CLIENT_DROPOUT	False	If True, a client drops out of simulation when each weight in the federated model is within `config.tolerance` of the client's weight. The simulation continues without that client. If False, no client dropouts take place.
SIMULATE_LATENCIES	True	If True, the system displays the time it would take for each step in the protocol to complete. If False, this information is not displayed.
USING_CUMULATIVE	True	If using our data partitioning module, this flag is useful for experimenting between dataset options. If False, the dataset for each iteration includes only the new data available that iteration. If True, the size of the dataset grows each iteration by the amount of new data. The weights of the ith iteration are not used in producing the weights of the i + 1th iteration.

Table 2: Changeable parameters of the PrivacyFL simulator defined by the system designer for each experiment.

Parameter	Setting	Explanation
DP_ALGORITHM	Laplace, Gaussian or Staircase	Depending on the DP algorithm defined, the model will use the selected DP mechanism for training the FL model. Hence, when Laplace is selected the Laplace noise will be added to the model.
DATASET	mnist or cifar10	Depending on the dataset defined, the FL model will train and test the data on the selected dataset.
NUM_CLIENTS (C)	3 or 8	The number of clients that will participate in the FL training model.
ITERATIONS (N)	10	The amount of iterations the FL training model is ran before completed.
len_per_iteration (S)	100 or 200	How many data points each client gets per iteration (starts at 0). During the experiments, an equal size of data points for each client per iteration is used.
epsilon (ϵ)	0.1 or 1.0	The privacy budget, a small value of ϵ provides higher privacy preservation, but lower accuracy and vice versa.

3.1.2 Lifecycle

As outlined by Mugunthan, Peraire-Bueno, and Kagal [16], the steps carried out in the simulations life-cycle is as follows:

1. Specify the parameters in the `config.py` file, also shown in Section 3.1.1, and run the FL simulation in `run_simulation.py`.
2. `run_simulation.py` creates an instance of Initializer, which initializes the server agent and the client agents.
3. When the `run_simulation.py` file is ran, the `run_simulation()` function is started which calls the server agent's `request_values()` function.
4. The following steps are repeated `num_iterations` number of times.
    - 4.1 The server agent requests the weights from the clients by calling the `produce_weights()` function for each of the clients.
    - 4.2 In the `produce_weights()` function, the ML model trains each client on the defined dataset for that iteration. Each clients saves and copies the weights locally. On the copied weights, the security and DP noise are added, if the configurations are set to true. The modified weights are returned to the server agent.
    - 4.3 The server agent averages the received weights and returns the federated weights to each client through the `receive_weights()` function.
    - 4.4 At the end of each iteration, each client computes the accuracy of the federated model versus its local model on the test-set.

3.2 Programming Environment & Hardware

To research the performance of the three differential privacy mechanisms with the aforementioned simulator the following environment must be set up. Python is used as the programming language in the simulator PrivacyFL. In order to run the given simulator, Python 3.8 is used. For the IDE, I chose PyCharm Professional 2020.2 made by JetBrians.

The analysis of the performance were done on a 2019 MacBook Pro with 2.4 GHz Quad-Core Intel Core i5 processor and 8GB of RAM.

3.3 The Data Set

For the experiments the clients are trained on the MNIST and CIFAR-10 datasets. Below both of the datasets are explained and a sample of the dataset is shown.

3.3.1 The MNIST Dataset

A classic dataset for ML is the MNIST database. It is a large database of handwritten digits that is used for training and testing a ML model. MNIST is a low-complexity dataset and consists of 60.000 images for training and 10.000 images for testing. The splitting of the images ensures that the trained model can accurately be tested on images preciously not examined. The dataset is used to train and test various supervised ML algorithms. In Figure 5, you see an example of the 28 x 28 black and white images representing the digits zero through nine. The simulator PrivacyFL uses the free ML library Scikit-learn. Therefore, the `sklearn.datasets` package is introduced in order to initialise the MNIST dataset from OpenML [21].

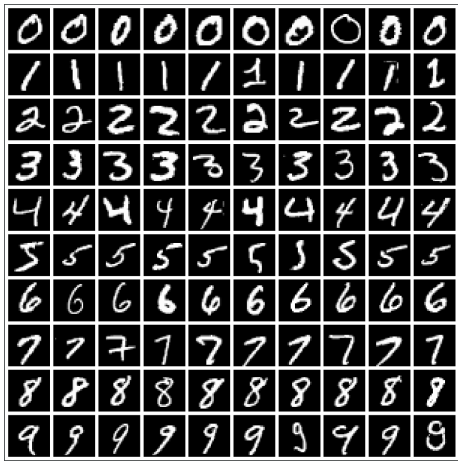

Figure 5: Sample images from the MNIST dataset.

3.3.2 The CIFAR-10 Dataset

The second dataset that is used in order to analyze the performance of the DP mechanisms, is CIFAR-10. This dataset is chosen because it has a higher complexity than the MNIST dataset. Therefore, the experiments will receive a more reliable performance overview. The CIFAR-10 contains 60.000, 32 x 32 color images in 10 different classes. Each class contains 6.000 images. The classes in the range from zero through nine correspond to the following class labels: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck, respectively. In Figure 6, you see an example of the color images for each different class. The dataset is split into five training batches and one test batch, each containing 10.000 images. Each batch contains 1.000 randomly-selected images from each class.

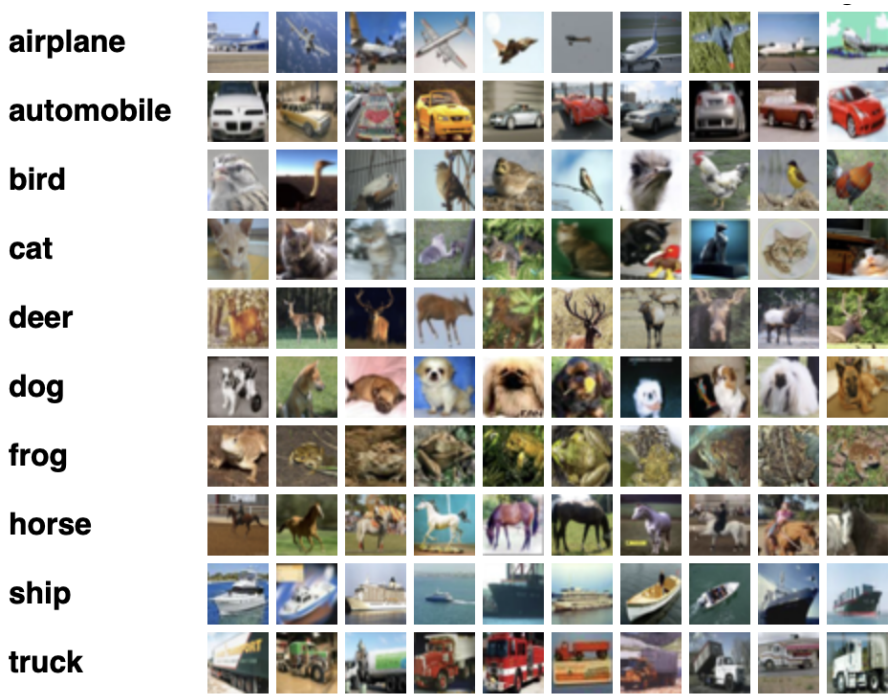

Figure 6: Sample images of the CIFAR-10 dataset, with 10 randomly selected images from each class. [9].

The CIFAR-10 dataset is collected by Alex Krizhevsky [9]. The dataset will be retrieved from [9] in case that the PrivacyFL simulator is ran and the CIFAR-10 dataset is not downloaded on that specific computer. Otherwise, the simulator will start running the simulation right away.

3.4 Implementation of the Mechanisms

In this section is described how the introduced methodologies above are applied in the experiment to investigate the trade-off between FL model accuracy and privacy preservation.

During the experiment, the following questions will be answered.

- What is the trade-off between FL model accuracy and privacy preservation?
- Among the Laplace, Gaussian, and Staircase mechanisms, which DP mechanism applied to the logistic regression model yields a better accuracy?

During the experiments with the different DP mechanisms and datasets, different performance metrics are measured. The accuracy of the FL model is measured for the prediction performance. For the privacy preservation, different values of the privacy loss parameter ϵ are applied, which will vary the amount of DP noise added to the DP mechanisms. As explained in Section 2.3, a small value of ϵ provides higher privacy preservation and vice versa. Therefore, the two different values for the privacy budget are chosen as: $\epsilon \in \{0.1, 1.0\}$, with $\epsilon = 0.1$ as a small value. Each implementation across the range of privacy budgets and datasets are evaluated, holding that each implementation has the same mathematical privacy guarantee.

4 Results

4.1 The Base Case

For the base case, the results show the accuracy for the training process of the FL logistic regression model without the use of DP, hence `USE_DP_PRIVACY` is set to False. In Figure 7, the accuracy of the base case is compared to the number of iterations for the training process. It shows the results of the four different cases without DP in this research for both datasets MNIST and CIFAR-10. The results of the base case provide a basis to compare the accuracy of the training process of the FL model with and without the use of DP.

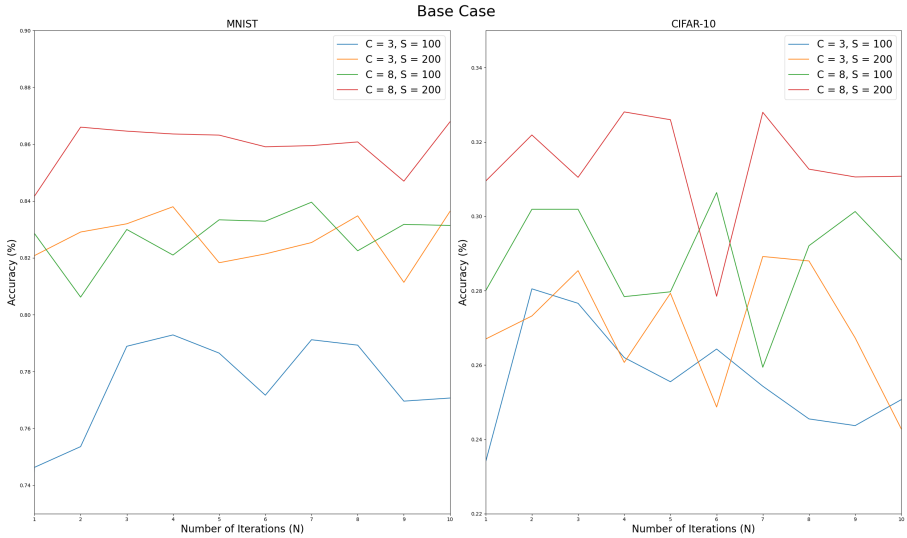

Figure 7: The base case, the training process of the FL logistic regression model without DP.

In Table 3, the average accuracy for the base case, the training of the FL logistic regression model without DP, is shown. Hence, when training the model with three clients, each with a batch size of 100, the average accuracy for the MNIST dataset equals to 78% and the CIFAR-10 dataset equals to 26%. The difference in (average) accuracy can be explained by the complexity of the dataset. The MNIST dataset is a low-complexity dataset, while the CIFAR-10 has a higher complexity. Therefore, the FL model discovers the training on the MNIST dataset to be easier and results in a higher accuracy.

Table 3: The average accuracy for the training of the FL logistic regression model without DP, with different parameters.

Parameters			**Average accuracy**	
Clients	Iterations	Sample size	MNIST	CIFAR-10
3	10	100	0,78	0,26
3	10	200	0,83	0,27
8	10	100	0,83	0,29
8	10	200	0,86	0,31

4.2 The Laplace Mechanism

As previously outlined, there are two scenarios to be considered. First, when the privacy loss parameter (ϵ) is equal to 0.1. Second, when ϵ is equal to 1.0.

4.2.1 Scenario 1: epsilon = 0.1

In scenario 1, the privacy loss parameter is set to 0.1. The following experiments will show the effect on the model accuracy compared to three settings. Each of the three settings are compared to the two dataset, MNIST and CIFAR-10. First, when the number of clients (C) participating in the FL process changes. Second, when the size of the sample dataset for each client (S) in the training process changes. Finally, when the number of iterations in the FL process changes.

Model Accuracy vs. Number of Clients Participating in the FL Process.
Figure 8 illustrates the accuracy of the ML model trained with FL, using the Laplace mechanism when $C \in \{3, 8\}$ and $\epsilon = 0.1$.

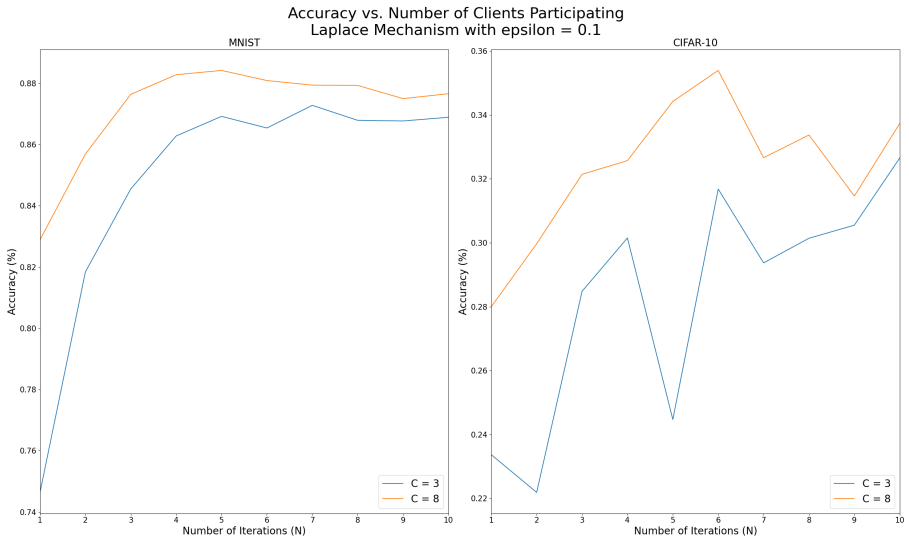

Figure 8: The training process of the FL logistic regression model with the Laplace mechanism. Where the accuracy is compared with different amount of clients participating in the training and epsilon=0.1.

Model Accuracy vs. Size of Dataset per Client.
Figure 9 illustrates the accuracy of the ML model trained with FL and using the Laplace mechanism when $S \in \{100, 200\}$, $C \in \{3, 8\}$ and $\epsilon = 0.1$.

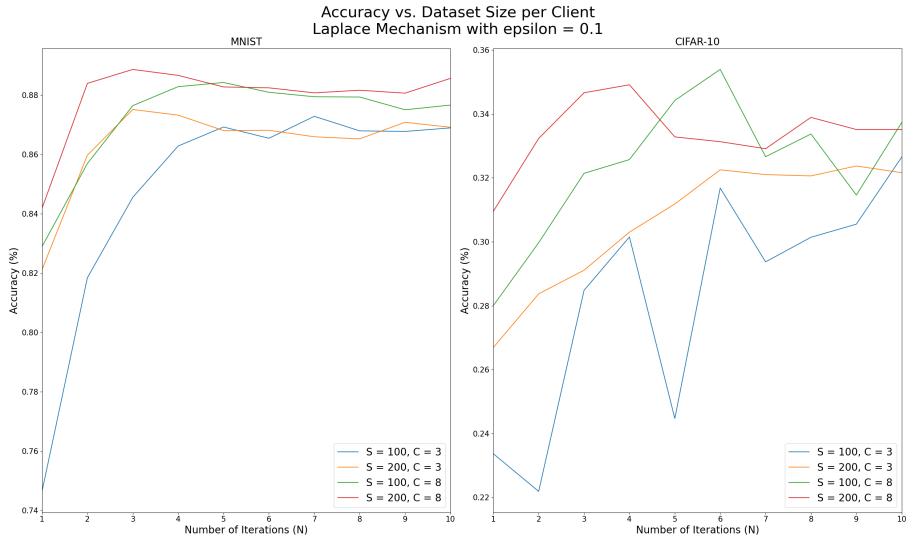

Figure 9: The training process of the FL logistic regression model with the Laplace mechanism. Where the accuracy is compared with different sample sizes of the dataset and different amount of clients participating in the training for epsilon=0.1.

Model Accuracy vs. Number of Iterations.
For both dataset, as the number of iterations increases, so does the accuracy of the model. For the MNIST dataset, the accuracy for each setting pretty much stagnate after three iterations. While, for the CIFAR-10 dataset, the accuracy differs with each iteration. For both datasets, the most favourable settings are 2 adjacent settings, namely $C = 8$ with $S = 100$ and $S = 200$.

4.2.2 Scenario 2: epsilon = 1.0

Similarly to Scenario 1 (Section 4.2.1), the same parameters for each simulation with an exception of the privacy loss parameters are set the same. In this scenario the privacy loss parameter is set to 1.0.

Model Accuracy vs. Number of Clients Participating in the FL Process.
Figure 10 illustrates the accuracy of the ML model trained with FL, using the Laplace mechanism when $C \in \{3, 8\}$ and $\epsilon = 1.0$.

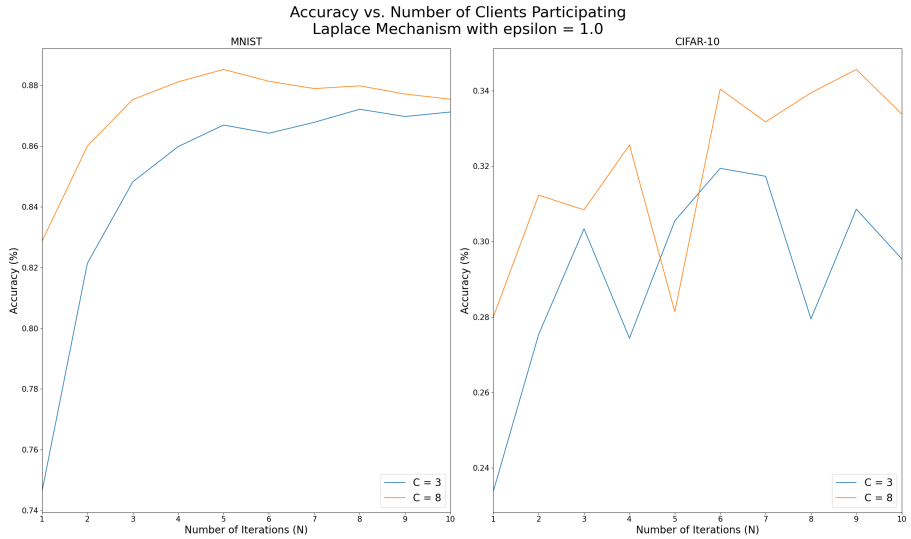

Figure 10: The training process of the FL logistic regression model with the Laplace mechanism. Where the accuracy is compared with different amount of clients participating in the training and epsilon=1.0.

Model Accuracy vs. Size of Dataset per Client.
Figure 11 illustrates the accuracy of the ML model trained with FL and using the Laplace mechanism when $S \in \{100, 200\}$, $C \in \{3, 8\}$ and $\epsilon = 1.0$.

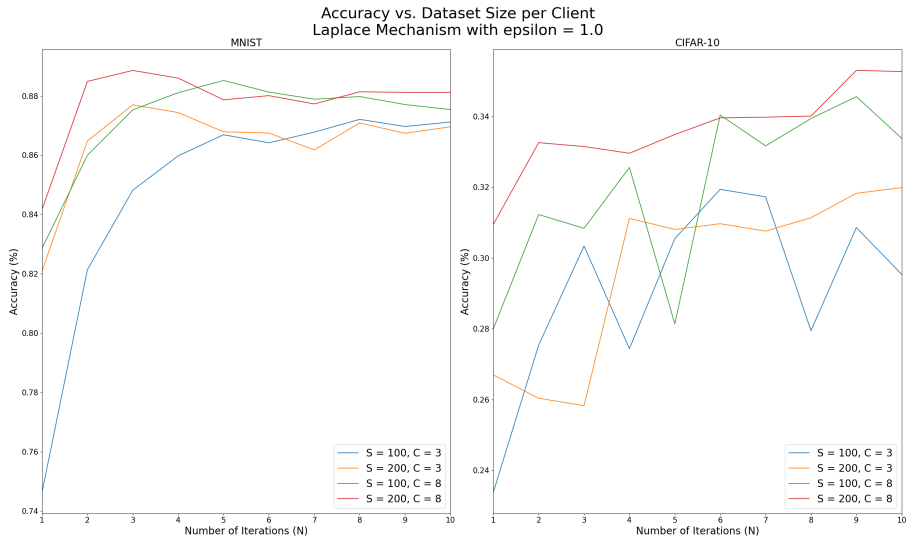

Figure 11: The training process of the FL logistic regression model with the Laplace mechanism. Where the accuracy is compared with different sample sizes of the dataset and different amount of clients participating in the training for epsilon=1.0.

Model Accuracy vs. Number of Iterations.
For both dataset, as the number of iterations increases, so does the accuracy of the model. For the MNIST dataset, the accuracy for each setting pretty much stagnate after three iterations. While, for the CIFAR-10 dataset, the accuracy differs with each iteration. For both datasets, the most favourable settings are 2 adjacent settings, namely $C = 8$ with $S = 100$ and $S = 200$.

4.2.3 Model Accuracy vs. Privacy Loss

In Table 4 and 5, it results that the accuracy of the FL logistic regression model with DP performs a fraction higher than without the use of DP (base case) in both datasets. Therefore, it is not obvious that the privacy loss parameter significantly affects the model accuracy, as we achieve very similar accuracy's.

Table 4: Summary of the average accuracy for each scenario of the MNIST dataset with the Laplace mechanism.

Parameters			**Average accuracy**		
C	N	S	Base case	Scenario 1	Scenario 2
3	10	100	0,78	0,85	0,85
3	10	200	0,83	0,86	0,86
8	10	100	0,83	0,87	0,87
8	10	200	0,86	0,88	0,88

Table 5: Summary of the average accuracy for each scenario of the CIFAR-10 dataset with the Laplace mechanism.

Parameters			**Average accuracy**		
C	N	S	Base case	Scenario 1	Scenario 2
3	10	100	0,26	0,28	0,29
3	10	200	0,27	0,31	0,30
8	10	100	0,29	0,32	0,32
8	10	200	0,31	0,33	0,34

4.3 The Gaussian Mechanism

Similarly, as the Laplace mechanism experiment, the two scenarios are considered. First, when the privacy loss parameter (ϵ) is equal to 0.1. Second, when the ϵ is equal to 1.0.

4.3.1 Scenario 1: epsilon = 0.1

In scenario 1, the privacy loss parameter is set to 0.1. The following experiments will show the effect on the model accuracy compared to three settings. Each of the three settings are compared to the two dataset, MNIST and CIFAR-10. First, when the number of clients (C) participating in the FL process changes. Second, when the size of the sample dataset for each client (S) in the training process changes. Finally, when the number of iterations in the FL process changes.

Model Accuracy vs. Number of Clients Participating in the FL Process.
Figure 12 illustrates the accuracy of the ML model trained with FL, using the Gaussian mechanism when $C \in \{3, 8\}$ and $\epsilon = 0.1$.

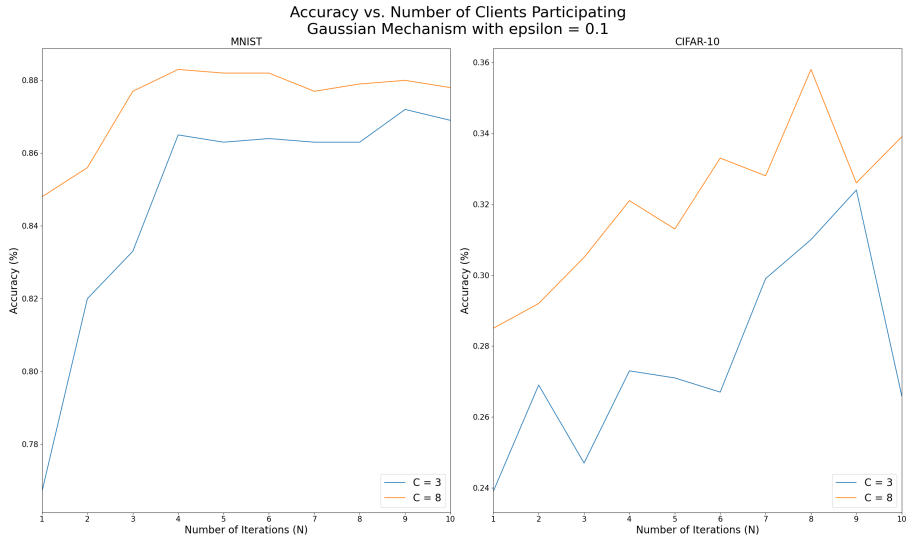

Figure 12: The training process of the FL logistic regression model with the Gaussian mechanism. Where the accuracy is compared with different amount of clients participating in the training and epsilon=0.1.

Model Accuracy vs. Size of Dataset per Client.
Figure 13 illustrates the accuracy of the ML model trained with FL and using the Gaussian mechanism when $S \in \{100, 200\}$, $C \in \{3, 8\}$ and $\epsilon = 0.1$.

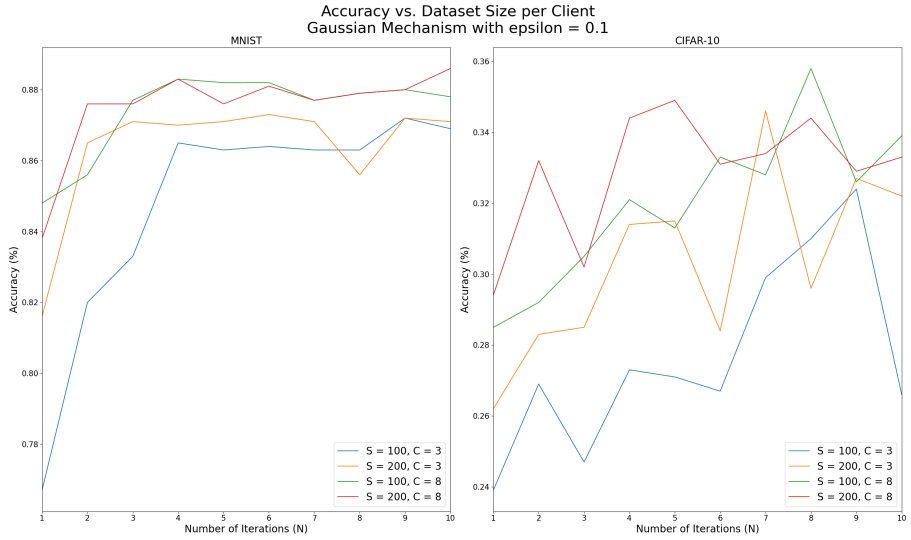

Figure 13: The training process of the FL logistic regression model with the Gaussian mechanism. Where the accuracy is compared with different sample sizes of the dataset and different amount of clients participating in the training for epsilon=0.1.

Model Accuracy vs. Number of Iterations.
For the MNIST dataset, as the number of iterations increases, so does the accuracy of the model. Only for the experiment where the settings are $S = 200$ and $C = 3$ there is a drop at iteration 8. For the CIFAR-10 dataset, the accuracy differs significantly. The settings for the experiment with $C = 3$ and $S = 100$ is distinctly lower than the other experiments.

For the MNIST dataset, there are two settings adjacent to the most favourable setting, namely $C = 8$ with $S = 100$ and $S = 200$.

4.3.2 Scenario 2: epsilon = 1.0

Similarly to Scenario 1 (Section 4.3.1), the same parameters for each simulation with an exception of the privacy loss parameters are set the same. In this scenario the privacy loss parameter is set to 1.0.

Model Accuracy vs. Number of Clients Participating in the FL Process.
Figure 14 illustrates the accuracy of the ML model trained with FL, using the Gaussian mechanism when $C \in \{3, 8\}$ and $\epsilon = 1.0$.

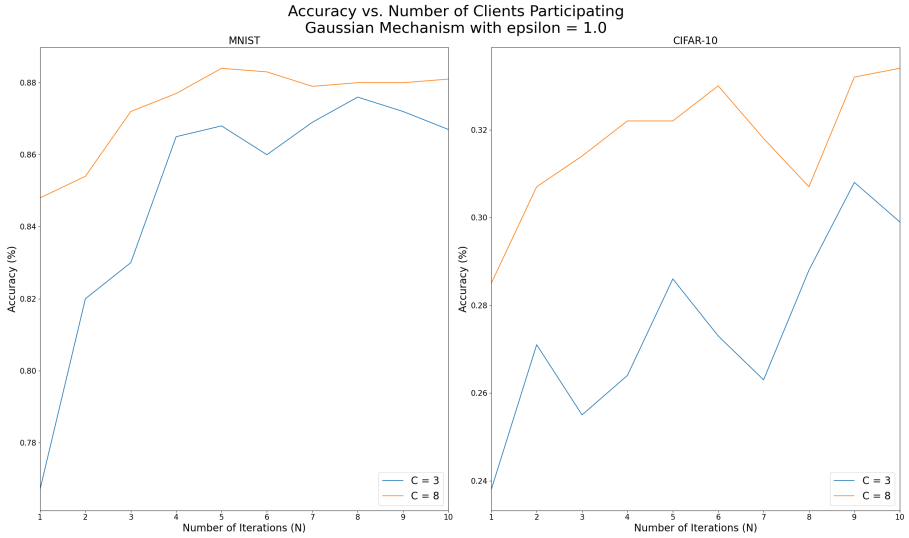

Figure 14: The training process of the FL logistic regression model with the Gaussian mechanism. Where the accuracy is compared with different amount of clients participating in the training and epsilon=1.0.

Model Accuracy vs. Size of Dataset per Client.
Figure 15 illustrates the accuracy of the ML model trained with FL and using the Gaussian mechanism when $S \in \{100, 200\}$, $C \in \{3, 8\}$ and $\epsilon = 1.0$.

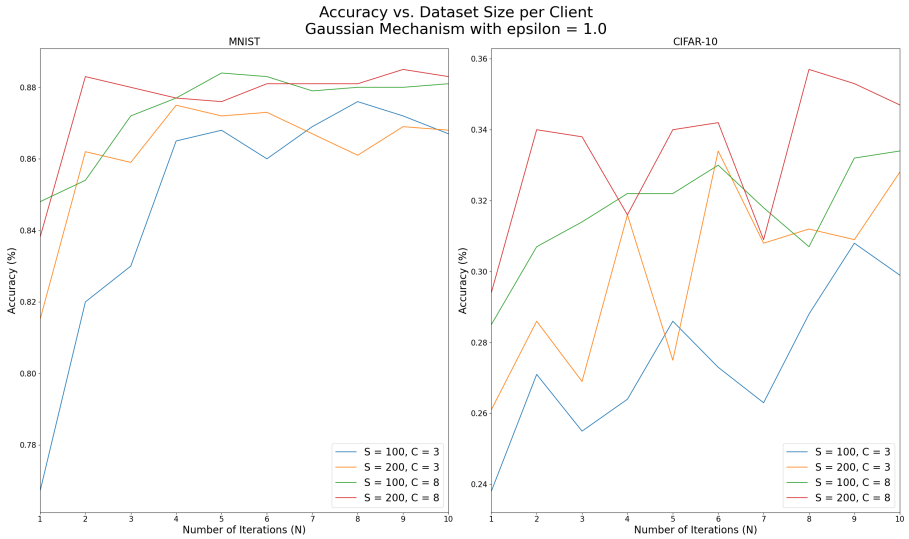

Figure 15: The training process of the FL logistic regression model with the Gaussian mechanism. Where the accuracy is compared with different sample sizes of the dataset and different amount of clients participating in the training for epsilon=1.0.

Model Accuracy vs. Number of Iterations.
For the MNIST dataset, the settings of the experiment with $C = 3$ increases significantly from 77% to 87% in ten iterations. The difference for the $C = 8$ is significantly lower. Still $C = 8$ outperforms the $C = 3$. For the CIFAR-10 dataset, the settings of the experiment with $C = 3$ is again outperformed by all other settings for the experiments. For both datasets, the most favourable setting for ten iterations is $C = 8$ and $S = 200$.

4.3.3 Model Accuracy vs. Privacy Loss

In Table 6 and 7, it results that the accuracy of the FL logistic regression model with DP performs higher than without the use of DP (base case) in both datasets. Therefore, it is not obvious that the privacy loss parameter significantly affects the model accuracy, as we achieve very similar accuracy's.

Table 6: Summary of the average accuracy for each scenario of the MNIST dataset with the Gaussian mechanism.

Parameters			**Average accuracy**		
C	N	S	Base case	Scenario 1	Scenario 2
3	10	100	0,78	0,85	0,85
3	10	200	0,83	0,86	0,86
8	10	100	0,83	0,87	0,87
8	10	200	0,86	0,88	0,88

Table 7: Summary of the average accuracy for each scenario of the CIFAR-10 dataset with the Gaussian mechanism.

Parameters			Average accuracy		
C	N	S	Base case	Scenario 1	Scenario 2
3	10	100	0,26	0,28	0,27
3	10	200	0,27	0,30	0,30
8	10	100	0,29	0,32	0,32
8	10	200	0,31	0,33	0,33

4.4 The Staircase Mechanism

Finally, similarly as the Laplace and Gaussian mechanism experiments, the two scenarios are considered. First, when the privacy loss parameter (ϵ) is equal to 0.1. Second, when the ϵ is equal to 1.0.

4.4.1 Scenario 1: epsilon = 0.1

In scenario 1, the privacy loss parameter is set to 0.1. The following experiments will show the effect on the model accuracy compared to three settings. Each of the three settings are compared to the two dataset, MNIST and CIFAR-10. First, when the number of clients (C) participating in the FL process changes. Second, when the size of the sample dataset for each client (S) in the training process changes. Finally, when the number of iterations in the FL process changes.

Model Accuracy vs. Number of Clients Participating in the FL Process.
Figure 16 illustrates the accuracy of the ML model trained with FL, using the Staircase mechanism when $C \in \{3, 8\}$ and $\epsilon = 0.1$.

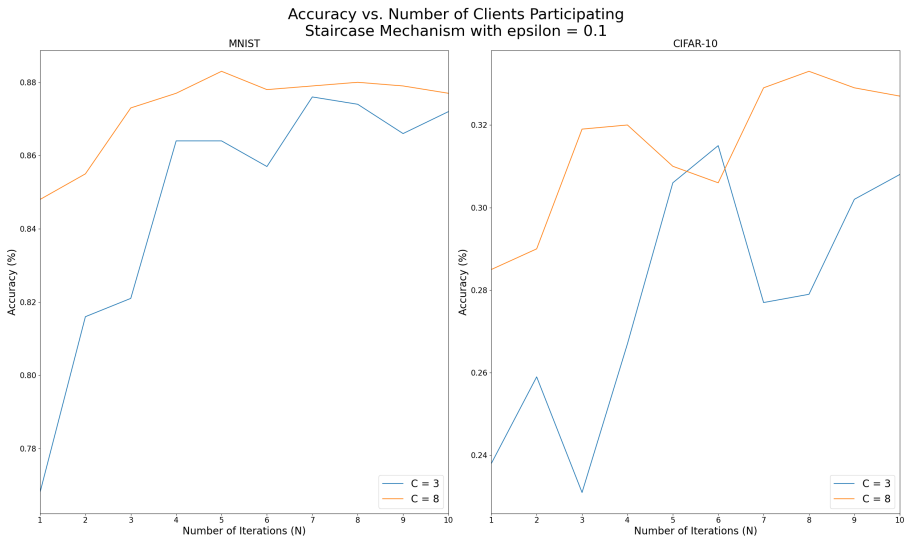

Figure 16: The training process of the FL logistic regression model with the Staircase mechanism. Where the accuracy is compared with different amount of clients participating in the training and epsilon=0.1.

Model Accuracy vs. Size of Dataset per Client.
Figure 17 illustrates the accuracy of the ML model trained with FL and using the Staircase mechanism when $S \in \{100, 200\}$, $C \in \{3, 8\}$ and $\epsilon = 0.1$.

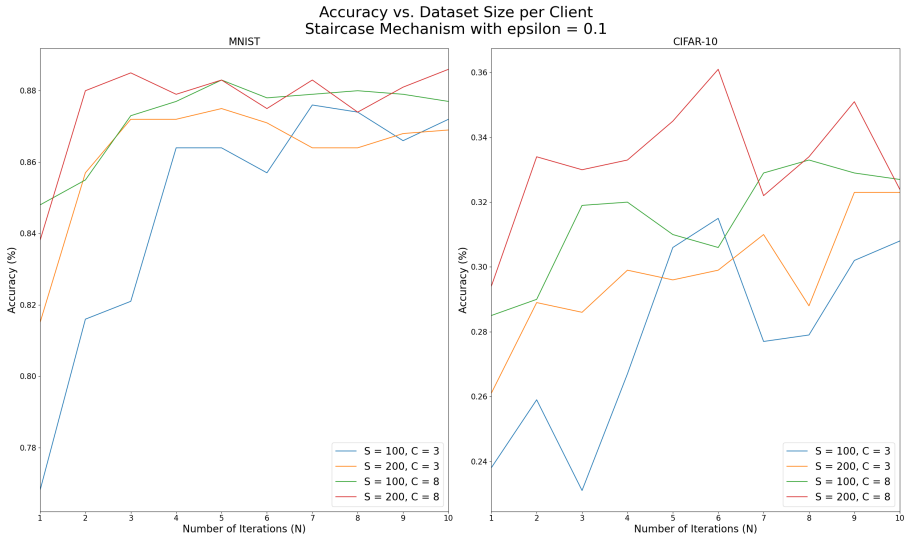

Figure 17: The training process of the FL logistic regression model with the Staircase mechanism. Where the accuracy is compared with different sample sizes of the dataset and different amount of clients participating in the training for epsilon=0.1.

Model Accuracy vs. Number of Iterations.
For both dataset, as the number of iterations increases, so does the accuracy of the model. In the CIFAR-10 dataset there are however a few drops around iteration 7 and 8. For the MNIST dataset, at the end of iteration 10, the accuracy of the four experiments do not differ a great deal. For the CIFAR-10 dataset, at the end of iteration, the experiment with $C = 3$ and $S = 100$ settings is distinctly lower than the other settings.

4.4.2 Scenario 2: epsilon = 1.0

Similarly to Scenario 1 (Section 4.4.1), the same parameters for each simulation with an exception of the privacy loss parameters are set the same. In this scenario the privacy loss parameter is set to 1.0.

Model Accuracy vs. Number of Clients Participating in the FL Process.
Figure 18 illustrates the accuracy of the ML model trained with FL, using the Staircase mechanism when $C \in \{3, 8\}$ and $\epsilon = 1.0$.

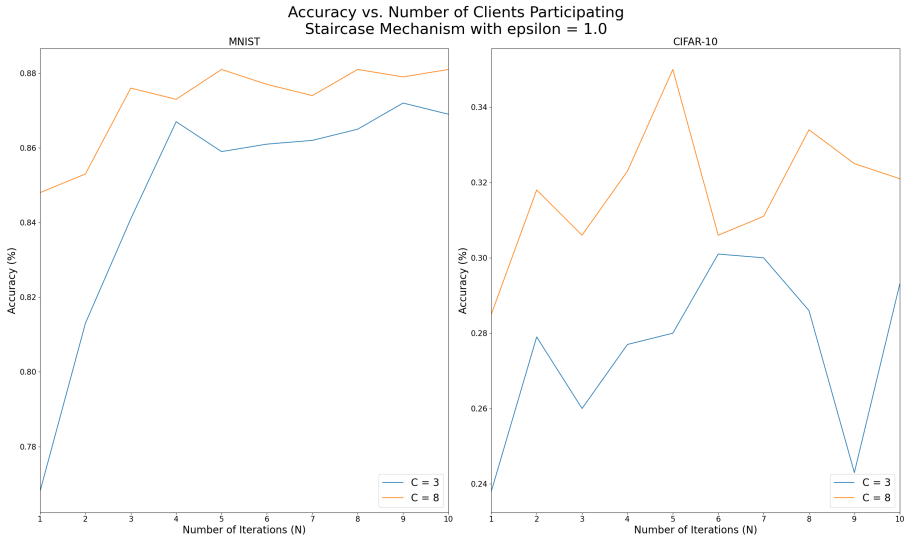

Figure 18: The training process of the FL logistic regression model with the Staircase mechanism. Where the accuracy is compared with different amount of clients participating in the training and epsilon=1.0.

Model Accuracy vs. Size of Dataset per Client.
Figure 19 illustrates the accuracy of the ML model trained with FL and using the Staircase mechanism when $S \in \{100, 200\}$, $C \in \{3, 8\}$ and $\epsilon = 1.0$.

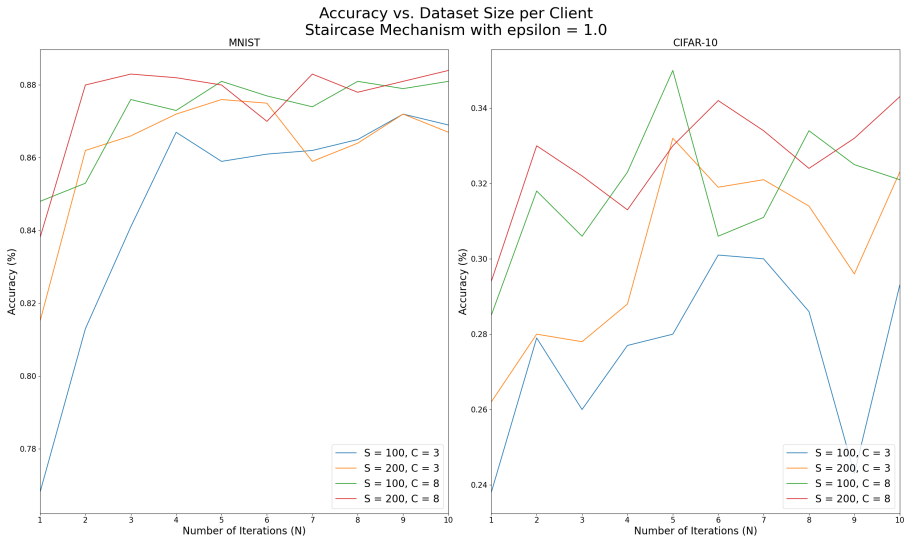

Figure 19: The training process of the FL logistic regression model with the Staircase mechanism. Where the accuracy is compared with different sample sizes of the dataset and different amount of clients participating in the training for epsilon=1.0.

Model Accuracy vs. Number of Iterations.
For the MNIST dataset, as the number of iterations increases, the accuracy of the model increases simultaneously. For the CIFAR-10 dataset this is not the case. At iteration 9, the $C = 3$ has a tremendous drop of the accuracy of the model. For the MNIST dataset, the two most favourable setting for ten iterations are $C = 8$ with $S = 100$ and $S = 200$. For the CIFAR-10 dataset, after ten iterations the most favourable setting is $C = 8$ and $S = 200$. But at iteration five, the accuracy of the model is the highest for $C = 8$ and $S = 100$.

4.4.3 Model Accuracy vs. Privacy Loss

In Table 8 and 9, it results that the accuracy of the FL logistic regression model with DP performs higher than without the use of DP (base case) in both datasets. Therefore, it is not obvious that the privacy loss parameter significantly affects the model accuracy.

Table 8: Summary of the average accuracy for each scenario of the MNIST dataset with the Staircase mechanism.

Parameters			**Average accuracy**		
C	N	S	Base case	Scenario 1	Scenario 2
3	10	100	0,78	0,85	0,85
3	10	200	0,83	0,86	0,86
8	10	100	0,83	0,87	0,87
8	10	200	0,86	0,88	0,88

Table 9: Summary of the average accuracy for each scenario of the CIFAR-10 dataset with the Staircase mechanism.

Parameters			**Average accuracy**		
C	N	S	Base case	Scenario 1	Scenario 2
3	10	100	0,26	0,28	0,28
3	10	200	0,27	0,30	0,30
8	10	100	0,29	0,31	0,32
8	10	200	0,31	0,33	0,33

4.5 Summary of the Results

FL is a promising solution to the analysis of privacy-sensitive data distributed globally across clients. At the core of FL is Federated Averaging, an aggregation algorithm that consolidates the weighted average of distributed ML models into a global model shared with every client participating in the learning model.

In this section, the summary of the results (shown in Table 10 and 11) from the experiments conducted in this thesis are discussed. With the help of empirical studies, it is possible to demonstrate that DP on a client level is feasible and a high model accuracy can be reached when adequate many parties are involved in the training process. When the number of clients participating in the FL model is increased, hence from three to eight, it is observed that the accuracy of all models using the various DP mechanism tends to improve with the number of clients.

Similarly, when the amount of data per client per round is increased, it is observed that the accuracy of all the DP mechanisms tend to improve. Furthermore, as the number of iteration increases from the first to the tenth iteration, the accuracy of all models using the different DP mechanisms tend to improve as well.

To go into more detail, for the Laplace mechanism, the highest achieved accuracy for the MNIST dataset is 88%. There are two cases that achieve this accuracy. The values $\epsilon = 0.1$ and $\epsilon = 1.0$ with $C = 8$ and $S = 200$. Since the two cases of ϵ yield the same accuracy, the most desirable setting is with a higher privacy preservation, hence $\epsilon = 0.1$. For the Laplace mechanism, the highest achieved accuracy for the CIFAR-10 dataset is 34%. Although, with a higher privacy preservation ($\epsilon = 0.1$) the model achieves 33%. The most desirable setting depends on the amount of privacy preservation the system designer wants to provide to the participating clients.

For the Gaussian mechanism, the highest achieved accuracy for the MNIST dataset is 88%. There are two cases that achieve this accuracy. The values $\epsilon = 0.1$ and $\epsilon = 1.0$ with $C = 8$ and $S = 200$. Since the two cases of ϵ yield the same accuracy, the most desirable setting is with a higher privacy preservation, hence $\epsilon = 0.1$. For the Gaussian mechanism, with the CIFAR-10 dataset, there are two cases that achieve the highest accuracy of 33%. The values $\epsilon = 0.1$ and $\epsilon = 1.0$ with $C = 8$ and $S = 200$. Since the two cases of ϵ yield the same accuracy, the most desirable setting is with a higher privacy preservation, hence $\epsilon = 0.1$.

For the Staircase mechanism, the highest achieved accuracy for the MNIST dataset is 88%. There are two cases that achieve this accuracy. The values $\epsilon = 0.1$ and $\epsilon = 1.0$ with $C = 8$ and $S = 200$. Since the two cases of ϵ yield the same accuracy, the most desirable setting is with a higher privacy preservation, hence $\epsilon = 0.1$. For the Staircase mechanism, with the CIFAR-10 dataset, there are two cases that achieve the highest accuracy of 33%. The values $\epsilon = 0.1$ and $\epsilon = 1.0$ with $C = 8$ and $S = 200$. Since the two cases of ϵ yield the same accuracy, the most desirable setting is with a higher privacy preservation, hence $\epsilon = 0.1$.

As described above, all of the DP mechanisms yield very similar average accuracy's in all scenarios across the different datasets. Surprisingly, however, almost all models trained with DP outperform the base case model trained without DP noise. This can be attributed to the nature of the dataset or the choice of the ML algorithm, which in this research is the logistic regression model. The addition of the DP noise in this research actually produces better accuracy. The results found in this research are consistent with prior related work of Sun, Qian, and Chen [20], where their FL model achieves 96% and 14% accuracy for the MNIST and CIFAR-10 datasets respectively. Including the work of Reyes et al. [19], the results show an 85% and 40% accuracy for the MNIST and CIFAR-10 datasets respectively.

Table 10: Summary of the MNIST dataset for all three of the DP mechanism and all settings used during the experiments.

MNIST						
Parameters				**Average accuracy per mechanism**		
epsilon	Clients	Iterations	Sample size	Laplace	Gaussian	Staircase
0,1	3	10	100	0,85	0,85	0,85
0,1	3	10	200	0,86	0,86	0,86
0,1	8	10	100	0,87	0,87	0,87
0,1	8	10	200	0,88	0,88	0,88
1,0	3	10	100	0,85	0,85	0,85
1,0	3	10	200	0,86	0,86	0,86
1,0	8	10	100	0,87	0,87	0,87
1,0	8	10	200	0,88	0,88	0,88

Table 11: Summary of the CIFAR-10 dataset for all three of the DP mechanism and all settings used during the experiments.

CIFAR-10						
Parameters				**Average accuracy per mechanism**		
epsilon	Clients	Iterations	Sample size	Laplace	Gaussian	Staircase
0,1	3	10	100	0,28	0,28	0,28
0,1	3	10	200	0,31	0,30	0,30
0,1	8	10	100	0,32	0,32	0,31
0,1	8	10	200	0,33	0,33	0,33
1,0	3	10	100	0,29	0,27	0,28
1,0	3	10	200	0,30	0,30	0,30
1,0	8	10	100	0,32	0,32	0,32
1,0	8	10	200	0,34	0,33	0,33

5 Conclusion & Future Work

This thesis compares three different approaches of DP mechanisms applied to FL, namely the Laplace, Gaussian and Staircase mechanism. I conclude by highlighting a subset of new and exciting challenges that this work opens up. A study into the area of choosing an appropriate value for the privacy loss parameter in DP FL is necessary. However, to achieve optimal accuracy and privacy preservation, it is unlikely that one value for the privacy parameter would fit all FL models. Naturally, one would have to experiment with different values of the privacy loss parameter for each specific learning task. Understanding and balancing these trade-offs, both theoretically and empirically, is considerable challenge in realizing private FL systems. In conclusion, it is possible to say that DP has both negative and positive effect. Hence, a higher level of privacy, small ϵ, can bring a higher accuracy when the number of clients participating, the sample size per client and the number of iterations are big enough. When the parameters are too small it can have an impact on the accuracy of the model. With a lower level of privacy, bigger ϵ, idem ditto.

While privacy is an important aspect for many ML applications, privacy-preserving methods for FL can be challenging to rigorously assert. This due to the statistical variation in the data and maybe even more difficult to implement due to systems constraints on each device and across the potentially massive network. Developing standard evaluation metrics and establishing standard benchmark datasets for different FL settings remain highly important directions for ongoing work. The privacy impact of these compromises must be well understood when DP is deployed to protect sensitive data. The results shown in this thesis are a step towards improving that understanding and reveal that the application of DP in FL is feasible.

FL presents an opportunity to leverage uniquely diverse datasets by providing efficient decentralized training protocols along with privacy and non-identifiability guarantees for the resulting models. This means that FL enables training on multi-institutional datasets in many domains where this was previously not possible. This provides a practical opportunity to leverage larger, more diverse datasets and explore the generalizability of models which were previously limited to small populations.

Along with ML challenges related to bias, fairness, and accuracy, FL has to content with the challenges of secure multiparty communications, the privacy-accuracy trade-off, and data heterogeneity. The importance of these considerations will likely continue to grow as the real-world deployment of FL expends to more users, domains, and applications. Li et al. [10] states that the evolution of privacy attacks in FL also need to be taken into consideration when considering deploying FL. The general attack types are mainly divided into three categories. First, data poisoning attack, which is a way of tainting with the data integrity of the training results. Second, model poisoning, designing a specific input which will generate a wrong result of the ML model. Finally, inferring attack, mainly used to restore training data through a white or black box [10]. A more extensive research in the different types of privacy attacks in FL is an important direction for future work. Deploying FL should be considered when massive amount of data and participants' private information are involved. However, deploying FL does not always guarantee sufficient privacy preservation [22]. Therefore, for each application a thorough research is needed before deploying FL.

Acknowledgements

I'd like to express my gratitude to Majid Lotfian Delouee and Boris Koldehofe. Without their advice, suggestions and guidance this research would not have been possible.

References

[1] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. "Towards Federated Learning at Scale: System Design". In: *Proceedings of Machine Learning and Systems*. Ed. by A. Talwalkar, V. Smith, and M. Zaharia. Vol. 1. 2019, pp. 374–388 (cit. on p. 5).

[2] Cynthia Dwork. "Differential Privacy". In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Springer, Berlin, Heidelberg, 2006, pp. 1–12. DOI: 10.1007/11787006_1 (cit. on p. 6).

[3] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Springer, Berlin, Heidelberg, 2006, pp. 265–284. DOI: 10.1007/11681878_14 (cit. on p. 8).

[4] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Foundations and Trends in Theoretical Computer Science*. Vol. 9. 3–4. Aug. 2014, pp. 211–407. DOI: 10.1561/0400000042 (cit. on pp. 6, 8–10).

[5] Quan Geng, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. "The Staircase Mechanism in Differential Privacy". In: *IEEE Journal of Selected Topics in Signal Processing*. Vol. 9. 7. 2015, pp. 1176–1184. DOI: 10.1109/JSTSP.2015.2425831 (cit. on pp. 11, 12).

[6] Quan Geng and Pramod Viswanath. "The optimal mechanism in differential privacy". In: *2014 IEEE International Symposium on Information Theory*. 2014, pp. 2371–2375. DOI: 10.1109/ISIT.2014.6875258 (cit. on pp. 7, 11).

[7] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. "Extremal Mechanisms for Local Differential Privacy". In: *The Journal of Machine Learning Research*. Vol. 17. 1. Jan. 2016, pp. 492–542 (cit. on p. 12).

[8] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. *Federated Learning: Strategies for Improving Communication Efficiency*. Tech. rep. 2017. arXiv: 1610.05492 [cs.LG] (cit. on p. 5).

[9] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". Apr. 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (cit. on pp. 16, 17).

[10] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. "A review of applications in federated learning". In: *Computers Industrial Engineering*. Vol. 149. 2020, p. 106854. DOI: 10.1016/j.cie.2020.106854 (cit. on p. 32).

[11] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine*. Vol. 37. 3. May 2020, pp. 50–60. DOI: 10.1109/msp.2020.2975749 (cit. on p. 5).

[12] Brendan McMahan and Daniel Ramage. *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. Apr. 2017. URL: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (cit. on p. 5).

[13] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. Tech. rep. 2017. arXiv: 1602.05629 [cs.LG] (cit. on p. 4).

[14] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. *Learning Differentially Private Recurrent Language Models*. Tech. rep. 2018. arXiv: 1710.06963 [cs.LG] (cit. on p. 5).

[15] Donald Michie, D. J. Spiegelhalter, C. C. Taylor, and John Campbell, eds. *Machine Learning, Neural and Statistical Classification*. Englewood Cliffs, N.J: Prentice Hall, 1994 (cit. on p. 4).

[16] Vaikkunth Mugunthan, Anton Peraire-Bueno, and Lalana Kagal. "PrivacyFL: A Simulator for Privacy-Preserving and Secure Federated Learning". In: Association for Computing Machinery, 2020, pp. 3085–3092. DOI: 10.1145/3340531.3412771 (cit. on pp. 13–15).

[17] Joseph P. Near and Chiké Abuah. *Programming Differential Privacy*. Vol. 1. 2021. URL: https://uvm-plaid.github.io/programming-dp/ (cit. on p. 11).

[18] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. "Smooth Sensitivity and Sampling in Private Data Analysis". In: *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 2007, pp. 75–84. DOI: 10.1145/1250790.1250803 (cit. on p. 8).

[19] Jonatan Reyes, Lisa Di Jorio, Cecile Low-Kam, and Marta Kersten-Oertel. *Precision-Weighted Federated Learning*. Tech. rep. 2021. arXiv: 2107.09627 [cs.LG] (cit. on p. 30).

[20] Lichao Sun, Jianwei Qian, and Xun Chen. "LDP-FL: Practical Private Aggregation in Federated Learning with Local Differential Privacy". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. Ed. by Zhi-Hua Zhou. Aug. 2021, pp. 1571–1578. DOI: 10.24963/ijcai.2021/217 (cit. on p. 30).

[21] Christopher J.C. Burges Yann Lecun Corinna Cortes. *The MNIST database of handwritten digits with 784 features, raw data available at: http://yann.lecun.com/exdb/mnist/*. 2014. URL: https://www.openml.org/d/554 (cit. on p. 15).

[22] Xuefei Yin, Yanming Zhu, and Jiankun Hu. "A Comprehensive Survey of Privacy-Preserving Federated Learning: A Taxonomy, Review, and Future Directions". In: *ACM Computing Survey*. Vol. 54. 6. Association for Computing Machinery, July 2021. DOI: 10.1145/3460427 (cit. on p. 32).