Malicious & Cooperative Client Behavior Under Federated Learning with Score-Based Aggregation and Cluster Elimination

by

Murat Arda Onsu

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the
degree of Master of Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Conventional AI-based service flow remains a challenge for IoT-enabled devices since data collected by local clients are transferred to a centralized server, which contains a global machine learning (ML) model. However, this introduces privacy and security concerns for the clients. Federated Learning is positioned to overcome this problem where each client trains a local model with its local data and shares its model parameters with the centralized server instead of sharing data. Upon receiving all parameters, it aggregates them and generates a new global model. Later this global model is distributed among the clients.

Various aggregation methods have been published for increasing the global model's accuracy performance after aggregation. However, those new aggregation algorithms must be thoroughly investigated under malicious and collaborative environments. A malicious environment is a scenario where malicious clients are present and can share parameters to degrade the aggregated model performance. On the other hand, the collaborative environment is another scenario in which some clients can share information to collaborate. Therefore, if benign clients receive data from malicious clients in collaborations, it degrades its performance and makes it partially malicious, which is also harmful to server aggregation.

To tackle this issue, we investigate a new aggregation method called Score Based Aggregation (SBA) That aims to mitigate the impact of the model parameters from such malicious clients without compromising the training accuracy. Also, we develop another method for eliminating entirely malicious clients from aggregation by the unsupervised clustering algorithm. Finally, we compare our result to a baseline approach where the malicious client varies from 20% to 50%. Numerical results suggest that the SBA aggregation and unsupervised clustering help the model maintain the convergence of accuracy at higher levels in comparison to the baseline approach.

Acknowledgements

I want to express my deepest gratitude to my supervisor, Dr. Burak Kantarci and Prof. Azzedine Boukerche, for their guidance and endless support. Since our first meeting, they have always been patient and understanding when dealing with research problems and other challenges. It is remarkable that they are accessible for communication at any moment, whether it be by email, online chat, or in person. I am grateful for the opportunity to work with them and for everything they have done for me.

I also want to thank Dr. Murat Simsek for motivating me to pursue a master's degree in a different field. His constructive criticism, ideas, and assistance play an important role in my work. I would also like to thank Cem Mumtaz Eris and Nahid Parvaresh for their help and efforts and for providing me with sources.

Lastly, I would like to express my gratitude to my parents, Ayse Ceyda Onsu and Murat Onsu, for their material aid and spiritual support. They have stood by my shoulder through all of my challenges and have been my strongest supporters in every decision I have made. Also, I owe my deepest gratitude to my dear brother Kemal Alper Onsu and sister-in-law Yongling Wu. They provided a great effort for me to get used to Canada and have been my greatest supporters and motivators in Ottawa. Finally, I want to give my great thanks to my colleagues and lab friends Didem Cicek Simsek, Yakup Akkaya, Ömer Melih Gül, Samhita Kuili, Ahmed Mohamed Elsayed Omara, Mohsen Shahbazi Dastjerdi, Ghazal Asemian, Bin Xiao, Parisa Fard Moshiri, Zhiyan Chen.

Table of Contents

List of Tables

List of Figures

Publications of the Candidate During MASc Studies

Outcomes of the thesis:

- **M. A. Onsu**, B. Kantarci, and A. Boukerche "On the Impact of Malicious and Cooperative Clients on Validation Score-Based Model Aggregation for Federated Learning" *2023 IEEE International Conference on Communications (ICC): Communication QoS, Reliability and Modeling Symposium* (Accepted)
- **M. A. Onsu**, B. Kantarci, and A. Boukerche "How to Cope with Malicious Federated Learning Clients: An Unsupervised Learning-Based Approach" *Elsevier Computer Network* (Under Review)

Chapter 1

Introduction

Federated learning has emerged as a distributed learning paradigm to remediate the security and privacy-related concerns in conventional machine learning [1, 2]. Unlike centralized machine learning, in a federated learning setup, clients are not expected to share their data for training, but trained models that build on their local data [3]. All the clients share an initial global model at a centralized server to train locally with their data and aid the centralized model's convergence and synchronization throughout numerous communication rounds by sharing model parameters. The central server updates its global model using a predetermined aggregation process after receiving all model parameters from the clients. The clients can access the updated global model to begin a new training round. This iteration continues until the global model converges [4].

As a result of these procedures, data on the local system is no longer shared with the outside world, protecting both the security and privacy of the client's data. These features allow these methods to be applied in a variety of fields, including intelligent transportation systems [5], medical analysis [6], cybersecurity [7, 8], the Internet of Things [9, 10, 11, 12], Industrial Internet of Things (IIoT) [13], image processing such as brain tumor segmentation, medical databases-meta analysis [14], and more. Especially Hospitals [15] are one example of a company or service that can benefit from this distributed learning that cares about the privacy of its clients. Moreover, Personal smart devices with considerable processing power are a top candidate to combine with Federated Learning in Mobile Crowdsensing [16]. The latest research also concentrates on improving privacy [17] and accuracy performance under a scarcity of data [18].

The federated averaging (FedAvg) method [3] is used in baseline federated learning, where each client contributes equally to the aggregation. However, in some cases, this

type of joining is occasionally ineffective for global model training. Numerous different aggregation methods have been put out in recent research to increase the accuracy performance of federated learning [19]. Different aggregation algorithms are used for different use cases, for example, heterogeneity of data [20]," forgetting event" cases [21], and the optimal model converges [22]. Several federated aggregation methods are tested in [23], and a new aggregation method is proposed for the overall model to deal with data heterogeneity and enable fair federated learning.

Furthermore, there are studies on malicious environments where some clients are malicious and may compromise aggregation or privacy, like [24, 25, 26]. The training process's invisibility and the data's inaccessibility limit federated learning. Due to these issues, evaluating and analyzing the training process is challenging, such as detecting potential client abnormalities [27]. Since FL act as a "Black Box during the training process, Malicious clients can easily attack" the training process, such as sending malicious weights to the server, to influence the behavior of the global model [28]. Therefore, it is difficult for the expert to increase global model accuracy in malicious environments.

Many types of research have been done to increase clients' accuracy under a scarcity of data, non-IID distributed data, or increase the validation accuracy of global models over federated averaging. They mainly apply the weighted averaging of each client to aggregation. These researches mostly get more promising results compared to baseline aggregation. These algorithms, however, primarily focus on the existence of heterogeneity, non-IID, and imbalanced clients [29]. In this research, we assume that the system contains malicious clients who obtained harmful weights that, after aggregation, reduce the server's overall model accuracy performance.

We also assume that there is a collaboration in the environment, meaning clients can exchange some amount of data. Thus, there are two cases. First, it changes the clients' data distributions. All clients start training with identical IID-distributed data during the first communication round. However, as a consequence of the cooperation, different clients will hold different amounts of data in the environment. So baseline federating algorithm got some drawbacks since letting all clients join the participation equally is not reasonable. Additionally, as previously noted, we consider some malicious clients in the environments. Data corruption is present in specific malicious clients. If benign clients receive those data in collaboration, it would harm the client and convert it into a partially malicious one.

In the first phase, we investigate clients' models and the server's global model behaviors under non-collaboration IID-distributed data. Still, we assume that there are malicious clients in the environment, and their rate in the environment varies from 20% to 50%. In order to improve accuracy in a malicious environment, we adopt the reputation-enabled

aggregation in [30] and adapt it to a dynamic mobile network setting by considering the local model validation scores under unseen data. We refer to this model as Score-Based Aggregation (SBA). Compared to federated averaging, SBA is more resilient and produces a more stable accuracy level. Thus, SBA seeks to keep accuracy at a target level even when federated averaging accuracy rapidly decreases due to high corruption rates. Model parameters of every client contribute to the accuracy in a weighted sum manner where the weight is a function of the local training model accuracy. This aggregation method makes the federated aggregation more resilient to malicious submissions.

In the second phase, we investigate client accuracy changes in collaboration. This time clients, including malicious ones, exchange small amounts for their data. After each communication round in collaboration, specific clients might share their information with other clients. Since some clients have more data than others, the environment is no longer IID. Additionally, malicious clients can share their data with other clients. Clients that receive corrupted data from malicious clients can no longer be non-corrupted but partially malicious. Thus, corruption can partially spread to other clients, degrading the global model accuracy after aggregation.

In the third phase, we offer two-step methods to overcome the problem below. The first step is client selection, inspired by [31, 32, 33], for discriminating the malicious clients from the aggregation so that they will not harm server aggregation. However, we need to be sure that the environment contains malicious and corrupted benign clients. Therefore, we first attempt to do an outlier detection method [34]. We first find outlier clients according to their validation accuracy score. If there is no outlier, we assume all clients are benign or malicious, or these are similarly distributed according to their scores. We can understand this by looking at their average score; Then, we use either the unsupervised K-Means [35] algorithm for clustering clients or omit the clustering phase. If outliers in the environment or the average of clients are not appropriate, we apply K-Means Clustering and collect the clients into two groups according to their validation Scores. Clusters with a low validation score average will be assumed malicious clients and discarded from aggregation. We inherit the SBA method as an alternative aggregation algorithm in the second step. This method generates a participation rate for each client based on their validation scores instead of equally including all clients in the aggregation. Thus, Clients join the aggregation based on their participation rate ratio.

1.1 Motivation

Federated learning is novel research from distributed machine learning that can keep all data from local devices and let the centralized server's global model train with only the parameters of clients' models [36]. The main benefits of Keeping local data with the client devices are security and privacy protection for the machine learning domain. For example, some clients that want to keep their data private, such as hospitals, can benefit from federated learning and train their models without data sharing.

However, since the centralized model is not trained directly, it is difficult for us to track the model process, making it vulnerable to malicious attacks. Therefore, are many ways to attack the federated learning models, mostly done for leaking information [12] or degrading the global model accuracy [37]. This work focuses on attacks that reduce global model accuracy and how to overcome them. Making our global model more resilient to the malicious environment will give us more time to detect errors and make the process run smoothly.

Also, we increase the global model training performance under non-IID data distributions. Initially, all clients are initialized with the same amount of data. However, when collaboration starts, some clients hold much more data than others. Since machine learning depends on data size significantly, clients' learning and validation performance vary in the environment. So, it does not make sense for each client to participate in the aggregation equally. Therefore, our project also considers the data distributions as well as the malicious clients.

1.2 Contributions

Following a thorough literature review, we aim to make a server's global model more resilient to the malicious environment by preserving its accuracy. We propose Score-Based-Aggregation and then use K-Means clustering for malicious client elimination.

Our main contributions can be summarized as follows:

- We identify a recent score-based weighted aggregation method, improve it in terms of resiliency and learning performance under malicious environments, and introduce the impact of cooperation against malicious or corrupted clients.
- we investigate the accuracy performance of score-based aggregation in federated learning under malicious environments where malicious client rate changes from 20%

to 50%. Then we analyze the impact of collaboration between federated learning clients on the global model performance. After evaluating the global model accuracy in a non-collaborative setting under each malicious client rate, experiments are pursued in a collaborative setting.

- We identify K-Means clustering methods for discriminating the malicious clients from the aggregation and investigating the impact of the SBA algorithm on aggregation in the malicious and collaboration environment with selected clients.
- We compare the validation accuracy performance of the server and losses of clients after aggregation in malicious and collaboration environments with baseline federated averaging, SBA, and K-Means clustering & SBA aggregation methods where malicious client rate changes from 20% to 50%.

1.3 Thesis Outline

This thesis is comprised of six chapters. In the second chapter, we will investigate what kind of research is done for federated learning in the literature to let it guide us. In federated learning, much research is based on overcoming the non-IID problem, or privacy and security. Since federated learning does not work on collected data in the centralized server, some clients can suffer from non-IID or fewer data problems, making the system converge difficulty [38]. On the other hand, although federated learning provides us with the privacy of data, it is sensitive the malicious attacks that attack the global model training or leak the information of clients [39, 40].

Moreover, a Malicious attack is difficult to track in a non-IID environment since many malicious client detection algorithms depend on the clients' validation scores. In a non-IID environment, their validation scores might be lower, even if clients are benign, which can be mistaken for malicious clients. Therefore, we put related works into two categories: Malicious Clients and non-IID Problems and Heterogeneous Data Distribution Problems. Since our work mainly depends on the aggregation phase, we collect all aggregation methods in the last section of the related works and compare our proposed method with them.

In the third chapter, we describe the Score-Based-Environment and observe its effect in a malicious environment. In this part of the project, it is assumed that there are no collaborating clients in the environment, so clients will not exchange their data. Also, data is IID distributed, and there are malicious clients in the environment. We increase the malicious client rate in the environment from 20% to 50% in each case and compare

SBA with baseline federated averaging. Comparison is made by global model validation accuracy. Moreover, we investigate the clients' (best, worst, standard) loss behavior from the first to the last.

In the fourth chapter, we allow clients to collaborate in the environment. Therefore clients can exchange their data with each other. However, since clients do not hold the same data, it changes the environment's data distribution from IID to non-IID. Also, since malicious clients can share the corrupted data with others, partially malicious clients occur in the next communication rounds. Thus, global model accuracy performance decreases. We also investigate the behavior of best, worst, and average clients by looking at their loss values and global model performance by examining their validation scores after aggregation.

In the fifth chapter, we develop a new method with two steps to increase the accuracy in the malicious and collaboration environment. In the first step, we eliminate fully malicious clients by unsupervised K-Means algorithm [41] according to their accuracy. Then, We let other clients join the aggregation with the SBA method, so instead of aggregating them equally, we consider their validation scores. This process is done in case of partially malicious clients that can not be eliminated by K-Means clustering. After that, we investigate the clients' loss values for three cases: Federated Averaging, SBA, and K-Means clustering with SBA.

Finally, in the last chapter, we wrap up the approaches described in the thesis and bestow a conclusion and future works.

Chapter 2

Literature Survey and Background

Federated learning is developed for privacy and security issues in centralized machine learning. In traditional centralized machine learning, each device is expected to share its local data with the server to perform training on a global model [42]. However, many companies and clients that care about their privacy concerns [43] do not agree to share all of their information [44]. Therefore, it causes the degradation of training since the server cannot collect all data. Another problem of traditional machine learning is the communication cost. Since a centralized server tries to collect all incoming data, it might cause a bandwidth bottleneck because of the massive volume of data in a limited wireless channel [45]. Especially some tasks that require real-time decision makings, such as intelligent autonomous vehicular, may suffer from latency issues [46].

Federated learning comes with a distributed learning idea that instead of centralized global model training, clients receive a copy of the initial global model and train it with their local data independently [16]. Then, they send their local model's parameter instead of data. When the server receives all updates, it combines all models, called aggregation, and generates a new global model. This new global model is distributed among each client, and clients start a new local training. This process is called communication rounds, repeated until the global model converges [47]. Federated learning steps are given below.

1. Server generates the initial global model.
2. Global model is shared with the clients.
3. Clients start local training with local data.
4. Clients send their updated model parameters to the server.

5. Server receives all local updates and starts aggregation.

6. new global model is distributed among the clients.

7. if the global model does not converge, return to step-3.

Since local data is not shared, federated learning ensures the privacy and security of clients' information. However, federated learning needs help with two main issues in the environments. First, since assuming that the central server and clients already have a preexisting trust relationship is unrealistic in a federated learning setting, the system and the clients might be attacked by malicious and corrupted clients that may decrease the global model performance or do data leakage from other clients indirectly. So it is vital to create a defense mechanism to improve the federated model resiliency in malicious environments.

Data distribution, on the other hand, is the second problem for federated learning. Since the server collects all client data in traditional centralized learning, the global model usually does not suffer from non-IID data distributions, or scarcity [48]. However, in federated learning, the training is done inside the local clients' devices with restricted data. Also, we cannot assume that all clients hold the IID data for training. Therefore, Federated learning needs to work on data heterogeneity and scarcity in the training phase. The server's main aim is to maximize the global model's performance in such situations [49].

In this section, we first give brief descriptions and background of federated learning and security. Then, we examine the previous literature on security and privacy. Malicious environments and corrupted clients are investigated in that part. Also, some studies additionally include non-IID data since, in heterogeneous environments, If attacks intend to reduce the global model's accuracy, it is much more challenging to distinguish malicious clients from benign clients as many methods for detecting malicious clients rely on the clients' validation ratings and in non-IID environment client may have low validation scores because of data distributions. Then we have to look into non-IID data distribution and malicious client issues and propose solutions in the domain. These studies inspire most of our work, and we try to develop new solutions that move one step further from those studies. The roadway of this Chapter is given in figure 2.1.

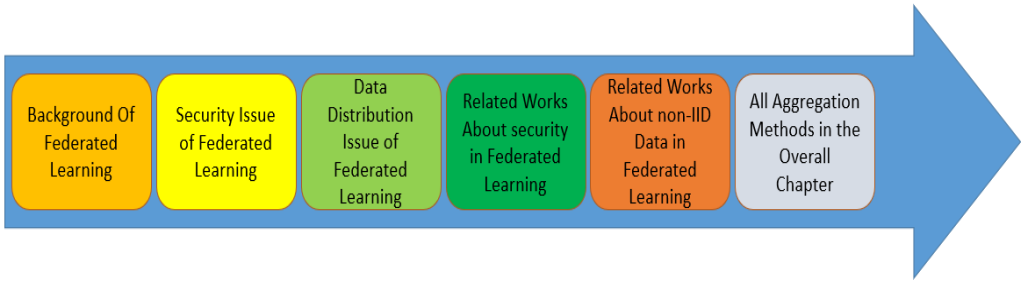

Figure 2.1: Roadway of the Chapter 2

2.1 Background of Federated Learning

Federated learning is the idea of protecting local clients' data privacy and security. It is a decentralized learning approach where, instead of centralized global model training with collected data from clients, clients train their local models with their data and send only local model parameters to the server [50]. The server aggregates these parameters and updates global models. This approach allows sensitive data to be kept on local devices and reduces communication costs because clients only share model updates instead of sharing actual data. Peer-to-peer (P2P) or centralized parameter server communication is an option for agents. Hierarchical structures are also an option to consider various model aggregation scales [51].

In federated learning, the network topology is divided into two categories: centralized federated learning and peer-to-peer federated learning. In a centralized network, All clients surround one centralized server and do communication with it. Although training is done in local clients, all clients' model parameters are collected in a centralized server. The server updates the global model with the aggregation method and distributes it to clients. On the other hand, in a completely decentralized architecture, there is no longer a central server functioning as an initiator, coordinator, and model aggregator. Peer-to-peer communication takes the place of communication with the central server. It provides resilience for a single point of failure because in centralized federated learning, if the centralized server fails, the overall training phase degrades; but, in decentralized federated learning, more than one factor plays an important role in the training phase. However, decentralized learning makes global monitoring difficult, so detecting malicious environmental factors is much more challenging. In this research, we focus on centralized federated learning. Topologies are illustrated in figure 2.2.

On the other hand, types of federated learning are divided into three categories: Hor-

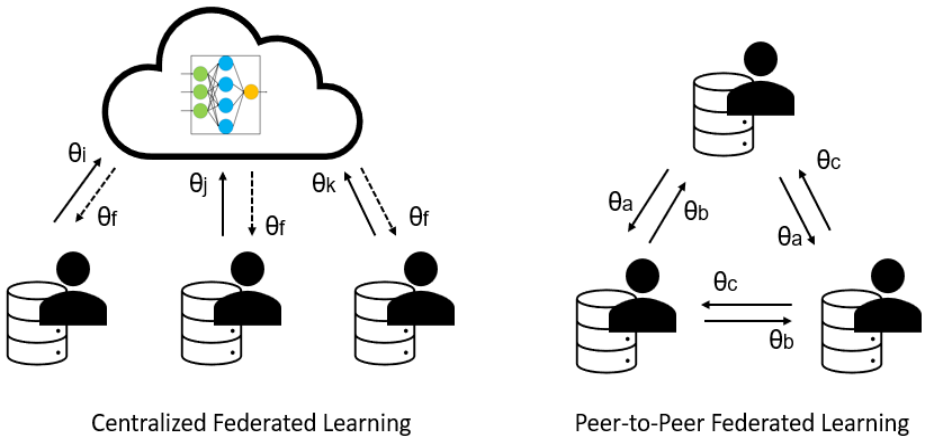

Figure 2.2: Topologies of Federated Averaging

izontal Federated Learning (HFL), Vertical Federated Learning (VFL), and Federated transfer learning (FTL). In HFL, clients have different data samples, but the dataset's attributes are identical. Different markets that sell the same supplies can be given as an example of HFL. Although their datasets are different, their data types are overlapped. In VFL, participants' data samples overlap; however, the data attributes in the samples differ. For instance, there is a bank and an online retailer nearby. The data samples from these two institutions overlap significantly because their user groups include some locals. Furthermore, the bank's data collection characteristics are connected to the user's credit score. Additionally, the qualities of the data gathered by the online retailer are connected to consumer habits. Therefore, these two institutions' data collection methods yield different features. This case is appropriate for the VFL. Finally, In FTL, The concepts of FL and transfer learning are combined. Transfer learning is a machine learning technique that applies information acquired in one domain or task to another domain or task. FTL uses transfer learning to get over the issue of non-overlapping samples, and non-overlapping features [52]. Our project considers the HFL types [53].

2.2 Security in Federated Learning

Federated Learning is nevertheless vulnerable to attacks that might harm the data integrity and the learned model's relevance, although being more privacy-friendly than the centralized method [37]. For instance, a poisoning attack [54, 55, 56] intends to bias the

model by delivering faulty training data. Moreover, since training is done directly, it is not possible the track and verify the data. Also, by using generative adversarial networks (GANs)-based attacks, on the other hand, [57] to recreate data from local node models, user data integrity might also be jeopardized.

Three common attacks in federated learning impact the global model's training phase and reduce its robustness [37]. These attack types are derived from *poisoning attacks*. This attack aims to change or bias the training data, for example, by mislabeling labels or noising inputs. As a result, it creates biases in the global model and makes convergence difficult [58]. This kind of poisoning attack is called *Data Poisoning*. *Model poisoning* aims to directly influence the global model without using harmful data points to be inserted. Since the number of participating clients number large, it is possible that the global model can receive a poisoning model from Byzantine nodes.

The second type of attack uses *Generative Adversarial Network (GAN)* to create poisoned models. PoisonGAN is one example of this kind of attack [59]. The attacker uses GAN and generates a synthetic and realistic dataset. This dataset is under the control of the attacker. To manipulate the parameters of the global model, a GAN is optimized on a specific client side, relying on model updates. Since generated data is realistic, GAN-based attacks are challenging to track. The last attack type is called the *Backdoor Attack*. Instead of aiming to decrease the accuracy of the model overall, this kind of approach produces a very narrow bias that is concentrated on particular labels. In [59], the backdoor attack is used in federated learning image detection tasks where the image classifier is altered to assign the attacker's label of choice. figure 2.3 shows vulnerability and attack types that reduce the performance of the global model of federated learning.

Many approaches are proposed in the literature to overcome these attacks. Since tracking the training phase is not possible in federated learning, researchers focus on following the clients' behavior in the environments to detect the malicious clients and eject them from the server aggregation so that they can not harm the global model training. Their validation accuracy or local model update history can track clients' behavior in each communication round. Another example of defense against the attack is model pruning which reduces the model's complexity by minimizing its size. This kind of defense mechanism is used for backdoor attacks. Therefore backdoor attacks need more work to execute since the resulting model is less expressive. Another approach is to make the global model more durable against malicious clients by reducing their impact on aggregation participation.

Our baseline aggregation approach of federated learning is federated averaging, in short fedAvg. In fedAvg, the global model takes the average of all incoming client model parameters. Although it is a simple approach in FL, It has the same vulnerabilities, such

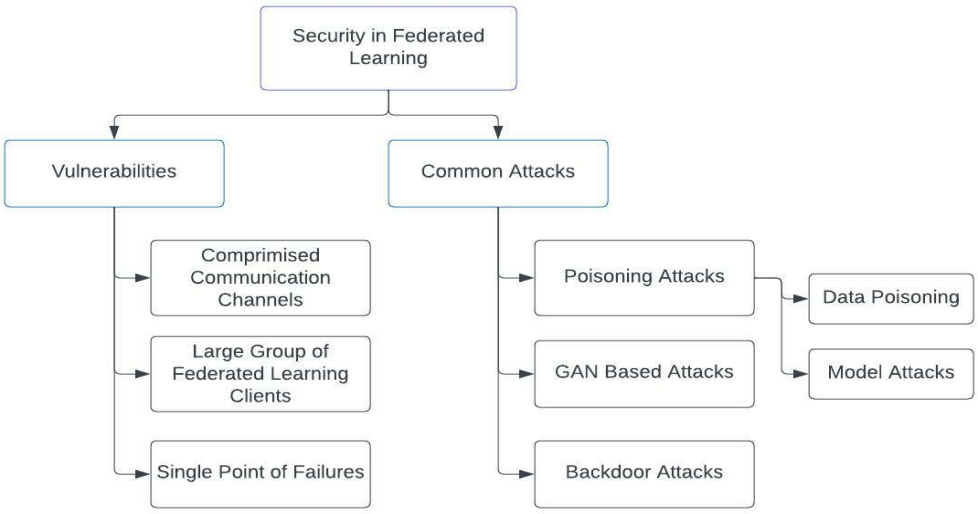

Figure 2.3: Vulnerabilities and Attack Types of Federated Learning

as treating all clients equally, whether benign or malicious. In the following sections, we investigate the related work in the literature for federated learning and security. Also, we look into the non-IID data distribution issue since it is also one of the problems alongside the security in federated learning. Finally, each proposed approach in the following research is compared with the baseline approach.

2.3 Data Distribution Issue in Federated Learning

Moreover, despite being a more privacy-friendly technique than centralized machine learning, the federated learning approach is challenging because end users expect to get good results as much as centralized model training. However, due to the distributed manner of federated learning, clients can get stuck with non-IID data distributions in the training phase [37]. Therefore, it reduces the model accuracy performance because of the need for appropriate amounts and distribution data.

The data distribution of particular local devices in federated learning differs from each other in the non-IID scenario, which is much more realistic than the IID scenario. Particularly for parametric models in horizontal FL, this phenomenon may result in substantial model divergence [60]. Categories of non-IID Data from the perspective of attribute x, la-

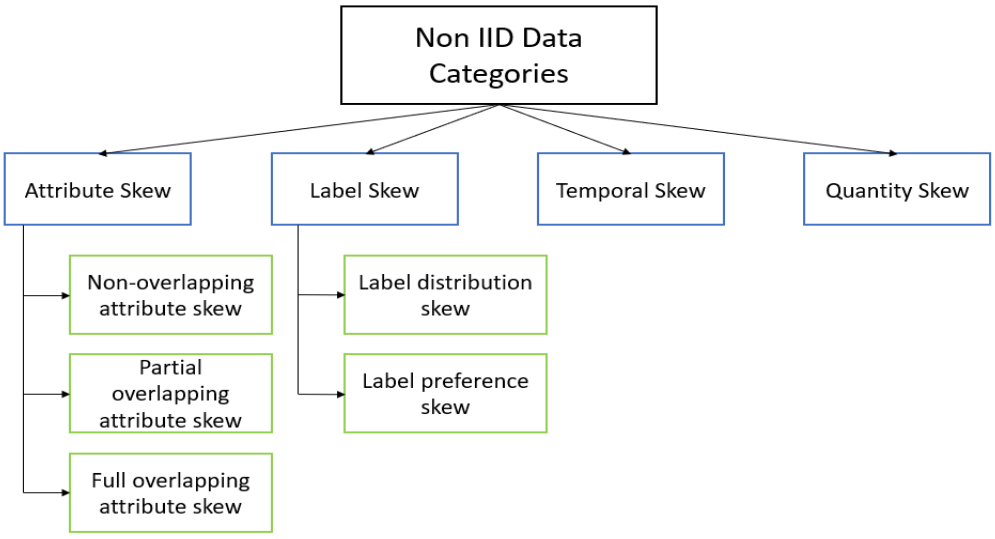

Figure 2.4: Categories of Non-IID Data

bel y, and other more complicated FL scenarios are Attribute skew, Label skew, Temporal skew, and Quantity skew. Non-IID Categories are shown in figure 2.4.

In attribute skew, each client has a varied feature distribution across their characteristics. It is divided into three parts according to their data attributes: non-overlapping attribute skew, partial overlapping attribute skew, and whole overlapping attribute skew. When an attribute skew is non-overlapping, it indicates that the data features across the clients are mutually exclusive. On the other hand, partial overlapping attribute skew, the second category of attribute skew, allows some data aspects to be shared. For example, multi-view images shot from different angles can be an example of partial overlapping attribute skew. Finally, In a Full overlapping attribute skew case, known as horizontal data partition, the whole data is horizontally partitioned among the clients without overlap.

Label skew, the second category of non-IID data categories, represents the situations where the label distribution varies from client to client. It is divided into two parts: Label distribution skew and Label preference skew. Label distribution skew is a common Non-IID category, where label distributions on the clients are different from the other clients in the environments, Label distribution imbalance. On the other hand, label preference skew considers the problems with client data sample intersection that are frequently present in real-world applications where the conditional distribution may change between clients, although attribute distributions are the same. In other words, even if the input data is the

same, the label varies in label preference skew.

Temporal skew is the term used to describe the skew in the distribution of temporal data, such as spatiotemporal data and time-series data, often known as data that has been time-stamped [60]. This type of data accounts for many real-world FL applications that need to be categorized separately. Unlike attribute and label skew, temporal skew refers to the inner correlation of data observations in the time domain. Mainly for time series data, The data distributions on each client are dynamic over time.

Finally, in the Quantity skew, Different clients have different numbers of training data, which can happen in any of the circumstances mentioned earlier. In our project, we consider the combination of the label distribution skew and quantity skew, where each client may hold a dataset with different size and label distributions than the other clients. In chapter 4 and 5, each client starts with IID distributed data. However, after collaboration, their data sizes and label distributions vary after every communication round. Non-IID data distributions cause the global model to fail to converge in the training phase since training is data-dependent in deep learning and machine learning [61]. Distribution prevents the model from receiving well-distributed data.

2.4 Research Related to Malicious Clients and non-IID Problem

Research that considers privacy and security in federated learning mainly focus on detecting and eliminating the malicious client from aggregation so it can no longer harm the global model performance. For example, to identify and track harmful parameters in a blockchain context, the study in [62] suggests a Blockchain-Based Grouped FL Scheme against malicious clients (BGFLS). Since federated learning shame can not judge the client parameter's correctness, millions of clients exist, and many might be malicious, so it is vital to handle this problem for global model performance. They said that the significant drawbacks of Blockchains: are the communication cost and tracking the malicious clients due to many clients. To detect and backtrack malicious clients, BGFLS is proposed where the scheme structure is composed of multiple groups, and the central server connects with each group of clients rather than the clients directly one-to-one.

Another study in [63] assumes that malicious clients try to leak confidential or sensitive information while the data is being gathered. In federated learning, data is not shared with the central server in case of privacy. However, it is still possible to leak data indirectly by clients' model parameters. It can take some information about the client if parameters are

appropriately investigated. Therefore, they propose that Meta Operation Neural Network (MONN) provide efficient and secure federated learning. Privacy is done by encryption, and MONN allows them to manually manipulate encrypted data without reassembling the original data. Although it is not related to degrading global model performance, it also assumes malicious attackers.

In [18], federated few-shot learning is suggested to solve the data shortage. It is known that machine learning suffers from two main problems: data privacy and few data in related fields. Federated learning and few-shot learning are proposed to overcome privacy and data scarcity problems. Data distribution differences are another problem for federated learning since machine learning is a data-driven system that can learn better provided by the vast volume of data. Lastly, this research focuses on malicious client presence that affects the performance while building the model. They propose three different methods: Federated Few-shot Learning (FedFSL), Weighted Federated Learning (WFedL), and Client Selection Federated Learning(CSedL) to solve data scarcity, non-IID data distributions, and malicious client presence by excluding malicious clients, respectively. In the last part of the paper, they combine these methods, namely Client Selection based WFedL (CSWFedL). Our work is mainly inspired by this, mostly in the Weighted Federated Learning part, and method comparison with federated averaging is shown in figure 2.5.

Another study in [64] works on detecting malicious clients in federated learning. Since federated learning is an indirect learning scheme and the global model is constructed by only aggregation, it is impossible to track the training procedure, making learning vulnerable to adversarial attacks by harmful model updates. This research investigates two different attack types, namely targetted and non-targeted adversarial attacks. The purpose of untargeted attacks is to reduce the performance of the centralized model. On the other hand, targeted attacks attempting to change the model's behaviors on a few certain data instances that the attackers have selected. Researchers propose a spectral anomaly detection-based framework that detects the abnormal model updates based on their low-dimensional embeddings by removing the unnecessary and noisy features while keeping the important ones. When unnecessary features from the model are removed, regular and abnormal data instances can be easily differentiated in low-dimensional latent space. A variational autoencoder, unsupervised machine learning, does spectral anomaly detection. It is essential for our project because we apply unsupervised learning to detect malicious clients.

In research [65], researchers study Byzantine Attacks in malicious environments. This attack happens due to federated learning's distributed feature. Malicious clients may create harmful weights by contaminating data in the training phase or directly tampering with the local updates. These harmful weights are used to damage the global model performance.

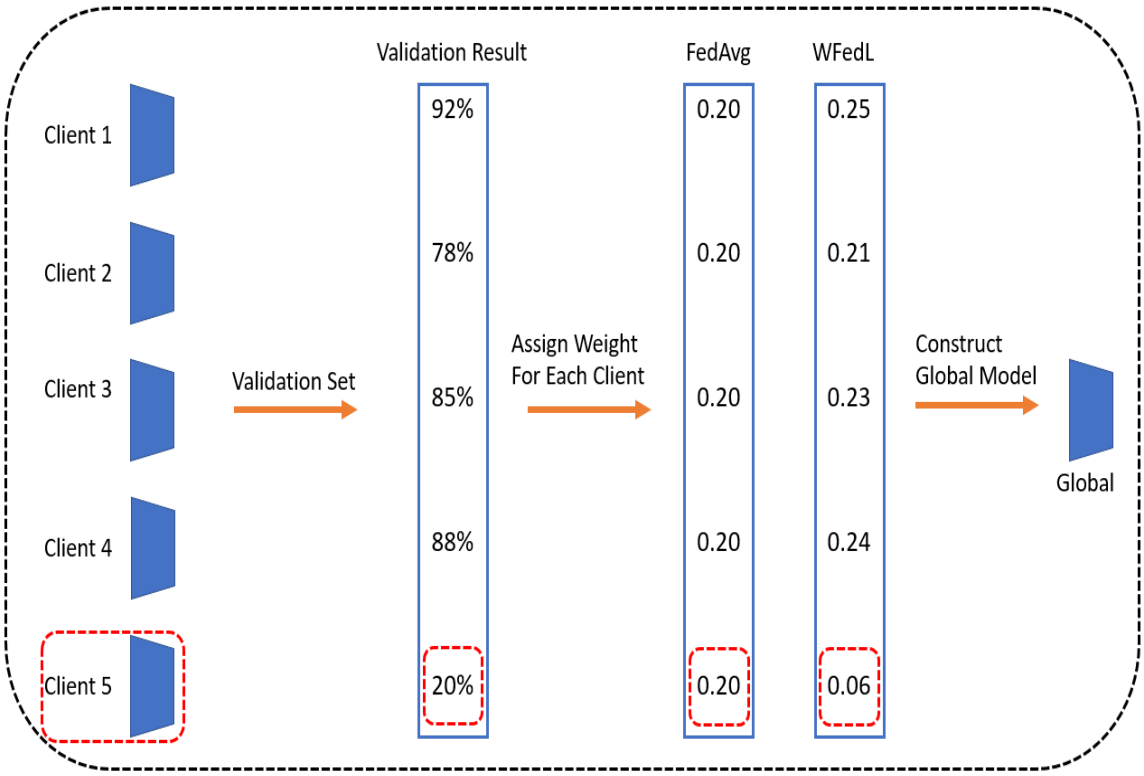

Figure 2.5: Comparison Between Federated Averaging and Weighted Federated Averaging

In order to address this issue, they proposed a multi-armed bandit-based robust federated learning scheme (MAB-RFL). This scheme allows the server to adaptively choose more likely benign updates by modeling the client selection process in federated learning as an extended MAB issue. In each step in the fundamental MAB issue, just one arm is chosen; however, in federated learning, many clients are selected to increase the global model accuracy. Therefore, they provide an adaptive client selection technique to choose the participants in the current round. Also, to evaluate each decision of client selection and identify the malicious clients from Sybil and non-Sybil attacks, they propose two approaches based on which rewards can be easily computed.

In [24], researchers focus on client selection strategy to discriminate the malicious clients from the federation learning aggregation phase. There are three kinds of clients in their environment: IID data distributed clients, Non-IID data distributed clients, and malicious clients. Non-IID and IID data-distributed clients are benign clients that can train their models properly. Malicious clients are suffered from the targetted adversarial attack, which aims to control a model that will be trained on attacker-signal data that leads to calling this attack a backdoor. This attack makes the centralized global model learn features related to the main and backdoor tasks. The attacker generates backdoor samples with unique features like drawing dashed lines on the image. This generated samples map to the wrong labels. Since data access is restricted in Federated learning, it is not easy to distinguish if data is malicious or benign. Therefore, researchers present a framework for discriminating malicious clients from benign clients. In this framework, an innocent client selection module inspects clients' weights with PCA clipping, ReLU clipping, and Norm clipping. The malicious clients can be excluded at this point with a thorough inspection. Then, the global server receives just the parameters that have been deemed to be benign for aggregation.

In [66], researchers consider poisoning attacks in federated learning in which malicious clients send manipulated model updates to the server for aggregation. The distributed nature of FL makes it susceptible to model poisoning attacks. Model poisoning attacks can be classified as targeted and untargeted. Many malicious client detection algorithms depend on the clean server validation dataset. On the other hand, they propose a new method to detect malicious-client called FLDetector. They suggest this procedure to address the limitations of current detection techniques, like the need for clean validation datasets. Moreover, FLDetector can be combined with Byzantine-robust FL methods to improve the model resilience in a malicious environment. The critical insight is that, while malicious clients craft the model updates rather than following the FL algorithm, benign clients calculate their model updates based on the FL algorithm and their local training data. As a result, a malicious client's model updates are inconsistent across iterations.

By examining the consistency of their model changes, FLDetector detects malicious clients based on intuition. Consequently, based on previous model updates, this method offers the server to predict each client's model update in each iteration.

Moreover, the authors in [67] also tackle poisoning attacks towards federated learning of object detection tasks. To this end, three poisoning attacks are considered: label poison attack, which causes a misdetection label; bounding box poison attack, which misaligns the location and size of the bounding box; and objects existence poison attacks. The defense mechanism against poisoning attacks is to separate poisoned gradients from benign gradients via spatial signature analysis.

Federated learning with object detection tasks is searched in [68]. This paper also considers the malicious and Non-IID data distributions and offers the FedDetectionBC framework for object detection in the blockchain. Since Data sources such as clients participate in model training, each client can influence the training process. Non-IID data distributed or malicious clients using poisoning attacks may harm the training process, even prevent it from converging. In order to apply federated learning to object detection while resolving the security and data heterogeneity issues in conventional federated learning, they propose FedDetectionBC. This framework incorporates federated learning, Ethereum blockchain, smart contract, and interplanetary file system (IPFS). They also try to use a different system architecture rather than the centralized architecture of conventional FL problems since a single point of failure may break the whole system, and the training will fail. In order to address the single point of failure issue, aggregators and verifiers are used in place of the central server in this study. Verifiers serve as an extra security measure, and local models are accepted if they get approvals from verifiers. On the other hand, IPFS stores model files, and smart contract offers the required functionalities for the training process. FedDetectionBC ensures traceability and auditability, In federated learning. For non-IID data distribution problems, they suggest Exchange-FedAvg for minimizing the effect of data heterogeneity by allowing local models to exchange to learn from different datasets before aggregation. This paper is an excellent example of a computer vision task in federated learning in non-IID and malicious environments.

Another researcher is also working on federated learning in malicious and Non-IID data distribution environments in [69]. Although federated learning provides security and prevents direct leakage of data privacy, it encounters potential severe attacks because it has weak control over the clients. Their environment assumes non-IID distributions, making tracking malicious clients difficult. In non-IID distributed environments, even the average gradients uploaded by the clients might be inconsistent when we use the inconsistency of gradients factor to identify malicious clients. The robust algorithm, in that case, mistakes regular clients for malicious clients. To handle these issues, they propose a new FL workflow

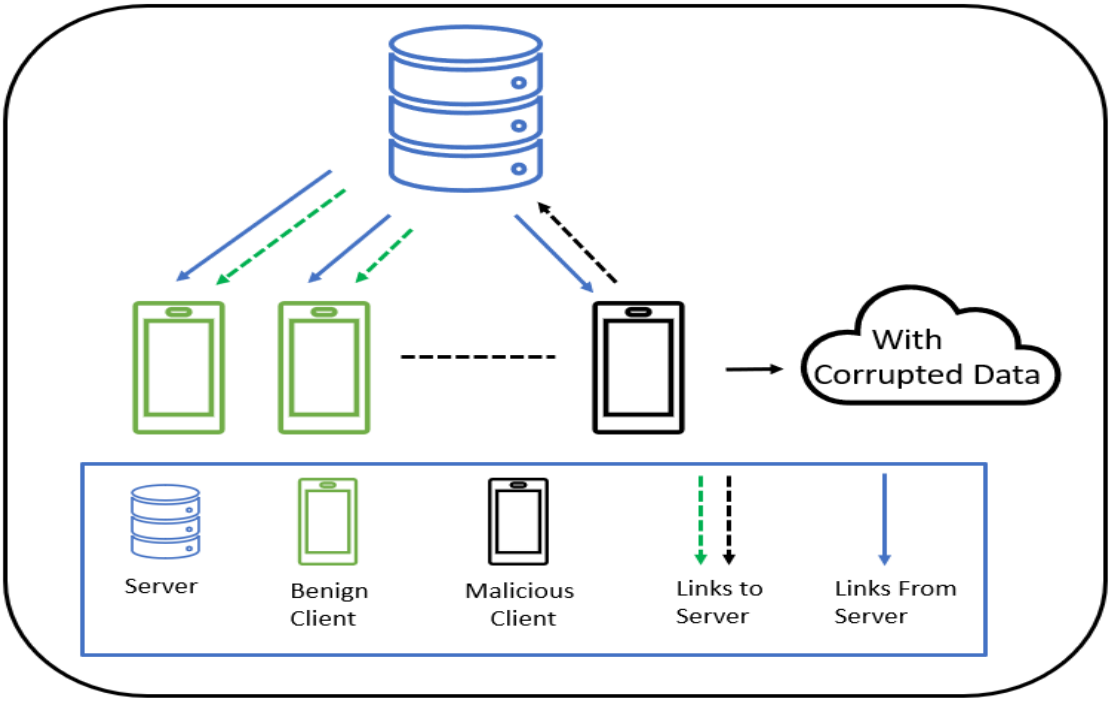

Figure 2.6: Malicious Environment in Federated Learning

named Cominer. First, they suggest the Label Cluster (LC) approach to address the low accuracy issue by non-IID. With robust algorithms applied in FL, LC can group clients with comparable data labels into a single cluster to save more regular clients. Second, to achieve robustness, Vertical Comparison (VC) is suggested to exclude malicious clients by looking up their gradient distance between adjacent rounds of each client. Our work also uses clustering methods to detect malicious clients from the client pool.

The clustering algorithm is also used for clients with mislabeled corrupted datasets in [70]. In that research, it is assumed that there is incorrect data labeling as in the real world. Faulty labels that map false input cause poisoned weight updates, negatively impacting global model performance. In the malicious environment, clients with mislabelled data generate abnormal weight compared to benign clients with the perfect dataset. Also, When malicious clients predominate, it might be challenging to distinguish corrupted weight updates. Additionally, zero-weighting the contribution of corrupted clients makes the model difficult to converge, even producing biased global models, since the contributions of both the correctly labeled and incorrectly labeled dataset are deleted in the aggregation phase. Therefore, they develop a novel robust FL solution that deletes the possible fault-labeled

dataset on local clients. They propose two methods. Firstly, it encodes the local data to low-embedding embedding using metrics learning. Secondly, Voting to identify potentially incorrectly labeled datasets and eliminate them from the local training phase. K-Means clustering is used in our project but for malicious client detection. The malicious environment that contains malicious clients sending their harmful data generated in the training phase with mislabeled data is illustrated in figure 2.6

Mislabeled problem is also considered in [71]. Since the labels on the data samples are manually given, they may be inaccurate. Also, treating all labels equally in existing federated learning methods aggregation, such as federated averaging, leads to degradation of performance due to wrong labels. Due to the learned models' overfitting of the label noise, generalization performance is poor. To locate incorrectly labeled data, they suggest a learning-based data cleaning process. Instead of simply relying on servers' pure validation dataset, their research takes a step forward and handles the noisy label problem as a bilevel optimization problem. This method is divided into two categories: Implicit Differentiation (AID) and Iterative Differentiation (ITD). They provide two new, practical compression-based algorithms to reduce communication loss: Iterative and Non-iterative algorithms. The lower level problem is roughly solved first using ITD methods, and the hypergradient is then calculated using backward (forward) automatic differentiation, whereas AID techniques only roughly simulate the hypergradient. Handling mislabeled data is essential since our corruption depends on incorrect labels in clients.

2.5 Research Related to Non-IID Data Distribution Problem

On the aggregation side, in [20], In order to increase the resilience of the centralized model, regardless of the distribution of the data, researchers suggest precision-weighted federated learning, which makes use of the heterogeneity of the data by penalizing the model uncertainty at the client level. Precision-weighted Federated Learning algorithm is another variance-based averaging scheme that aggregates model parameters across clients. It is expected to penalize model uncertainty at the client level to increase the centralized model's accuracy, Irrespective of how the data are distributed, whether IID or Non-IID. Their method also computes the weighted average at each communication step using the uncentered variance of the gradient estimator from the Adam optimizer. Although they do not concern the malicious clients in the environment, proposing precision weighted aggregation and dealing with non-IID data is inspirational for our project.

In [21], researchers study to improve the robustness of federated learning in the "forgetting event" case. It happens in a continuous learning setting; when a model fails a task, they have already mastered it after learning a new task. For example, if the model is to learn two consecutive tasks, the model performance on the first task degrades after it learns the second task. A forgetting event occurs when a model loses its prior understanding of some data during the training. Unfortunately, forgetting event cases is seen more often than traditional centralized learning. Therefore, they propose a new aggregation method called federated weighted averaging, FedWAvg. According to the number of forgettable examples in each round, this approach reorganizes each client's weight. They plan to gain better performance in highly unbalanced data with a new aggregation method.

Another research [22] uses the model personalization technique to deal with heterogeneous non-IID data distribution in the environment. They suggested a novel federated averaging method called WAFFLE, Weighted Averaging For Federated LEarning, as a new personalized, collaborative ML algorithm. This model is suggested to prevent client drift and quicken the convergence to the ideal personalized model. In addition, it aims to reduce the loss from the personalized model on the unique agent of interest. This approach is inherited from two different approaches, Weight Erosion and APFL, two personalized federated learning methods.

Additionally, in [30], researchers show new reputation-based aggregation methods to increase the accuracy and reduce to converge time of the global model in non-IID scenarios. Therefore, the contribution of local clients' models is evaluated by their reputation scores, which are calculated by three performance metrics. Reputation-based participation is our main inspiration for score-based aggregation in this project.

2.6 All Aggregation Methods

This project mainly uses different aggregation algorithms to overcome malicious and non-IID problems. All aggregation methods from previous sections 2.5 and 2.4 are given in table 2.1.

According to table 2.1, many new aggregation methods are used for heterogeneity and non-IID features. Research [18] and [68] also work in the malicious environment; however, the aggregation method proposed in [68] is for Handling non-IID data distribution problems. Moreover, just like us, some research does not only offer new aggregation methods but uses this new algorithm with other methods. For example, in [18], they also apply federated few-shot learning and client selection.

Table 2.1: All Aggregations Methods in Related Works

Research	Aggregation	Method	Dataset	Accuracy (%)
[18]	Client Selection Based WFedL (CSWFedL)	Assigning different weights to different clients models by the validation result	CIFAR-10 CIFAR-100	81.39 50.15
[20]	Precision-Weighted Federated	Aggregating the weights from each client into a globally shared model by averaging the weights by the predicted variance's inverse.	MNIST CIFAR-10 Fashion-MNIST	98.5 59.4 86.5
[21]	Federated Weighted Averaging (FedWAvg)	Reorganizing each client's weight according to the number of forgettable examples in each round.	SVHN CIFAR-10	92.37 80.83
[22]	Weighted Averaging For Federated Learning (WAFFLE)	Reducing the loss from the personalized model on the unique agent of interest	MNIST CIFAR-10	99.69 91.40
[30]	Reputation-Based Aggregation Methods	Contributing models from local clients are evaluated using their reputation scores derived from three performance measures.	MNIST Fashion-MNIST	98.68 92.83
[68]	Exchange-FedAvg	Enabling the sharing of local models to gain knowledge from various datasets before aggregation	VOC dataset	71.64

We apply a score-based aggregation method that considers each client's validation scores calculated by unseen data. Although much research work on federated learning in malicious environment, non of them consider if malicious clients enter aggregation by passing the elimination method. Also, weighted federated aggregation methods are used chiefly to overcome the non-IID data distribution in the environment. Moreover, making one-to-one communication and sharing data is another challenge not considered in previous research. It causes non-IID data distribution and spreads malicious factors among benign clients. Although we use a similar weighted averaging method in this research, we also observe the

malicious rate's effects on the aggregation and how effective score-based aggregation is in case malicious clients enter the aggregation and non-IID distributions. The score-based aggregation method's main aim is to mitigate the effect of malicious clients even if it enters the aggregation. This method is done by looking up their validation scores. After we ensure the effects of the SBA method, we move next step, which is eliminating malicious clients.

Validation scores are also used in research [18] for assigning weights to different local models. In research [20], Precision-Weighted Federated, which aggregates the weights from each client into a globally shared model, is applied. Moreover, in [21] primary purpose is to deal with forgetting events, which is done by Reorganizing each client's weight according to the number of forgettable examples. In addition, research [68] calculates reputation scores for each client and lets clients join the aggregation by considering these scores. On the other hand, [68], instead of arranging weight for each client for aggregation, allows clients to exchange information from various datasets before aggregation so that it would overcome the low accuracy and data scarcity problem. Finally, WAFFLE in [22] is proposed for heterogeneous non-IID data problems, and it minimizes the loss caused by the personalized model on the particular agent of interest. All research datasets and their test results are also shown in 2.1. We apply Fashion-MNIST dataset for federated learning training and testing which is used in research [20, 30].

Chapter 3

Score-Based-Aggregation in IID Environment

In this section, we first describe Score-Based-Aggregation (SBA) and its advantage of federated averaging (fedAvg). Then we describe our new score-based aggregation method and its differences from fedAvg. Following this, We describe our system and model. Then we construct the clients' local and server's global model and investigate its architectures. Next, we create our simulation environment by NS3 and apply NS3-AI for deep learning. Finally, we train our federated learning in the fedAvg and SBA methods and compare the results.

3.1 Score Based Aggregation

Score-Based-Aggregation is proposed for overcoming non-IID and malicious clients issues of federated averaging. In baseline federated averaging, all clients participate the aggregation equally, which means that we consider malicious clients and benign clients, or clients with extensive data and clients with fewer data, as the same. Therefore treating each client the same causes trouble with aggregation and reduces the aggregation. That is why we need to increase the impact of benign clients in the aggregation while decreasing the effect of benign clients.

Each client's federated aggregation participation rate is dynamically configured by the score-based federated averaging. Thus, when the server receives client weights, it validates the client model with unseen IID distributed data held only by the server. The unseen

data is randomly allocated from the server repository of many unseen data. So each client receives different IID data and calculates their validation scores. Each client's validation accuracy is mapped onto a participation rate for that client, and it takes value from the range of [0,1]. Given P_i denoting the participation rate of client i, $\sum P_i = 1$ whereas the participation rate of each client is calculated as follows:

$$P_i = \frac{V_i}{\sum_{j=1}^{N}(V_j)} \tag{3.1}$$

Where P_i and V_i are the participation rate and the validation scores of the client i, respectively, the Score-Based Aggregation Algorithm obtains the weights of the aggregated model in the server ($W'global$) using these normalized participation rates as follows:

$$W'_{global} = \sum_{i=1}^{N}(W_i) * P_i \tag{3.2}$$

Consequently, the impacts of malicious clients with corrupted weights degrading the aggregation are reduced. As a result, the convergence of the global model after the aggregation is maintained for a while, leaving sufficient time to detect malicious or corrupted clients in the network.

3.2 System Model

Federated learning is done in distributed, decentralized environments [72]. Unlike centralized training, clients hold their data and send only model updates to the server [73]. Each client does training, and the server aggregates and updates the global model in the final step of each iteration. The ideal scenario for federated learning is one in which clients are given access to IID-distributed training data, and there are no malicious or communication-related environmental concerns. Federated learning can therefore conduct proper training and generate satisfactory results.

However, in a real-world scenario, these cases are not possible. Environmental drawbacks always exist, such as adversarial attacks or Non-IID distributed data. The system model considers the coexistence of malicious and benign clients in a federated learning environment. Some clients in the system suffer from *data poisoning*, which is harmful to the training steps of clients [74]. Data poisoning is an adversarial attack that corrupts clients'

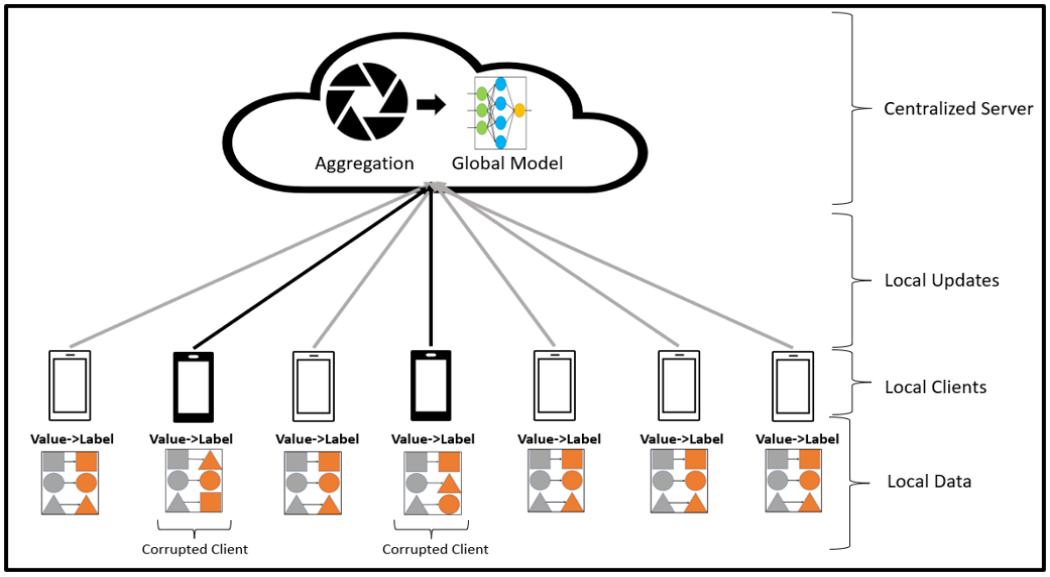

Figure 3.1: Malicious Federated Learning Environment.

training data and degrades the training performance [75, 40]. To emulate data poisoning on clients, instead of teaching clients the wrong label for one class, client data are corrupted by shuffling l out of $\mathcal{L}$ of its data labels. Therefore, one input class in the data is matched to more than one label. Thus, their validation accuracy can not reach more than a certain level. For instance, when a client's data is corrupted, malicious clients' accuracy ranges between 20% and 35% while benign clients' are between 88% to 92%. Consequently, the global model suffers from a model poisoning attack by receiving those updates [66]. The malicious environment is given in figure 3.1

In this research, we have N clients under one central server. Each client starts with the same amount of IID data. However, there are p proportions of malicious clients where their k the number of data labels are shuffled, so in their training set, not every input data is matched with its corresponding label. As a result, p value changes from 0.2 to 0.5 for each scenario. In this case, the centralized server receives corrupted weights from malicious clients that degrade the accuracy of the global model after aggregation. That is why we apply the SBA method to aggregation to reduce the impact of malicious clients' weights.

3.3 Client's Model

The expected task for the client's and server's model in this work is to make image classification, which is an essential task in many areas, such as intelligent transportation systems [76], with federated learning in a distribution environment. Image processing tasks in federated learning are also researched in literature such as [77], which is CNN-based driving action recognition methods. Those studies are also proposed for privacy and security issues.

Table 3.1 lists all clients' model parameters. Each local client and global model uses Convolutional Neural Network (CNN) architecture [78, 79, 80]. The CNN model, therefore, includes three convolutional layers and three linear layers. Convolutional layers use ELU [81] activation function while linear layers use *tanh* [82]; since *tanh* limits the output to a range from -1 to 1, it is useful for preventing an excessive number of results and gradients from vanishing. On the other hand, ELU (Extended Linear Unit) is an activation that is used as an alternative for ReLU (Rectified Linear Unit) activation function. Except for negative inputs, ELU and RELU are quite similar. For non-negative inputs, they are both in the form of an identity function, $\{f(x) = x\}$. In the case of negative inputs, as opposed to RELU, which smoothens sharply, ELU becomes smooth gradually. Unlike ReLU, ELU can produce negative outputs. Activation functions is shown in figure 3.2

Table 3.1: Environment Variables & Initial Parameters in IID Environment

	IID Environment
Malicious Rate [%]	{20,30,40,50}
Malicious Label Shuffle	80%
Initial Data Instances	600
Dataset Name	Fashion-MNIST
# Convolutional Blocks	3
# Linear Blocks	3
Optimizer	Adam
Loss Function	Cross entropy
Learning Rate	0.001
Activation Functions	ELU (Conv.), Tanh (Lin.)

Each convolutional layer is followed by batch normalization [83], and maxpooling2D layers [84]. These layers normalize the output of each convolutional layer and reduction of

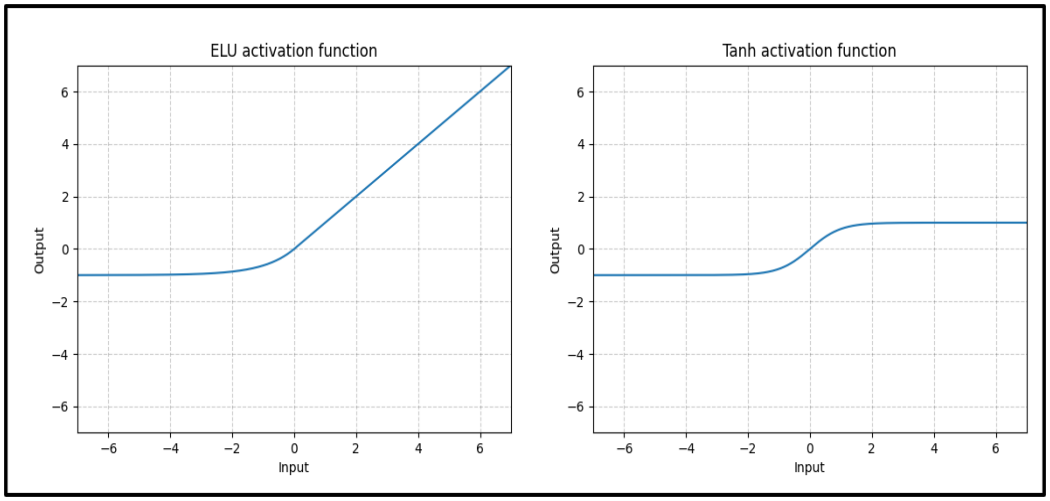

Figure 3.2: ELU activation function & tanh activation function

the size of the inputs. This paper refers to layer groups consisting of a convolutional layer, maxpool2D layer, and Batch normalization as Convolutional blocks. Also, the layer group consisting of only the Linear Layer is called Linear Block. The global model and each local model consist of 3 convolutional groups and three linear groups; overall architecture is illustrated in Fig. 3.3

We choose the Adam optimizer [85] to update the CNN model's weights and parameters to reduce the error value. According to research, the Adam method for efficient stochastic optimization optimizer provides improved training results for CNN classification applications [86]. We set the learning rate to a minimum value, which is 0.001 since learning is intended to be federated. Each client has a relatively small volume of data (determined empirically after hyperparameter-tuning).

Cross entropy loss is selected as the loss function as it is a standard loss function used in multi-label classification [87, 88, 89]. The loss function is a function that shows how efficiently the model performs after each epoch. Greater loss values mean the model needs to be optimized better and produce relevant results. In deep learning and machine learning, the objective is to minimize the loss of values through proper training and better parameter selection [90, 91]. Cross-Entropy loss is a significant lost function. It is used to optimize classification models. The cross-entropy loss function is applied after the softmax activation function in the last layer. Formulation of the Cross-Entropy loss function is given in 3.4

Since horizontal (parallel) federated learning is assumed, such as in [18], all client models

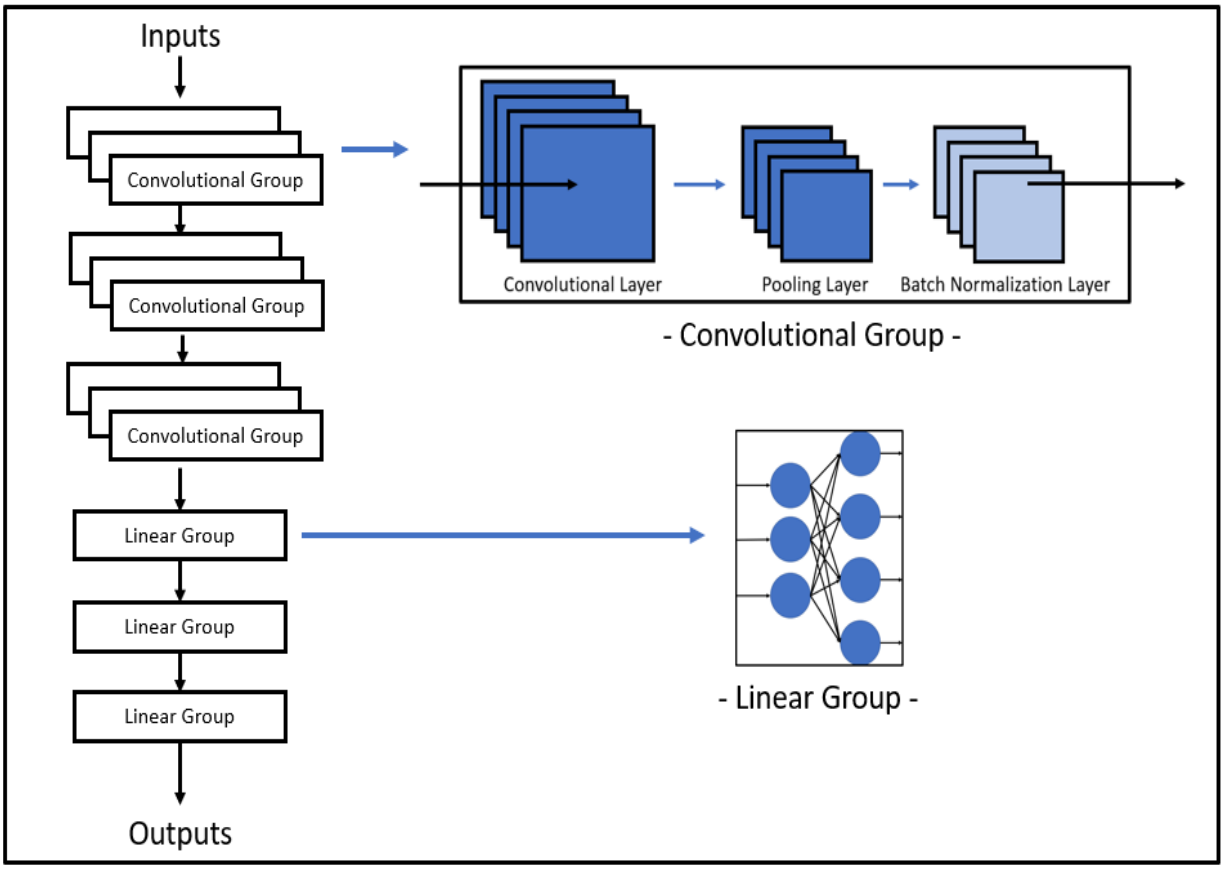

Figure 3.3: CNN model and Convolutional & Linear Groups

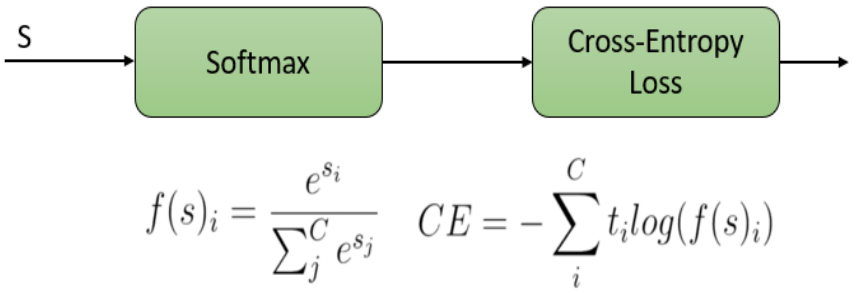

Figure 3.4: Cross-Entropy Loss Function

and the centralized server's global model must have the same deep learning architecture. The server model is initiated by using randomly created parameters. When each client synchronizes with the server, it receives the centralized CNN model at the beginning of each communication round.

3.4 Simulation Environments with NS-3 Simulation

We create our federated learning environment, including clients, servers, and communication links throughout the *ns-3* simulation. ns-3 is a discrete-event simulator typically run from the command line [92]. It is written directly in C++. It is primarily intended for use in research and education. ns-3 provides simulation engines for users to run virtual experiments as well as simulations of how packet data networks function. It is designed as a set of libraries, and In order to conduct research that is challenging or impossible to do with existing systems, ns-3 is used. In this section, we described the network architecture of our environment and communications. Then we explain how we apply deep learning [93] and AI modules in the system.

3.4.1 Basic NS-3 Simulation

In ns3, the primary computing device is called a node. The node can be imagined as a computer or a host device that connects to a network. In this simulation environment, nodes represent clients' devices and a centralized server. We have N number of client nodes connected to one centralized server node. We use *NetAnim* visualization tools for visualizing simulation, and the output is shown in figure 3.5. The node in the leftmost corner stands for the server, and the rests stand for clients.

At the beginning of the scenario, the server, standing left corner of the figure, shares the initial global model with each client. After receiving the global model, clients start training the model with their local data. Then all clients send back the updated global model to the server. Once the server receives all global models, it starts aggregation and generates a new global model. Finally, this global model is distributed among the clients. Communication is done by 802.11p communication standards [94, 95]. While we set the parameters of the physical layer, we apply the OFDM rate of this communication to 27 Mbps and Bandwidth to 10 MHz. Finally, our bit rate is 27 Mbps.

Although we select vehicular communication, we do not apply mobility to the clients since it is the basic scenario. The main aim is to find and observe better aggregation

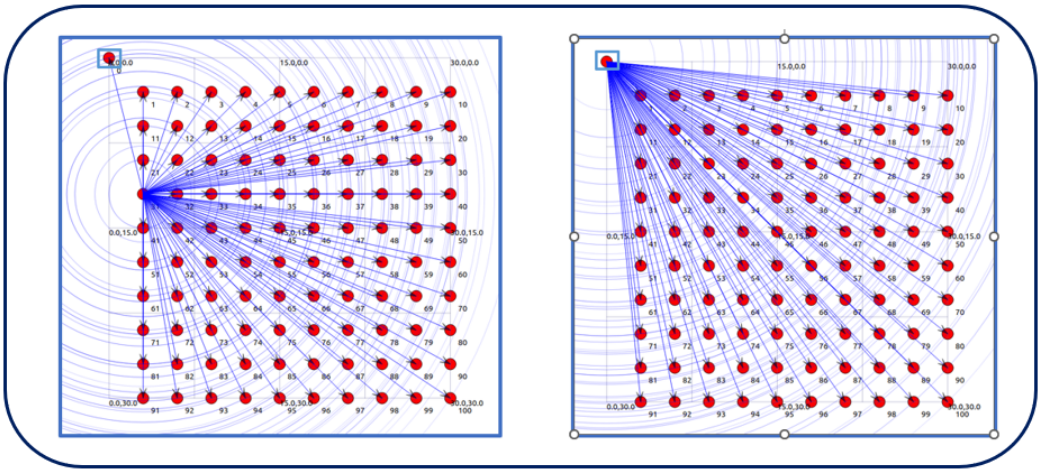

Figure 3.5: NetAnim Simulation Environments

algorithms in malicious and collaborating environments. On the other hand, even if there is mobility in the environment, if clients do not get far from the server, they continue to send their packets, and we do not consider other base stations and hangover situations; it is assumed that the server has a wide coverage area that covers all clients. Therefore, mobility does not affect the system.

Figure 3.5 shows the one state in the simulation environment where in the left part of the figure, the server gives an update to 30^{th} client, and that clients inform the others. All clients send their updates to the server in the right part of the image. Each update contains parameters of the client's model after local training, and they are sent by k number of packets. We use WAVE in the system architecture for wireless-based communication specified by the IEEE.

For inference time, on the other hand, if model uses GPU for training, model parameters sending and training take around 18 seconds taking into account that small dataset for each client. Model aggregation and model distributions take approximately 10 seconds. If considering 100 clients in the environments, 1 communication round lasts 40 minutes. However, if GPU is not used, model training takes 36 seconds and total 1 communication round duration extends to around 1 hour. In this research, we select number of communication round is 20.

3.4.2 NS-3 AI

With the increase in research on ns3 simulation, more and more researchers are willing to apply artificial intelligence tools and algorithms to network research to increase efficiency and performance. However, These two parts are developed independently and extremely hard to merge, so connecting these two tasks with data interaction is more reasonable and convenient. As a result, *ns3-ai* module comes into the picture. it is Inspired by ns3-gym, but using a different approach which is faster and more flexible. In order to facilitate data connection between ns-3 and other python-based AI frameworks, our module offers a highly effective solution. This module provides a Python module that enables AI to interconnect. The framework of this module is shown in figure 3.8 [92].

Ns3-ai does not provide any AI algorithm, but it provides a Python module that enables ns3 to interconnect with the AI module. AI and Python frameworks for model training and data processing, such as PyTorch, sklearn, matplotlib, OpenCV, and so on, need to be separately installed. All it takes is to download the ns3-ai project and use it to exchange data between ns-3 and the AI algorithm. Data received by the ns3 framework is called **environment data**, and data received by the AI framework is called **action data**.

AI frameworks and the ns-3 simulator operate in separate processes in this system. Communication between frameworks is done through data transfer, and it is required to send data to the AI model for training and to test the model in the ns-3. The network and topology are configured using the ns-3 simulator, which produces data for the training of AI algorithms. The AI frameworks include methods that enable us to train models with data from ns-3 and then send the trained model's data back to ns-3 for testing. In brief, ns3 sends data to the AI framework as environment observations, and ns3-ai sends updated model's parameters and predictions in return. illustration is shown in figure 3.7.

Data transfer between frameworks is done by *Shared Memory Pool*. Both sides can access the memory and control mainly in ns-3 because it is convenient and straightforward to operate memory in C++ directly. The structure of the shared memory pool is shown in figure [92]. The memory blocks, control blocks, and main control blocks are the three components that comprise the shared memory pool in continuous memory. The main control block oversees the whole memory pool and changes its version after any modification, such as requiring a memory block that takes place in the memory pool. A new control block will be automatically formed in the memory pool's tail for each new memory block added from the head since there is a one-to-one relationship between the two types of blocks.

The control block includes information such as the *size* and the *address* of the memory block. It is readable but cannot be changed until the free block. The memory block

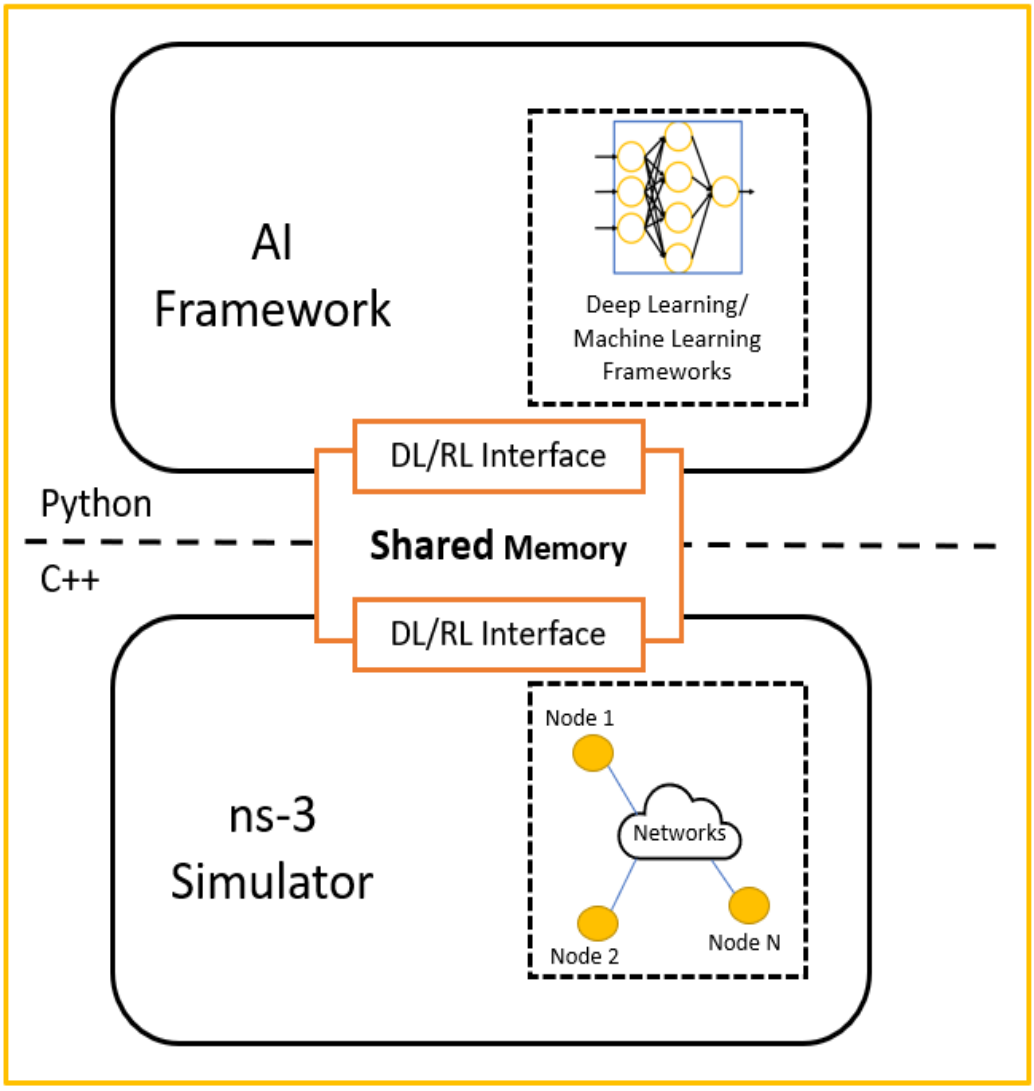

Figure 3.6: NS3-AI System Architecture

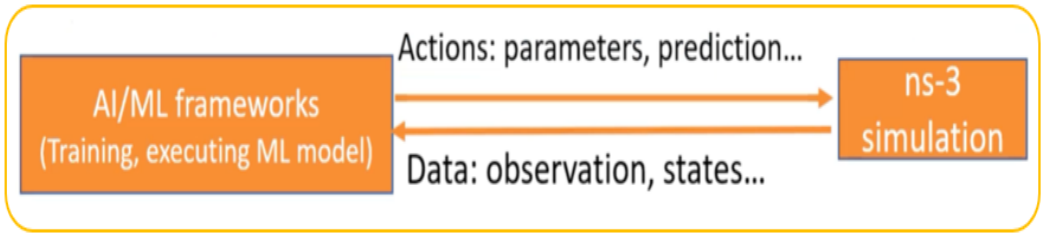

Figure 3.7: Data interactions between NS3 and AI/ML-Framework

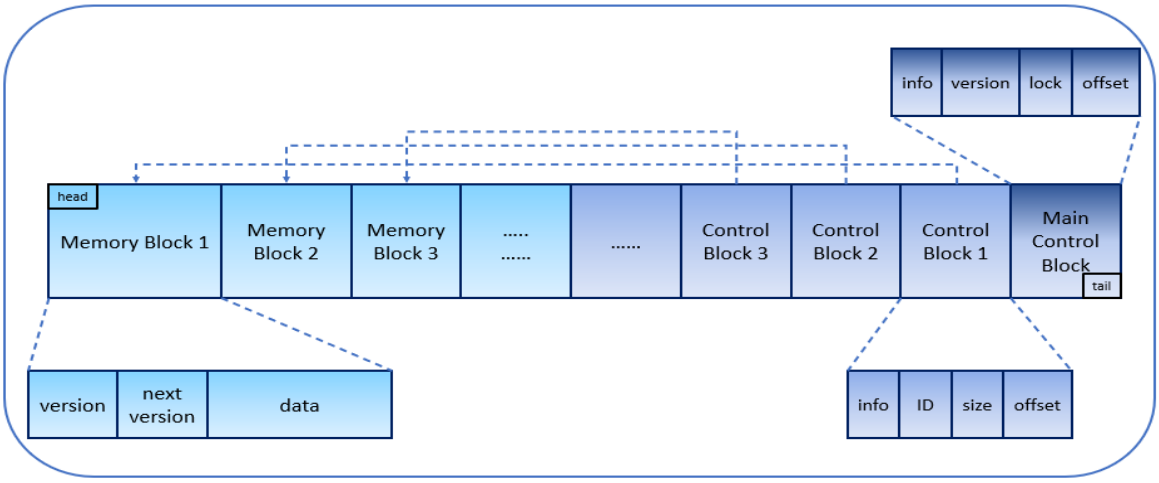

Figure 3.8: NS3-AI Memory structure

is readable and writable memory with the *lock indicator*. For every process that wants to access or modify the data, the lock indicator compares the version variable and the nextVersion variable. If they match, it indicates that the memory is accessible. After that, it will atomically add +1 to the following version to lock the memory and +1 to the version after its operation to unlock the memory. We got memory pool keys and memory block keys. The memory block key needs to keep the same in the ns-3 and ns3-AI.

Each client in this simulation shares their data with the ns3-AI module throughout the shared memory pool. It starts local training by the Python module by using *pytorch* framework [96]. After training is complete, the AI module sends the updated module to the clients, and in the ns3 simulation environment, clients send their model updates to the server. All client models are aggregated inside the server, creating a new global model. In the last step, the server distributes the updated global model to the clients for further local training. This process calls *Communication Round*, and it continues until the global model converges. The environment parameter is given in the table 3.2

Table 3.2: NS3-AI Python Parameters

	NS3-AI Parameters
Memory Pool Key	1234
Memory size (bytes)	4096
Memory Block Key	1294
# Clients	100
Learning Rate	0.001
Initial Data size	600
Client Validation Data Size	1000
Server Validation Data Size	2000
# Communication Rounds	40
# Local Epochs	20
Batch Size	64
Input Size	32x32

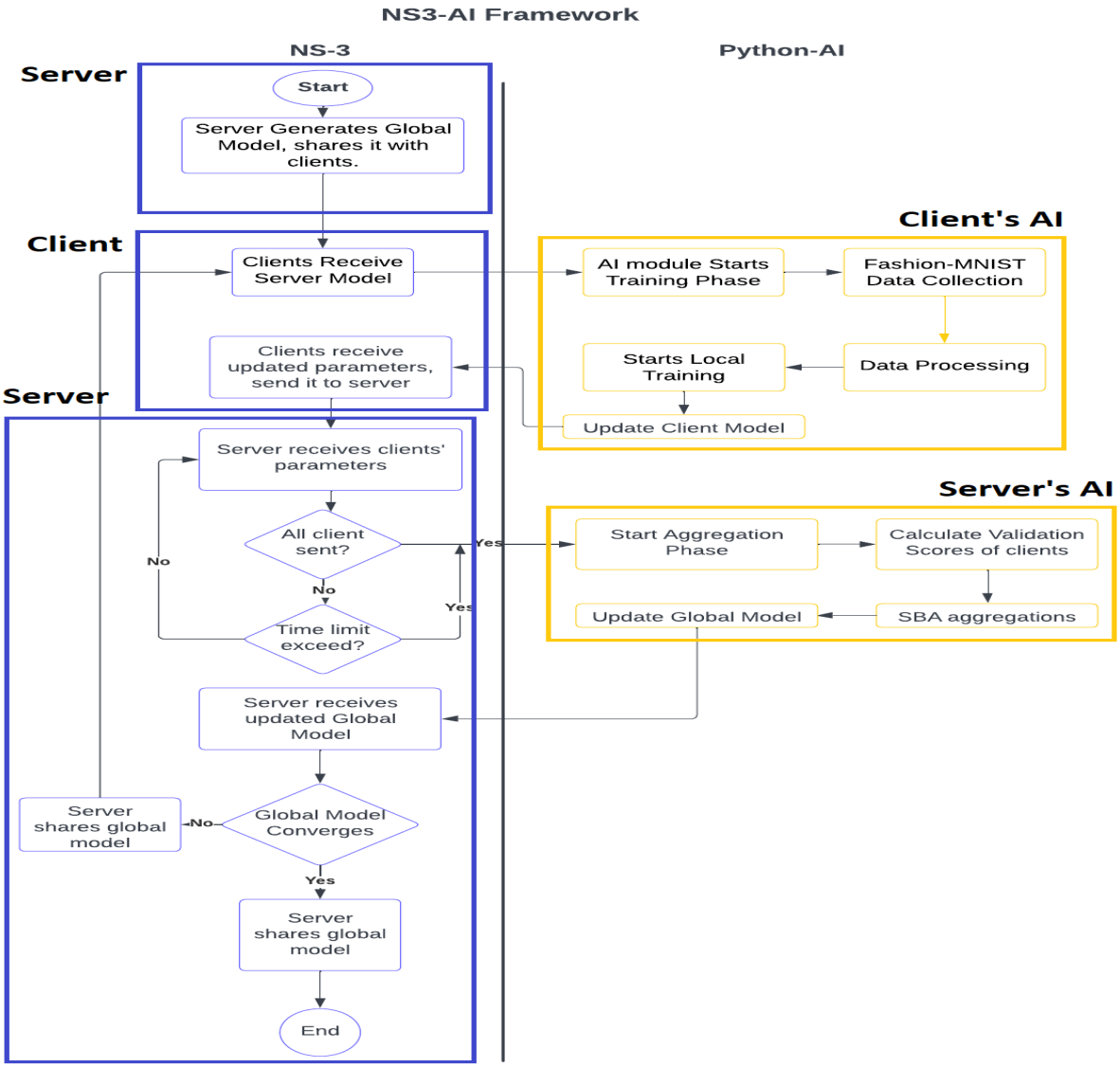

Figure 3.9: Flowchart of Federated learning with SBA Aggregations

3.5 Flow Diagram of Training

The flow diagram of federated learning with SBA aggregation is given in figure 3.9. The flowchart is split into client, Server, Client's AI, and Server's AI. The system is built into the ns3- AI framework; the left part shows the ns-3 simulation where all the networking applications and communications are made. The right part of the figure shows the python part of the system where data collection and processing, AI module creation, training, and updates are made. The flowchart starts with the initialization of the global model and ends when the global model converges. The server part takes place in both the initialization and finalization of the diagram. There is no connection between the server's and the client's AI part.

3.6 Dataset: fashionMNIST

We use the Fashion-MNIST dataset in [97] for model training and testing. It is a grayscale image dataset having 60.000 training and 10.000 test data with sizes of 28×28. It has ten classes: T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. In the initial round, 60.000 images are reshaped to 32x32 and distributed among the clients equally. A global server holds 10.000 test datasets That are used for validation of the client's model and the server's global model.

3.7 Experimental Results

In this section, we start federated learning with an IID distributed environment. We compare our results with both SBA and Baseline Federated Averaging aggregation methods and show the advantages of our proposed method. There are four different cases for this simulation. In the first case, we assume that 10% of the environment contains malicious clients. In each case, the malicious rate increases to 10%. We first investigate the benign client's behavior in the system by looking at their loss values; then, we compare the SBA and FedAvg aggregation methods by the centralized server's global model.

3.7.1 Client's Losses

In this part of the thesis, we investigate the benign clients' behaviors that achieve the worst, best and average validation accuracy scores. We collect their loss values per local

Figure 3.10: Fashion MNIST with Labels

epoch and draw the plot line of the first and last communication round's losses. Also, we show their validation score that has been calculated in the server with unseen data. As mentioned in 3.3, we use the Categorical Cross-Entropy Loss function to calculate loss values. Plot lines are shown in figure 3.11.

The left part of the figures shows the losses of best, worst, and average models in the first communication round, and the Right part of the figures show the same values in the 40^{th} communication round. According to graphs, the worst performance in the first communication round is done by 82^{th} client with a 74.2% validation score. The average score is done by 4^{th} client with 81%, and the best score is achieved by 16^{th} client with 84.5%. Since all clients start with the same global model parameters, Their initial loss values start almost identical, around 1.80. At the end of the local training, the Average model has a loss value closest to zero, followed by the best and worst models. All models seem to converge at 16^{th} local epoch; for an exception, best model loss values seem to increase slightly at 15^{th} epoch and then continue reducing.

At the end of the last communication round, clients achieving the best, worst and average validation scores differ. the worst performance is done by 21^{st} client with 88.3% validation score. The average score is done by 19^{th} client with 90%, and the best score is

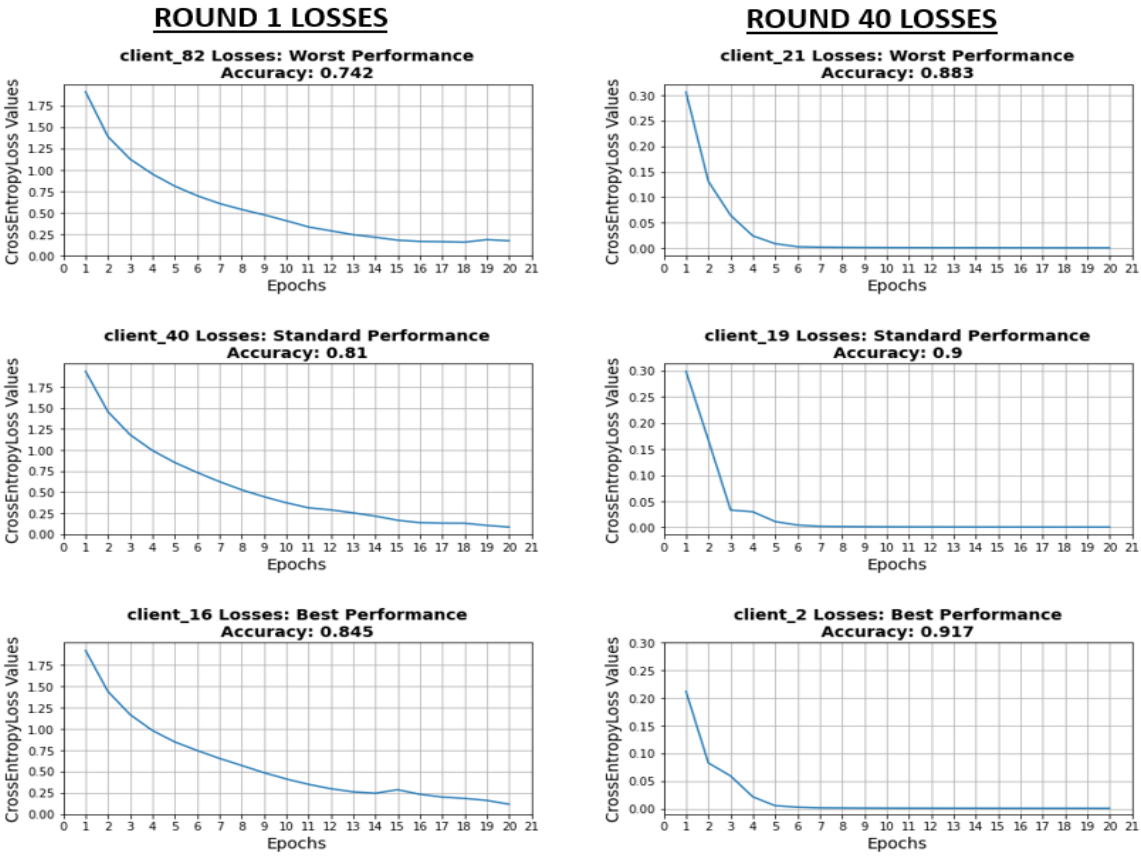

Figure 3.11: Clients Loss Behaviour

achieved by 2^{nd} client with 91.7%. We can see that all clients converge at 6^{th} local epoch, which is almost zero. Clients with the worst and average validation scores start with 0.30 loss values, while the best client starts with 0.20 values. The worst performance client's loss values behavior always tends to fall. The loss value of the client with an average score reduces rapidly in the first three local epochs. Then it slightly increases at the fourth epoch. After that, it always reduces and converges at 6^{th} epoch. Finally, the loss value of the best client reduces rapidly in the first and second epochs. Then the reduction rate reduces for the next two epochs. After that, it continues reducing until it gets converges.

3.7.2 Global Model Accuracy

In this section, on the other hand, we compare the validation accuracy results of fedAvg and SBA aggregation methods. Validation Scores are calculated after the aggregation and are done with the server's unseen validation data. The results graph is shown below in figure 3.12

In the figure, from left to right malicious rate increases from 20% to 50%. According to the figure, when the malicious rate increases, validation accuracy score differences between score-based aggregation (SBA) and federated averaging get advanced for both IID and non-IID environments. Under the IID scenario, it is clear that the global model accuracy remains higher compared to fedAvg, and under every malicious environment, both of them converge. Under the 20% malicious client rate, the SBA validation accuracy score is higher than 90%. As the malicious rate increases, the performance of SBA and federated averaging degrades. However, Baseline fedAvg degrades accuracy score faster than SBA. For this reason, the accuracy differences between SBA and federated averaging increase as the malicious client rate increases. In the end, under 50% malicious rate, SBA accuracy is 7.3% higher than the baseline. The numeric accuracy results of validations are shown in table 3.3

Table 3.3: Experimental Results In IID Environment

Malicious Rate	FedAvg IID	SBA IID
20%	0.897	0.914
30%	0.876	0.897
40%	0.845	0.889
50%	0.806	0.879

In more detail, according to Table 3.3, both validation scores are significantly close

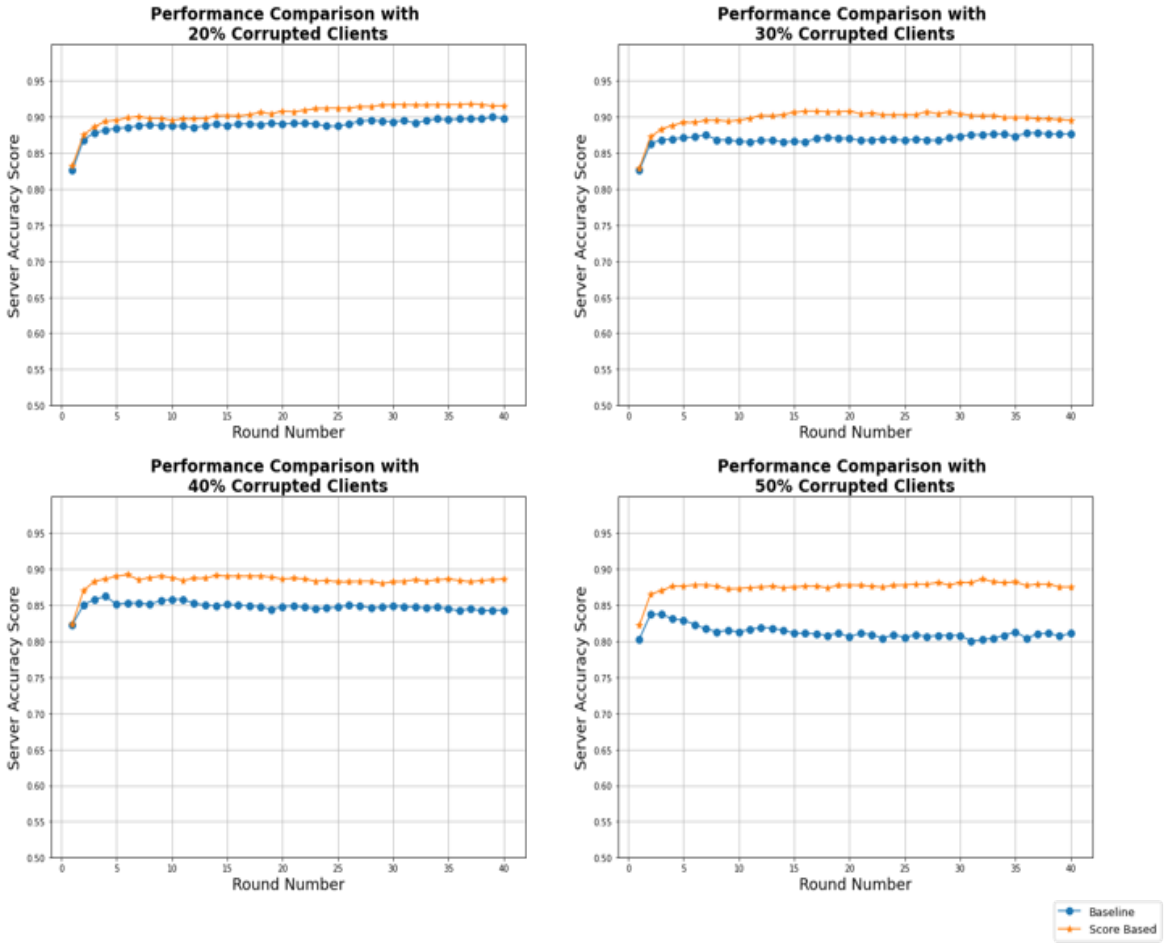

Figure 3.12: Global Model Accuracy Results in IID

to each other under the 20% malicious client rate. Accuracy scores under unseen data after FedAvg is 89.7% while SBA is 91.4%. When the malicious client rate is increased to 30%, the validation score after FedAvg is reduced by 2.1%; meanwhile, The validation score after SBA is reduced by 1.7% and dropped to 89.7%. After the malicious clients in the environment increase to 40%, validation accuracy scores are reduced to 84.5% and 88.9% after fedAvg and SBD, respectively. Finally, when we increase the malicious client's rate to 50%, the validation accuracy score differences between 20% and 50% malicious environment are 9.1% and 3.5% for IID-distributed fedAvg and SBA.

Chapter 4

Score-Based-Aggregation in Non-IID Environment with Collaborating Clients

In the previous chapter 3, we investigate the federated learning and clients' behavior in the IID-distributed malicious environment. Also, we propose a new aggregation method called Score-Based-Aggregation. In this part of the project, the distributed environment becomes more complex, and a new factor comes, namely, **collaboration**. Collaboration allows clients to share their information with other clients after every communication round.

In this chapter, we first describe what collaboration is and how it affects the environment. After that, we investigate the insufficiency of our current aggregation methods, fedAvg, and SBA. Then we decide on our environment's new parameters and show them in the table. Lastly, we compare baseline federated averaging and SBA methods in the collaboration environment with different cases.

4.1 Another Case: Client Collaboration

In our former case, clients in the federated learning environment work in the non-collaborative mode. Each client uses the same volume and class distribution in their dataset, so aggregation at the central server can be more stable. However, in this part, we add collaboration in the environment to make our environment more complex and real since, in a real-world environment, it is expected for people to share some amount of data.

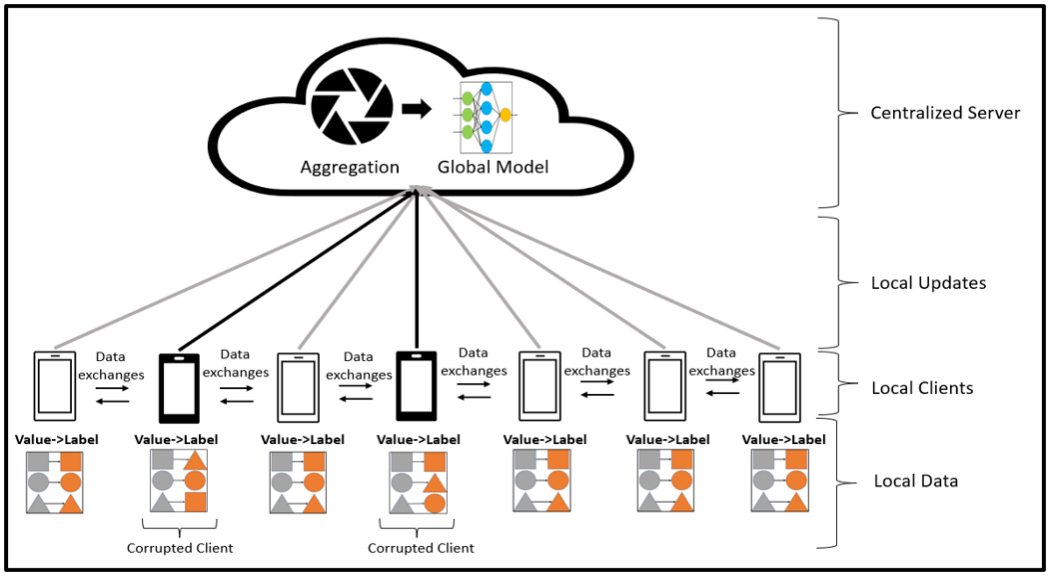

Figure 4.1: Malicious Federated Learning Environment.

In our collaboration scenario, The system model considers that some clients can share their information with other clients after every communication round. Therefore, it makes the environment non-IID since some clients have more data while others have less. To imitate malicious behavior, in each communication round $\mathcal{C}$, a portion of the clients (denoted by c) share $\mathcal{S}$ of their data with $c \times N$ clients. Cooperation leads to a non-IID distribution across federated learning clients since the data volume and distribution at the clients change in each communication round. A new malicious federated learning environment is shown in figure 4.1.

Our environment's baseline architecture does not change, however. Parallel federated learning is presumed, where each client trains their local model concurrently and sends them to the central server for aggregation. Moreover, malicious clients still hold corrupted, mislabelled data, reducing clients' local training performance. The primary purpose of malicious clients is to degrade the centralized global model's validation accuracy.

However, different from previous environments, collaboration brings some issues to our new environment. First, it leads to a non-IID distribution across federated learning clients since the data volume and distribution at the clients change in each communication round. Since machine learning is a data-driven learning system, more data leads local clients to train and generate better results than a low data volume. In the non-IID scenario of our environment, the environment's data distribution does not change dramatically in the first

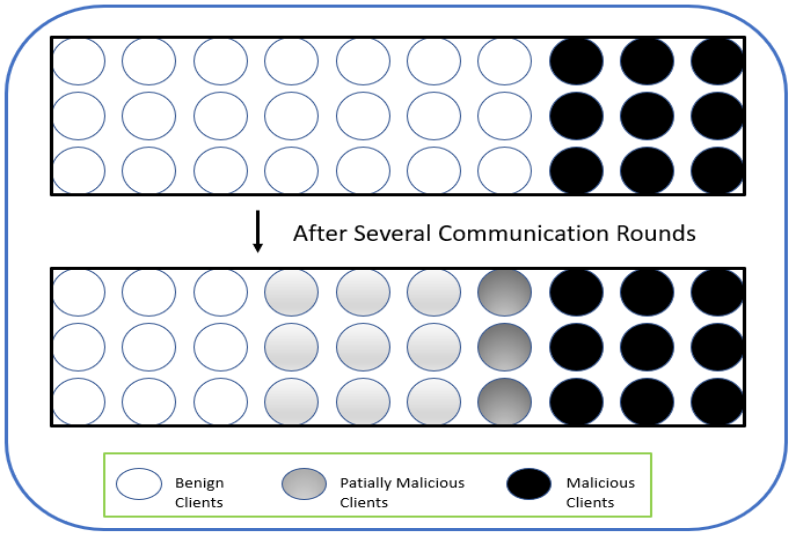

Figure 4.2: Client Malicious State Changes During Several Communication Rounds

few communication rounds since only a few amounts of data are shared. However, at the end of the last communication round, the data sizes of clients that hold maximum data in the environment are five times larger than the client's data size that holds the minimum number of data. Therefore, they get different validation scores for training, and it is not reasonable to participate equally.

Also the second issue is that malicious clients may share corrupted data with other clients during the cooperation. This exchange leads to partially corrupted clients in the federated learning environment. At the beginning of the training, we can easily extract the malicious clients by looking up their validation scores. However, when cooperation begins and continues, some clients may be affected by corrupted data if they exchange data with malicious clients. So, their training data no longer consists of completely benign data but includes some corrupt data. Therefore we can not distinguish clients with benign and malicious categories since partially malicious clients will occur in the following epoch. Clients' malicious state changes from the first iteration to the last iteration are illustrated in Fig. 4.2.

The Score-Based-Aggregation can solve the non-IID problem since it allows clients to join server aggregations depending on their participation rates which are calculated by comparing and normalizing their validation scores. Collaboration does not properly affect aggregation's regularization when it comes into the picture. In the previous case, validation

score differences between malicious and benign clients were huge, so their participation rates allowed them to join the aggregation minimal or maximum. However, if there are malicious clients in the aggregation, and with the non-IID case, clients' differences between malicious and benign will not be apparent anymore, making the malicious clients join the aggregation more.

4.2 New Parameters in Environments

Many factors get into the picture in the new malicious and collaborative environment. We first had to decide the ratio of clients participating in the collaboration in each communication round. In each communication round, some ratio of clients sends their few amounts of data to the same amount of clients. The connection between clients in each communication round are one-to-one connections; in other words, one client can send data to only one client, and one client can receive data from only one client for every round. Connections happen randomly, which means the receiver and sender are selected arbitrarily. Therefore, clients' data size can stay above the number given before running the experiments. Also every experiment, we increase the malicious client ratio by 10%. Besides, other clients' parameters in the NS3-AI part, such as local models, loss functions, and optimizers Dataset, remain the same as in 3.2. All parameters in the system are given in table 4.1.

4.3 Experimental Results

This section starts federated learning with a non-IID distributed collaborative environment. The system's initial state is similar to the IID-distributed environment described in chapter 3. However, as soon as collaboration begins, clients who are partially malicious occur, and the environment's IID attribute shifts to non-IID. First, we investigate clients' behavior in the first and last communication rounds by investigating their loss values. Then We compare the validation scores of the server with both SBA and Baseline Federated Averaging aggregation methods. These results will be lower than the previous results in 3.4 since now we create interactive behavior in the environment that changes all dynamics, including data distributions and malicious ratio. In the first scenario, we suppose that 10% of the environment's clients are malicious. The malicious rate rises to 10% in each case.

Table 4.1: Environment Variables & Initial Parameters in Non-IID Environment

	non-IID Environment
Malicious Rate [%]	{20,30,40,50}
Malicious Label Shuffle	80%
Initial Data Instances	600
Sender Clients Ratio [%]	0.25
Receiver Clients Ratio [%]	0.25
Data Exchange Ratio [%]	0.1
Minimum Data	200
Dataset Name	Fashion-MNIST
# Convolutional Blocks	3
# Linear Blocks	3
Optimizer	Adam
Loss Function	Cross entropy
Learning Rate	0.001
Activation Functions	ELU (Conv.), Tanh (Lin.)

4.3.1 Client's Losses

The behavior of the clients who provide the worst, best, and average validation accuracy scores with SBA in malicious environments are examined in this section of the thesis. We collect the values of their losses for each local epoch, and for the first and last communication rounds, we plot the losses. Then, we present their validation score, computed on the server using unseen data. We use the Categorical Cross-Entropy Loss function to calculate loss values just like section 3.7.1. The loss value changes of clients are illustrated in figures 4.3 and 4.4.

The loss values of clients, in a 20% malicious rate, are shown in figure 4.3 with the highest, worst, and average accuracy scores. The loss values of clients in 50% malicious rate are displayed in figure 4.4 together with their best, worst, and average accuracy scores. The X-axis denotes the communication round, and the Y-axis denotes the categorical cross-

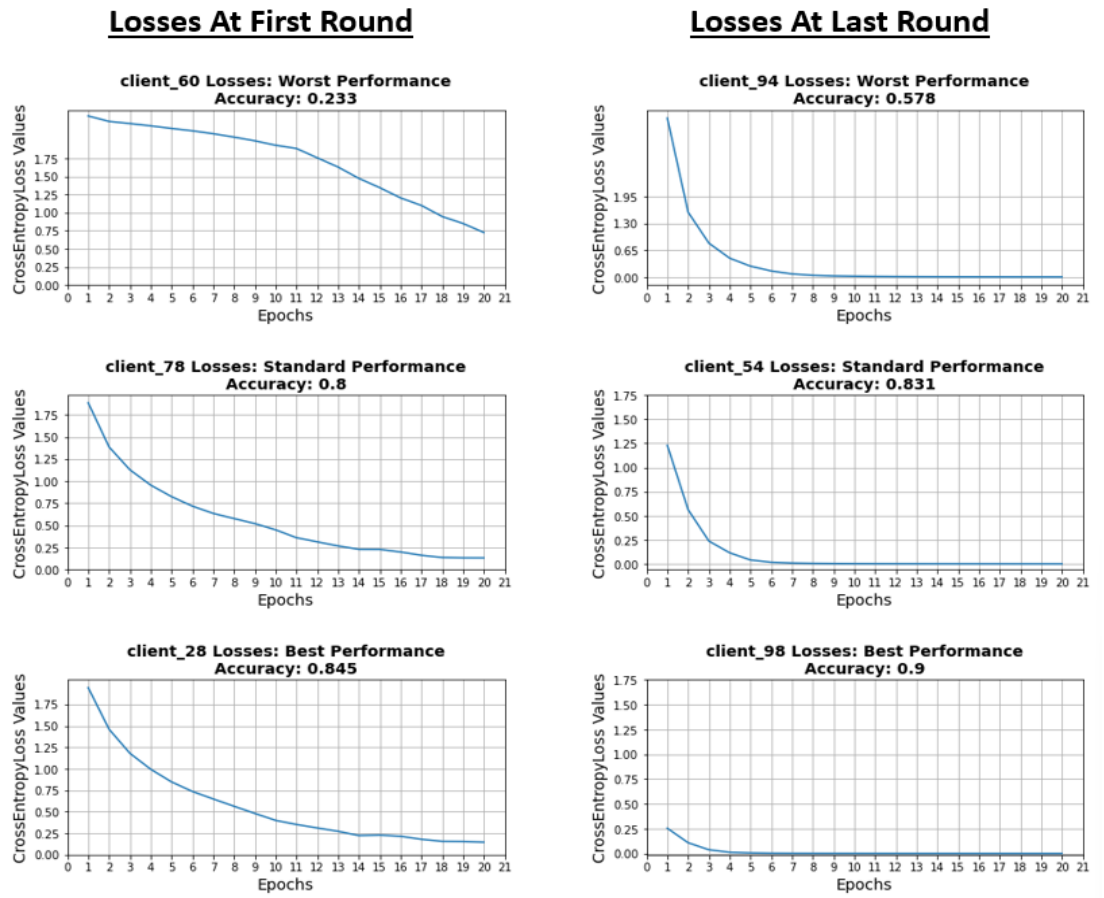

Figure 4.3: Loss Values of Clients Under 20% Malicious Rate.

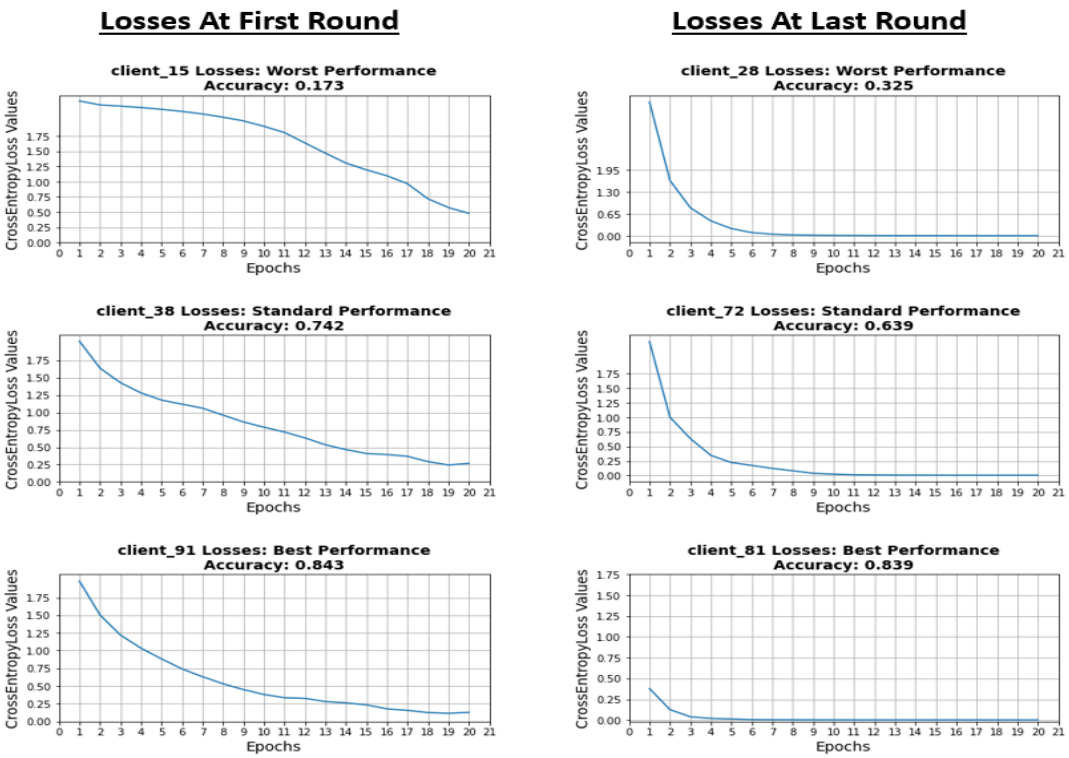

Figure 4.4: Loss Values of Clients Under 50% Malicious Rate.

entropy result for the specified customers. Data is IID-distributed among the clients in the first communication round when only benign and fully malicious clients are present in the environment. Under 20% of malicious clients in the first round, the worst accuracy performance is done by 60^{th} client with 23.3%, and it is followed by 78^{th} and 28^{th} clients with 80% and 84.5% validation accuracy. As observed, other clients converged with loss values of around 0.1, whereas a fully malicious client could not. Under 50% of the malicious rate worst performance is done by 15^{th} clients with 17.3% validation scores. 38^{th} client is with standard validation score which is 74.2%, and best validation score is achieved by 91^{th} clients with 84.3%. The fully malicious client is not converged in terms of loss values, similar to the 20% malicious rate case, while others converged around 0.2 cross-entropy losses.

On the other hand, the IID characteristic of environments is changed to non-IID in the most recent communication round since some clients share their data in every communication round. Additionally, partially malicious clients appear in the following communication rounds because some clients accept corrupted data from malicious clients. It seems that, when 20% of malicious clients are in the environment, the worst performance is done by 94^{th} clients with 57.8% accuracy scores. The middle point of validation scores is made by 54^{th} clients, and 55^{th} clients achieved the best performance with 83.1% and 90.0% validation scores. When the malicious rate of the environment is increased to 50%, the worst accuracy performance is done by 28^{th} clients with 32.5% validation scores. it is followed by 63.9% and 83.9% validation scores that are achieved by 72^{nd} and 81^{th} clients. All clients' losses converged remarkably close to zero when the malicious rate was between 20% and 50%.

It seems that even if there are collaborating clients in the environments, the worst accuracy score in the 20% malicious rate is 57.8% which is at a partially malicious clients level. However, when the malicious rate increased to 50%, the worst accuracy performance dropped to below 32.5%. When malicious clients distribute their corrupted data to other clients, it causes them to lessen their corrupted data, which causes an increase the accuracy. Also, it increases its accuracy when it receives pure data from benign clients. Moreover, If the environment has a low malicious rate, like 20%, the corrupted data from malicious clients may be dispersed across the clients sufficiently thinly to have little to no impact on accurate performance until the malicious clients run out of damaged data and their performance suffers. That is why we see only partially malicious and benign clients in the 20% malicious environment. However, if the rate of malicious clients rises, their data can adversely impact the training of other clients and the environment. For example, in a 50% malicious environment, although all clients converged at the end of their training, there are fully malicious, partially malicious, and benign clients in the environments. These loss

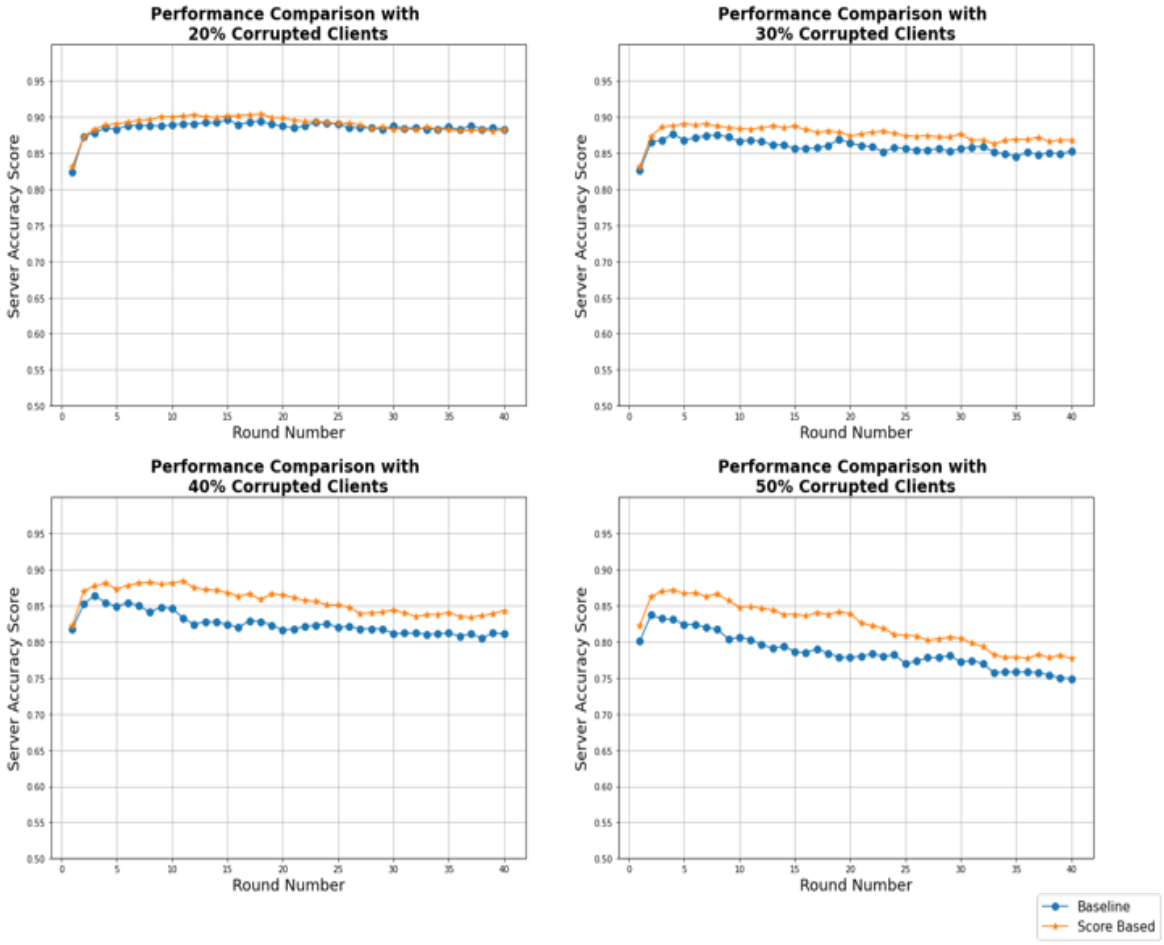

Figure 4.5: Global Model Accuracy Results in non-IID

values are collected from training. Therefore, it can memorize the training data distributions even if there is mislabelling data inside the local device. The overall consequence is that collaborating with clients allows malicious factors to flaw the system dramatically.

4.3.2 Global Model Accuracy

In this part, we are just looking into the global models' accuracy comparison between SBA and baseline fedAvg. Validation scores are calculated after aggregation with unseen data. The results graph is shown below in figure 4.5.

In figure 4.5, from left to right malicious rate increases from 20% to 50%. According to the figure, when the malicious rate increases, validation accuracy score differences between score-based aggregation (SBA) and federated averaging get advanced like 3.4.2. Under the 20% malicious client rate, the SBA validation accuracy score is slightly lower than 90%. As the malicious rate increases, the performance of SBA and federated averaging degrades. However, Baseline fedAvg degrades accuracy score faster than SBA. In the brief of the chart, SBA aggregation scores are still higher than the fedAvg. However, the accuracy difference is not as much as the IID environment, and the degradation of accuracy performance is significantly high when the malicious client rate increases.

Furthermore, under 50% malicious client rates, Models do not converge at the end of the last communication round. Because of the collaboration of clients, there is an imbalance of data across clients, and the number of instances assigned to a client can be more than 1200, while the data size of another client is smaller than 200. Moreover, it is possible that malicious clients can send their corrupted data to other benign clients when collaboration is allowed. It reduces overall performance, for instance, in 20% malicious client cases. In the communication round between 10 and 20, global model validation accuracy is higher than 90%. However, since the corruption data exchanges between malicious and benign clients, the accuracy score decreases in the rest of the communications. It is because clients that receive corrupted data can no longer be considered benign clients but partially malicious ones. Hence, aggregation alone does not work as well as in the IID-distributed environment; that is why accuracy performances tend to decrease dramatically in 40% and 50% of malicious client rate scenarios. Numeric values at the end of the communication round are given in table 4.2

Table 4.2: Experimental Results In non-IID Environment

Malicious Rate	FedAvg non-IID	SBA non-IID
20%	0.883	0.882
30%	0.845	0.862
40%	0.807	0.833
50%	0.752	0.779

In more detail, according to Table 4.2, both validation scores are significantly close to each other under the 20% malicious client rate. Accuracy scores under unseen data after FedAvg is 88.3% while SBA is 88.2%. When the malicious client rate is increased to 30%, the validation score after FedAvg is reduced by 3.8%; meanwhile, The validation score after SBA is reduced by 2.0% and dropped to 86.2%. After the malicious clients in the environment increase to 40%, validation accuracy scores are reduced to 80.7% and

83.3% after fedAvg and SBD, respectively. Finally, when we increase the malicious client's rate to 50%, the validation accuracy score differences between 20% and 50% malicious environment are 13.1% and 10.3% for IID-distributed fedAvg and SBA. In this case, their accuracy scores reduce to under 80%, and validations of federated Averaging is 75.2% while federated Averaging is 77.9%.

In the previous case, validation scores difference between 20% and 50% malicious rate with the SBA method is 3.5%. However, in a non-IID collaboration environment, this difference grows to 10.3%. Also, In the non-IID scenario, SBA is better than fedAvg by 7.3%, and in this chapter, this result reduces to 2.7%. Although SBA achieves better results than fedAvg, in non-IID cases, its performance gets closer to the baseline approach and is not tolerable for model resiliency. That is why in the further chapter, we take one step further on this approach and come up with better aggregation methods, namely "SBA with K-Means Clusters."

Chapter 5

Score-Based-Aggregation and Cluster Elimination in Non-IID Environment with Collaborating Clients

In the previous chapter, we introduced collaboration in the environment and tested the SBA and fedAvg aggregation methods. However, unlike the non-Collaboration scenario in chapter 3, we did not get promising results for both algorithms. Our proposed method's results get closer to the baseline algorithm, which is not resilient enough for the distribution non-IID environment. Therefore we propose a new aggregation method, namely, Score Based Aggregation with K-Means Clustering. This method eliminates fully malicious clients and clients that dramatically receive amounts of corrupted data before the specific aggregation. In the rest of this chapter, we first introduce our new method. Then we will focus on its advantages and test it in the non-IID environment with a collaborating client. Finally, we compare this method with SBA and FedAvg aggregation methods.

5.1 Drawbacks of Collaboration

In the previous chapter, SBA lost its robustness because of the collaboration factor. Since clients can share their data with each other, the IID-distribution structure of the environment is changed to non-IID, and malicious clients can affect the other clients' performance negatively. Although clients receive new data, it is not ensured that it will improve their training because it is unknown if the data is pure or corrupted. If the client

receives data from the benign client, it will improve its accuracy but reduce the sender client's performance because of losing its data. On the other hand, if the clients receive corrupted data, it will degrade its performance since they can not learn properly from that data and gets confused. These kinds of clients are called partially malicious clients. Moreover, if one client sends its data to a malicious client, even if the data is pure, it loses its effectiveness because of the Presence of fully corrupted data in the malicious clients.

Therefore, at further communication rounds, our clients can be classified into three categories according to their validation scores: benign clients, partially malicious clients, and malicious clients. In standard cases, if clients' accuracies are around 85% or more, they are probably counted as benign clients. On the other hand, if this value is around 25%, we can easily assume that they are malicious clients. However, suppose client accuracy is around 60% or 70%. In that case, we can not clearly say that it is partially malicious clients since if clients give so much data away, even if it was full pure data, it may get stuck with this accuracy score because of data shortage.

In an IID-distributed environment, the accuracy scores of clients are either around 85% or 25%. Therefore there is a clear distinction between clients, and SBA can easily reduce the malicious clients' participation rate minimally so that they can not affect the aggregation gravely. However, this reduction can no longer be appropriately made when collaboration comes into the picture. Normalized validation scores calculate the participation rate to zero and one, and malicious clients' participation rate is very close to zero, while benign clients are almost one. However, in non-IID scenarios comes into the picture, those ratios will change. For example, if one benign client becomes a partially malicious client, its accuracy score reduces to 85% to 65%, increasing other malicious clients' participation rates. It also causes benign clients' participation rate increases too; however, it can not make the global model performance stable because the system already loses one benign client and increases the malicious clients' participation to aggregation.

5.2 Elimination of Malicious Clients with Unsupervised Method: K-Means Clustering

We propose our new algorithm to increase the robustness of federated learning and make it more resilient in the collaborative environment. This new method consists of two steps. The first step tries to eject fully malicious clients from the aggregation. Although SBA reduces participation rates of malicious clients in the aggregation, it still affects the model training and reduces benign clients' participation. The participation rate depends on

the clients' validation rates and the number of clients. If the client number increases, each client's participation decreases. Hence, if we remove malicious clients in the system, benign and partially malicious clients' participation rates increase since the number of clients in the system reduces. Although the partially malicious client joining rate increases, it does not negatively affect the global model aggregation since benign client joining rates also increase, and malicious clients can no longer harm the global model. After that, we apply our second step, described in section 5.3.

We use K-Means clustering methods to eliminate malicious clients in the environment. The clustering algorithm is used in federated learning for many research, such as clustering clients based on their data distributions in Non-IID distributed environment [98]. In this work, We use different clustering algorithms based on clients' validation scores since validation scores show how trustworthy the client is. [99], which presents a method for calculating an estimated trust score for each model update, aggregates those changes weighted by those scores, and then updates the global model, is the source of inspiration for using clients' trust scores. Although the K-Means method has some limitations, such as finding optimal cluster number k, and performance degradation in high dimensional data, the reason for selecting this algorithm is the simple data structure we use in an unsupervised method, which is a 1-dimensional validation score array and predefined cluster number which is 2: benign, malicious.

In the environment, malicious and benign clients can be distributed in 5 cases. Firstly, in the system, there might not be benign clients. In the second case, all clients can be malicious. Moreover, there are some outlier malicious clients among massive benign clients distribution or vice versa. In the last case, malicious and benign clients can be homogeneously distributed. Therefore, we will decide if the unsupervised cluster method is applied in the current case. Also, an unsupervised Cluster algorithm must properly discriminate malicious clients from benign clients. Since data structure has a simple architecture, basic unsupervised learning algorithms such as K-Means can perform better than complex methods. We test many case examples with complex methods, such as SVM, which might eliminate outlier benign clients from the aggregation.

The validation score of each client is calculated by the server's unseen validation data, just like 3.1. All clients are tested with different validation IID data, and their validation scores are stored in server data. Before starting the malicious client elimination, we must ensure that there are malicious clients in the environment. So, we first calculate the average of all clients' validation scores. Then we try to find outliers of clients according to this score. If there is no outlier, we assume all clients are benign or malicious, or benign and malicious clients are similarly distributed in the system according to their scores. Therefore, we look at the average, if the average is below a certain *lower threshold*, we assume that all clients

are malicious, and we use previous global model updates. On the other hand, if the average is above the *upper threshold*, we assume that all clients are benign clients, so we do not need to do elimination methods; it is omitted and continues to SBA aggregation directly. All thresholds are shown in table 5.1. However, if there are outliers in the system or the average is not below the malicious threshold or above the benign threshold, we assume that the environment consists of partially malicious clients, malicious clients, or both, along with being clients. Then We apply **K-Means Clustering** considering These scores.

Table 5.1: malicious and benign thresholds

Lower Thresholds	%30
Upper Thresholds	%85

K-Means algorithm is an unsupervised clustering algorithm that uses distance measure to separate data points into clusters [100]. In this method, iterative learning is used to find k number of clusters. k is initially defined, and in this paper, we set it 2: malicious and non-malicious clusters. Locations of cluster centroids are given randomly at the initial state. The learning process has two stages. In the first stage, the training samples belong to one of the clusters. In the second stage, the locations of cluster centroids are updated. Until it meets the stopping conditions, these two stages are repeated over and over again in training.

Input space in this algorithm is clients' validation scores. *Euclidean distance* is used to calculate the measure between centroids and each data point. Calculation of the K-Means algorithm according to [100] is given below:

After we cluster clients into two groups according to their validation scores, we calculate the average of each client cluster. Then, we select the cluster that has a higher average score. Clients under a low average score cluster are considered fully malicious clients and are immediately discarded from the aggregation. Figure representation of clustering clients is shown in 5.1

5.3 SBA After Elimination

Although eliminating fully malicious clients seems reasonable, protecting the global model performance is insufficient because partially malicious clients receive small amounts of corrupted data. Though their corrupted data do not dramatically reduce their validation accuracy score (because they are in the non-malicious client cluster), they might affect the

Algorithm 1 Clustering of Client Validation Scores

definition
    C is the list of clients,
    D is server validation data,
    d is ratio of data clients take from server for validation,
    k is the number of clusters,
    c_j , the j^{th} cluster centroid, j = {1,2,...k}
    s_j, the number of samples associated to the j^{th} cluster
    $Score_i$, the i^{th} element in Score list
end definition
$Scores \leftarrow []$
for each $c_i \in \boldsymbol{C}$ **do**
    $D_i \leftarrow Randomly\_Selects(D, d = 0.10)$
    $score \leftarrow Validation(c_i, D_i)$
    $Scores.append(score)$
end for
$k \leftarrow 2$
/* Starting to K-Means */
repeat
    $c_j = c_j^+, \forall j$
    $\forall i$, find j for which $|Score_i - c_j|^2$ is minimal $\forall$j and set $Score_i \Rightarrow jth$ cluster
    $\forall$j, $c_j^+ = \frac{1}{s_j} \sum_{k=1}^{s_j}$ where $Score_k \Rightarrow jth$ cluster
until no samples are re-associated to another cluster

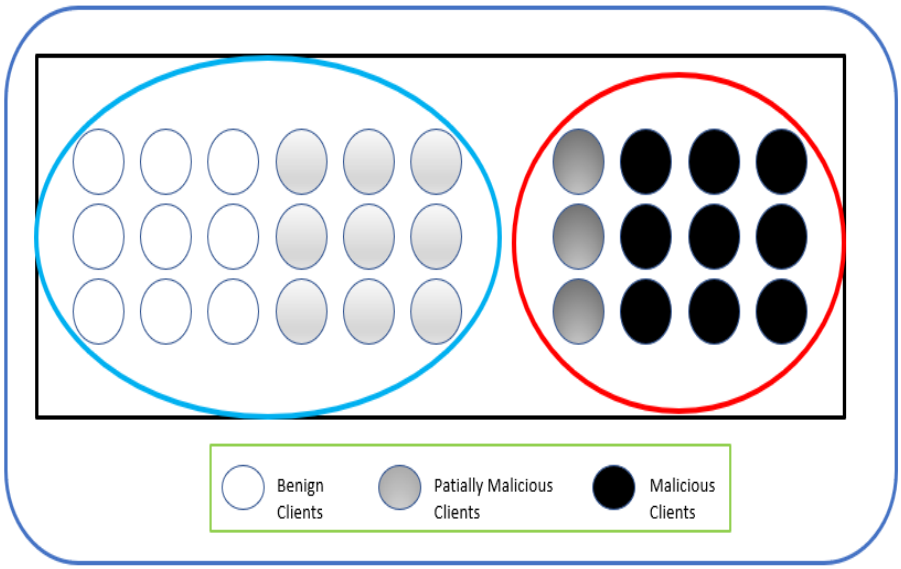

Figure 5.1: KMeans Clustering Among Clients.

server aggregation since their performances are still not as good as benign clients. Therefore, we need to do weighted aggregation in the server by considering clients' validation scores.

In the second step, we apply SBA aggregation to the clients accepted to the aggregation, which is mentioned in section 3.1. Since partially malicious clients can not provide good results like benign clients, Letting them join the aggregation equally is not reasonable. Therefore, we apply SBA to adjust their participation rates. The SBA aggregation method looks at each client's validation score and arranges the participation rate for each client in the range [0-1].

5.4 Flow Diagram of SBA with K-Means Clustering

The flow diagram of SBA with K-Means Clustering is given in figure 5.2. The diagram starts when the server receives all model updates to be ready for aggregations. First, server unseen validation data do client validation. After collecting a score for each client, we look if there are outlier clients according to validation scores. If there are outliers, we assume malicious and benign clients in the system and apply K-Means Clustering elimination. Otherwise, the system consists of either fully malicious or benign clients, similarly

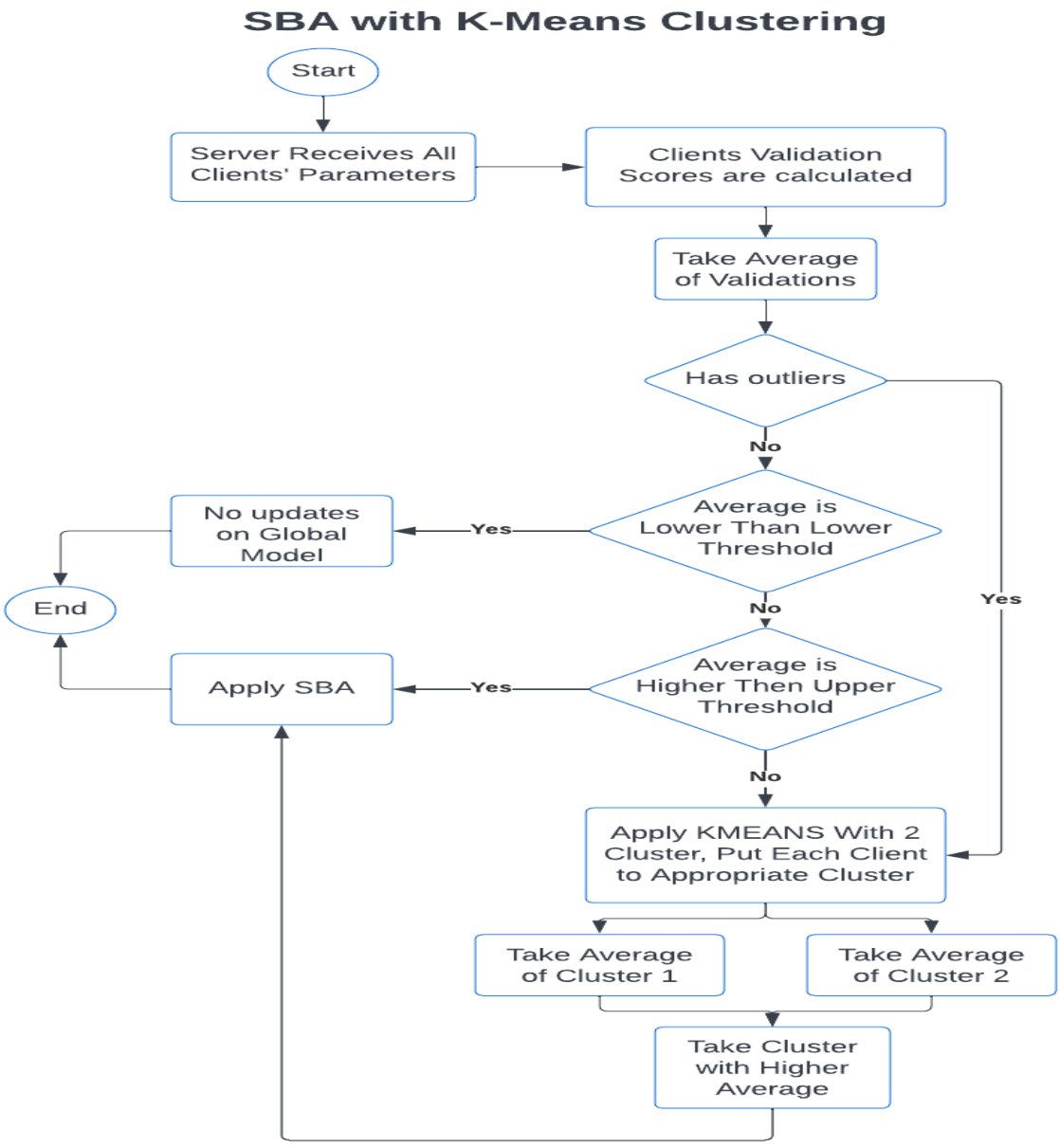

Figure 5.2: Flowchart of SBA with K-Means Clustering

distributed in the system. We can understand this by taking the average validation scores. The system includes one kind of client if the average is higher than the upper threshold or less than the lower thresholds in 5.1. Then it goes either SBA aggregation or no aggregation state. Otherwise, we apply K-Means Cluster elimination. K-Means creates two clusters and puts each client into that cluster according to their validation scores. Clients under clusters having higher validation score averages are selected to go into the SBA aggregation state. Aggregation finishes after SBA aggregation.

5.5 Complexity Analysis

On the other hand, complexity analysis is another essential factor in comparing this methodology with other studies. Environments include N number of clients, and each does CNN training. Considering client number, CNN training is done by N times, and clients send their model parameters to the server once training is done. After receiving all models from clients, the server does the K-Means clustering. The rest of this methodology is composed of arithmetic operators such as summation, division, and multiplication. The time complexity of CNN, Linear, and K-Means is given below:

$$O(T_{CNN}) = O(N^2 * F^2 * C_in * C_out * H * W)[101] \tag{5.1}$$

$$O(T_{linear}) = O(N * M)[102] \tag{5.2}$$

$$O(T_{K-Means}) = O(I * K * N * D)[103] \tag{5.3}$$

The equation 5.1 shows the time complexity of Convolutional Layer, where N is the number of examples in a batch, F is the size of the filter, C_in and C_out are the number of input and output channels, and H and W are the height and width of the input feature map. The dominant factor is the filter size, which is usually much smaller than the input size. Therefore, the time complexity of CNN is usually much lower than fully connected neural networks, and CNNs can efficiently process large input data.

The equation 5.2 shows the time complexity of Linear Layer, where N is the number of input features and M is the number of output features. The dominant factor is the number of input features, and the time complexity increases linearly with it. Therefore,

the computational cost of a linear layer can be high when the number of input features is large, and it can become a bottleneck in the neural network.

The equation 5.3 shows the time complexity of Linear Layer, where I is the number of iterations, K is the number of clusters, N is the number of data points, and D is the number of dimensions in each data point. The dominant factor is the number of iterations, which can be as high as the maximum number of allowed iterations or until convergence is reached. The algorithm involves assigning each data point to its nearest cluster center, and then updating the cluster centers based on the new assignments. This process is repeated until convergence or until the maximum number of iterations is reached. Values of Variables are given in table 5.2.

Table 5.2: Time Complexity Parameters

Variables	Values
N_{CNN}	64
F_{CNN}	[64, 32, 16]
C_in_{CNN}	1
C_out_{CNN}	1
H_{CNN}	32
W_{CNN}	32
N_{Linear}	64
M_{Linear}	[128, 32, 10]
$I_{K-Means}$	Max Value 300
$K_{K-Means}$	2
$N_{K-Means}$	100
$D_{K-Means}$	1

5.6 Experimental Results

In this section, we use federated learning with a non-IID distributed collaborative environment, the same as the section 4.3. The system's initial state is the same as the IID-distributed environment in chapter 3. However, when collaboration starts, the IID attribute of the environment changes to non-IID with the presence of partially malicious clients. We compare our results with Baseline Federated Averaging, SBA, and SBA with K-Means Clustering aggregation methods and show the advantages of our proposed method.

Four cases vary by malicious client rate in the environment from 20% to 50%. Our simulation and AI parameters are the same as in the previous sections. We first look into the system behavior of the benign client By examining their loss values; Then, we compare all aggregation methods by the centralized server's global model.

5.6.1 Client's Losses

In this part of the thesis, we investigate the clients' behaviors that achieve the worst, best and average validation accuracy scores with Score Based Aggregation with K-Means Clustering. We gather the values of their losses for each local epoch and plot the losses for the first and last communication rounds. Then, we present their validation score, computed on the server using unseen data. We use the Categorical Cross-Entropy Loss function to calculate loss values just like section 3.7.1. The loss value changes of clients are illustrated in figures 5.3 and 5.4.

Figure 5.3 shows the loss values of clients, in 20% malicious rate, with best, worst, and middle accuracy scores. Figure 5.4 shows the loss values of clients, in a 50% malicious rate, with best, worst, and middle accuracy scores. X-axes indicate the communication round, and Y-axes indicate the categorical cross-entropy result of given clients. In the first communication round, there are only benign and fully malicious clients in the environment, and data is IID distributed among the clients. Under 20% of malicious clients in the first round, the worst accuracy performance is done by 49^{th} client with 23.6%, and it is followed by 5^{th} and 96^{th} clients with 80% and 84% validation accuracy. As can be seen, a fully malicious client could not be converged by loss values, while others converged with around 0.1 loss values. Under 50% of the malicious rate worst performance is done by 15^{th} clients with 18.8% validation scores. 44^{th} client is with standard validation score which is 70.7%, and best validation score is achieved by 47^{th} clients with 85.2%. like the 20% malicious rate scenario, the fully malicious client is not converged according to loss values, while others converged around 0.2 cross-entropy losses.

In the last communication round, on the other hand, since some clients share their data in every communication round, the IID feature of environments is changed to non-IID. Also, because some clients accept corrupted data from malicious clients, partially malicious clients occur in the following communication rounds. It seems that, when 20% of malicious clients are in the environment, the worst performance is done by 94^{th} clients with 61.7% accuracy scores. The middle point of validation scores is made by 54^{th} clients, and 55^{th} clients achieved the best performance with 84.0% and 90.6% validation scores. When the malicious rate of the environment is increased to 50%, the worst accuracy performance

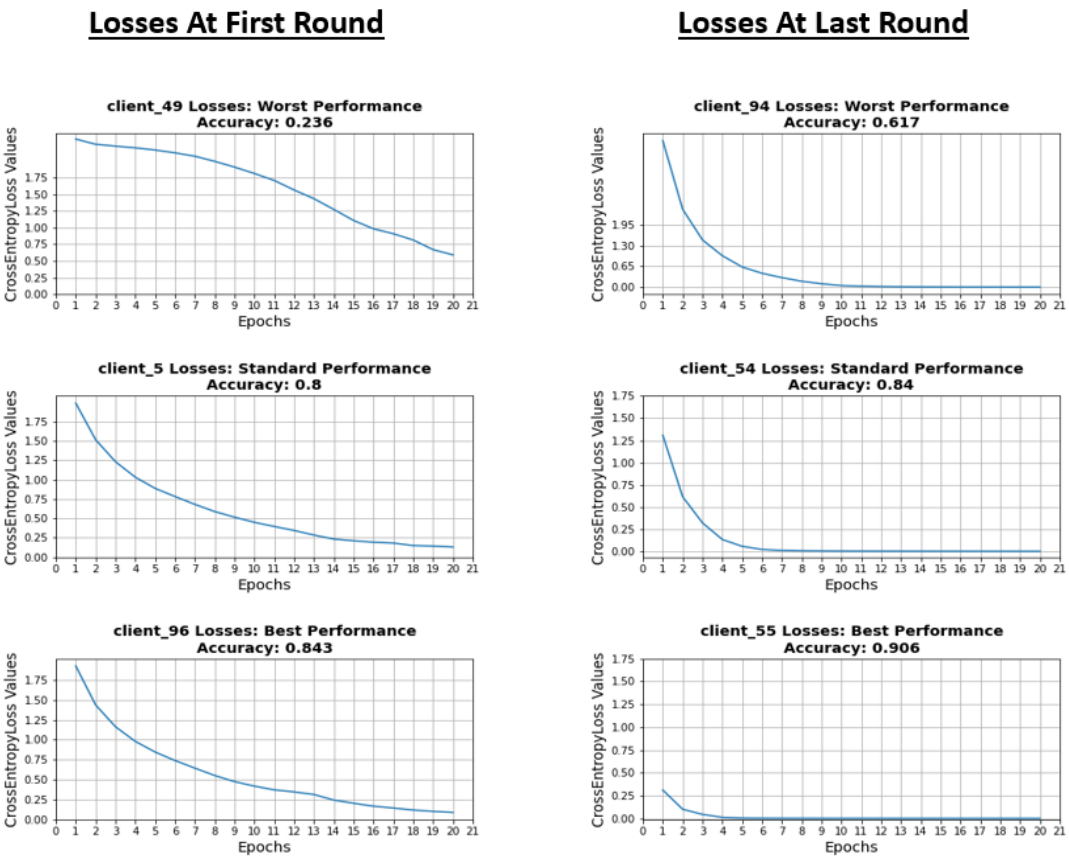

Figure 5.3: Loss Values of Clients Under 20% Malicious Rate.

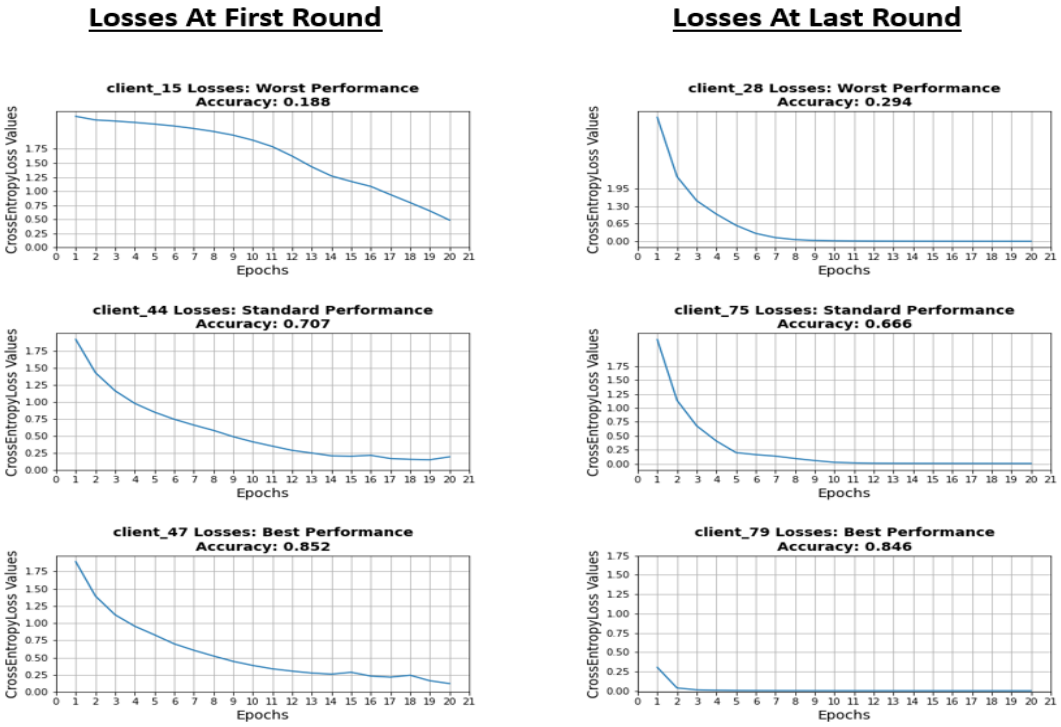

Figure 5.4: Loss Values of Clients Under 50% Malicious Rate.

is done by 28^{th} clients with 29.4% validation scores. it is followed by 66.6% and 84.6% validation scores that are achieved by 75^{th} and 79^{th} clients. under 20% and 50% malicious rate, all clients' losses converged very close to zero.

Although there is a collaboration in the environments and malicious clients send their corrupted data to benign clients, under 20% of the malicious rate, the worst performance is 61.7% which is above 50%. Though it is not benign client-level performance, it is much better than malicious clients' first iteration. However, when the malicious rate increased to 50%, the worst accuracy performance dropped to below 30%, but it was still greater than 50% first communication round accuracy performance. This is because when malicious clients distribute their corrupted data to other clients, it causes them to lessen their corrupted data. In addition, giving their corrupted data away, receiving pure data, and training with better initial parameters from the server model after aggregation with the proposed method increase their performance. So if the malicious rate of the environment is less, such as 20%, those corrupted data from malicious clients may be spread too thin among the clients to affect almost nothing accuracy performance until malicious clients do not have enough corrupted data to make their performance degrades. That is why we see only partially malicious and benign clients in the 20% malicious environment. However, if the malicious rate increases, malicious clients' data may harmfully affect environments and other clients' training. For example, in a 50% malicious environment, although all clients converged at the end of their training, there are fully malicious, partially malicious, and benign clients in the environments. Though the worst accuracy score in the environment is increased due to having pure data from benign clients, the best performance of clients at the end of the communication round is lesser than the first iteration since, this time, pure data from benign clients is spread too thin among the malicious clients on the opposite of environment with 20% malicious rate. Also, it is clear that although accuracy scores are slightly higher, malicious clients' loss values start much higher in the last communication round than benign clients.

5.6.2 Global Model Accuracy

In this section, on the other hand, we compare the validation accuracy results of fedAvg, SBA, and SBA with K-Means Clustering aggregation methods. Validation Scores are calculated after the aggregation, and they are done with the server's unseen validation data. The results graph is shown below in figure 5.5.

In Fig. 5.5, we compare the servers' accuracy score in a malicious and collaborative environment with the baseline algorithm, SBA, and our proposed method, K-Means clusters

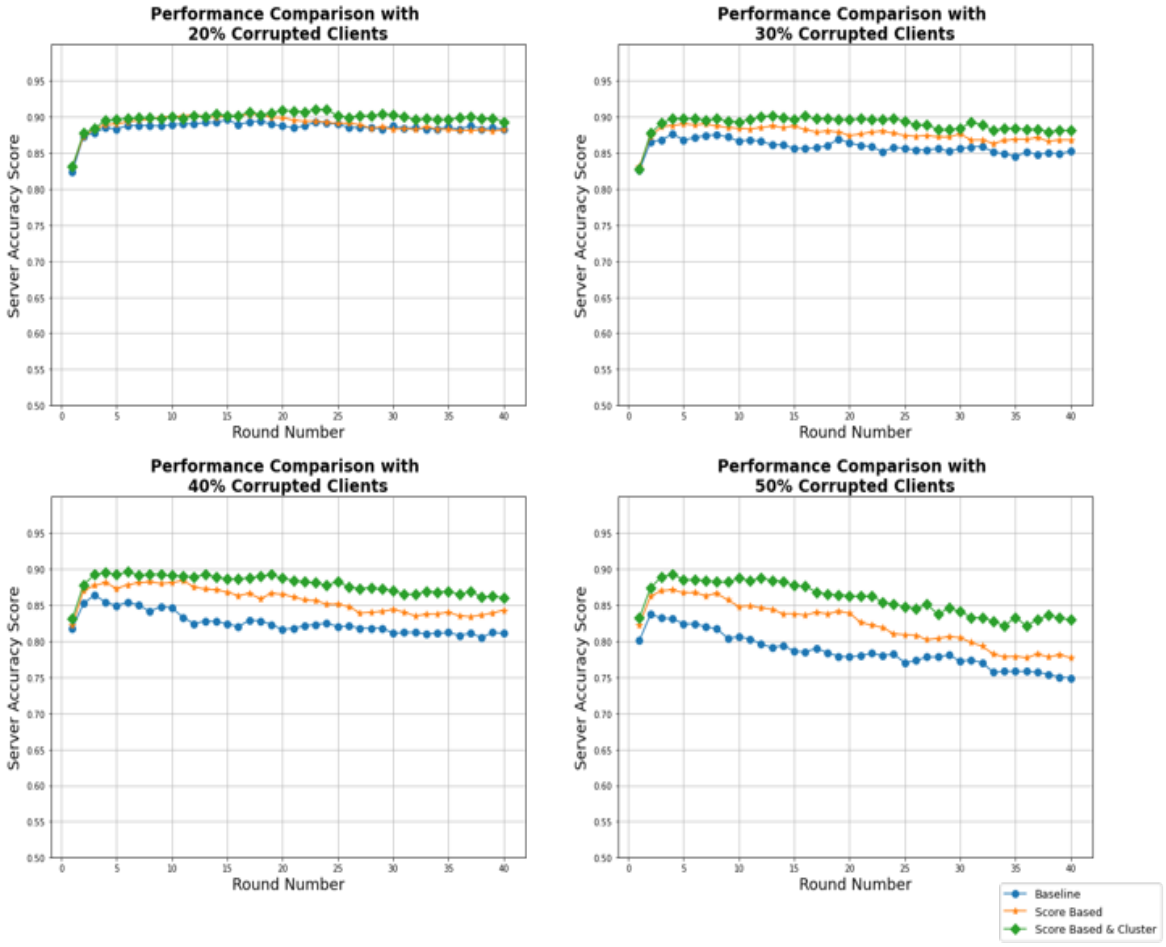

Figure 5.5: Global Model Accuracy Results in non-IID

with SBA. X-axes show the number of communication rounds, and Y-axes show the server validation score after the end of the communication round. It can be seen our proposed method surpasses the rest of the other methods according to validation scores, and it is followed by SBA and Baseline algorithm. Also, The chart illustrates that when the malicious rate rises, our proposed methods score-based aggregate (SBA) with K-Means clustering and other methods' validation accuracies score differences advance. When 20% of clients are malicious in the environment, there are not so many differences between those three methods. However, when the malicious rate increases to 30%, validation differences widen. At a 40% malicious rate, the Baseline method's accuracy score degrades fast while others try to converge their scores. Finally, in the 50% malicious rate, our proposed method's validation score is still above 80% while those two reduce dramatically to around 75%. Detailed results are given in table 5.3.

Table 5.3: Experimental Results in Non-IID Environment

Malicious Rate	FedAvg non-IID	SBA non-IID	SBA + K-Means Cluster non-IID
20%	0.883	0.882	0.898
30%	0.845	0.862	0.878
40%	0.807	0.833	0.867
50%	0.752	0.779	0.820

As it can be seen in the table 5.3 when the malicious client rate is 20% in the environment, Baseline and SBA are very close together, and the baseline is slightly greater, 88.3% and 88.2% In our approach, it is 15% greater than them and reaches 89.8%. When there are 30% of malicious clients, the baseline degrades dramatically and drops 84.5% while SBA and our approach's accuracies reduce the same, 20%, and their validation scores are 86.2% and 87.8%. When the malicious client rate increases to 40% baseline, have 80.7%, and SBA has 88.3% accuracy scores. SBA with K-Means clustering is 6% and 3.4% greater than baseline and SBA methods with 86.7% validation accuracy. Finally, when half of the environment is malicious, baseline and SBA accuracy scores reduce below 80% and drop to 75.2% and 77.9% while our proposed approach stays at 82% of accuracy scores.

Chapter 6

Conclusion

Federated learning is a novel distributed machine learning method used to overcome centralized machine learning techniques' security and privacy issue. In centralized machine learning, All data residing in clients' local devices are sent to the centralized server. The global model inside the centralized server is trained with these data and then distributed to the clients. However, giving local data away causes privacy and security issues in centralized training. For example, some companies that care about data privacy, such as hospitals, may want to keep their data private from a second user. So it causes drawbacks for training the global models.

In federated learning, instead of centralized training with collected data, all clients train their models inside their devices with their local data. Then, send their model parameters to the centralized server instead of data. The server aggregates all these parameters and updates the global model. This new global model is shared with clients, and this process is called a communication round. It is repeated until the global model converges. Since the local client does not share data with the centralized server, it protects the user's privacy.

However, because of its distributed behavior and indirect training of the global model, federated learning is vulnerable to malicious attacks that affect the security or privacy of a system. Attacks' purpose can be either data leakage or degrade global model performance. In this, we focused on creating a new aggregation method resilient to malicious attacks that harm the global model performance.

A malicious attack on the global model was made on the server indirectly; Instead of attacking to global model directly, malicious attacks were attempted on local clients by manipulating their training. Manipulation was done by data poisoning, assigning wrong labels for input data. This attack degraded the local model's training, so local clients

generated harmful parameters. These parameters are sent to the centralized server just like other clients' model parameters to aggregations. Letting these harmful parameters join the aggregation directly caused performance reduction on global models; Moreover, since the attack was made indirectly, it was impossible to track the malicious attacks.

We devised a new idea to improve the resiliency of the global model for this attack, which was a new aggregation method. Instead of allowing all clients to participate in the aggregation, we generated a participation rate for each client. The participation rate was calculated by looking up clients' validation scores which were calculated in the centralized server, with unseen data, after receiving all clients' model parameters. Then, we put each client's participation rate in the range of zero to one ([0-1]), and all participation rates' summation was equal to 1. So instead of letting all clients join the aggregation equally, clients with fewer validation scores joined the aggregation less, while clients with high accuracy joined the aggregation high. This aggregation method was called Score-Based-Aggregation (SBA).

We tested SBA on a federated learning environment with IID data distribution. We had four cases that varied each other by different malicious client rates in the environment. For example, the first case was a system with a 20% malicious client rate, and in each case, this rate increased by 10%; the last case's malicious rate was 50%. We compared our new aggregation methods with the baseline federated averaging method (fedAvg). According to table 3.3, the SBA method surpassed the fedAvg in every case. For example, in 20% malicious client rate, SBA got the maximum accuracy score with 91.4% while fedAvg had 89.7% validation accuracy.

Then we added collaboration to the environment to make our system more complex and realistic. In collaboration, clients were allowed to share their data with one-to-one communication links. However, this attribute changed the system's IID feature to non-IID since clients' data sizes would not be the same after the information exchange. Also, malicious clients could share their corrupted data with the benign client. As a result, after several communication rounds, the system would contain non-IID data distributions with benign, malicious, and partially malicious clients.

We tested SBA on a federated learning environment with IID data distribution. We had four cases that varied each other by different malicious client rates in the environment. The first case was a system with a 20% malicious client rate, and in each case, this rate increased by 10%; the last case's malicious last case was 50%. We compared our new aggregation methods with the baseline federated averaging method (fedAvg). According to table 3.3, the SBA method surpassed the fedAvg in every case. For example, in the 20% malicious client rate, SBA got the maximum accuracy score with 91.4%, while fedAvg had

89.7% validation accuracy.

This method consisted of two steps. In the first step, after calculating clients' validation accuracy in the server, we cluster clients into two groups considering their scores. Then we select the cluster with a higher average and disband clients inside the other cluster, considered fully malicious, from aggregation. In the second step, we apply SBA aggregation because of partially malicious clients inside the selected cluster. For example, according to table 5.3, in 50% malicious client rate, new methods achieved more than 80% validation score, in contrast to SBA and fedAvg, which was 82.0%, and got converged according to 5.5.

Moreover, we observed clients' loss values in all three scenarios, which were shown in 3.11, 5.3, 5.4, 4.3, 4.4. Only clients with the best, the worst, and the average performance were illustrated and converged. It is clear that clients were affected by the global model's performance, and if the global model's performance was high, their score was high too. The system's malicious clients could not converge in the first communication round. However, In the last communication round, although they had lower than benign clients' loss results, their loss values converged.

6.1 Future Works

In this Research, we focused on the collaboration of clients and non-IID scenarios of the environments. We developed a new aggregation method that considers clients' validation scores so that it would either join the aggregation with some ratio or be ejected. Partially malicious clients could not be separated from malicious clients so that they might join the aggregation. Fortunately, the SBA method can mitigate their effects considering their malicious rate. However, since the malicious rate is calculated by looking up the validation scores with unseen data if malicious clients somehow generate better results with this dataset, it might be selected for aggregation and harm the global model. Therefore, considering another value for calculating malicious rate must be necessary for future works.

Moreover, in real-world scenarios, more than these updates will be needed. We must consider other cases that will make global model training difficult and make the environment more realistic. Also, the attack type's main aim is not only degrading server performance but also stealing users' information. Therefore, we must focus on both cases and move one more step in our future work.

More Harmful Malicious Clients

The First thing we will focus on is the malicious clients. When a malicious client shares their data with other clients, they also lose its corrupted data, improving its local model's training since the less corrupted data. However, if a malicious client stores a copy of its corrupted data inside, it will still corrupt the model parameter updates. Also, if malicious clients can corrupt incoming data from benign clients, it will negatively affect its training and the number of pure data inside the environment. Therefore benign clients need to be conscious while giving their data away.

Intelligence defense Mechanism for Benign Clients

Moreover, we need to track malicious client detection at the client level. If clients can understand which clients are malicious, it will not accept data from them and may warn the other clients so that they will not receive this corrupted data. This process can be done by tracking accuracy changes for each local device itself. For example, if a client's local training validation accuracy reduces after receiving data from a particular client, it may understand that this is a malicious client. Also, sufficiently unseen validation data may not be available on the server. In this case, we have to use different methods to calculate validation scores for clients and the global model.

New Malicious Attack Types

Moreover, data poisoning is not the only attack type in federated learning. In section 2.3 we have seen other attack types, such as GAN-based attacks and Backdoor attacks. So, our global model must be conscious of these attacks, and we may develop additional methods for improving resiliency. For example, this method may work alongside SBA with K-Means Clustering. However, we must consider that the computational power should not exceed the proper limit. Otherwise, it will make the global server's job difficult.

Data leakage Attacks

Lastly, Federated learning has to consider the attacks that steal clients" information. This research only considered attack types that degrade server performance by corrupting the training data. The main aim of federated learning is securing data privacy. This feature separates federated learning from traditional centralized learning. Therefore, we

will investigate the data leakage attack in future works and how they steal the information indirectly from the user. Then we will apply two-step security methods. The first step will consider data privacy; the second will consider global model aggregation security. Data security is usually done by encrypting the clients' model weights. Decryption is done on the server side. We will enhance the encryption procedure and use it in our next project. Finally, we will combine overall previous ideas and make a highly secure method that secures user privacy and aggregation from malicious factors in our future works.

References

[1] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[2] S. Bharati, M. R. H. Mondal, P. Podder, and V. S. Prasath, "Federated learning:: Applications, challenges and future directions," *International Journal of Hybrid Intelligent Systems*, vol. 18, no. 1-2, pp. 19–35, 2022.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[5] D. M. Manias and A. Shami, "Making a case for federated learning in the internet of vehicles and intelligent transportation systems," *IEEE Network*, vol. 35, no. 3, pp. 88–94, 2021.

[6] E. Darzidehkalani, M. Ghasemi-Rad, and P. van Ooijen, "Federated learning in medical imaging: Part ii: Methods, challenges, and considerations," *Journal of the American College of Radiology*, 2022.

[7] B. Ghimire and D. B. Rawat, "Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things," *IEEE Internet of Things Journal*, 2022.

[8] B. Ghimire and D. B. Rawat, "Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things," *IEEE Internet of Things Journal*, 2022.

[9] T. Zhang, C. He, T. Ma, L. Gao, M. Ma, and S. Avestimehr, "Federated learning for internet of things," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 413–419, 2021.

[10] P. García Santaclara, A. Fernández Vilas, and R. P. Díaz Redondo, "Prototype of deployment of federated learning with iot devices," in *Proceedings of the 19th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pp. 9–16, 2022.

[11] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[12] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[13] P. Boopalan, S. P. Ramu, Q.-V. Pham, K. Dev, P. K. R. Maddikunta, T. R. Gadekallu, T. Huynh-The, *et al.*, "Fusion of federated learning and industrial internet of things: A survey," *Computer Networks*, p. 109048, 2022.

[14] F. A. KhoKhar, J. H. Shah, M. A. Khan, M. Sharif, U. Tariq, and S. Kadry, "A review on federated learning towards image processing," *Computers and Electrical Engineering*, vol. 99, p. 107818, 2022.

[15] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–37, 2022.

[16] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, "Federated learning in smart city sensing: Challenges and opportunities," *Sensors*, vol. 20, no. 21, p. 6230, 2020.

[17] Z. Li, "A personalized privacy-preserving scheme for federated learning," in *2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, pp. 1352–1356, IEEE, 2022.

[18] X. Xu, S. Niu, Z. Wang, D. Li, H. Yang, and W. Du, "Client selection based weighted federated few-shot learning," *Applied Soft Computing*, vol. 128, p. 109488, 2022.

[19] Y. Lu and L. Fan, "An efficient and robust aggregation algorithm for learning federated cnn," in *Proceedings of the 2020 3rd International Conference on Signal Processing and Machine Learning*, pp. 1–7, 2020.

[20] J. Reyes, L. Di Jorio, C. Low-Kam, and M. Kersten-Oertel, "Precision-weighted federated learning," *arXiv preprint arXiv:2107.09627*, 2021.

[21] M. Hong, S.-K. Kang, and J.-H. Lee, "Weighted averaging federated learning based on example forgetting events in label imbalanced non-iid," *Applied Sciences*, vol. 12, no. 12, p. 5806, 2022.

[22] M. Beaussart, F. Grimberg, M.-A. Hartley, and M. Jaggi, "Waffle: Weighted averaging for personalized federated learning," *arXiv preprint arXiv:2110.06978*, 2021.

[23] S. Kanaparthy, M. Padala, S. Damle, and S. Gujar, "Fair federated learning for heterogeneous data," in *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*, pp. 298–299, 2022.

[24] H. Jeong, J. An, and J. Jeong, "Are you a good client? client classification in federated learning," in *Intl. Conf. on Information and Communication Technology Convergence (ICTC)*, pp. 1691–1696, 2021.

[25] W. Yi Ming, L. Ge Hao, F. Li Yu, and P. Mao, "Research on block chain defense against malicious attack in federated learning," in *2021 The 3rd International Conference on Blockchain Technology*, pp. 67–72, 2021.

[26] E. Isik-Polat, G. Polat, and A. Kocyigit, "Arfed: Attack-resistant federated averaging based on outlier elimination," *Future Generation Computer Systems*, vol. 141, pp. 626–650, 2023.

[27] L. Meng, Y. Wei, R. Pan, S. Zhou, J. Zhang, and W. Chen, "Vadaf: visualization for abnormal client detection and analysis in federated learning," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 11, no. 3-4, pp. 1–23, 2021.

[28] L. Meng, Y. Wei, R. Pan, S. Zhou, J. Zhang, and W. Chen, "Vadaf: visualization for abnormal client detection and analysis in federated learning," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 11, no. 3-4, pp. 1–23, 2021.

[29] W. Huang, T. Li, D. Wang, S. Du, and J. Zhang, "Fairness and accuracy in federated learning," *arXiv preprint arXiv:2012.10069*, 2020.

[30] Y. Wang and B. Kantarci, "Reputation-enabled federated learning model aggregation in mobile platforms," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2021.

[31] S. Zhang, Z. Li, Q. Chen, W. Zheng, J. Leng, and M. Guo, "Dubhe: Towards data unbiasedness with homomorphic encryption in federated learning client selection," in *50th International Conference on Parallel Processing*, pp. 1–10, 2021.

[32] Z. Du, C. Wu, T. Yoshinage, L. Zhong, and Y. Ji, "On-device federated learning with fuzzy logic based client selection," in *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, pp. 64–70, 2022.

[33] N. Rodríguez-Barroso, E. Martínez-Cámara, M. V. Luzón, and F. Herrera, "Dynamic defense against byzantine poisoning attacks in federated learning," *Future Generation Computer Systems*, vol. 133, pp. 1–9, 2022.

[34] A. Boukerche, L. Zheng, and O. Alfandi, "Outlier detection: Methods, models, and classification," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–37, 2020.

[35] I. B. Mohamad and D. Usman, "Standardization and its effects on k-means clustering algorithm," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, no. 17, pp. 3299–3303, 2013.

[36] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[37] R. Gosselin, L. Vieu, F. Loukil, and A. Benoit, "Privacy and security in federated learning: A survey," *Applied Sciences*, vol. 12, no. 19, p. 9901, 2022.

[38] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[39] M. P. Sah and A. Singh, "Aggregation techniques in federated learning: Comprehensive survey, challenges and opportunities," in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 1962–1967, 2022.

[40] Z. Wang, J. Ma, X. Wang, J. Hu, Z. Qin, and K. Ren, "Threats to training: A survey of poisoning attacks and defenses on machine learning systems," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–36, 2022.

[41] F. Al-deen and F. Ba-Alwi, "A survey on unsupervised k-means algorithm in big data environment," *Asian Journal of Research in Computer Science*, pp. 1–8, 08 2021.

[42] D. Hassan Mahlool and M. Hamzah Abed, "A comprehensive survey on federated learning: Concept and applications," *arXiv e-prints*, pp. arXiv–2201, 2022.

[43] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021.

[44] L. Li, Y. Fan, and K.-Y. Lin, "A survey on federated learning," in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, pp. 791–796, IEEE, 2020.

[45] M. C. Eriş, B. Kantarci, and S. Oktug, "Unveiling the wireless network limitations in federated learning," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 262–267, IEEE, 2021.

[46] M. Aloqaily, O. Bouachir, A. Boukerche, and I. Al Ridhawi, "Design guidelines for blockchain-assisted 5g-uav networks," *IEEE network*, vol. 35, no. 1, pp. 64–71, 2021.

[47] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1759–1799, 2021.

[48] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[49] Y. Wang, B. Kantarci, and W. Mardini, "Aggregation of incentivized learning models in mobile federated learning environments," *IEEE Networking Letters*, vol. 3, no. 4, pp. 196–200, 2021.

[50] M. Asad, A. Moustafa, and T. Ito, "Federated learning versus classical machine learning: A convergence comparison," *arXiv preprint arXiv:2107.10976*, 2021.

[51] M. Shayan, C. Fung, C. J. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2020.

[52] Y. Chen, Y. Gui, H. Lin, W. Gan, and Y. Wu, "Federated learning attacks and defenses: A survey," *arXiv preprint arXiv:2211.14952*, 2022.

[53] D. Liu, L. Bai, T. Yu, and A. Zhang, "Towards method of horizontal federated learning: A survey," in *2022 8th International Conference on Big Data and Information Analytics (BigDIA)*, pp. 259–266, IEEE, 2022.

[54] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[55] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 374–380, IEEE, 2019.

[56] A. Prud'Homme and B. Kantarci, "Poisoning attack anticipation in mobile crowdsensing: A competitive learning-based study," in *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, pp. 73–78, 2021.

[57] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[58] G. Xia, J. Chen, C. Yu, and J. Ma, "Poisoning attacks in federated learning: A survey," *IEEE Access*, pp. 1–1, 2023.

[59] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, "Poisongan: Generative poisoning attacks against federated learning in edge computing systems," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3310–3322, 2020.

[60] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[61] G. J. Hussain and G. Manoj, "Federated learning: A survey of a new approach to machine learning," in *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, pp. 1–8, IEEE, 2022.

[62] H. Guo, Y. Mao, X. He, and H. Nie, "A blockchain-based grouped federated learning scheme against malicious clients," in *2021 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2021.

[63] D. Meng, H. Li, F. Zhu, and X. Li, "Fedmonn: meta operation neural network for secure federated aggregation," in *IEEE Intl. Conf. on High Performance Computing and Communications; IEEE Intl. Conf. on Smart City; IEEE Intl. Conf. on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 579–584, 2020.

[64] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[65] W. Wan, S. Hu, J. Lu, L. Y. Zhang, H. Jin, and Y. He, "Shielding federated learning: Robust aggregation with adaptive client selection," *arXiv preprint arXiv:2204.13256*, 2022.

[66] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2545–2555, 2022.

[67] K.-H. Chow and L. Liu, "Perception poisoning attacks in federated learning," in *IEEE Intl. Conf. on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pp. 146–155, 2021.

[68] Y. Ge and J. Zhou, "Blockchain based federated learning for object detection," in *Proceedings of the 2022 2nd International Conference on Control and Intelligent Robotics*, pp. 608–612, 2022.

[69] H. Zou, Y. Zhang, X. Que, Y. Liang, and J. Crowcroft, "Efficient federated learning under non-iid conditions with attackers," in *Proceedings of the 1st ACM Workshop on Data Privacy and Federated Learning Technologies for Mobile Edge Network*, pp. 13–18, 2022.

[70] J. Li, X. Zhang, and L. Zhao, "Robust federated learning based on metrics learning and unsupervised clustering for malicious data detection," in *Proceedings of the 2022 ACM Southeast Conference*, pp. 238–242, 2022.

[71] J. Li, J. Pei, and H. Huang, "Communication-efficient robust federated learning with noisy labels," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 914–924, 2022.

[72] L. Bhatia and S. Samet, "Decentralized federated learning: A comprehensive survey and a new blockchain-based data evaluation scheme," in *2022 Fourth International Conference on Blockchain Computing and Applications (BCCA)*, pp. 289–296, IEEE, 2022.

[73] L. Witt, M. Heyer, K. Toyoda, W. Samek, and D. Li, "Decentral and incentivized federated learning frameworks: A systematic literature review," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[74] F. Nuding and R. Mayer, "Data poisoning in sequential and parallel federated learning," in *Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics*, pp. 24–34, 2022.

[75] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–35, 2022.

[76] A. Boukerche, A. J. Siddiqui, and A. Mammeri, "Automated vehicle detection and classification: Models, methods, and techniques," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–39, 2017.

[77] B. Zhang, J. Wang, J. Fu, and J. Xia, "Driver action recognition using federated learning," in *2021 the 7th International Conference on Communication and Information Processing (ICCIP)*, pp. 74–77, 2021.

[78] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[79] A. Boukerche and Z. Hou, "Object detection using deep learning methods in traffic scenarios," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–35, 2021.

[80] S. Bhatnagar, D. Ghosal, and M. H. Kolekar, "Classification of fashion article images using convolutional neural networks," in *Intl. Conf. on Image Information Processing (ICIIP)*, pp. 1–6, 2017.

[81] M. Cococcioni, F. Rossi, E. Ruffaldi, and S. Saponara, "A novel posit-based fast approximation of elu activation function for deep neural networks," in *IEEE Intl. Conf. Smart Computing*, pp. 244–246, 2020.

[82] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, 2022.

[83] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[84] B. Graham, "Fractional max-pooling," *arXiv preprint arXiv:1412.6071*, 2014.

[85] Z. Zhu, H. Sun, and C. Zhang, "Effectiveness of optimization algorithms in deep image classification," *arXiv preprint arXiv:2110.01598*, 2021.

[86] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[87] H. Yessou, G. Sumbul, and B. Demir, "A comparative study of deep learning loss functions for multi-label remote sensing image classification," in *Int. Geoscience and Remote Sens. Symp.*, pp. 1349–1352, 2020.

[88] S. Kornblith, T. Chen, H. Lee, and M. Norouzi, "Why do better loss functions lead to less transferable features?," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28648–28662, 2021.

[89] B. Lucena, "Loss functions for classification using structured entropy," *arXiv preprint arXiv:2206.07122*, 2022.

[90] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: a new paradigm to machine learning," *Archives of Computational Methods in Engineering*, vol. 27, pp. 1071–1092, 2020.

[91] S. Kornblith, T. Chen, H. Lee, and M. Norouzi, "Why do better loss functions lead to less transferable features?," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28648–28662, 2021.

[92] H. Yin, P. Liu, K. Liu, L. Cao, L. Zhang, Y. Gao, and X. Hei, "ns3-ai: Fostering artificial intelligence algorithms for networking research," in *Proceedings of the 2020 Workshop on Ns-3*, pp. 57–64, 2020.

[93] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

[94] T. Begin, A. Busson, I. Guérin Lassous, and A. Boukerche, "Video on demand in ieee 802.11 p-based vehicular networks: Analysis and dimensioning," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 303–310, 2018.

[95] Z. Zhang, A. Boukerche, and H. Ramadan, "Design of a lightweight authentication scheme for ieee 802.11 p vehicular networks," *Ad Hoc Networks*, vol. 10, no. 2, pp. 243–252, 2012.

[96] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[97] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[98] L. Yu, W. Nie, L. Xin, and M. Guo, "Clustered federated learning based on data distribution," in *2021 3rd International Conference on Advanced Information Science and System (AISS 2021)*, pp. 1–5, 2021.

[99] N. Wang, Y. Xiao, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Flare: Defending federated learning against model poisoning attacks via latent space representations," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pp. 946–958, 2022.

[100] A. I. Károly, R. Fullér, and P. Galambos, "Unsupervised clustering for deep learning: A tutorial survey," *Acta Polytechnica Hungarica*, vol. 15, no. 8, pp. 29–53, 2018.

[101] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[102] J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618," *Genetic Programming and Evolvable Machines*, vol. 19, no. 1-2, pp. 305–307, 2018.

[103] M. K. Pakhira, "A linear time-complexity k-means algorithm using cluster shifting," in *2014 international conference on computational intelligence and communication networks*, pp. 1047–1051, IEEE, 2014.