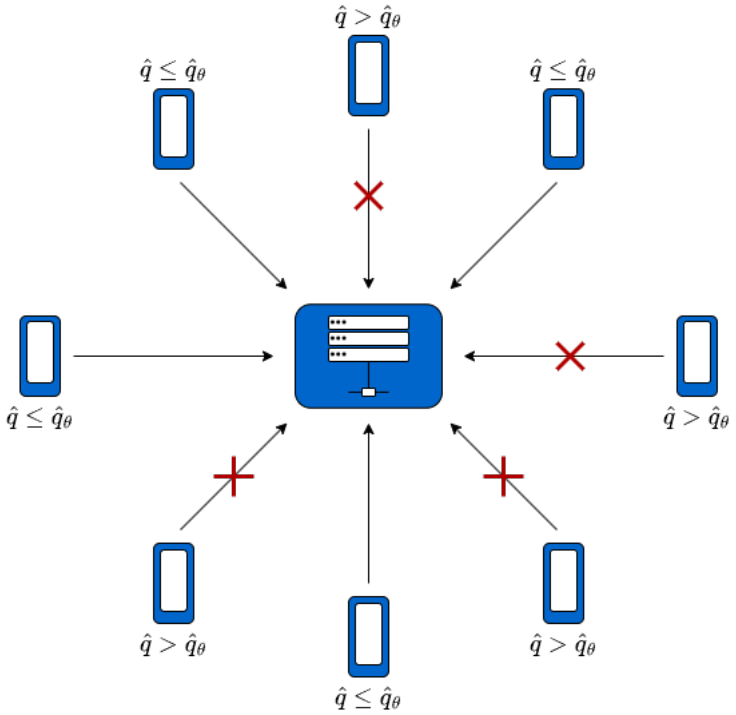
Distributed Learning for Privacy-Aware Clients

Client Selection in Federated Learning on Non-IID Data using Conformal Prediction

Master's Thesis
MATTEK10 - 4.105c



Aalborg University
Mathematical Engineering

This project is written in LaTeX and compiled in Overleaf. For simulations Python 3.7 has been used. For figures the TikZ and Pgfplot packages in LaTeX and Geogebra has been used.

Mathematical Sciences - Aalborg
University Skjernvej 4A, 9220 Aalborg Ø
http://math.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Theme:
Distributed Learning for Privacy-Aware Clients

Title:
Client Selection in Federated Learning on Non-IID Data using Conformal Prediction

Project Period:
Spring Semester 2023

Project Group:
1.405c

Participants:
Sisse Hyldig Pedersen
Frederik Christensen
Benjamin Knudsen

Supervisors:
Shashi Raj Pandey
Christophe Biscio
Aysegül Kivilcim

Copies: 1

Number of Pages: 51

Date of Completion:
June 2, 2023

Abstract:

Federated learning (FL) is a recent paradigm, which aims to preserve privacy in distributed networks, where locally collected data is stored client-side. The training of local models necessitates, that the client's data distribution support the objective of the application. Therefore, selecting the appropriate clients for aggregation is essential to a successful network. In this thesis, we propose the use of conformal prediction (CP) in a client selection method in a FL setting. We compare our proposed method of using a metric connected to CP as an evaluation criteria for when a client is finished training locally. Using this criteria, the clients determine whether or not to participate in aggregation for a specific global training round. We evaluate three different classification datasets: `MNIST`, `EMNIST` and `CIFAR10` and distributes these datasets among clients both IID and non-IID. Using different error rates and number of clients, we can conclude that our proposed method outperforms the standard method of aggregating clients in terms of prediction accuracy for a convolutional neural network for non-IID data.

Preface

This thesis is written by a group of master students studying Mathematical Engineering at the Department of Mathematical Sciences at Aalborg university. The group would like to thank the supervisors Christophe Biscio, Shashi Raj Pandey and Aysegül Kivilcim for their supervision throughout the semester.

The references throughout this thesis have been handled with the numerical IEEE-method with specified page numbers, sections or chapters of the respective source. Further information about the sources can be found in the bibliography in the end of this thesis. Abbreviations used throughout this thesis is listed in the `Abbreviation List`.

The figures and tables have been made by the authors of the thesis unless otherwise stated. These figures and tables were created using the `matplotlib.pyplot` package in Python and using the Ti*k*Z package in LaTeX. The scripts that have been used to obtain the results in this thesis are written i Python 3.8 and can be shared by the authors upon request.

Aalborg Universitet, June 2, 2023.

Benjamin Bitsch Knudsen	Frederik Christensen	Sisse Hyldig Pedersen
<bknuds18@student.aau.dk>	<fchr18@student.aau.dk>	<shpe16@student.aau.dk>

Resumé på Dansk

Med de teknologiske udviklinger, har maskinlæring og kunstig intelligens haft en banebrydende udvikling med endeløse applikationer. Dertil kræves der en stor indsamling af data til at træne disse maskinlæringsmodeller. Et problem opstår ved indsamlingen af data, da det kommer fra forskellige personer/institutter, hvor det måske ikke er ønsket eller muligt at indsamle dataet til en central lokation pga. sikkerhedsmæssige årsager. Dertil har fødereret læring vist sig at være et muligt værktøj til at træne maskinlæringsmodeller uden indsamling af data.

Fødereret læring benytter sig af at indsamle og distribuere maskinlæringsmodellen's parametre mellem en central lokation, kaldet en aggregator, og et udvalg af enheder, kaldet klienter. Hver klient træner lokalt på deres egen maskinlæringsmodel og sender derefter deres parametre til aggregatoren. Efter indsamlingen af klienternes parametre opdaterer aggregatoren parametrene ved brug af `FedAvg` metoden. Derefter udsendes de nye parametre tilbage til klienterne og processen fortsætter på ny. Det betyder altså at fødereret læring ikke deler klienternes private data, men i stedet deres modelparametre. Der findes forskellige metoder til at vælge hvilke klienter der udtages til aggregation. Standardmetoden er, at tilfældigt vælge en delmængde af alle klienterne som udtages til aggregation. I denne specialeafhandling, undersøger vi en udtagningsproces af klienter ved brug af konform prædiktion ved at tilknytte en metrik baseret på klienternes nøjagtighed til at prædiktere rigtigt. Vi ser på indflydelsen af både ligeligt fordelt data mellem klienter, samt ikke-ligeligt fordelt data mellem klienter.

Konform prædiktion er et statistisk værktøj der benyttes efter træning af maskinlæringsmodeller til at lave prædiktionssæt, hvortil man med en bestemt statistisk konfidens kan sige, at den korrekte prædiktion ligger i sættet.

Til at vurdere om vores foreslåede udtagningsproces er bedre end en standard udtagningsproces har vi evalueret præcisionen af maskinlæringsmodellen for begge metoder. Dertil har vi evalueret forskellige parametre tilknyttet konform prædiktion, samt evalueret indflydelse af antallet af klienter. Ud fra disse forsøg kan der konkluderes at vores foreslåede udtagningsproces med ikke-ligeligt fordelt data udkonkurrerer standardmetoden. Derudover kan der konkluderes at passende parametre fra konform prædiktion har stor indflydelse på præcisionen af vores foreslåede metode.

Abbreviation List

AI	Artificial Intelligence.
CE	Cross-entropy.
CNN	Convolutional Neural Network.
CP	Conformal Prediction.
ERM	Empirical Risk Minimisation.
FedSGD	Federated Stochastic Gradient Descent.
FedAVG	Federated Averaging.
FL	Federated Learning.
GTR	Global Training Round.
IID	Independent and Identically Distributed.
IoT	Internet of Things.
LTR	Local Training Round.
ML	Machine Learning.
PoC	Power-Of-Choice.
SGD	Stochastic Gradient Descent.

Contents

1. Problem Analysis

1.1 Background and Motivation

Modern mobile phones, smartwatches and autonomous vehicles are just a few of the many modern inventions which gather a wealth of data each day [36, p. 1]. This constant availability of massively distributed data has pivoted the success of machine learning (ML), which further increased the usability of intelligent application, e.g. speech recognition, next-word prediction, computer vision etc. [34, p. 2]. In order to obtain insight in the big amount of data, which is generated in the ubiquitous internet of things (IoT) devices, artificial intelligence (AI) techniques such as deep learning methods can be utilised [6]. However, a prerequisite of traditional AI techniques is, that the data is collected and processed at a centralised location. This may not be feasible in practice due to the high scalability of IoT networks, while also neglecting the user's privacy as a result of centralised data storage [6].

For this purpose, federated learning (FL) is a recent ML paradigm, which aims to preserve privacy in a decentralised ML setting. FL has been used in various applications such as in healthcare, automation, logistics, etc [9]. As the FL paradigm serves to perform local training on devices (clients), it necessitates that, the distributed training supports the objective of the application [9]. It is therefore essential that the clients, which are beneficial in supporting the application, are weighed higher than ones that are less beneficial. Throughout this thesis, the focus will be to evaluate the performance of a proposed client selection strategy for a privacy-aware FL setting. The following problem analysis provides the context for this thesis.

1.2 Supervised Learning

Consider sampling a stream of labeled data $\{X_i, Y_i\}_{i=1}^{n}$, where X_i are the covariates (e.g. images), Y_i are the corresponding labels and n is the amount of data samples collected. Based on this previously observed data and the covariate X_{n+1}, it is desired to determine its label Y_{n+1}. A method of predicting the label Y_{n+1} is the use of supervised ML algorithms, e.g. neural networks, decision trees, logistic regression etc. Although these models are complex and are subject to prolonged training time, a substantial amount of empirical evidence shows, that these models can give highly accurate predictions using IID data samples for both training and testing [18, p. 1]. A method to decrease the computational time of a supervised learning algorithm is

to distribute the model to multiple machines, which is known as distributed learning.

1.3 Distributed Learning and Federated Learning

Distributed ML is becoming increasingly relevant in modern technology. This is both due to the increasing complexity of ML models and the increasing amount of training data [20, p. 4].

Increasing the complexity of the ML models in return increases their representational capacity. Considering a neural network, this can be done by either making the network 'wider', meaning adding more parameters, or 'deeper' by adding layers. These increasingly complex ML models yield significantly more expensive computational costs, for which distributed learning could be a solution [20, p. 4].

Some challenges occur when considering the amount of training data, which negatively impacts the performance of the ML model, some of which include overfitting and underfitting. Underfitting and overfitting means that the ML model is unable to accurately capture the relations between the input and output. Underfitting is a result of a limited amount of training data and overfitting occurs due to an overflow of training rounds, and both result in the model being incapable of generalising to new data [20, p. 4]. With the increase in data volume these issues are addressed, however the computational cost of processing the larger data volumes becomes a relevant issue. For this distributed ML could be a possible solution by distributing the data among multiple machines and training the same model architecture on each machine [20, p. 4].

The distributed learning model can be seen in Figure 1.1, which depicts the decentralised setting. In this context, decentralised refers to the ML model being distributed with data to multiple devices (referred to as clients), as opposed to a centralised setting, where nothing is distributed.

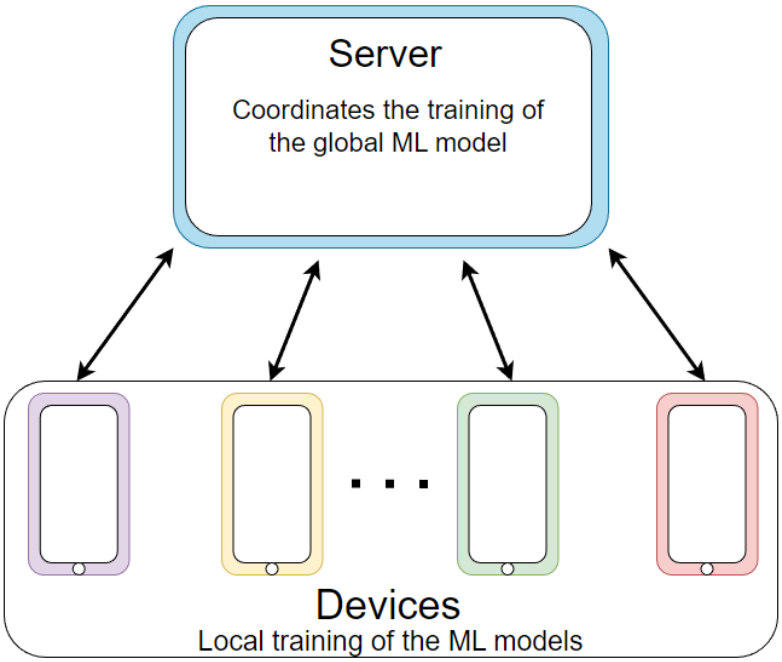


Figure 1.1: Distributed Learning.

In the decentralised setting, the training data is shared amongst the devices for them to locally train on. This can however pose a major privacy concern and security risk.

Consider a smart phone, which has access to an unprecedented amount of data with its powerful sensors, e.g. GPS, microphones, cameras etc. The data collected from these sensors are valuable in the sense that ML models trained on this data can greatly improve the usability of intelligent applications. Examples of this are speech recognition and text entry, which can be improved using language models [7, p. 1]. However, this data may be of sensitive nature, and the user may not always be willing to share their data for storage in a centralised location. A solution for this is FL, which aims to locally train the ML model of each client using local data. In this setting, the clients only receive and transmit model parameters, thereby preserving their personal data [36, p. 2].

FL relies on stochastic gradient descent (`SGD`), for which the IID property of training data is important for yielding an unbiased estimate of the gradient [39, p. 2]. In practice, the data distribution cannot be assumed to be IID both across clients and in time, which disrupts the convergence rate of optimisation in the FL algorithms. The data distribution cannot be assumed to be IID, since both the amount and distribution of data of a client varies significantly. Subsequently, each client would then be biased to optimise its own ML model, instead of cooperating towards optimising the global ML model. These issues are known as concept drift, client drift or model drift caused by statistical heterogeneity [21, p. 1][34, p. 1]. Principles and

challenges of FL is further explained in Chapter 2.

In this thesis a possible solution to model drift is proposed, where the influence of each client is regulated based on a metric regardless of their data distribution. The goal of implementing a metric to regulate client influence is to counteract the model drift as a result of a possible skewed data distribution across clients. In this context, the metric ultimately decides, which clients should participate in the FL training process.

For this purpose, conformal prediction (CP), or conformal inference, is an attractive paradigm, in which a metric is calculated based on the training data and the ML model. The framework of CP and the validity of a metric for client regulation is examined in Chapters 3 and 6, respectively.

The metric is used as an indicator, as to which clients should transmit its model parameters. This client sampling strategy is designed to sort out the specific clients, for which the metric is high enough. Our proposed client sampling strategy and alternatives are further explained in Chapter 4.

1.4 State of the Art

There exists an abundance of previous research on the convergence rate of FL, different client selection strategies for FL and research of non-IID data distributions of clients. In [15], a Power-Of-Choice (PoC) algorithm for client selection is considered. A theoretical proof showcases, that selecting clients with the highest local loss for training instead of unbiased selection, increases the rate of convergence. It was shown that the PoC algorithm lead to three times faster convergence rate with higher test performance than randomly picking clients for training.

In [14], a convergence guarantee for strongly convex and smooth problems for FL using the Federated Averaging (`FedAvg`) algorithm is shown without making assumptions that the data is IID or that all clients participate in training. It is shown that the sampling strategy, number of local training rounds (LTRs) and that the heterogeneity of training data is crucial for the convergence rate of `FedAvg`.

In [25], CP is used in a FL setting. A convolutional neural network (CNN) is trained on different medical image datasets (`BloodMnist`, `DermaMNIST`, `PathMNIST`, `TissueMNIST`, `RetinaMNIST`, `OrganMNIST3D`) in which each client calculates a quantile. They proposed an algorithm in which the quantile is averaged to improve the coverage. The coverage convergence rate of the proposed algorithm is compared to the coverage convergence rate of a local quantile. Both methods achieve similar empirical coverage, however the proposed algorithm achieved smaller CP sets.

To the best of our knowledge using CP scores for client participation has not been done.

1.5 Problem Statement

In this thesis, we examine the performance of a proposed client sampling strategy utilising different scoring functions from the field of CP in a FL setting. Furthermore our proposed client sampling method is compared to the standard method of sampling clients in FL. This leads to the following problem statement:

> How does a client sampling strategy utilising CP perform compared to other sampling strategies in a FL setting?

1.6 Thesis Scope and Limitations

In this thesis we limit ourselves to the following:

1. The aggregator's test set is drawn from the global data distribution.
2. Each client has a distinct dataset with non overlapping data between clients.
3. Only image classification is examined, and tested on the `MNIST`, `EMNIST` and `CIFAR10` datasets.
4. A CNN is used as the ML model for image classification.

2. Federated Learning

In this chapter, the concept of FL is explained. The purpose of the chapter is to tackle the privacy issue, which is the result of data sharing in a distributed environment. In Section 2.1 the general ML optimisation problem is explained. In Section 2.2 the general ML optimisation problem is expanded into a decentralised ML optimisation problem and some FL methods are presented as ways to solve this optimisation problem. Multiple FL algorithms are compared using a logistic regression example in Section 2.3.
Finally, in Section 2.4 the remarks on FL is discussed including the model drift which derives from a decentralised setting.

FL is a decentralised ML technique where multiple clients collaborate to train a global model without exchanging the client's datasets [20, p. 167-168]. This is done using a central server, known as the aggregator, which initialises the multiple clients with the same initial parameters. Each client has the same model architecture and their own private datasets. After initialisation, each client trains their model and afterwards sends their updated model parameters to the aggregator. The aggregator then updates the global model, and distributes the updated parameters to the clients once again. The training of clients and aggregation process is then repeated until completion, which is determined either after a specific number of training and aggregation rounds, or a sufficiently low loss (given some loss function) for the global model. When clients are training locally, it will be called the LTR, and the number of aggregation rounds will be called the global training rounds (GTR).

2.1 Supervised ML Optimisation

A supervised ML problem can be posed as an empirical risk minimisation (ERM) problem, which has the form:

$$\min_{\boldsymbol{\omega}\ \mathsf{R}^d} f(\boldsymbol{\omega}) \qquad \text{where} \qquad f(\boldsymbol{\omega})\ ,\ \frac{1}{n}\sum_{i=1}^{n} \mathsf{L}(\boldsymbol{x}_i, y_i, \boldsymbol{\omega}), \tag{2.1}$$

where $\boldsymbol{\omega}$ is the parameter of the model, which is optimised over, $\boldsymbol{x}_i \quad \mathsf{R}^d$ is the $i\,th$ training sample and y_i being its corresponding label, n being the number of the training samples and L is the loss function. Furthermore L is considered to be a convex differentiable function for all inputs, which in return means that (2.1) is a convex smooth optimisation problem [4, p. 79]. To solve this, several optimisation

algorithms exist, however a widely used algorithm is the stochastic gradient descent (SGD) algorithm.

2.1.1 Stochastic Gradient Descent

SGD is an optimisation algorithm that converges towards the optimal set of parameters of a model, which minimises a given loss function. It is a variant of the standard gradient descent algorithm, which updates the model parameters in the direction of the negative gradient of the loss function with respect to the entire training data [3, p. 240]. A parameter update for the standard gradient descent is

$$\boldsymbol{\omega}^{(\tau+1)} = \boldsymbol{\omega}^{(\tau)} - \eta \quad f(\boldsymbol{\omega}^{(\tau)}), \tag{2.2}$$

where the parameter $\eta > 0$ is called the learning rate, τ denotes the iteration index, and f is the loss function [3, p. 240].
However, as the standard gradient descent computes the gradients of the entire training set, its computational time scales with the size of the training set. To reduce computational time, the SGD updates the model parameters for each training sample in a random order, using only a small random subset of the training data at a time. Furthermore, this allows for faster convergence and improves the models ability to adapt to new data [19, p. 275]. A parameter update for the SGD algorithm is

$$\boldsymbol{\omega}^{(\tau+1)} = \boldsymbol{\omega}^{(\tau)} - \eta \quad L_i(\boldsymbol{\omega}^{(\tau)}), \tag{2.3}$$

where L_i is the loss function for the $i\,th$ training sample. The algorithm iteratively adjusts the parameters in the direction of the negative gradient of the loss function for each random subset of the training data. The SGD has especially proven useful in practice when training neural networks on large datasets [3, p. 240].

2.2 Supervised FL Optimisation

The standard supervised ML optimisation problem in (2.1) can be realised in a FL setting as follows. Let the global dataset be the collection of data from all clients, and let P_k be the set of indices of data samples of client k from the global dataset. Furthermore, let the number of data samples of client k be expressed as $n_k = |P_k|$. Then given that each client optimises their own model, the FL optimisation problem for client k is given as the ERM problem

$$F_k(\boldsymbol{\omega}) = \frac{1}{n_k} \sum_{i \ P_k} L\,(\boldsymbol{x}_i, y_i, \boldsymbol{\omega}). \tag{2.4}$$

Using (2.4) to rewrite (2.1), the global FL optimisation problem, i.e. the optimisation problem in the aggregator, can be expressed as an ERM problem

$$f(\boldsymbol{\omega}) = \frac{1}{n}\sum_{i=1}^{n} L_i(\boldsymbol{x}_i, y_i, \boldsymbol{\omega}) \tag{2.5}$$

$$= \frac{1}{n}\sum_{k=1}^{K}\sum_{i \in P_k} L_i(\boldsymbol{x}_i, y_i, \boldsymbol{\omega}) \tag{2.6}$$

$$= \frac{1}{n}\sum_{k=1}^{K} n_k F_k(\boldsymbol{\omega}). \tag{2.7}$$

Thereby the FL optimisation problem is

$$\min_{\boldsymbol{\omega} \in \mathbb{R}^d} \quad f(\boldsymbol{\omega}) = \frac{1}{n}\sum_{k=1}^{K} n_k F_K(\boldsymbol{\omega}). \tag{2.8}$$

Similar to the standard supervised ML problem, the FL optimisation problem can be solved by its own version of `SGD`, called federated `SGD`.

2.2.1 Federated Stochastic Gradient Descent

Federated `SGD`, denoted as `FedSGD`, selects a fraction $C \in [0, 1]$, of the K clients to perform the gradient computations on each local ML model, which repeats for each GTR.
Denoting the gradient of the k'th client for the t'th GTR as

$$\mathbf{g}_k = \nabla F_k(\boldsymbol{\omega}^{(t)}), \tag{2.9}$$

the `FedSGD` algorithm for $C = 1$ can be computed [7, p. 4]. Firstly `FedSGD` computes the gradient $\mathbf{g}_k$ for all K clients, then the aggregator aggregates all K gradients and applies an update to the parameters as

$$\boldsymbol{\omega}^{(t+1)} \leftarrow \boldsymbol{\omega}^{(t)} - \eta \nabla f(\boldsymbol{\omega}^{(t)}) = \boldsymbol{\omega}^{(t)} - \eta \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{g}_k. \tag{2.10}$$

This is then repeated for each GTR [7, p. 4].

An equivalent method would be updating the parameters on each client and aggregating them instead of aggregating the gradients. Updating the parameters for each client can be expressed as

$$\boldsymbol{\omega}_k^{(t+1)} \leftarrow \boldsymbol{\omega}^{(t)} - \eta \mathbf{g}_k \qquad \text{for} \qquad k = 1, \ldots, K. \tag{2.11}$$

The weights of the aggregator would then be updated as

$$\boldsymbol{\omega}^{(t+1)} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \boldsymbol{\omega}_k^{(t+1)}. \tag{2.12}$$

This repeats for each GTR [7, p. 4].

2.2.2 Federated Averaging

Expanding the `FedSGD` algorithm such that each client computes multiple local updates on their parameters, i.e. such that

$$\boldsymbol{\omega}_k \leftarrow \boldsymbol{\omega}_k - \eta \mathbf{g}_k \tag{2.13}$$

for multiple LTRs before being aggregated yields the `FedAvg` algorithm. The number of LTRs each client performs before getting aggregated is usually denoted by E. Furthermore B is the local batch size, i.e. the number of training samples at each LTR before the local parameters are updated. $B = \infty$ indicates that the entire local dataset is treated as one batch [7, p. 4].

The entire FL algorithm for `FedAvg` is shown in Algorithm 1.

Algorithm 1: Federated Averaging Algorithm

INPUT:

Number of LTRs E,
Learning rate η,
Local batch size B,
The fraction C of the K clients,
Initial parameters $\boldsymbol{\omega}^{(0)}$.

OUTPUT:

The parameters for the global model $\boldsymbol{\omega}$.

Aggregator:

0 : Distribute $\boldsymbol{\omega}^{(0)}$ to Clients
1 : **for** each round $t = 1, 2 \ldots$ **do**
2 : $m \leftarrow \max\ (C \cdot K, 1)$
3 : $S^{(t)} \leftarrow$ (random set of m clients).
4 : **for** each client $k \quad S^{(t)}$ in parallel **do**
5 : $\boldsymbol{\omega}_k^{(t+1)} \leftarrow$ ClientUpdate$(k, \boldsymbol{\omega}^{(t)})$

6 : **end for**
7 : $m^{(t)} \leftarrow \sum_{k \ S^{(t)}} n_k$
8 : $\boldsymbol{\omega}^{(t+1)} \leftarrow \sum_{k \ S^{(t)}} \frac{n_k}{m_t} \boldsymbol{\omega}_k^{(t+1)}$
9 : **end for**

ClientUpdate$(k, \boldsymbol{\omega})$: //Run on client k
5a : $B \leftarrow$ (randomly spilt P_k into batches of size B)
5b : **for** each local epoch $i = 1, 2 \ldots E$ **do**
5c : **for** each batch **b** B **do**
5d : $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta \ L(\boldsymbol{\omega}; \mathbf{b})$
5e : **end for**
5f : **end for**
5g : return $\boldsymbol{\omega}$

[7, $p.$ 5]

2.3 Example: Comparison between SGD, FedSGD and FedAvg for Logistic Regression

To compare SGD, FedSGD and FedAvg an example using logistic regression is presented. The data used for the example is the breast cancer dataset from sklearn[5]. In Figure 2.1 the test accuracy is shown for all three methods.

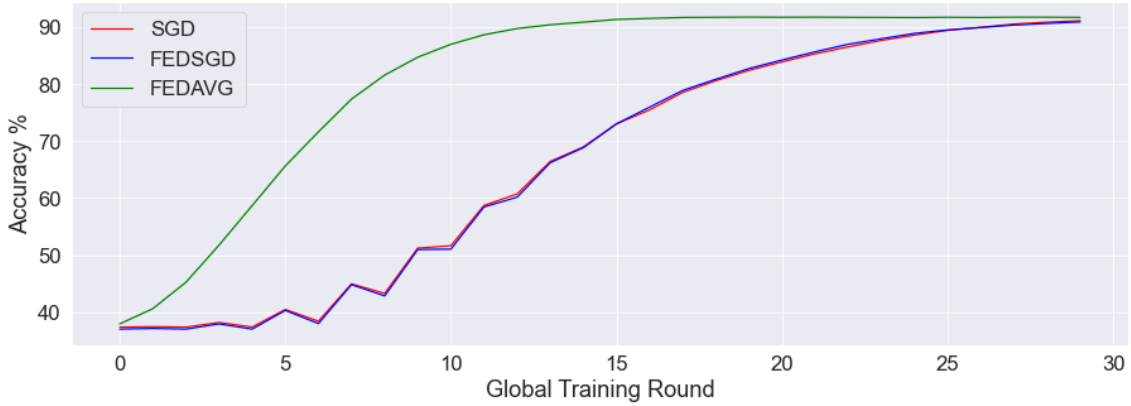


Figure 2.1: Comparison of SGD, FedSGD and FedAvg. FedAvg utilises two LTRs.

In Figure 2.1, the data of the two FL algorithms is IID among 10 clients, where FedAvg uses two LTRs before aggregation. Furthermore, the results are mean values from conducting the experiment 100 times.

The FedAvg algorithm in Figure 2.1 is seen to achieve a higher accuracy quicker than FedSGD and SGD, which is due to the increased number of LTRs before aggregation. Although the FedAvg has double the computational time than FedSGD, the accuracy is significantly higher per GTR. This means that the FedAvg is more beneficial in a communication cost context, compared to the two other methods. The FedSGD and SGD are identical as expected, due to algorithms utilising the same datasets and the data distribution being IID across clients.

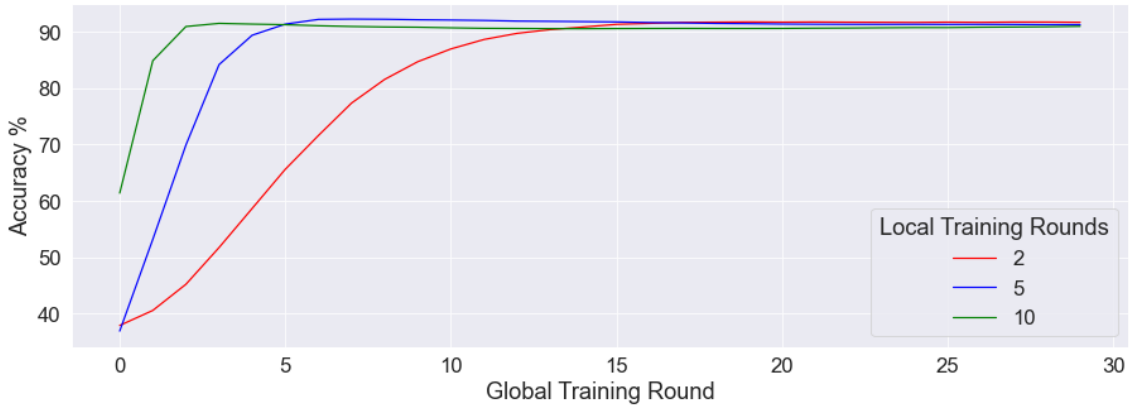


Figure 2.2: Performance of FedAvg utilising various amounts of LTRs.

A comparison between the number of LTRs before aggregation in the FedAvg algorithm is shown in Figure 2.2. An increase in the amount of LTRs is observed to increase the accuracy per GTR quicker.

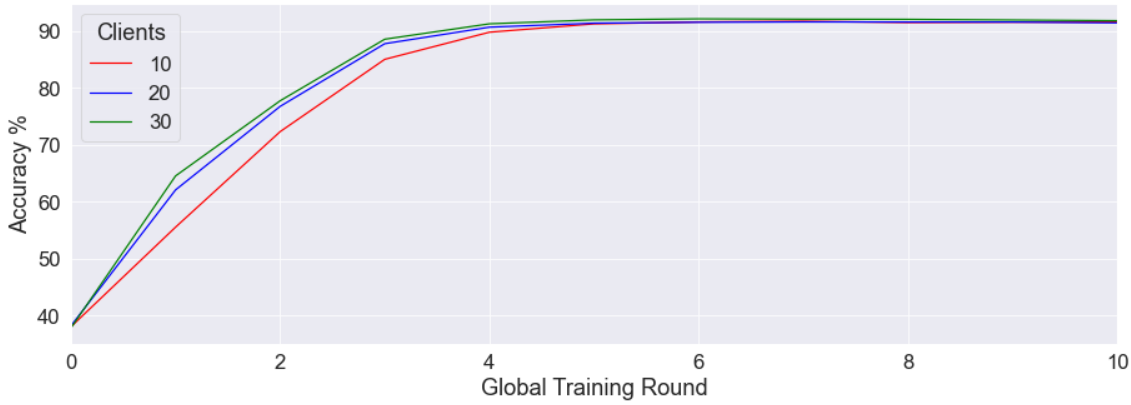


Figure 2.3: Performance of FedAvg utilising various amounts of clients.

In Figure 2.3, the FedAvg algorithm is plotted using various amounts of clients and 5 LTRs. In this case, the impact of increasing clients improves the accuracy of the aggregator, with decreasing variance at further aggregations. However, this result may vary significantly depending on the ML model and data distribution.

2.4 Remark on Federated Learning

A remark on the importance of the data distribution in FL has to be addressed. Since each client has its own locally sampled data, it cannot be assumed that the clients data distributions reflects the global data distribution. This means that each client will be biased against their own data, and does not necessarily generalise well to solve the global optimisation problem. Therefore sampling strategies have to be considered, when picking which clients should be chosen in each GTR. So far, in this project, only picking random clients have been considered, however some metric could be used as a sampling strategy. We propose a method for sampling clients based on CP, which is further explained in Chapter 4.

3. Conformal Prediction

In this chapter, the framework of CP is presented. The purpose of the chapter is to introduce a metric, which is utilised for the proposed client sampling strategy in a FL setting. Most traditional prediction methods (such as supervised ML methods), lack the ability to estimate the uncertainty of the prediction. A way to provide a statistical confidence on the output for any ML model that outputs a prediction is CP [25, p. 2].

CP is a method for producing sets of predictive outputs without assumptions on the data distribution and the predictive ML model. Examples of predictive ML models include neural networks, decision trees and quantile regression, all of which output class or point predictions [25, p. 2]. Based on the ML model, the prediction sets are constructed to include the correct prediction with a certain probability $1 - \alpha$, where $\alpha \quad [0, 1]$ is a user-chosen error-rate parameter [1, p. 4]. For a regression task, the prediction set would consist of an interval and for a classification task, the prediction set would consist of one or multiple class labels.

The prediction set is constructed based on a conformal score function, which determines the relation between the model input and output. The score function is desired to reflect a large number, when the ML model predicts correctly, and a small number the model is uncertain or incorrect. The selection of a good score function is pivotal for creating useful prediction sets[1, p. 6]. A good score function is a relative term, and depends on the task of the ML model. Examples of score functions are showcased in Section 3.2.

Based on the scores, a quantile is calculated and used for creating the prediction sets. Depending on the type of CP, the method of calculating the quantile varies.

There are multiple adaptations of CP, which have different goals and complications. The most widely-used type of CP is split CP, and is much less computationally heavy than the version it originated from, known as full CP [1, p. 6].

3.1 Split Conformal Prediction

The split CP method splits the training data into a training set and a calibration set, where the latter is utilised to calibrate the predictive ML model. For the sake of clarity, an example of split CP will be introduced.

Consider a convolutional neural network (CNN), which takes image inputs and outputs an estimated probability for each class label. For simplicity's sake, let

the input to the CNN be the MNIST dataset, which consists of images containing handwritten digits ranging from zero to nine. The CNN would then output values for each class, which can be converted to estimated probabilities using the softmax function [35, p. 64]

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad \text{for} \quad i = 1, 2, \ldots, K, \tag{3.1}$$

where $\mathbf{z}^K = [z_1, z_2, \ldots, z_K]$ is the CNN's output layer vector and K is the number of classes. The softmax output of the CNN is denoted $\hat{f}(X_i)$, where examples of softmax outputs are '47 % - Class 3' or '83 % - Class 0'. Some of the images in the training set are reserved for calibration, i.e. calibration data is unseen during training. In this case, the calibration data is a sequence of image and class pairs $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, where X_i are the images, Y_i are class labels of the respective images and n being the length of the calibration set.

In the calibration step the conformal scores are calculated with a possible conformal score function, which could be

$$s_i = 1 - \hat{f}(X_i)_{Y_i}, \tag{3.2}$$

where $\hat{f}(X_i)_{Y_i}$ is the softmax score (estimated probability) for the true class. In this case, a high conformal score would reflect a low estimated probability, which means the CNN's prediction is incorrect or unlikely, and vice versa. On the left in Figure 3.1, the CNN's output for a handwritten '1' is illustrated.

The next step of CP is to define $\hat{q}$ as the $\frac{(n+1)(1-\alpha)}{n+1}$ empirical quantile of the scores $s_1, \ldots, s_n$, where $\lceil \cdot \rceil$ is the ceil function [1, p. 4]. Using the score function (3.2), a low $\hat{q}$ is the result of low scores, which can be interpreted as the ML model is well adjusted to the characteristics of the calibration data. Subsequently, a high $\hat{q}$ is the result of high scores, which can be interpreted as the ML model incorrectly identifying the classes of the calibration data. In the middle of Figure 3.1, $\hat{q}$ is determined with $n = 500$ and $\alpha = 0.1$.

Finally, the prediction set $C(X_{\text{test}})$ of a new data input X_{test} is calculated as

$$C(X_{\text{test}}) = \{y : \hat{f}(X_{\text{test}}) \geq 1 - \hat{q}\}. \tag{3.3}$$

This prediction set includes every class label, for which the estimated probability is large enough, i.e. larger than $1 - \hat{q}$. On the right in Figure 3.1, a test sample yields the prediction set $\{3, 8, 9, 6\}$ sorted by softmax score.

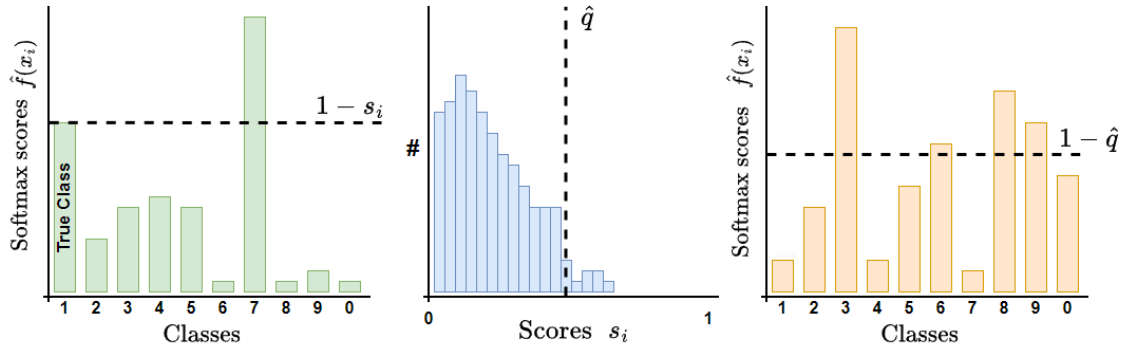


Figure 3.1: The steps of CP using MNIST data: (1) Compute conformal scores on calibration data, (2) Sort scores and determine $\hat{q}$ and (3) Determine whether to include classes in the prediction sets from test samples.

A useful statistical property of constructing prediction sets using this method is the coverage guarantee that it provides. If the size of the prediction sets are considered an important part of the sampling strategy, the coverage will affect the quantile $\hat{q}$.

3.1.1 Coverage

The property of coverage is, that a prediction set contains the true class label or point prediction with high probability. The exchangeability of a sequence of random variables is a prerequisite for formally defining the coverage guarantee for split CP.

Definition 3.1 (Exchangeable Sequence of Random Variables)
Let $X_1, X_2, \ldots$ be an exchangeable sequence of random variables (which may be finitely or infinitely long), and let π be a finite permutation. Then the joint probability distribution of $X_1, X_2, \ldots$ and the joint probability distribution of the permuted sequence $X_{\pi(1)}, X_{\pi(2)}, \ldots$ are equivalent [2, p. 473].

Intuitively Definition 3.1 states that finitely exchanging positions of random variables in the sequences does not change the joint distribution of the sequence. Exchangeability of the dataset is a necessary condition for the coverage property, while IID is a sufficient condition. In other terms, IID infers exchangeability, since the joint pdf of an IID sequence is the product of the marginals. The cumulative property implies that the joint pdf of the permutation of such sequence is equivalent to the original sequence. Now the conformal coverage guarantee can be stated.

Theorem 3.1 (Conformal Coverage Guarantee)
Suppose $\{X_i, Y_i\}_{i=1}^n$ is the calibration set, $(X_{\text{test}}, Y_{\text{test}})$ is a new test point and they are IID (or exchangeable). Let the scoring function be $s_i = 1 - \hat{f}(X_i)_{Y_i}$. Then define the quantile of the calibration scores as

$$\hat{q} = \inf\left\{q : \frac{|\{i : s(X_i, Y_i) \leq q\}|}{n} \geq \frac{(n+1)(1-\alpha)}{n+1}, \quad q \quad [0,1]\right\} \tag{3.4}$$

and the resulting prediction sets as

$$C(X_{\text{test}}) = \{y : s(X_{\text{test}}, y) \leq \hat{q}\} = \{y : \hat{f}(X_{\text{test}})_y \geq 1 - \hat{q}\}. \tag{3.5}$$

Then the following holds:

$$P(Y_{\text{test}} \quad C(X_{\text{test}})) \geq 1 - \alpha. \tag{3.6}$$

[1, p. 50]

The quantile $\hat{q}$ is chosen as the infimum of the set of possible quantiles q, which satisfy the condition in (3.4). If $\frac{(n+1)(1-\alpha)}{n+1} = 0.95$, then the set of quantiles would consist of every q with a value greater than 95 % or more of the calibration scores. A prerequisite for the coverage property is that the score function is required to be monotonic [25, p. 2].

Proof of Theorem 3.1.
Let $s_i = s(X_i, Y_i)$ for $i = 1, \ldots, n$ and $s_{test} = s(X_{test}, Y_{test})$. Consider the case where the $s_i = s_j$ for $i = j$. The calibration scores are assumed to be sorted such that $s_1 < \cdots < s_n$. In this case if $\alpha \geq \frac{1}{n+1}$ then $\hat{q} = s_{\ (n+1)(1-\alpha)}$ and when $\alpha < \frac{1}{n+1}$ then $\hat{q} = \infty$. However, in the case where $\hat{q} = \infty$, then $C(X_{test}) = Y$, meaning that the coverage property is satisfied (Y denotes the entire label set). Therefore only the case where $\hat{q} = s_{\ (n+1)(1-\alpha)}$ has to be proven. Note that the following two events are equal

$$\{Y_{test} \quad C(X_{test})\} = \{s_{test} \leq \hat{q}\}. \tag{3.7}$$

Inserting the definition of $\hat{q}$

$$\{Y_{test} \quad C(X_{test})\} = \{s_{test} \leq s_{\ (n+1)(1-\alpha)}\}. \tag{3.8}$$

> Since the variables $(X_1, Y_1), \ldots, (X_{test}, Y_{test})$ are exchangeable,
>
> $$P(s_{test} \leq s_k) = \frac{k}{n+1} \qquad k \quad \mathsf{Z}. \tag{3.9}$$
>
> This means that s_{test} is equally likely to be anywhere between the calibration points $s_1, \ldots, s_n$, thereby yielding the desired result
>
> $$P(s_{test} \leq s_{\ (n+1)(1-\alpha)}\ ) = \frac{(n+1)(1-\alpha)}{(n+1)} \geq 1 - \alpha. \tag{3.10}$$
>
> [1, pp. 50-51]

From [1, p. 51], if the scores $s_1, \ldots, s_n$ have a continuous joint distribution, (3.6) can be proved to be upper bounded as

$$P(Y_{test} \quad \mathsf{C}(X_{test})) \leq 1 - \alpha + \frac{1}{n+1}. \tag{3.11}$$

However, in practice the scores does not need to have a continuous joint distribution, since a vanishing amount of white noise can be added to the score to make a discrete joint distribution continuous [1, p. 51]. The coverage bounds from (3.6) and (3.11) serve as performance guarantees of CP

$$1 - \alpha \leq P(Y_{test} \quad \mathsf{C}(X_{test})) \leq 1 - \alpha + \frac{1}{n+1}. \tag{3.12}$$

Choosing $\alpha = 0$ would guarantee $Y_{test} \quad \mathsf{C}(X_{test})$, but would result in $\mathsf{C}(X_{test}) = \mathsf{Y}$, which is already known beforehand. Finding the optimal α is a task for experimentation, and will be examined in Chapter 6.

3.2 Examples of CP score functions

As previously mentioned, the choice of score function is essential to the performance of CP, since it affects the $\hat{q}$-values. This section will showcase multiple score functions, which are utilised in experimentation. The purpose of introducing multiple score functions is to potentially increase the performance of the FL network.

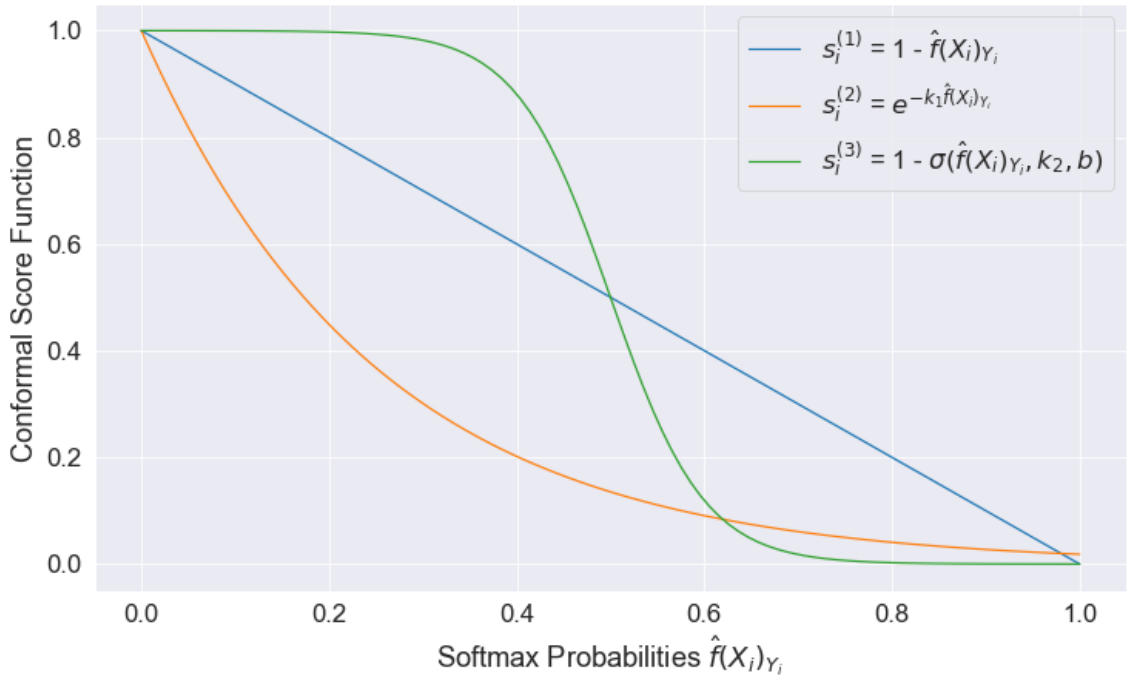


Figure 3.2: Three score functions are plotted with $k_1 = 4$, $k_2 = 20$ and $b = 10$.

The score functions evaluated are

$$s_i^{(1)} = 1 - \hat{f}(X_i)_{Y_i}, \tag{3.13}$$

$$s_i^{(2)} = e^{-k_1 \hat{f}(X_i)_{Y_i}}, \tag{3.14}$$

$$s_i^{(3)} = 1 - \sigma(\hat{f}(X_i)_{Y_i}, k_2, b), \tag{3.15}$$

where σ is a modified sigmoid function defined as

$$\sigma(x) = \frac{1}{1 + e^{-k_2 x + b}}. \tag{3.16}$$

In (3.14), $k_1 > 0$ determines the slope of the function. In (3.15), $k_2 > 0$ determines the slope (a high k results in the function converging towards a step-function) and b determines where the center of the slope is [22, p. 71]. These three score functions are illustrated in Figure 3.2.

The score function $s^{(1)}$ from [1, p. 4] is a linear function of the softmax probability $\hat{f}(X_i)_{Y_i}$ of the true class for image X_i, hence the conformal scores of the calibration data scale linearly. Alternatives to this score function are $s^{(2)}$ and $s^{(3)}$, which are proposed to include biases towards specific softmax probabilities. Using $k_1 = 4$, the score function $s^{(2)}$ is designed to generally assign softmax probabilities to lower conformal scores in comparison to $s^{(1)}$. Subsequently, the quantile $\hat{q}$ is also lower, which in turn impacts the client selection process.

Using $k_2 = 20$ and $b = 10$, the score function $s^{(3)}$ is designed to penalise lower softmax probabilities with higher conformal scores and reward higher softmax probabilities with lower conformal scores. In comparison to $s^{(1)}$, the score function $s^{(3)}$ significantly distinguishes between low and high scores, which 'rewards' good models for accurate predictions and 'punishes' bad models for inaccurate predictions. The influence of using alternative score functions in client selection is examined in Chapter 6.

4. Client Sampling Strategies

In an FL setting a sampling strategy determines which clients that will be participating in the training process and to what extent [pp. 10-11][13]. If the sampling strategy is not chosen appropriately, it can lead to biased results. If the sampling strategy selects clients that contain skewed data distributions, may make the resulting global model perform worse generally. Choosing an appropriate sampling strategy can lead to an improved performance and robustness of the FL model, by ensuring that the training process is representative and balanced across all participating clients [33; 16]. This makes the choice of a sampling strategy very important, since each client may have different data distributions, computational resources and dataset sizes [16].

Some of the sampling strategies that are commonly used in an FL setting are *uniform sampling* [12], *stratified sampling* [26], *clustered sampling* [24] and *importance sampling* [33].

4.1 Uniform Sampling

Uniform sampling is where a fraction of the clients are sampled uniformly at random from the quantity of all the clients [16].
Let K be the set of all clients, let C K that is selected for training and let $|K|$ be the cardinality of K. Then for uniform sampling the following is obtained

$$P(c_i \quad C) = \frac{1}{|K|} \qquad i \quad K \tag{4.1}$$

where $P(c_i \quad C)$ is the probability of the i'th client being one of the selected clients in the training process [12; 16].
Hence all the clients have an equal probability to be selected for the training process and thereby contribute equally to the training process. This selection process does not take into account that clients may have different data distributions or amount of data, which may lead to biased results. In this case a sampling strategy that takes heterogeneity of the data into account may be needed in order to ensure an unbiased training process [24, p. 25].

4.2 Stratified Sampling

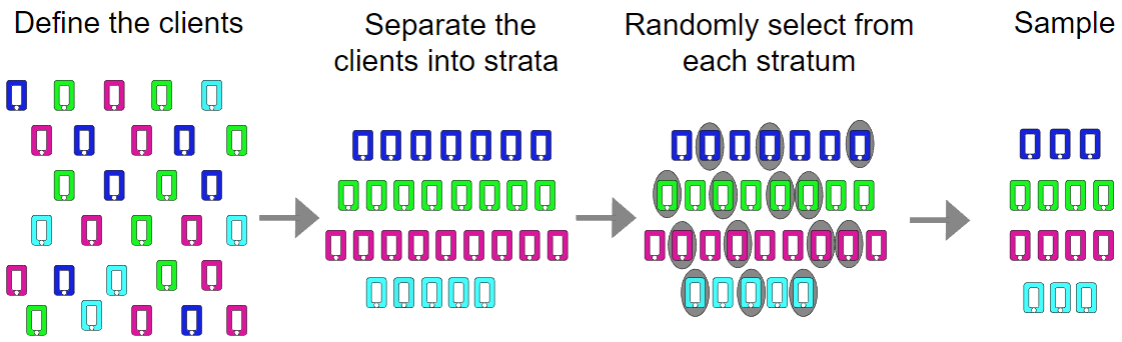


Figure 4.1: Stratified Sampling.

Stratified sampling is used when the clients have different data distributions. In stratified sampling the clients are grouped into strata based on the characteristics that they share (e.g. gender, education etc.) with each stratum consisting of only one common characteristic [26]. A fraction of clients are then selected from each stratum to participate in the training process ensuring that the training process includes all the characteristics. In Figure 4.1 the concept of stratified sampling is illustrated. A trade-off in stratified sampling compared to uniform sampling is higher precision at the cost of higher complexity, due to the division of the clients into strata. It should be noted that the characteristics of the clients might not be accessible [24, Chp. 3].

4.3 Clustered Sampling

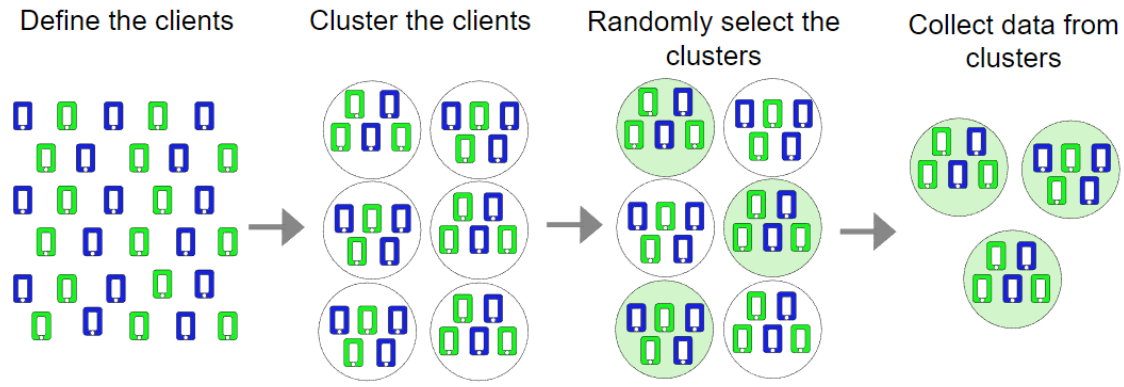


Figure 4.2: Clustered sampling.

In clustered sampling the clients are divided into smaller groups called clusters, and a fraction of these clusters are selected to participate in the training process.

Usually clustered sampling is used when some pre-existing groups, such as ages, cities etc. are visible from the data [17]. A key difference between stratified and clustered sampling is the selection process. In stratified sampling, random samples are selected from each stratum, whereas in clustered sampling, a random set of clusters is selected. Generally clustered sampling decreases the precision compared to uniform and stratified sampling. This is due to each cluster being biased in some form [24, Chp. 5]. In practice, clustered sampling is used because of the convenience and lower cost to sample in clusters than using uniform sampling.
In Figure 4.2 the concept of clustered sampling is illustrated.

4.4 Importance Sampling

In importance sampling the clients are weighted according to the importance of their data in the overall training objective [16]. This means that, clients that contain more important data are given higher weights, while clients with less important data are given lower weights.

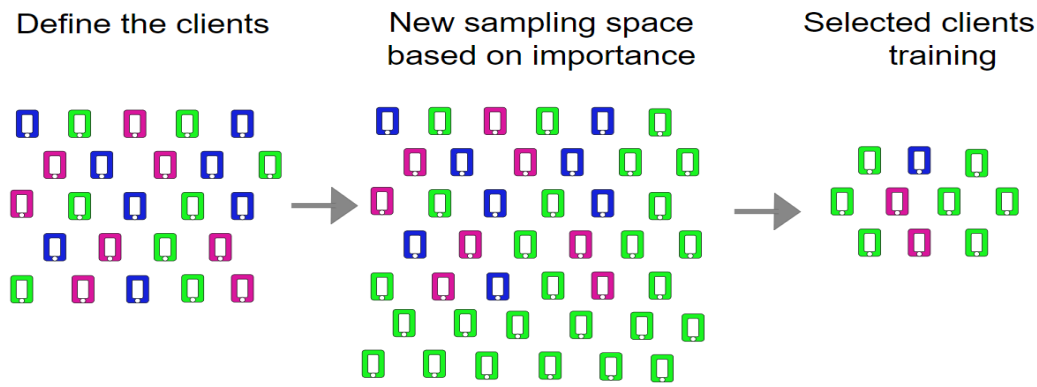


Figure 4.3: Importance sampling. Left: The di□erent clients are defined. Middle: The green clients are determined to be of higher importance, illustrated by making the majority of clients in the sampling space the green clients. Right: A subset of the new sampling space is selected for training. More clients of a specific color means higher importance weight.

Consider client c_i, then the importance weight assigned to client c_i is denoted as ω_i. In importance sampling, a subset $C = \{c_1, c_2, \ldots c_m\}$ of all the clients K is selected, such that the probability of selecting client c_i is proportional to its importance weight ω_i. This gives the following probability of selecting client c_i [33]

$$P(c_i \quad C) = \frac{\omega_i}{\sum_{j=1}^{N} \omega_j}. \tag{4.2}$$

Choosing appropriate importance weights are crucial for correct sampling. One example of weighting clients is by their amount of data, i.e. clients with more data is weighted higher than clients with less data. The concept of importance sampling is illustrated in Figure 4.3.

Overall, the choice of sampling strategy depends on the application of the FL task and the characteristics of the clients. Hence it is important to choose a sampling strategy that produces representative data for the training model, while also considering scalability and efficiency [16; 33].

4.5 The Usage of CP in Client Sampling

Our proposal for a sampling strategy, is to include the CP framework to select specific clients, which positively contributes to counteract model drift. In the CP framework, conformal scores are calculated based on a calibration set and an error-rate α, which are used to determine the quantile $\hat{q}$. A large quantile $\hat{q}$ is the result of larger scores, which encodes a worse relation between X and Y, i.e. the client's model is bad at correctly predicting.

The error rate α affects the amount of correct predictions a client should have and the quantile threshold $\hat{q}_\theta$ then determines how certain that client should be in its prediction. Therefore a low error rate and a low quantile threshold are desired. This infers that the client's ML model correctly predicts with a high softmax probability consistently. We propose, that each client trains until it satisfies a quantile threshold $\hat{q}_\theta$, which will be specified in Chapter 5. To ensure that a client does not train infinitely, the number of LTRs is limited. Subsequently, if a client does not satisfy the threshold after the maximum number of LTRs, the client is not participating in aggregation this GTR. After each aggregation all clients are updated with the new parameters and begin training locally again. This is repeated for a predetermined number of GTR.

There could be several reasons why a client fails to satisfy the threshold, some of which are: the dataset, the model architecture, the choice of α, the size of the calibration set or if the client's model is underfitted due to lack of data. If none of the clients satisfy $\hat{q} \leq \hat{q}_\theta$, the choice of α. $\hat{q}_\theta$ or the number of clients has to be reconsidered.

5. Experimental Setup

In this chapter, our proposed methods to incorporate CP in client selection for FL are described with specific procedures both client-side and aggregator-side. The ML model and datasets used to conduct experiments are presented, as well as the method for distributing data among clients IID vs. non-IID.

5.1 Proposed Method

In Section 4.5, we proposed a method for selecting clients for aggregation. The entire training process can be clearly explained by separating the training into an aggregator side and a client said.

Aggregator Side

The objective for the aggregator is simply to aggregate and distribute model parameters. It distributes an initial set of parameters to the clients, and after the clients are done training locally, the parameters of the clients which satisfy $\hat{q} \leq \hat{q}_\theta$ will be aggregated. The model parameters from the aggregated clients are then averaged using the `FedAvg` algorithm and distributed back to all clients.

Client Side

After receiving model parameters from the aggregator, the client starts training on its local data. The local data is split into a training set (80 %) and calibration set (20 %). Between each LTR, the calibration set is used to calculate the quantile $\hat{q}$ based on a predetermined error rate α. If a client satisfies that $\hat{q} \leq \hat{q}_\theta$, the client is said to be done training. To assure that a client does not training an infinite number of LTRs, a maximum number of LTRs of 40 epochs has been determined. If none of the clients satisfy $\hat{q} \leq \hat{q}_\theta$, the error rate α or the threshold $\hat{q}_\theta$ has to be reconsidered. Throughout the experiments $\hat{q}_\theta = 0.25$, meaning based on the error rate α, we require that $\approx 1 - \alpha\,\%$ of all scores has to be below 0.25. This value has been chosen empirically.

5.2 The Datasets

The datasets used in the experiments are the `MNIST` [31; 30], `EMNIST` [29; 37] and `CIFAR-10` [28; 23] datasets from the Pytorch librabries.
The `MNIST` dataset is chosen due to it being widely used as a benchmark tool. To extend the problem from a 10 class classifier, the `EMNIST` dataset was chosen. The `CIFAR10` dataset was chosen to see how our proposed model performs with a dataset different from the characteristics (other classes) of `MNIST` and `EMNIST`.

The `MNIST` dataset contains a large database of handwritten digits between 0 to 9. The digits are converted to 28 × 28 pixel images and the dataset contains $60,000$ training images and $10,000$ test images [30].

The `EMNIST` dataset is an extended version of the `MNIST` dataset and as well as containing images of handwritten digits 0-9, it contains images of handwritten lower and upper case letters from the English alphabet. We consider the balanced version of `EMNIST`, which $112,800$ training images and $18,800$ test images from 47 classes [37].

The `CIFAR-10` dataset contains color images in the 32 × 32 pixel format and it contains the following 10 classes: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship* and *truck*. The dataset contains $50,000$ training images, $5,000$ images of each class and the test set consists of $10,000$ images, $1,000$ of each class [23].

5.3 The ML Model

To conduct the experiments, we use a convolutional neural network (CNN), since they have been shown to achieve high accuracy for image classification problems [38; 25, p. 40-42; p. 1]. The architecture of our CNN consists of a mixture of convolutional layers, pooling and fully-connected layers. The specific architecture is shown in Figure 5.1. Remark: This architecture considers the `MNIST` and `EMNIST` 28×28 image inputs. The dimensions of each convolutional layer differs slightly when using the 32×32 image input from the `CIFAR10` dataset.

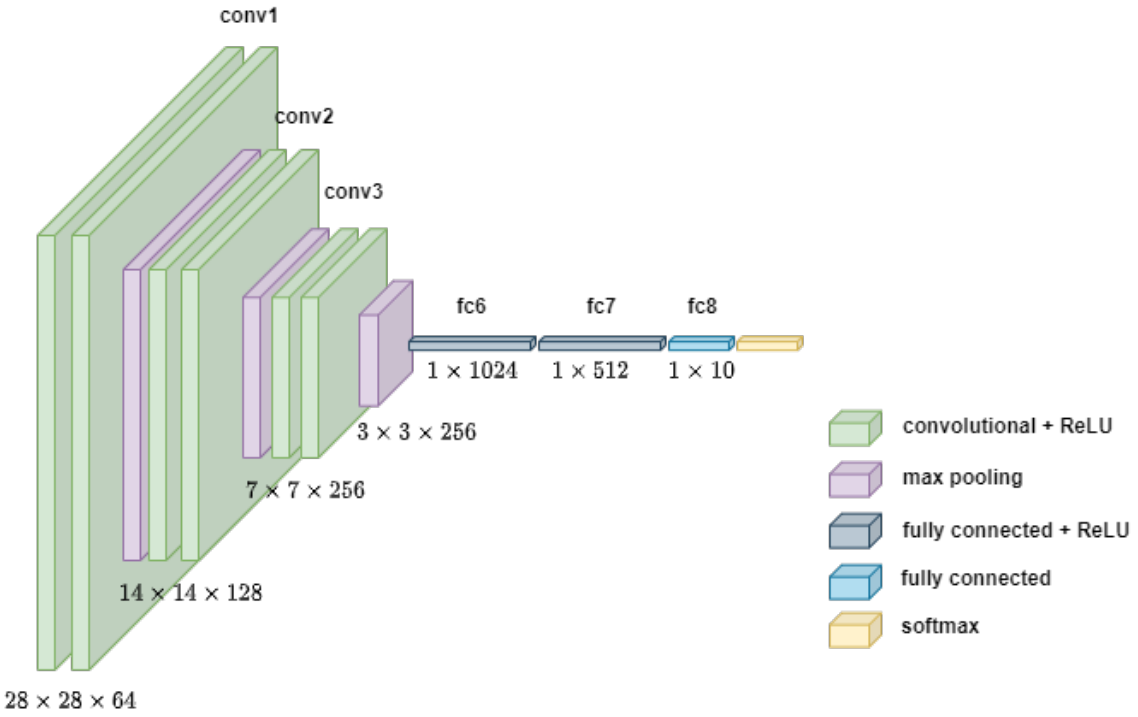


Figure 5.1: Architecture of our CNN. Output returns softmax probabilities.

This architecture has been chosen due to its high performance accuracy on the chosen datasets as shown in Section 6.1.

5.3.1 Hyperparameters

The hyperparameters of the CNN are the optimiser, the learning rate, the weight decay, the batch size and the loss function. The optimiser is the Adam optimiser, which is an extension of `SGD`. This optimiser has been chosen due to its fast performance and good results. More about the optimiser is found in [20, p. 10]. The learning rate is set to 0.001 with a weight decay of 0. These values are the default values from the Pytorch Adam Optimiser [27]. The batch size is set to 128 and has been chosen through experimentation.
For training, the cross-entropy (CE) function is used as the loss function. The CE function measures how close the predicted output $\hat{f}(x)$ is to the true class. The CE loss function for a specific sample (x, y) is given as

$$L_{CE} = -\sum_{i=1}^{n} y_i \log\left(\hat{f}(x)_{y_i}\right) \tag{5.1}$$

where $\hat{f}(x)_{y_i}$ is the softmax probability for the i'th class, n is total number of classes and y_i is the true probability for the i'th class. The CE loss function is minimised

when all predictions are correct. More about CE can be found in [20, p. 64].

5.4 Data Distribution

In this thesis, the effect that the data distribution has on the performance of the ML model in a FL setting is examined. Two scenarios are considered in the FL setting: balanced IID data among clients and unbalanced non-IID data among clients.

5.4.1 IID Data

In order to make the data balanced and IID among the clients, each client draws an equivalent amount of samples from the training set. This means that each client has the same amount of data (balanced), and it the same distribution. This is done by Pytorch's build-in function `random_split` [32]. Each client then generates a training set and calibration set. The test set from the dataset is reserved for the aggregator, and therefore not partitioned among the clients.

5.4.2 Non-IID Data

In order to make the data Non-IID and unbalanced among the K clients, the Dirichlet mean-parameter has been utilised. Similar methods has been used in [8; 10; 11]. The partitioning a dataset of n classes is illustrated in Figure 5.2.

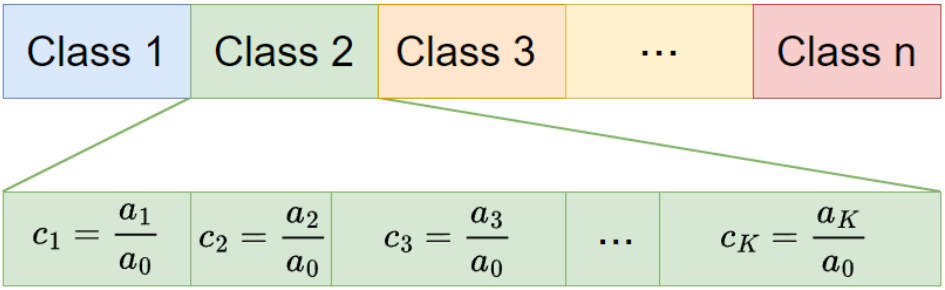


Figure 5.2: Visualisation of non-IID partitioning of the dataset.

The data is sorted by class and each class is divided into K groups, K being the total number of clients. To divide a class, K random numbers $a_1, a_2, \ldots, a_K$ are picked, where each number is divided by their sum $a_0 = \sum_{i=1}^{K} a_i$. This yields random fractions, for which each client is assigned a corresponding fraction of the class. This process is repeated for each class and results in each client having a different distribution and amount of data.

Similarly to IID data, the test set from the dataset is reserved for the aggregator, and therefore not partitioned among the clients.

6. Results

In this chapter, experiments of three different ML scenarios are conducted. The ML scenarios that we have tested and compared are:

(1) A centralised setting,

(2) A FL setting, where the data is balanced and IID among the clients,

(3) A FL setting, where the data is unbalanced and Non-IID among the clients.

Scenario (1) and (2) are considered as baselines to compare them with the third scenario. Scenario (3) is considered the realistic scenario, based on the clients' non-IID and unbalanced data distribution [7; 34, p. 2; p. 2]. The main objective is to evaluate the performance of our proposed client sampling strategy in a FL setting, as explained in Section 4.5. The experiments are conducted using the setup from Chapter 5.

6.1 Scenario (1) – Centralised Setting

The first scenario is the centralised ML setting and is considered a standard supervised ML problem. In this setting, all the data is collected and stored at one location, and is used to train a single ML model. The training set is separated into a smaller training set (80 %) and a calibration set (20 %). The calibration set is used post training to calculate the CP quantile $\hat{q}$ using the score function $s^{(1)}$, and will be evaluated at each training training round. The accuracy of the model is calculated for the training set and test set to evaluate how the model performs on known and unknown data. The accuracy is calculated as the number of correct predictions (the class with the highest softmax probability) divided with the length of the entire training/test set.

Using the datasets `MNIST`, `EMNIST` and `CIFAR10`, the training accuracy and test accuracy of the CNN are plotted Figure 6.1.

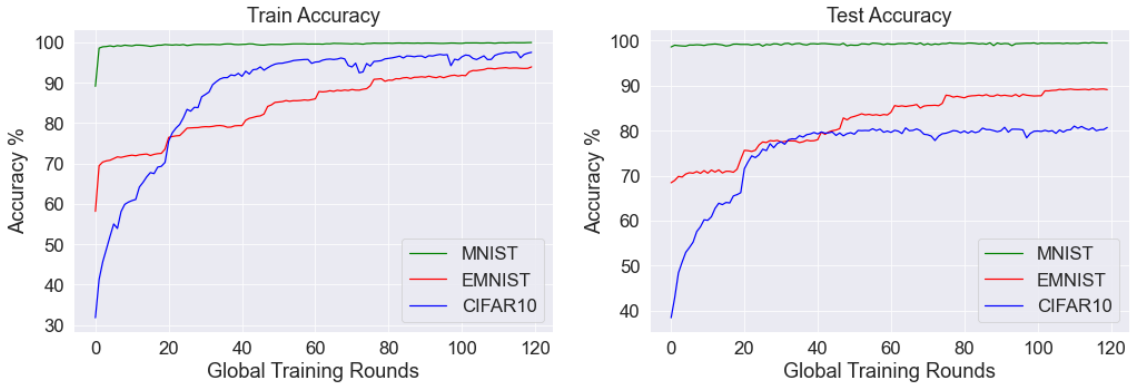


Figure 6.1: Accuracy for `MNIST`, `EMNIST` and `CIFAR10` datasets using our proposed CNN.

From Figure 6.1 it is seen that the CNN achieves an accuracy of $\approx 99\,\%$ in training and testing for the `MNIST` dataset, a training and testing accuracy of $\approx 95\,\%$ and $\approx 90\,\%$ for the `EMNIST` dataset, and a training and test accuracy of $\approx 97\,\%$ and $\approx 81\,\%$ for the `CIFAR10` dataset. A reason for the higher accuracy using the `MNIST` dataset is the low number of classes and the fact that the images are grey-scaled. A reduction in accuracy is observed using the `EMNIST` dataset, which is possibly due to the increase in classes from 10 to 47. The accuracy using the `CIFAR10` dataset achieves similar results to the `EMNIST` dataset, although with a reduction in testing accuracy. This could be a consequence of the `CIFAR10` dataset containing a higher variety of images in the different classes and the fact that it is RGB colored images. In Figure 6.2 the CNN's CE loss is plotted for each dataset.

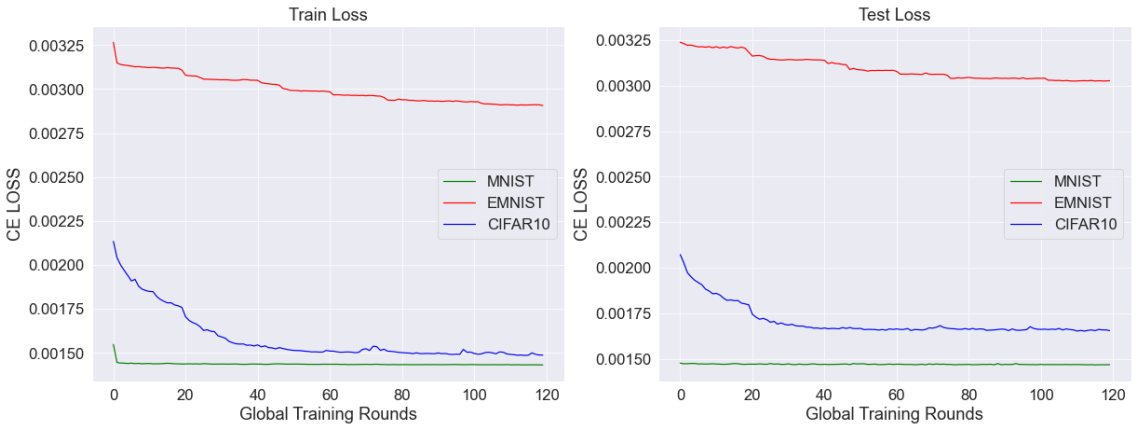


Figure 6.2: CE loss for `MNIST`, `EMNIST` and `CIFAR10` datasets using the CNN.

Notably, the test CE loss using the `EMNIST` dataset is higher than using `CIFAR10` dataset, although it has a higher accuracy. This could be the result of the CE loss function being a sum and the `EMNIST` dataset consisting of more classes than the `CIFAR10` dataset, hence the CE loss is greater. Therefore, we will only evaluate the accuracy.

The quantiles for different error rates α were also calculated to give an indication of which α-values are possible in the centralised setting. The results are shown in Figures 6.3 and 6.4 for the `MNIST` dataset, and the `EMNIST` and `CIFAR10` datasets, respectively.

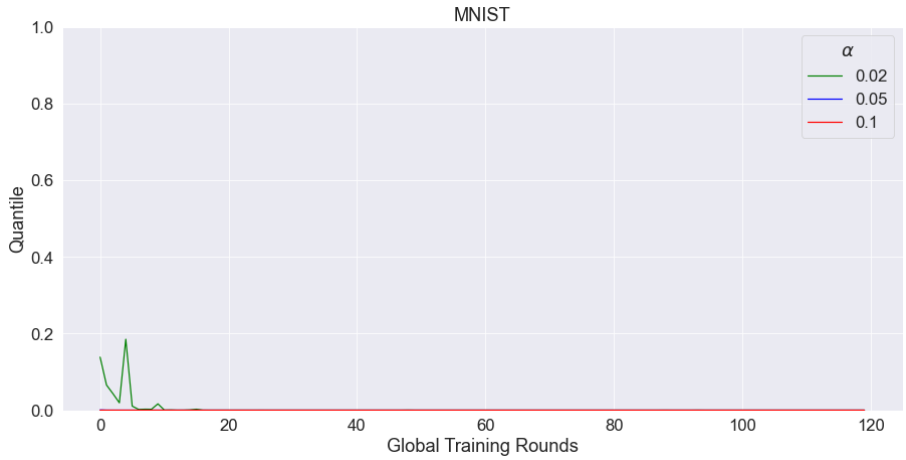


Figure 6.3: The quantile for different error rates for the `MNIST` dataset.

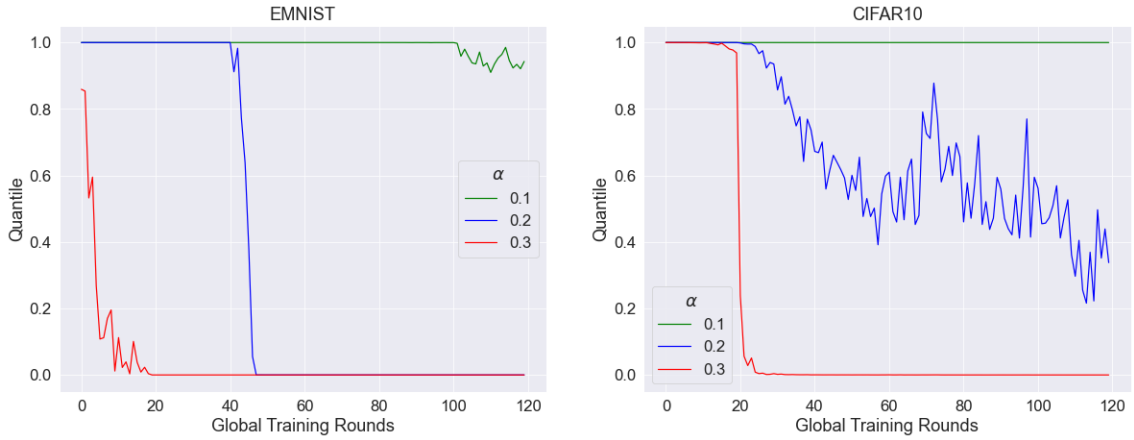


Figure 6.4: Left: The quantile for different error rates for the `EMNIST` dataset. Right: the quantile for different error rates for the `CIFAR10` dataset.

From Figure 6.3, it is possible to achieve a low quantile with an error rate of 0.02, after a reasonable number of training rounds. These error rates (0.02, 0.05, 0.1) will be evaluated for scenario (2) and (3) to see if they are achievable in a FL setting. From Figure 6.4 it is seen that a low quantile is achievable for an error rate of 0.2 and 0.3 for both the `EMNIST` and `CIFAR10` datasets.

The low quantile in Figure 6.3 in combination with a very low α is the result of the CNN consistently predicting correct. The spike in the quantile using $\alpha = 0.02$ is possibly a result of the `SGD` updating the parameters in the training round. Setting

$\alpha = 0.01$ yielded the quantile $\hat{q} = 1$ for all GTRs, hence we determine $\alpha = 0.02$ as the lower bound for our CNN.

From Figures 6.4 and 6.1, there is observed to be a relation between the test accuracy of our model and the value of the quantile $\hat{q}$. When the model exceeds a test accuracy of $1 - \alpha$, the quantile starts decreasing. As an example, when the test accuracy for `EMNIST` reaches $\approx 80\,\%$, the quantile for $\alpha = 0.2$ drops immediately. This means that not only do we get a high accuracy, but we also predict the true class with a high softmax probability.

6.2 Scenario (2) – FL setting with Balanced IID Data

In this scenario, the influence of clients and error rates on the performance of the aggregator is evaluated. The accuracy of the aggregator is calculated for 5, 10 and 15 clients, with three different error rates for each. The quantile threshold $\hat{q}_\theta$ is set to 0.25 and the score function $s^{(1)}$ is used.
In the standard FL method, a random subset of clients participate in aggregation each GTR, which is denoted as 'Frac' in Table 6.1. A quarter of the clients participate in aggregation each GTR. However, since such few clients are considered in this thesis, we also compare our proposed method with aggregating all clients.
When using our proposed method, clients were allowed up to 40 LTRs, but ceases training when they have reached a quantile below the threshold. When not using our proposed method, clients were allowed 20 LTRs.
For each number of clients, the data distribution among the clients is fixed, e.g. for 5 clients, each client have the same local data for all three methods. The results are presented in Table 6.1, which shows the highest accuracy achieved in 10 GTRs.

From Table 6.1, aggregating all clients each GTR is observed to achieve the highest accuracy. Our proposed method achieves approximately the same accuracy, while only choosing a fraction of all clients yields a significantly lower accuracy. In Table 6.1 an 'X' means that our proposed method fails, which is due to none of the clients achieving a quantile below the threshold with the chosen error rate. This could be due to multiple reasons, which include: a lack of LTRs for each client, a lack of data for each client or the quantile threshold being too strict.

	5 Clients		10 Clients		15 Clients	
MNIST	α	Accuracy	α	Accuracy	α	Accuracy
	0.02	99.3 %	0.02	X	0.02	X
	0.05	99.5 %	0.05	99.2 %	0.05	99.1 %
	0.1	99.2 %	0.1	98.5 %	0.1	98.4 %
	All	99.7 %	All	99.4 %	All	99.6 %
	Frac	73.5 %	Frac	65.2 %	Frac	37.9 %
	5 Clients		10 Clients		15 Clients	
EMNIST	α	Accuracy	α	Accuracy	α	Accuracy
	0.4	68.7 %	0.4	X	0.4	X
	0.5	70.8 %	0.5	X	0.5	X
	0.6	63.8 %	0.6	X	0.6	X
	All	77.9 %	All	29.8 %	All	15.0 %
	Frac	3.3 %	Frac	2.5 %	Frac	2.9 %
	5 Clients		10 Clients		15 Clients	
CIFAR10	α	Accuracy	α	Accuracy	α	Accuracy
	0.5	68.5 %	0.5	X	0.5	X
	0.6	65.0 %	0.6	66.2 %	0.6	X
	All	81.4 %	All	68.7 %	All	44.7 %
	frac	21.6 %	Frac	12.8 %	Frac	17.9 %

Table 6.1: Accuracy for different error rates using IID data.

Choosing all clients for aggregation yields the highest accuracy, which is possibly due to the IID data of the clients. Since every client is included and the data distribution is identical, each client's bias may only contribute an insignificant amount to model drift. From Table 6.2, our method is shown to also aggregates all clients each GTR. This means that for IID data, there is no benefit in using our method as compared to aggregating all clients. Similar tables for `EMNIST` and `CIFAR10` are shown in

Appendix A.

	5 Clients			10 Clients			15 Clients		
GTR \ α	0.02	0.05	0.1	0.02	0.05	0.1	0.02	0.05	0.1
1	1	5	5	X	10	10	X	14	15
2	3	5	5	X	10	10	X	15	15
3	4	5	5	X	10	10	X	15	15
4	4	5	5	X	10	10	X	15	15
5	5	5	5	X	10	10	X	15	15
6	5	5	5	X	10	10	X	15	15
7	4	5	5	X	10	10	X	15	15
8	5	5	5	X	10	10	X	15	15
9	5	5	5	X	10	10	X	15	15
10	5	5	5	X	10	10	X	15	15

Table 6.2: Number of aggregated clients for `MNIST` IID data for each GTR.

6.3 Scenario (3) – FL Setting with Unbalanced Non-IID Data

Similar to Section 6.2, the influence of clients and error rates on the performance of the aggregator is evaluated. The results are calculated using identical setup parameters as in Section 6.2, where only the data distribution among clients differs. Different score functions are also evaluated to see if they have any effect on the performance of the aggregator. In Table 6.3 a comparison of accuracy for the different sampling methods for 5, 10 and 15 clients is shown.

	5 Clients		10 Clients		15 Clients	
MNIST	α	Accuracy	α	Accuracy	α	Accuracy
	0.02	98.6 %	0.02	95.1 %	0.02	97.4 %
	0.05	97.8 %	0.05	95.0 %	0.05	98.1 %
	0.1	96.9 %	0.1	95.3 %	0.1	96.6 %
	All	90.7 %	All	73.9 %	All	94.9 %
	Frac	11.3 %	Frac	11.6 %	Frac	12.8 %
	5 Clients		10 Clients		15 Clients	
EMNIST	α	Accuracy	α	Accuracy	α	Accuracy
	0.3	63.0 %	0.3	48.2 %	0.3	52.5 %
	0.4	60.3 %	0.4	66.2 %	0.4	57.5 %
	0.5	54.1 %	0.5	53.1 %	0.5	59.8 %
	All	28.5 %	All	11.4 %	All	6.5 %
	Frac	6.9 %	Frac	7.3 %	Frac	3.5 %
	5 Clients		10 Clients		15 Clients	
CIFAR10	α	Accuracy	α	Accuracy	α	Accuracy
	0.3	48.8 %	0.3	35.7 %	0.3	25.7 %
	0.4	47.2 %	0.4	44.7 %	0.4	58.8 %
	0.5	50.7 %	0.5	52.3 %	0.5	53.7 %
	All	64.3 %	All	50.3 %	All	51.8 %
	Frac	11.3 %	Frac	12.9 %	Frac	10.9 %

Table 6.3: Accuracy for di□erent error rates using non-IID data.

From Table 6.3, our proposed method is shown to achieve a higher accuracy when utilising the `MNIST` and `EMNIST` datasets compared to the other methods, and a similar accuracy to including all clients using the `CIFAR10` dataset. Compared to the results from Table 6.1, non-IID data improves the applicability of our method,

	5 Clients			10 Clients			15 Clients		
GTR \ α	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5
1	3	5	5	3	6	10	2	2	13
2	4	5	5	3	8	10	3	4	15
3	4	5	5	4	7	10	4	7	15
4	4	5	5	6	8	10	3	8	15
5	4	5	5	7	8	10	5	9	15
6	5	5	5	6	8	10	4	9	15
7	5	5	5	7	6	10	6	9	15
8	5	5	5	7	9	10	7	10	15
9	5	5	5	7	10	10	5	10	15
10	5	5	5	6	10	10	8	10	15

Table 6.4: Number of aggregated clients for `EMNIST` non-IID data for each GTR.

i.e. no failed aggregations for similar error rates. This is possibly due to some clients receiving more training data, or that the distribution they have does not include all classes. The possible reduction in classes reduces the complexity for the specific clients, i.e. fewer classes to predict.

To gain insight on the accuracy of our method, the number of clients participating in aggregation for the `EMNIST` dataset is examined and shown in Table 6.4. Similar tables for `MNIST` and `CIFAR10` are shown in Appendix A.

From Table 6.4, more clients participate in aggregation when increasing the error rate. For 10 clients and $\alpha = 0.5$, all clients participate in every GTR, but yield a significantly higher accuracy compared to including all clients as shown in Table 6.3. This is possibly due to our method being allowed up to 40 LTRs, whereas the other methods only use 20 LTRs. For 10 and 15 clients and $\alpha = 0.3$, the `CIFAR10` dataset achieves a significantly lower accuracy compared to the other error rates, which is a result of too few clients included in each GTR. This can also be seen in Table A.4 in Appendix A. The essential part of Table 6.4, is that as the GTRs increases, the number of aggregated clients increase. This means that the clients which were not considered for aggregation in earlier GTR, have improved and are participating.
An example of the evolution of the accuracy for the three client selection methods is

shown in Figure 6.5. From this we see that already in the first few GTR the accuracy using our proposed method exceeds the two other methods. This tendency is present in all the experiments with non-IID data. This further supports that choosing the appropriate clients leads to better results using our proposed CNN.

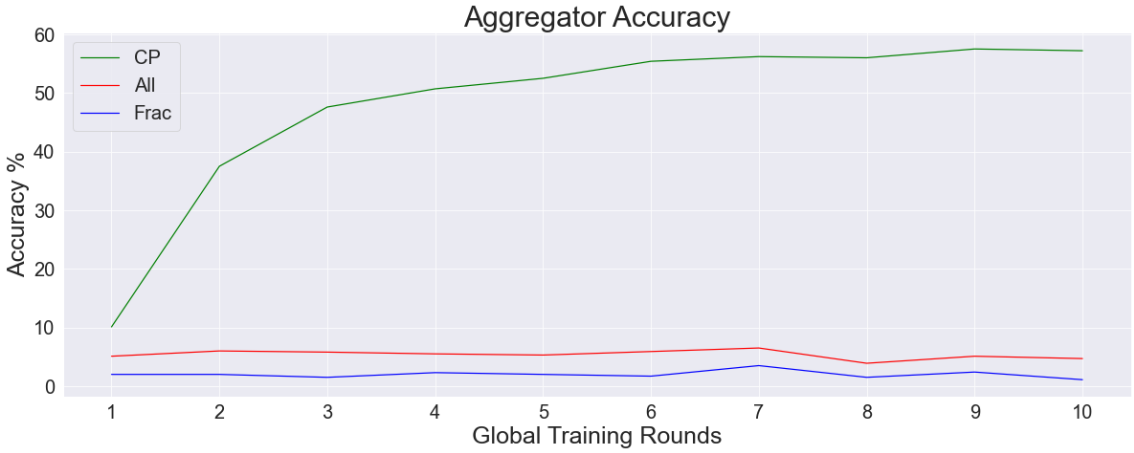


Figure 6.5: A comparison of the three different client selection methods, where the CP is for the EMNIST dataset with an error rate on 0.4 and for 15 clients.

6.3.1 Different CP Score Functions

Recall the examples of CP score functions presented in Section 3.2:

$$s_i^{(1)} = 1 - \hat{f}(X_i)_{Y_i}, \tag{6.1}$$

$$s_i^{(2)} = e^{-4\hat{f}(X_i)_{Y_i}}, \tag{6.2}$$

$$s_i^{(3)} = 1 - \sigma(\hat{f}(X_i)_{Y_i}, 20, 10). \tag{6.3}$$

The impact of the score function is examined in relation to the accuracy of the aggregator for scenario (3). The results are plotted in Figure 6.6 for the EMNIST dataset with an error rate $\alpha = 0.3$, a quantile threshold $\hat{q}_\theta = 0.25$ and 10 clients.

From Figure 6.6 it is seen that $s^{(2)}$ achieves the highest accuracy, $s^{(3)}$ achieves the second highest accuracy and $s^{(1)}$ achieves the lowest accuracy. With this error rate and quantile threshold, $s^{(1)}$ requires that 70 % of softmax probabilities are higher than 0.7, $s^{(2)}$ requires that 70 % of softmax probabilities are higher than ≈ 0.35, and $s^{(3)}$ requires that 70 % of softmax probabilities are higher than ≈ 0.55.

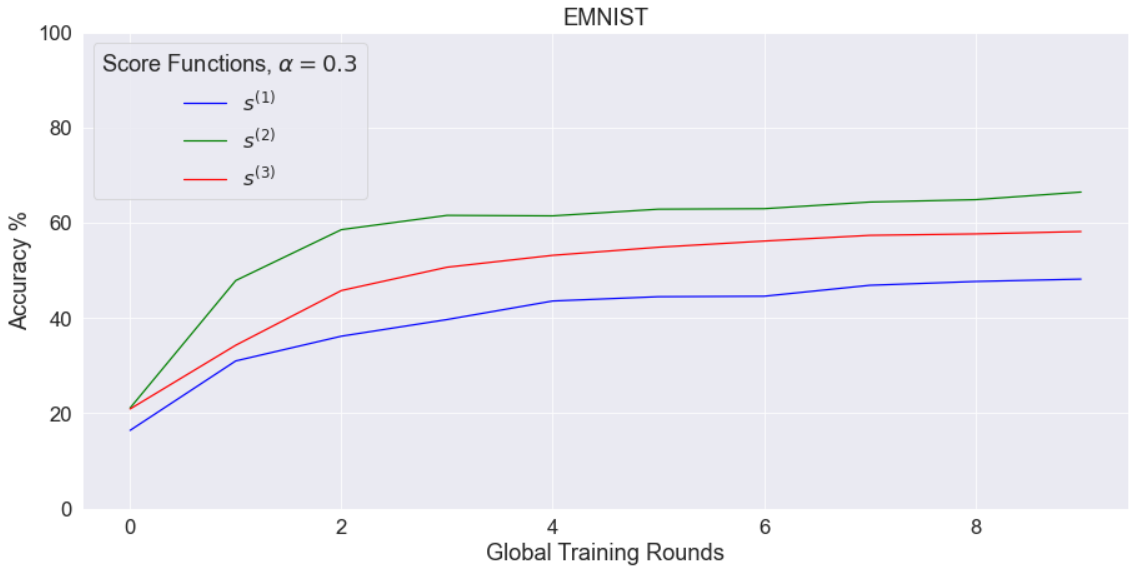


Figure 6.6: Accuracy plot utilising the di□erent score functions presented in Section 3.2.

Therefore, $s^{(2)}$ is more likely to include clients in aggregation, although they may become less certain in their predictions, i.e. lower softmax probabilities. Instead of using different score functions, altering the quantile threshold for $s^{(1)}$ would allow for more clients to participate in aggregation, which yields similar results. However, excessively increasing the threshold for clients, results in clients immediately ceasing training after a few or even the first LTR, and subsequently decreasing the accuracy. Therefore choosing an appropriate quantile threshold is essential.

6.4 Evaluation of Results

From comparing scenario (1) to scenario (2) and (3), we see that distributing the data among clients yields a lower accuracy than keeping it centralised. From Scenario (2), we see that aggregating all clients achieves the highest accuracy, hence our proposed method is worse. Furthermore due to lack of data in each client, our method fails to produce results for different error rates for a specific quantile threshold.

In Scenario (3), we see that our proposed method outperforms aggregating all clients and aggregating a random subset of all clients. We believe that parring each client with a performance metric (CP quantile) benefits the aggregation process.

Our proposed method is shown to improve all clients each GTR using non-IID data to the point that previously ignored clients can be considered for aggregation, while improving the accuracy of the aggregator. Therefore discarding clients that was previously ignored is not beneficial to the training process.

From evaluating different score functions, we see that it has the same effect as choosing a different quantile threshold. We see that increasing the quantile threshold can increase the accuracy of the aggregator, although too great of an increase yields a worse accuracy. Optimising the error rate and quantile threshold would likely significantly increase the accuracy of the aggregator.

In Scenario (3), it should be noted that the accuracy of 5 clients cannot be compared to that of 10 or 15, due to the distribution of data being generated differently when adding more clients. However, it is still possible to compare the methods used for each number of clients, since each client has the same distribution for all three methods (CP, All, Frac).

7. Conclusion

In this thesis, client selection methods for FL are examined in relation to their impact on the aggregator's accuracy. A method utilising the CP framework has been proposed, in which a quantile based on the prediction-capabilities of the clients was calculated. This quantile serves as a threshold for client participation in aggregation.

Our proposed method was compared to selecting all of the clients and a fraction of the clients for aggregation. The methods where evaluated for three different datasets, `MNIST`, `EMNIST` and `CIFAR10` for a different amount of clients and error rates. If the data is IID among clients, selecting all clients every GTR yielded the highest accuracy. Furthermore, our proposed method fails due to an insufficient amount of data at each client, which yields that the quantile from each client fails to reach the threshold in 40 LTR. However, if the data is non-IID among clients, our proposed method produced similar or higher accuracy than the other two client selection methods. Our method was shown to include previously ignored clients in later GTRs, which is due to the method improving the accuracy of both the aggregator as well as all the clients. Hence we believe that discarding ignored clients is not beneficial for the training process of the aggregator.

Tuning the error rate and quantile threshold was concluded to be essential to the accuracy of the aggregator. The error rate affects the amount of correct predictions a client should have. The quantile threshold then determines how certain that client should be in its prediction.

To summarise, our proposed client selection method for non-IID gives a good indication on when each client should stop their local training, however the resulting accuracy of the global optimisation problem is greatly depending on the error rate and quantile threshold in our method.

8. Further Research

From subjects and experiments examined in this thesis, we recommend the following topics for further research.

Adaptive Error Rate α and Quantile Threshold $\hat{q}_\theta$

In this thesis, the error rate and quantile threshold have been fixed throughout all GTRs. However, since the number of clients participating in aggregation each GTR is of importance, choosing the error rate and quantile threshold adaptively could be considered. A higher error rate or a higher quantile threshold in the earlier GTRs would allow for more clients to participate in the selection process. If the error rate or quantile threshold is lowered for the later GTRs, it would incentivise that clients should train more to be able to participate in client selection. It would be of interest to see how this affects the accuracy of the aggregator.

Examination of Participating vs. non-Participating Clients

When the data is non-IID among clients, it would be interesting to examine the difference between clients which participate in client selection and those that do not. Whether it is the amount of data or the distribution of data in a client that determines its participation is of interest.

Scalability

Due to the limited availability of computational power to process large amounts of data, our experiments cannot resemble the real-world application of FL. Therefore, the behaviour of our proposed method for larger networks and datasets is of great interest. Furthermore a larger amount of clients would show the applicability of our proposed method in a real world setting.

Including CP sets in Selection Process

Conformal prediction is primarily used to create sets of predictive outputs. However, our usage of CP is limited to the quantile of the calibration scores for each client. An interesting experiment would be to calculate the prediction set for each client every GTR, and inspect the relation between prediction set size and aggregated clients.

Bibliography

[1] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. arXiv:2107.07511v6, 2022.

[2] Patrick Billingsley. Probability and Measure. John Wiley & Sons, 1995. ISBN 0-471-00710-2.

[3] Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006. ISBN 978-0387-31073-2.

[4] Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, 2004. ISBN 978-0-521-83378-3.

[5] Sklearn Datasets. Sklearn datasets load breast cancer. URL `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html`.

[6] Dinh C. Nguyen et al. Federated learning for internet of things: A comprehensive survey. arXiv:2104.07914v1, 2021.

[7] H. Brendan McMahan et al. Communication-efficient learning of deep networks from decentralized data. arXiv:1602.05629 [cs.LG], 2023.

[8] Mi Luo et al. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. arXiv:2106.05001, 2021.

[9] Shashi Raj Pandey et al. Fedtoken: Tokenized incentives for data contribution in federated learning. arXiv:2209.09775v2, 2022.

[10] Tao Lin et al. Ensemble distillation for robust model fusion in federated learning. arXiv:2006.07242, 2021.

[11] Tzu-Ming Harry Hsu et al. Measuring the effects of non-identical data distribution for federated visual classification. arXiv:1909.06335, 2019.

[12] Vladimir Braverman et al. The power of uniform sampling for coresets. arXiv:2209.01901v2, 2022.

[13] Wenlin Chen et al. Optimal client sampling for federated learning. arXiv:2010.13723v3, 2022.

[14] Xiang Li et al. On the convergense of fedavg on non-iid data. arXiv:1907.02189v4, 2020.

[15] Yae Jee Cho et al. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. arXiv:2010.01243v1, 2020.

[16] Yann Fraboni et al. A general theory for client sampling in federated learning. arXiv:2107.12211v4, 2022.

[17] Yann Frabonin et al. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. arXiv:2105.05883v2, 2021.

[18] Isaac Gibbs and Emmanuel Candés. Conformal inference for online prediction with arbitrary distribution shifts. arXiv:2110.07661v2, 2022.

[19] Yoshua Bengio Ian Goodfellow and Aaron Courville. Deep Learning. MIT Press, 2016. ISBN 979-1-097-16044-9.

[20] Bin Cui Jiawei Jiang and Ce Zhang. Distributed Machine Learning and Gradient Optimization. Springer, 2022. ISBN 978-981-16-3420-8.

[21] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B. Gibbons. Federated learning under distributed concept drift. arXiv:2206.00799v2 [cs.LG], 2023.

[22] Pooja Kherwa, Saheel Ahmed, Pranay Berry, Sahil Khurana, Sonali Singh, Jaydip Sen, Sidra Mehtab, David W. W Cadotte, David W. Anderson, Kalum J. Ost, Racheal S. Akinbo, Oladunni A. Daramola, Bongs Lainjo, Rajdeep Sen, and Abhishek Dutta. Machine Learning Algorithms, Models and Applications. IntechOpen, 2021. ISBN 978-1-83969-486-8.

[23] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar10. URL `https://www.cs.toronto.edu/~kriz/cifar.html`.

[24] Sharon L. Lohr. Sampling: Design and Analysis. Taylor & Francis Group, 2019. ISBN 978-0-3672-7346-0.

[25] Charles Lu and Jayashree Kalpathy-Cramer. Distribution-free federated learning with conformal prediction. arXiv:2110.07661, 2022.

[26] Per Pettersson and Sebastian Krumscheid. Adaptive stratified sampling for non-smooth problems. arXiv:2107.01355, 2021.

[27] Pytorch. Adam, . URL https://pytorch.org/docs/stable/generated/torch.optim.Adam.html.

[28] Pytorch. Cifar10, . URL https://pytorch.org/vision/stable/generated/torchvision.datasets.CIFAR10.html.

[29] Pytorch. Emnist, . URL https://pytorch.org/vision/main/generated/torchvision.datasets.EMNIST.html.

[30] Pytorch. Yann lecun et al., . URL https://yann.lecun.com/exdb/mnist/.

[31] Pytorch. Mnist, . URL https://pytorch.org/vision/main/generated/torchvision.datasets.MNIST.html.

[32] Pytorch. Torch.utils.data, . URL https://pytorch.org/docs/stable/data.html.

[33] Elsa Rizk, Stefan Vlaski, and Ali H. Sayed. Federated learning under importance sampling. arXiv:2012.07383, 2020.

[34] Yong Shia, Yuanying Zhang, Yang Xiao, and Lingfeng Niu. Optimization strategies for client drift in federated learning: A review. ScienceDirfect: Procedia Computer Science vol. 214, 2022.

[35] Niladri Syam and Rajeeve Kaul. Machine Learning and Artifical Intelligence in Marketing and Sales. Emerald Publishing Limited, 2021. ISBN 978-1-80043-881-1.

[36] Li Tian, Anit Kumar Sahu, Virginia Smith, and Ameet Talwalkar. Federated learning: Challenges, methods, and future directions. arXiv:1908.07873v1 [cs.LG], 2019.

[37] Western Sydney University. Emnist. URL https://www.westernsydney.edu.au/icns/resources/reproducible_research3/publication_support_materials2/emnist.

[38] Bing Xue Ying Bi and Mengjie Zhang. Genetic Programming for Image Classification. Springer, 2021. ISBN 978-3-030-65927-1.

[39] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. arXiv:1806.00582v2, 2022.

Appendices

A. Appendix

A.1 Number of Clients Participating in each GTR for IID and nonIID data

This section includes Tables from Scenario 2 and 3 for the number of aggregated clients each GTR for different datasets with IID or non-IID data.

	5 Clients			10 Clients			15 Clients		
α / GTR	0.4	0.5	0.6	0.4	0.5	0.6	0.4	0.5	0.6
1	2	5	5	X	X	X	X	X	X
2	5	5	5	X	X	X	X	X	X
3	5	5	5	X	X	X	X	X	X
4	5	5	5	X	X	X	X	X	X
5	5	5	5	X	X	X	X	X	X
6	5	5	5	X	X	X	X	X	X
7	5	5	5	X	X	X	X	X	X
8	5	5	5	X	X	X	X	X	X
9	5	5	5	X	X	X	X	X	X
10	5	5	5	X	X	X	X	X	X

Table A.1: Number of aggregated clients for EMNIST IID data for each GTR

	5 Clients			10 Clients			15 Clients		
α / GTR	0.4	0.5	0.6	0.4	0.5	0.6	0.4	0.5	0.6
1	X	4	5	X	X	4	X	X	X
2	X	5	5	X	X	10	X	X	X
3	X	5	5	X	X	10	X	X	X
4	X	5	5	X	X	10	X	X	X
5	X	5	5	X	X	10	X	X	X
6	X	5	5	X	X	10	X	X	X
7	X	5	5	X	X	10	X	X	X
8	X	5	5	X	X	10	X	X	X
9	X	5	5	X	X	10	X	X	X
10	X	5	5	X	X	10	X	X	X

Table A.2: Number of aggregated clients for `CIFAR10` IID data for each GTR.

	5 Clients			10 Clients			15 Clients		
α / GTR	0.02	0.05	0.1	0.02	0.05	0.1	0.02	0.05	0.1
1	5	5	5	10	10	10	6	15	15
2	5	5	5	10	10	10	8	15	15
3	5	5	5	10	10	10	10	15	15
4	5	5	5	10	10	10	11	15	15
5	5	5	5	10	10	10	13	15	15
6	5	5	5	10	10	10	14	15	15
7	5	5	5	10	10	10	14	15	15
8	5	5	5	10	10	10	14	15	15
9	5	5	5	10	10	10	15	15	15
10	5	5	5	10	10	10	14	15	15

Table A.3: Number of aggregated clients for `MNIST` non-IID data for each GTR.

	5 Clients			10 Clients			15 Clients		
α / GTR	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5
1	3	4	5	2	6	9	3	7	13
2	4	5	5	2	8	10	3	11	15
3	4	5	5	3	9	10	3	12	15
4	4	5	5	3	9	10	3	14	15
5	4	5	5	2	10	10	3	14	15
6	4	5	5	3	10	10	3	14	15
7	4	5	5	3	10	10	3	14	15
8	4	5	5	2	10	10	3	15	15
9	5	5	5	3	10	10	3	15	15
10	4	5	5	3	10	10	3	15	15

Table A.4: Number of aggregated clients for `CIFAR10` non-IID data for each GTR.