TU WIEN Informatics

Achieving Sustainable Federated Edge Analytics by Using Incomplete Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Mr. Paul Joe Maliakel, BSc
Matrikelnummer 12012422

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Ivona Brandić
Mitwirkung: Univ.Ass. Dr. Shashikant Shankar Ilager

Wien, 4. September 2023

Paul Joe Maliakel	Ivona Brandić

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Achieving Sustainable Federated Edge Analytics by Using Incomplete Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Mr. Paul Joe Maliakel, BSc
Registration Number 12012422

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Ivona Brandić
Assistance: Univ.Ass. Dr. Shashikant Shankar Ilager

Vienna, 4th September, 2023

Paul Joe Maliakel	Ivona Brandić

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Mr. Paul Joe Maliakel, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. September 2023

Paul Joe Maliakel

Danksagung

Zuallererst möchte ich meine tiefste Dankbarkeit gegenüber Professorin Ivona Brandić zum Ausdruck bringen, die mich während der gesamten Zeit, in der ich an dieser Abschlussarbeit gearbeitet habe, unermüdlich unterstützt und wertvolle Anleitung geboten hat. Ihre Expertise und Einblicke waren ein Leitstern auf meinem Weg und haben mir die dringend benötigte Orientierung und Weisheit geboten. Ihr konstruktives Feedback und ihre Ermutigung haben nicht nur die Qualität dieser Forschung verbessert, sondern auch meine akademischen Bestrebungen geprägt.

Ein besonderer Dank gilt Shashikant Ilager, dessen Rückmeldungen und Ratschläge von unschätzbarem Wert waren, insbesondere in herausfordernden Phasen dieses Projekts. Seine pragmatischen Tipps und seine erfahrene Perspektive haben meine Arbeit erheblich unterstützt, und dafür bin ich zutiefst dankbar.

Ich schulde meiner Familie und meinen Freunden unermessliche Dankbarkeit. Meinen Eltern, deren Liebe, Weisheit und Opfer immer meine konstante Kraftquelle waren, spreche ich meine herzlichsten Dank aus. Ihr Glaube an meine Fähigkeiten und ihre anhaltende Ermutigung waren entscheidend, um diesen akademischen Meilenstein zu erreichen. Zudem haben die unerschütterliche Liebe und Unterstützung meiner Freunde als meine Säulen gedient, indem sie emotionalen Halt und Motivation geboten haben. Ihre Kameradschaft und ihr Glaube an meine Fähigkeiten haben die Höhen und Tiefen dieser akademischen Reise leichter erträglich gemacht.

Acknowledgements

First and foremost, I extend my deepest gratitude to Professor Ivona Brandić for her unflagging support, invaluable guidance, and mentorship throughout the entire journey of working on this thesis. Her expertise and insights have been a beacon that lighted my path, providing much-needed direction and wisdom. Her constructive feedback and encouragement have not only improved the quality of this research but also shaped my academic pursuits.

Special thanks go to Shashikant Ilager, whose feedback and counsel were invaluable, particularly during challenging phases of this project. His pragmatic advice and seasoned perspective greatly aided my work, and for that, I am profoundly grateful.

I owe an immeasurable debt of gratitude to my family and friends. To my parents, whose love, wisdom, and sacrifices have been my constant source of strength, I offer heartfelt thanks. Their belief in my abilities and their enduring encouragement have been instrumental in reaching this academic milestone. Additionally, the unwavering love and support from my friends have been my pillars, providing emotional sustenance and motivation. Their camaraderie and belief in my capabilities have made the highs and lows of this academic journey easier to bear.

Kurzfassung

Im rasanten Bereich des Edge Computings sticht das Föderierte Lernen als bedeutende Innovation hervor, um Datenanalytik in dezentralisierten Netzwerken durchzuführen. Dieser Ansatz ermöglicht nicht nur eine unmittelbare Datenverarbeitung, sondern steigert auch die Datenschutzanforderungen der beteiligten Daten. Die Nachhaltigkeit solcher föderierten Systeme wird jedoch durch das wiederkehrende Problem unvollständiger Daten gefährdet, insbesondere im Bereich tabellarischer Daten, die eine Mischung aus kategorialen und numerischen Variablen sowie unausgeglichene Klassendistributionen aufweisen. Dies erschwert die Erzielung genauer und verlässlicher Ergebnisse mit föderierten maschinellen Lernalgorithmen. Um die Nachhaltigkeit des Föderierten Edge-Analytics zu stärken, untersucht diese Arbeit den Einsatz von Generative Adversarial Networks (GANs), um die Leistung des föderierten Lernens in Umgebungen mit unvollständigen Daten zu verbessern. Drei spezifische GAN-Techniken werden eingeführt, von denen jede ihren eigenen Schwerpunkt hat. Das *Federated Classwise Sampling GAN* zielt auf Ungleichgewichte in der Klassendistribution ab und trainiert separate GAN-Modelle für jede Klassenbezeichnung. Das *Federated Classwise Sampling with Client Grouping GAN* fügt dem Training eine zusätzliche Stabilitätsebene hinzu, indem es Clients aufgrund ähnlicher Stichprobengrößen für bestimmte Klassenbezeichnungen gruppieret. Die von diesen GANs generierten synthetischen Daten werden verwendet, um föderierte Lernmodelle zu trainieren und ihre Effektivität bei der Milderung der Nachteile unvollständiger Daten zu erforschen. Die Arbeit wird anhand von drei unterschiedlichen tabellarischen Datensätzen ausgewertet. Die Ergebnisse der Studie haben gezeigt, dass das *Federated Classwise Sampling GAN* und das *Federated Classwise Sampling with Client Grouping GAN* die Modellgenauigkeit um 4% bzw. 17% für die Datensätze Ädultünd Ïntrusion"verbessert haben. Bemerkenswert ist, dass unsere vorgeschlagene Technik des *Federated Classwise Sampling with Client Grouping GAN* Stabilität in der Genauigkeit und weniger Ausführungszeit gezeigt hat, was sie besonders für nachhaltiges Föderiertes Edge-Analytics in realen Szenarien, in denen Datenunvollständigkeit eine Herausforderung darstellt, geeignet macht. Zusammenfassend hebt diese Arbeit das transformative Potenzial von GAN-Techniken zur Verbesserung von föderierten Lernmodellen im Kontext unvollständiger Daten hervor. Insbesondere zeigt unsere vorgeschlagene Methode des *Federated Classwise Sampling with Client Grouping GAN* sowohl Stabilität als auch Effizienz und positioniert sie als nachhaltige Lösung für reale Szenarien, in denen Datenunvollständigkeit eine Rolle spielt.

Abstract

In the fast-paced world of edge computing, Federated Learning stands out as a key innovation for performing data analytics across decentralized networks. This approach not only allows for immediate data processing but also enhances the privacy of the data involved. However, the sustainability of such federated systems is jeopardized by the recurring issue of incomplete data, particularly in the realm of tabular data featuring a mix of categorical and numerical variables, and imbalanced class distributions. This makes it difficult to achieve accurate and dependable results with federated machine learning algorithms. To fortify the sustainability of Federated Edge Analytics, this thesis investigates the application of Generative Adversarial Networks (GANs) to improve federated learning performance in environments characterized by incomplete data. Three specific GAN techniques are introduced, each with its unique focus. *Federated GAN* serves as a existing generalized approach without specialization, potentially less efficient for complex, imbalanced datasets. *Federated Classwise Sampling GAN* targets imbalances in class distribution, training separate GAN models for each class label. *Federated Classwise Sampling with Client Grouping GAN* adds an extra layer of stability to training by grouping clients based on similar sample counts for specific class labels. The synthetic data generated by these GANs are used to train federated learning models, exploring their effectiveness in mitigating the drawbacks of incomplete data. The thesis is evaluated across three distinct tabular datasets. The study's findings have revealed that, while *Federated Classwise Sampling GAN* and *Federated Classwise Sampling with Client Grouping GAN* improved model accuracy by 4% and 17% for the Adult and Intrusion datasets, respectively, the simple *Federated GAN* actually led to decreased performance. Remarkably, our proposed *Federated Classwise Sampling with Client Grouping GAN* technique demonstrated stability in accuracy and less execution time, making it highly suitable for sustainable Federated Edge Analytics in real-world scenarios where data incompleteness is a challenge. In summary, this thesis highlights the transformative potential of GAN techniques in enhancing federated learning models in the context of incomplete data. Notably, our proposed *Federated Classwise Sampling with Client Grouping GAN* demonstrates both stability and efficiency, positioning it as a sustainable solution for real-world scenarios where data incompleteness is a concern.

Keywords: Federated Learning, Incomplete Data, Generative Adversarial Networks (GANs), Federated Edge Analytics, Class Imbalance

Contents

CHAPTER 1

Introduction

1.1 Background and motivation

Edge computing [1] has evolved as a promising approach for efficient and decentralized data processing particularly in the field of Internet of Things (IoT) [2] applications. It enables data to be processed and analyzed at the network edge that is closer to the data sources therefore resulting in reduced latency and reduced burden on centralized servers. This approach facilitates real time data analysis, quick response times and enhanced privacy.

In this regard, federated learning [3] has drawn attention as a decentralized machine learning strategy suitable for sustainable edge computing. It enables collaborative model training without the need to share raw data. Multiple edge devices participate in the training process by contributing their local data while preserving data privacy. This collaborative approach facilitates the development of sustainable, robust and accurate models without compromising sensitive user data.

However, a significant challenge in federated edge analytics is the presence of incomplete data. In real-world edge environments, data incompleteness can arise due to network connectivity issues, device failures or privacy constraints. There is also a specific case where data samples are not enough for one or more labels in clients. This incomplete data poses a significant hurdle for training sustainable, accurate and reliable models at the edge.

In today's data concentrated world,we encounter a wide range of data types, from textual contents in linguistic projects for Natural Language Processing to images employed in computer vision applications. Tabular data is particularly prevalent in this broad domain and is found in industries including finance, retail, and e-commerce. This type of structured data usually comprises of both discrete and continuous columns with potential

challenge of uneven distributions. Such complexities make it arduous to achieve sustained model performance.

Training the models on incomplete or imbalanced data can lead to bad performance. Due to privacy concerns, as seen in industries like healthcare or finance, it is frequently impossible and even illegal to aggregate all the data on a single server for a more balanced training. Therefore working with incomplete data is a necessity leading to compromise in the model quality and sustainability.

Techniques like Random oversampling(ROS) [4] and SMOTE [5] are frequently used to handle data incompleteness. This kind of methods are straighforward and often doesn't capture the underlying data distribution leading to biased results. Recognizing these limitations of the sampling techniques the recent shift has been towards more sustainable techniques. Generative Adversarial Networks (GANs) [6] are a promising alternative which effectively look at these challenges. Despite generative models' success in text and image synthesis, there remains a glaring gap in federated generative models designed for generative tabular data. When considering tabular data encountered for example in hospital settings and other sectors, additional challenges arise. Tabular data often consist of a mix of discrete and continuous columns with imbalanced distributions, which complicates model training and generalization.

To address these challenges, this research focuses on leveraging Generative Adversarial Networks (GANs) [6] to tackle the problem of incomplete data in federated edge analytics. GANs are powerful generative models capable of learning the underlying data distribution and generating synthetic instances that can closely resemble real data. The quality and completeness of local datasets can be improved by using GANs to impute missing or incomplete data at the edge in the federated learning framework.

The goal of this study is to look into and suggest new approaches for imputing missing class labels and incomplete data in the context of federated edge analytics. This project attempts to produce synthetic data samples that capture the intricate patterns contained in the data by creating GAN-based imputation approaches. Then, more precise models can be trained using these imputed datasets at the edge.

The motivation behind this research stems from the need to address the limitations of traditional approaches that rely on sharing complete data for centralized training. In edge analytics, sharing raw data jeopardizes user privacy and presents issues with data security. By enabling edge devices to jointly train models without disclosing private information, federated learning offers a privacy-preserving approach. However, the issues with missing data in the context of tabular data analytics are not sufficiently addressed by the federated learning techniques now in use.

The significance of this research lies in its potential to improve model training, decision making and resource optimization in real-world edge computing scenarios by addressing the challenges of incomplete data in federated edge analytics and leveraging GAN-based imputation techniques. By effectively handling incomplete data, federated edge analytics can leverage the collective knowledge of distributed edge devices to train

accurate models thus leading to improved decision making, predictive capabilities and resource optimization. Furthermore, the privacy-preserving nature of federated learning, combined with GAN-based data imputation, ensures that sensitive data remains secure and protected, making the system both effective and sustainable.

The findings of this study could have applications in a number of fields, such as healthcare, banking, and IoT applications, where edge analytics is essential. The research findings can help to address important sustainability issues in federated edge analytics by guiding the creation of more effective and privacy-preserving federated learning systems.

1.2 Problem statement

The focus of this study is on the issue of incomplete data or missing data in the realm of sustainable federated edge analytics. In practical edge computing environments, incomplete data sets can emerge from a variety of factors like unstable network connections, equipment malfunctions or data privacy limitations. Such gaps in the data substantially hinder the ability to develop reliable and accurate machine learning models using federated learning. Classical techniques such as Batch Gradient Descent [7], Support Vector Machines [8] and k-Means Clustering [9], which depend on having full data sets for training are not well-suited for edge analytics, primarily because of concerns over data privacy and the necessity for localized data handling. In addition, structured data, often found in sectors like healthcare, finance, and retail, typically includes a complicated mix of both categorical and numerical variables along with skewed distributions. Due to this complexity, it is considerably more difficult to train models that can comprehend the data's underlying intricacies and generalize them to different contexts. These issues demand innovative solutions that can efficiently manage missing data while leveraging the advantages of federated learning, all while respecting data privacy and making sure the trained models are both strong and widely applicable. So, using a set of research questions, we define the parameters of our study.

1.3 Research questions

- **RQ1: How does the accuracy of a federated learning model change across different levels of data incompleteness??**
  To explore the influence of data incompleteness/non-IIDness on the accuracy of federated learning models: This research question delves into understanding the variability in the accuracy of federated learning models under varying degrees of data incompleteness or non-IIDness. Such conditions are simulated using different alpha values within a Dirichlet distribution.

- **RQ2: How will the accuracy of the federated learning model change with the addition of GAN-generated synthetic data to the original incomplete data??**

To assess the impact of GAN-generated data on the accuracy of federated learning models: The focus of this inquiry is to gauge the changes in model accuracy when GAN-generated synthetic data is introduced to the original dataset. This investigation aims to discern the potential advantages or drawbacks of integrating synthetic data into federated learning contexts.

- **RQ3: What is the quality of the generated dataset compared to the original dataset??**
  To evaluate the quality of GAN-generated datasets relative to original datasets: This research seeks to provide a comparative analysis between original datasets and their GAN-generated counterparts. By doing so, it intends to establish benchmarks for dataset quality and to discern the practicality and viability of using GANs for data generation in federated learning environments.

1.4 Overview of the Thesis Structure:

The thesis will be structured to address the research questions and achieve the research objectives outlined above. The Chapter background 2 provides provides an overview of problem domain, motivation behind the research, the problem statement, and the research objectives. The introduction will also highlight the challenges associated with handling incomplete data in the federated learning setting.

The Chapter literature survey 3 gives a comprehensive literature review, covering existing methods for handling incomplete data in federated learning, learning-based approaches for data imputation, and the challenges specific to tabular data with missing class labels. This review will provide a foundation for the proposed research and identify gaps in the existing literature.

The methodology chapter 4 will describe the approach taken to address the research objectives, including the specific techniques and algorithms employed for data imputation in the federated learning setting. This section will also outline the evaluation metrics and experimental setup used to assess the effectiveness and fairness of the proposed method.

The results chapter 5 will present the findings of the research, including the performance of the proposed approach compared to existing methods, the fairness of the imputed dataset, and the relationship between effectiveness and efficiency in the federated learning process.

Finally, the thesis will conclude 5 with a summary of the research findings, limitations of the research, and suggestions for future work in this area.

CHAPTER 2

Background

This chapter covers a comprehensive introduction to various concepts in machine learning, including supervised learning, unsupervised learning, reinforcement learning, and semi-supervised learning. It also introduces common machine learning models, such as linear regression [10], logistic regression [11], decision trees [12], random forests [13], support vector machines [14], k-nearest neighbors [15], naive Bayes [16], neural networks [17], gradient boosting [18] , and k-means clustering [9].

The content dives into regression and classification, explaining how regression predicts continuous output based on input features, while classification categorizes input data into predefined classes. It introduces key classification algorithms like decision tree classifiers, random forest classifiers, and support vector machines, along with visual examples of how they work.

Deep learning is introduced as a subset of machine learning, focusing on multi-layered neural networks that can learn complex patterns. The content explains the basics of neural networks, feedforward networks, and their training process.

The concept of federated learning is introduced, describing a decentralized approach to machine learning where devices collaborate to train a global model without sharing raw data. The federated learning process is explained using an example of next-word prediction on mobile phones.

The challenges of federated learning are discussed, including expensive communication, systems and statistical heterogeneity, and privacy concerns. The content also introduces Generative Adversarial Networks (GANs), describing their architecture, training process, and challenges like gradient vanishing and mode collapse.

The concept of Wasserstein GAN (WGAN) is introduced, explaining its use of the Wasserstein distance for stable training and its application in addressing gradient vanishing and mode collapse challenges. The WGAN with Gradient Penalty (WGAN-GP) variant

is discussed as an improvement over traditional WGAN, addressing weight clipping challenges.

Finally, the content introduces the Dirichlet distribution and its relevance in modeling proportions of categories in non-IID datasets within federated learning, emphasizing its potential applications in preserving data privacy and improving model updates

2.1 Introduction to Machine Learning

Machine learning consists a wide range of techniques and algorithms that allow computers to learn patterns and relationships from data. There are four different categories for machine learning:

In **supervised learning**, algorithms undergo training through labeled datasets, where each data point is linked to a known target or class label. Through this process, algorithms internalize the connection between input attributes and output labels, enabling them to predict outcomes for brand-new, unreported data occurrences.

Unsupervised learning pertains to identifying inherent patterns and structures within unlabeled data. Unsupervised learning include activities like dimensionality reduction and clustering, which use algorithms to find meaningful clusterings or reduce data complexity.

Semi-Supervised Learning learning combines elements of supervised and unsupervised learning. A well-balanced use of the information is achieved by combining a small amount of labeled data with a larger pool of unlabeled data, which improves the performance of the model.

The goal of the specialized method known as **Reinforcement learning** is to teach agents how to make decisions sequentially in a given environment in order to maximize rewards. This tactic is useful in situations where agents interact with their surroundings to learn new things, fine-tuning their activities in response to input received from the environment.

2.2 Classification

The task of classifying input data points into preset groupings or categories is an example of a supervised learning problem. Classification generates discrete output labels as opposed to regression, which produces continuous output. Each data point associated with a class label is used to train classification models on labeled data. Deep neural networks, decision trees, logistic regression, and support vector machines are examples of common categorization algorithms.

Table 2.1: Common Machine Learning Models

Model	Description
Linear Regression	Predicts continuous target variable based on input features
Logistic Regression	Classifies data into binary categories using logistic function
Decision Trees	Constructs tree-like structure for decisions based on features
Random Forest	Ensemble of decision trees for improved accuracy
Support Vector Machines	Finds hyperplane to separate data into classes
K-Nearest Neighbors	Classifies data based on majority class of k-nearest neighbors
Naive Bayes	Applies Bayes' theorem for probabilistic classification
Neural Networks	Multi-layered networks that learn complex patterns
Gradient Boosting	Iteratively builds models to correct errors
K-Means Clustering	Divides data into clusters based on similarity

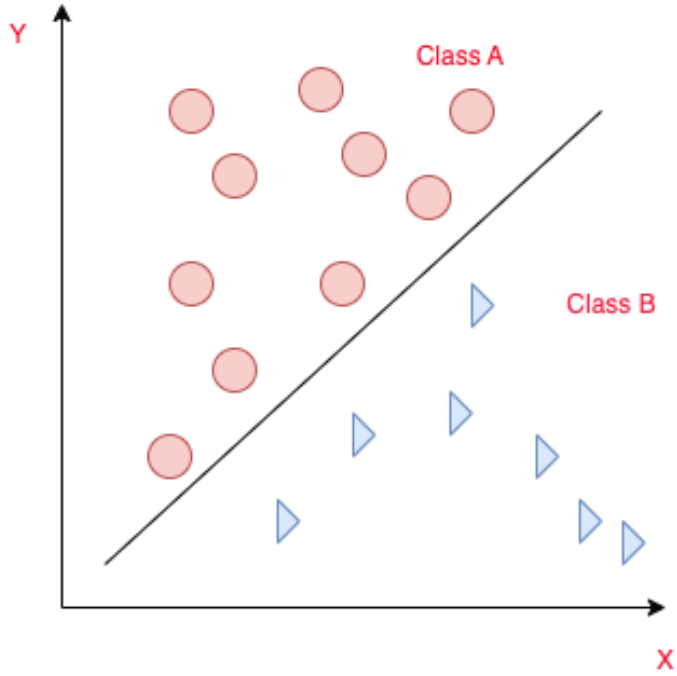

Figure 2.1: A simple example of how a classification algorithm classifies samples

2.2.1 Decision Tree Classifier

Consider yourself faced with a succession of decisions, each of which leads you down a different road. A decision tree classifier is like a practical tool that enables machines to make decisions similarly to human beings. It is frequently used in machine learning to group items. The tree has a starting point, similar to how you make your first choice. Then, it makes a decision after considering precise facts about the item you're trying to

sort at each step. Until it decides where your stuff belongs, these options keep diverging. This process helps the tree learn from examples so that it can make similar choices for new things in the future. The figure 2.2 is an example of a decision tree which uses 3 features to classify.

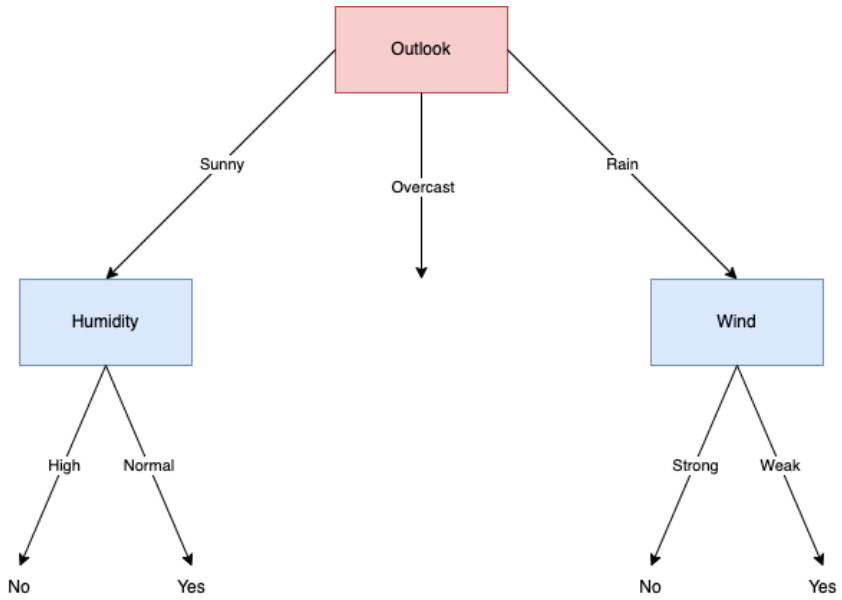

Figure 2.2: An example of a decision tree which uses 3 features to classify.

2.2.2 Random Forest Classifier

Visualize a forest filled with unique trees, each with its own distinct perspective. Think of a random forest classifier as a group of these trees collaborating to make more intelligent predictions. Instead of depending solely on a single tree, a random forest generates multiple trees. Every tree learns from different subsets of information and forms its own interpretations. When a prediction is needed, each tree provides its input, and the ultimate prediction reflects the majority consensus among the trees. This approach prevents the classifier from relying excessively on patterns specific to its training examples, enhancing its ability to make accurate predictions for novel, unseen scenarios.

2.2.3 Support Vector Machine (SVM)

Think about having two separate, obviously different sets of points. Imagine the task of determining the best line to draw between these sets. A support vector machine (SVM) achieves this exact goal. It's a smart tool made to figure out the best line (or perhaps a more complicated form) to use to maximize the distance between these point sets. The chosen line is positioned so that it keeps a maximum distance from each set's closest spots. These specific places act as the line's "support" from the ground up.

Even in situations where the points cannot be clearly separated by a simple line, SVMs perform well. They accomplish this by changing how they view the issues, which is akin to using a magnifying glass to have a clearer understanding. SVMs are capable of handling situations where the goal is to divide elements into two groups or more, and their efficiency is especially clear in complex circumstances when traditional patterns are missing.The figure 2.3 is an example of a SVM that classifies samples into 2 groups.

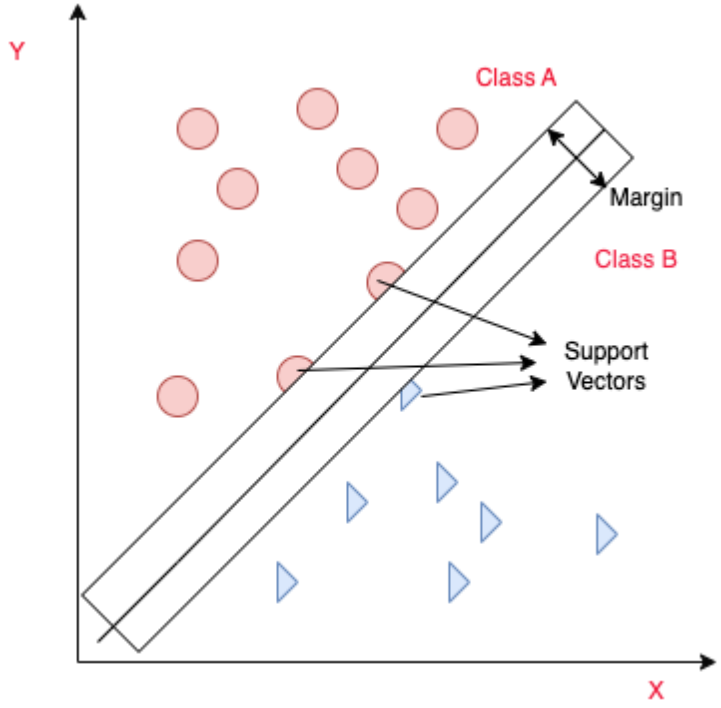

Figure 2.3: Support Vector Machine

2.3 Deep Learning

Deep learning is a specific type of machine learning that uses multi-layered, complicated neural networks to evaluate and comprehend complex data patterns. These networks, often referred to as deep neural networks, are made up of interconnected layers that process input data in a methodical manner via a sequence of operations. They can capture complex properties and representations present in the data thanks to this iterative approach. Deep learning research has achieved significant advances in a number of sectors, including but not limited to speech and image recognition, natural language processing, autonomous vehicles, and other areas.

2.3.1 Neural networks

In the field of machine learning, neural networks (NNs) are a powerful tool that are widely used in a variety of disciplines. They are excellent at recognizing complex data patterns, handling ambiguity, and accurately projecting multiple outcomes. However, a disadvantage is that in some situations, their necessity for significant, high-quality data makes it difficult to achieve exact learning. Although NNs are based on complex ideas, their use has been made simpler by the availability of free tools and a larger pool of experienced practitioners.

Feedforward Network

We use a feedforward network for our Federated classification tasks. These networks, commonly referred to as neural networks, play a pivotal role in both machine learning and deep learning. They are particularly adept at tasks such as classification and regression. Operating as a one-way conduit, data progresses through these networks from input to output, without any circular paths.There are three distinct layers in the network: input, hidden, and output. Through weights, which are honed during training, neurons within these layers link with one another. Intricacy is introduced by activation functions, which direct data transformation as it moves through the network. The network adjusts these weights throughout training using methods like gradient descent to improve predictions. You can refer to Figure 2.4 for a basic depiction of a feedforward neural network's architecture.

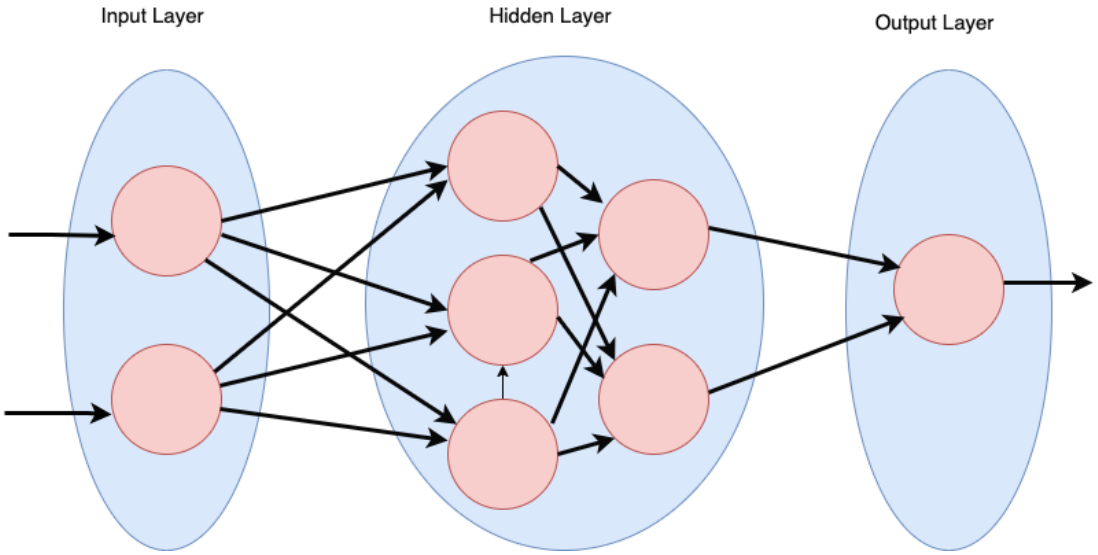

Figure 2.4: A simple feed-forward neural network with dense layers

2.4 Introduction to Federated Learning

Imagine an illustrative scenario where federated learning is applied to enhance next-word prediction on mobile devices. The goal is to develop a predictive system that can provide precise predictions for the next word a user might type in. Instead of sending raw data to a central server, the technique trains the model decentralized to maintain data secrecy and save network burden.

In this setup, remote devices or clients communicate with a central server on a regular basis to jointly improve a global model. A selected group of mobile phones uses their unique, non-uniformly distributed user data (in our instance, incomplete data) to conduct localized training during each communication cycle. The server receives the model updates from these remote devices, which reflect their local learning. The server incorporates these adjustments and then sends the improved global model to an additional group of devices.

This network-wide iterative training procedure continues until a convergence benchmark is reached or a predetermined stopping point is reached. Federated learning enables the development of an accurate next-word predictor by distributing the training process and just sending model updates as opposed to raw data. This strategy protects user privacy while reducing network load. The figure 2.5 is an example of a federated learning setup with 3 clients.

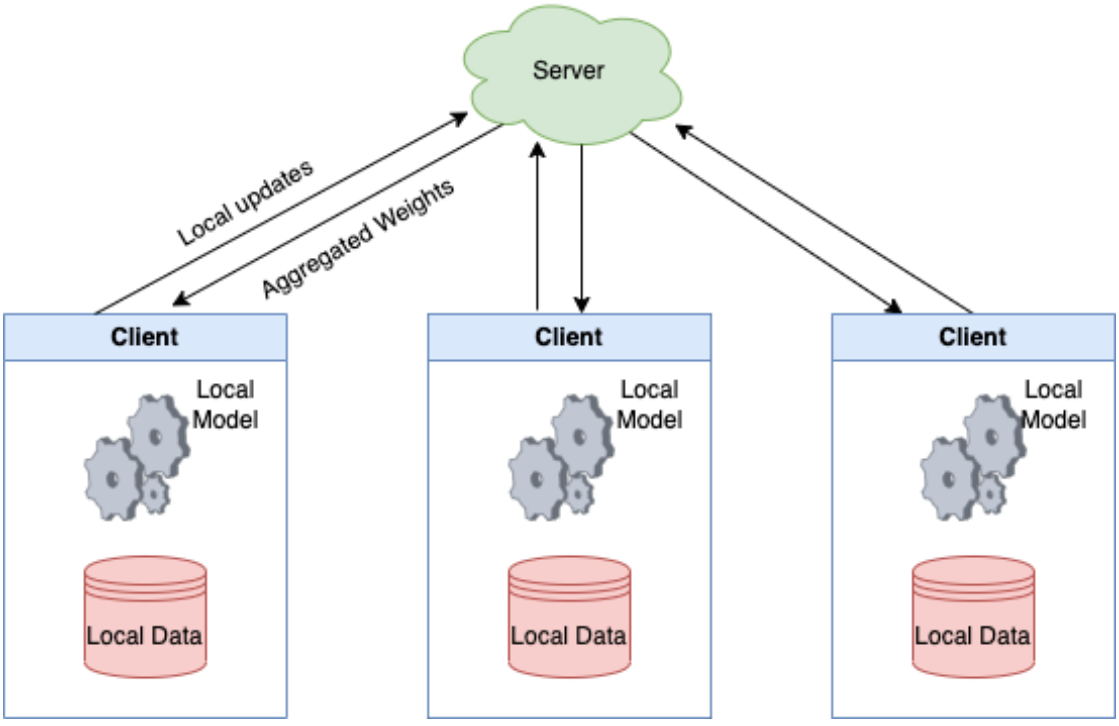

Figure 2.5: A Federated Learning setup

2.5 FL - Problem Formulation

Training a central model with data gathered from various client devices is the main problem in federated learning. The fundamental issue is that the data produced by these devices is localized to each of their specific locations, with sporadic updates provided to a centralized server. When minimizing an objective function that includes the comprehensive learning objective, inputs from each device's unique localized data are taken into account. This objective function embodies the need to accommodate the decentralized nature of the data while training a coherent global model.

$$\min_w F(w), \text{ where } F(w) := \sum_{k=1}^{m} p_k F_k(w) \tag{2.1}$$

In this context, let m denote the overall count of devices, where $p_k \geq 0$ and $\sum_k p_k = 1$. The variable F_k represents the local objective function for the k-th device. Typically, this local objective function is defined as the empirical risk calculated over the device's specific data. In other words, $F_k(w)$ is computed as $\frac{1}{n_k} \sum_{j=1}^{n_k} f_{j,k}(w; x_{j,k}, y_{j,k})$, where n_k stands for the number of locally available samples. The user-defined term p_k serves to determine the relative influence of each device, with common selections being $p_k = \frac{1}{n}$ or $p_k = \frac{n_k}{n}$, where $n = \sum_k n_k$ signifies the total sample count. It's important to recognize that while the problem defined as (1) holds as the conventional formulation, other objectives or modeling strategies might be appropriate depending on the specific application context.

2.6 FL - Incompleteness in Federated Learning

2.6.1 Defining Incompleteness

In federated learning, "data incompleteness" encapsulates two main aspects: uneven distribution of data classes across devices and disparity in data volume among different devices.

Uneven Distribution of Data Classes: Devices in federated networks often possess data that reflects the unique behavior or characteristics of their users. This can lead to scenarios where certain data classes are predominantly present on some devices while scarcely available or entirely missing on others.

Data Volume Disparity: Beyond the type or class of data, there's a disparity in the sheer volume of data across devices. Some devices, due to more frequent usage or longer operation times, might generate or collect larger volumes of data compared to others.

Both these facets of incompleteness present unique challenges in model training and can significantly influence the generalizability and performance of federated models [19].

2.6.2 Contrasting Incompleteness with Other FL Properties

Non-IID Data: Non-Identically and Independently Distributed (non-IID) data indicates that the distribution of data on each device differs from the overall data distribution. Although the uneven distribution of data classes (an element of inadequacy) can contribute to non-IIDness, non-IIDness can also exist without data inadequacy. For example, even if all devices contain instances of all classes, the nuances and specifics of the data can be distinct enough to result in non-IID characteristics.

Data Skewness vs. Uneven Distribution: While both terms refer to irregularities in data distribution, data skewness primarily addresses the balance among data classes on a specific device. Conversely, uneven distribution, encompassed by inadequacy, pertains to the complete absence or minimal occurrence of particular classes on devices.

Having dissected these concepts, it's clear that data incompleteness, with its multifaceted nature, is a critical aspect to consider in federated learning, distinct from other data-related challenges.

2.7 Other FL Challenges

For federated learning to be successfully implemented, a number of issues must be resolved. These difficulties include expensive communication, systems heterogeneity, statistical heterogeneity, and privacy concerns [19].

Expensive Communication: In federated networks, communication is a significant bottleneck, particularly when dealing with a high number of devices. The solution to this problem is effective communication techniques. Instead of broadcasting the complete dataset, these strategies concentrate on sending brief messages or model updates. The number of communication rounds and the amount of transmitted messages should be kept to a minimum.

Systems Heterogeneity: The range of storage capacities, computational abilities, communication features, and reliability across devices in federated networks is wide. This diversity creates challenges, particularly in dealing with slow participants, ensuring fault tolerance, and adapting to fluctuating hardware and network conditions. It's crucial for federated learning approaches to manage disconnected devices, adapt to a multitude of hardware configurations, and prepare for minimal device engagement.

Statistical Heterogeneity: The data contributed by devices in federated networks often vary in distribution, completeness, and volume. This disrupts the common assumptions of data being Independent and Identically Distributed (I.I.D.) or Incomplete often used in distributed optimization methods. The complexity of modeling, analyzing, and evaluating the system increases due to this type of variability. Techniques like multi-task learning and meta-learning can be applied to manage this diversity, as they can either tailor models to individual requirements or concurrently train multiple localized models.

Privacy Concerns: Privacy is a significant concern in federated learning. Sharing model updates rather than raw data offers some safety, but it still has the potential to divulge sensitive data. Though they may have an influence on model performance and system efficiency, techniques like differential privacy and secure multiparty computation strive to improve privacy. In private federated learning systems, finding a good balance between privacy and use is difficult.

2.8 Federated Averaging (FedAvg):

The Federated Averaging (FedAvg) method is the cornerstone of the Federated Learning field. The significant study "Communication-Efficient Learning of Deep Networks from Decentralized Data" by Google researchers introduced it. [20].

FedAvg's main goal is to cooperatively train a global model using data dispersed among a variety of clients, which could be devices or edge nodes, all the while avoiding sharing raw data.

All participating customers are given access to the global model throughout each training session. Without sending the data to a central server, each client does local model training using its own local data. The clients send their modified model weights back to the central server after local training. The acquired model weights are combined by the central server using a weighted average technique, which improves the overall model. This iterative process of local model training, client training, model aggregation, and global model update continues until convergence is attained or a preset stopping condition is satisfied.

Clients' actual raw data stays contained on their devices, protecting data integrity and boosting privacy. The Communication Efficiency is a significant benefit because it reduces communication costs because only model changes, which are often small in size, are sent between clients and the central server. This method is flexible enough to support a wide range of clients and data distribution patterns.

2.9 GAN - Generative adversarial networks

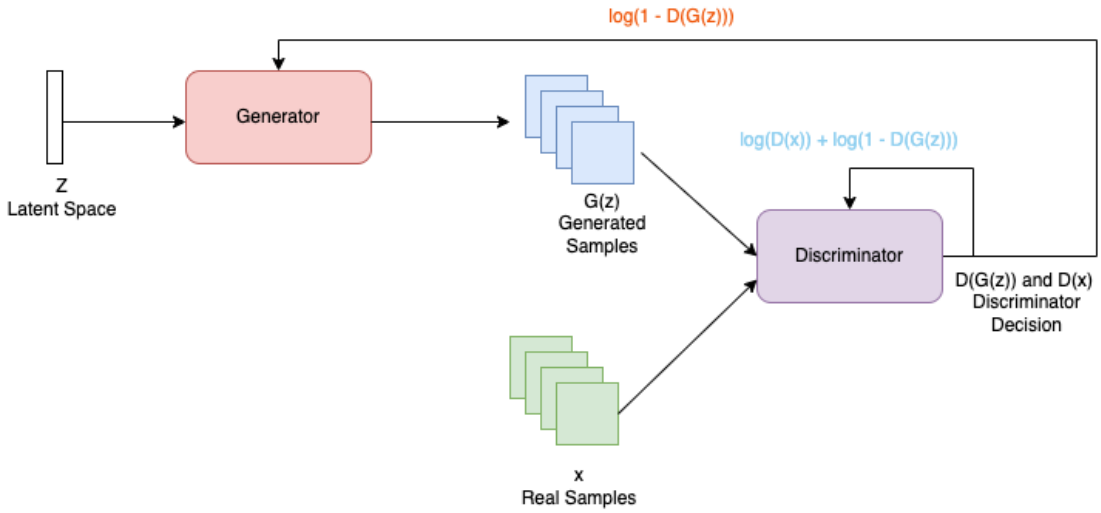

Figure 2.6: A simple GAN Architecture

In a landmark study titled "Generative Adversarial Networks," published in 2014 [6], Ian J. Goodfellow and colleagues established the idea of generative adversarial networks (GANs). The ability of GANs to generate data that closely resembles a training dataset has led to their significant rise in popularity in the deep learning community.

The essential idea underlying GANs entails training both the generator and discriminator neural networks using game theory. While the discriminator tries to tell the difference between created and real data, the generator aims to produce synthetic data instances that closely resemble real data. These networks undergo adversarial training, where the generator tries to trick the discriminator in order to improve its ability to create accurate data. The discriminator simultaneously strives to improve its capacity to distinguish created data from genuine data.

The generator gradually improves its capacity to produce data that appears authentic, while the discriminator becomes better at identifying real samples from fake ones throughout the training phase. The goal of this iterative procedure is to produce high-quality synthetic samples by having the generator produce data that is difficult for the discriminator to distinguish from actual data.

2.10 GAN - Training

The generator and discriminator networks are iteratively refined during the GAN training process in order to produce synthetic data that closely mimics real data. Giving random weights as starting points to the generator (G) and discriminator (D) networks constitutes the initialization step. The benchmark that the GAN attempts to mimic is the training data, which is made up of actual samples from a dataset.

A batch of synthetic samples are created during the discriminator training phase by running random noise (latent space vectors) through the generator G. The dataset's actual samples and these created fake samples are merged to form a batch that is used to train the discriminator. The created fake samples are given a target value of 0 to indicate their artificiality, while the real samples are given a target value of 1 to indicate their authenticity. Aiming to minimize the binary cross-entropy loss between its predictions and the related target values, the discriminator D is then trained on this batch. Here, the goal is to improve the discriminator's ability to tell authentic samples from false ones.

The generator's training is the main goal of the next phase. Using random noise, a new set of fake samples is created from G. All of the created fake samples are labeled with the goal value of 1, in contrast to the discriminator training. This labeling strategy was developed by the generator to deceive the discriminator into believing that the samples were real. In order to reduce the binary cross-entropy loss between the discriminator's predictions on the generated samples and the desired value of 1, the generator G is then trained on this batch. The goal of the generator is to provide samples that convincingly resemble real data in order to trick the discriminator.

Repeat the sequence a predetermined number of times (e.g., epochs). The discriminator and generator receive alternate training throughout each iteration, with the discriminator improving its capacity to distinguish between real and fake data samples and the generator enhancing its capacity to create realistic examples.

It's important to note that training GANs can be challenging due to issues like mode collapse (where the generator only learns to produce a limited variety of samples) and instability. Researchers have proposed various techniques and architectural modifications to address these challenges and improve GAN training stability and performance like wasserstein GAN [21].

2.10.1 Wasserstein GAN (WGAN)

In a conventional Generative Adversarial Network (GAN), the generator aims to deceive the discriminator by producing realistic-looking data. Meanwhile, the discriminator's job is to differentiate between real and artificially generated samples. To ensure effective training and avoid issues like the disappearance of gradients and mode collapse, metrics like Jensen-Shannonn [22] or Kullback-Leibler [23] divergence should be minimized.

Wasserstein GAN [21] introduces the Wasserstein distance as a metric for determining, the separation between the actual data distribution and the generated counterpart. The training procedure is more stable because the Wasserstein distance provides a more insightful measure of distribution dissimilarity.

The Wasserstein distance, also known as the Earth Mover's distance, measures how much "work" is required to transform one distribution into another. It provides a more meaningful metric of distribution dissimilarity. In WGAN, the Wasserstein distance is used to guide the training process.

For two distributions P and Q, the Wasserstein distance is given by:

$$W(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma} [\|x - y\|]$$

Where $\Pi(P,Q)$ is the set of all joint distributions with marginals P and Q.

The discriminator (also known as the critic) in WGAN generates real-valued scores as opposed to probabilities. Maximizing the mean score of genuine samples and lowering the mean score of created samples are the critic's training goals. The goal of the generator is to increase the mean score of the results. Furthermore, WGAN guarantees Lipschitz continuity by limiting the gradients of the critic, which is frequently accomplished using weight clipping.

Discriminator Loss:

$$L_{\text{critic}} = \mathbb{E}_{x\sim P_{\text{data}}}[D(x)] - \mathbb{E}_{z\sim P_z}[D(G(z))]$$

Generator Loss:

$$L_{\text{generator}} = -\mathbb{E}_{z\sim P_z}[D(G(z))]$$

Here, $D(x)$ represents the critic's score for real data x, $G(z)$ is the generated sample from noise z, and P_z is the noise distribution.

The Wasserstein GAN uses the Wasserstein distance as a gradient-friendly metric and adopts the critic-generator dynamic to promote smoother convergence in order to give more stable training.

Gradient Vanishing Problem

When deep neural networks are trained using gradient-based optimization techniques like gradient descent, the *gradient vanishing problem* presents a dilemma. These methods modify the weights of the network based on the gradient with respect to the error function. Gradients can, however, dramatically decline in some networks, particularly those that use activation functions like the sigmoid function. The sigmoid function condenses the input into a tiny range between 0 and 1, which frequently results in a derivative that is less than 0.25. These modest gradients in deep networks are multiplied numerous times as the error spreads backward through the network. As a result, the gradient gets less exponentially as it advances backward through the layers.

Mode Collapse Phenomenon

The phenomenon called *mode collapse* specifically emerges during the training of Generative Adversarial Networks (GANs). GANs include two key parts: a generator, in charge of creating fresh samples of data, and a discriminator, in charge of telling these created samples from actual data. The generator should ideally generate a wide range of

outputs that accurately reflect the training data. However, in real-world scenarios, the generator might start outputting a small number of outputs (or even the same output over and over again), which would only adequately capture a small number of modes of the data distribution. Mode collapse is the phrase used for this event. The main objective of the GAN is undermined when mode collapse takes place since the generator no longer produces a wide variety of outputs. Effectively addressing mode collapse poses a significant challenge in achieving robust and effective GAN training.

2.10.2 Wasserstein GAN with Gradient Penalty (WGAN-GP)

While Weighted Gradient Penalty (WGAN-GP) solves weight clipping's restrictions in WGAN to improve training stability, it also incorporates a gradient penalty term to get around optimization difficulties and less-than-ideal results. [24].

The primary objective of the discriminator is still the same in the setting of WGAN-GP as it is in WGAN. However, this strategy adds a gradient penalty into the discriminator's loss function rather than depending solely on weight clipping as a fix. Without the use of weight clipping, this gradient penalty enforces Lipschitz continuity within the discriminator. This method prevents gradient explosion or vanishing, during training by ensuring that the discriminator's gradients behave appropriately.

The discriminator's output gradients are penalized by the gradient penalty term with respect to random points sampled along the linear routes separating produced and real samples. This method fosters a smoother and more reliable training trajectory by inducing gradients to converge towards a value of 1.

By adopting the gradient penalty mechanism, WGAN-GP significantly enhances training stability, leading to the generation of higher-quality samples when compared to traditional WGAN approaches.

2.11 Dirichlet distribution

A Dirichlet distribution [25] is a type of multivariate probability distribution that is often used in modeling the proportions of different categories in a given population. It is mathematically defined as follows:

$$f(x;\alpha) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{k} x_i^{\alpha_i - 1}$$

Here, $x = (x_1, ..., x_k)$ is a k-dimensional vector representing the proportions of the k categories, $\alpha = (\alpha_1, ..., \alpha_k)$ is the vector of parameters of the distribution, and $\mathrm{B}(\alpha)$ is the multivariate Beta function.

The Dirichlet distribution can be used in the context of federated learning with non-identically and independently distributed (non-IID) datasets to simulate the percentages

of various classes in the dataset or data incompleteness as we defined it above. The class proportions in each client's local dataset can be modelled by a particular set of parameters called α for each client. These local Dirichlet distributions can then be used to estimate the global class proportions, which can give an idea of the general data distribution throughout the federation while protecting data privacy.

Moreover, the Dirichlet distribution can be useful in updating models in federated learning. Local updates can be calculated and combined in a manner weighted by the Dirichlet distribution of data, potentially leading to more effective and robust models.

CHAPTER 3

Literature survey

In our literature survey, we will commence by investigating Federated Learning (FL) algorithms tailored to overcome challenges arising from incomplete data. Subsequently, we will shift our focus to traditional methods for generating synthetic data, building a foundation for our exploration. Progressing further, our survey will delve into the realm of deep learning techniques, specifically exploring intricate methodologies that leverage deep neural networks to address data incompleteness. Our survey will then pivot to Generative Adversarial Networks (GANs), investigating their applications within the context of tabular data—a common format in structured datasets. Lastly, our survey will culminate in an exploration of existing comprehensive studies that tackle data incompleteness in the realm of Federated Learning.

3.1 FL Algorithms for Solving Incomplete Data Problems

Federated Learning (FL) algorithms can be applied to solve incomplete data problems where data is missing or incomplete across different clients or parties. Here's some top FL algorithms that can handle incomplete data settings.

3.1.1 FedProx

FedProx [26] is a prominent federated optimization algorithm designed to address the difficulties of non-IID data distribution in the area of Federated Learning. It extends the standard proximal gradient descent algorithm with the goal of improving model convergence and also the robustness. FedProx makes use of a regularization term that makes devices to update their local models towards a global model thus reducing the impact of data heterogeneity. By adding this regularization term, FedProx aims to balance the trade off between global and local model updates thus leading to improved convergence and performance with non IID data distribution.

The FedProx optimization problem can be given as below:

Given a global model w and a set of local models w_i corresponding to different clients and the goal is to minimize a loss function $F(w)$ that captures the over all performance of the model on the entire dataset:

$$\min_w F(w) = \frac{1}{N} \sum_{i=1}^{N} F_i(w) + \lambda \phi(w),$$

where N is the total number of devices, $F_i(w)$ is the loss function for the ith client's local model, λ is a hyperparameter that controls the regularization, $\phi(w)$ is a regularization term that makes the global model to be close to the local models.

The key insight of FedProx is in the regularization term $\phi(w)$ which can be defined as:

$$\phi(w) = \frac{1}{2} \sum_{i=1}^{N} \|w - w_i\|_2^2.$$

This regularization term makes or encourages each of the local model w_i to be close to the global model w thereby improving the consistency between local and global models. The parameter λ calculates the trade off between fitting the local data and staying close to the global model.

The optimization process in FedProx encompasses updating both the global model w and the local models w_i. In each round of training, the devices compute their local gradients and update their models using the following formula:

$$w_i^{(t+1)} = \arg\min_{w_i} \left\{ F_i(w_i) + \frac{\lambda}{2} \|w_i - w^{(t)}\|_2^2 \right\},$$

where t denotes the iteration.

The global model w is then updated by averaging the local models updates weighted by the number of data points on each device or the client:

$$w^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} n_i w_i^{(t+1)},$$

where n_i is the number of data points on the ith device or the client.

3.1.2 FedNova

FedNova [27] is an approach that improves the Federated Averaging (FedAvg) algorithm by addressing the challenges posed by varying numbers of local training steps taken by

different parties in federated learning. These challenges can arise due to variations in computational power, local dataset sizes and many other factors. Parties with more local steps have larger local updates and this can affect the global model if directly averaged. To counter this, FedNova introduces a normalization and scaling mechanism for local updates. This ensures that parties contributions are weighted based on the number of local steps they take and therefore preventing skewed influences on the global model. Despite its notable enhancement,the alterations made to FedNova are lightweight therefore leading to minimal computational burden during updates of the global model.

3.1.3 SCAFFOLD

SCAFFOLD [28] is an algorithm designed to address the challenges posed by non IID (non identically distributed) data in federated learning. It acknowledges that in non IID scenarios, the updates contributed by different parties can have high variance due to diverse data distributions (non-IIDness). To mitigate this variance, SCAFFOLD algorithms introduces the concept of control variates. These are auxiliary terms used to reduce variance in statistical estimations.

In SCAFFOLD, the control variates are introduced both for the global model (server) and for individual local models (clients). The algorithm estimates two update directions. One for the server model and the other for each individual client model. These update directions are found from the control variates. The update direction for the server model acts as a "target" direction and it helps to counteract the drift introduced by local training in non-IID settings.

By introducing control variates and estimating update directions of the local and global model, the SCAFFOLD algorithm aims to correct the variance and drift caused by non-IID data distributions. This approach enables more stable and efficient convergence of the global model in federated learning and even when faced with non-IDDness across different clients.

3.1.4 MOON

MOON (Model-Contrastive Federated Learning) [29] is a novel and effective federated learning framework designed to address the challenge of handling non-IIDness in local data distribution across multiple clients in the context of image datasets with deep learning models. The main objective of MOON is to enable collaborative model training without the need for parties to share their raw data.

The key idea behind MOON is to leverage the similarity between model representations to improve the local training of individual parties. It achieves this by performing contrastive learning at the model-level. In contrastive learning, the representations of similar data points are encouraged to be close to each other in the feature space while those of dissimilar data points are pushed apart. By applying this concept in a federated learning setting, the MOON algorithm aims to enhance the model representations of individual parties and improve the overall performance of the federated learning process.

Federated Learning algorithms, while effective in many scenarios, encounter challenges when addressing extreme non-IIDness in the data distribution. Extreme non-IIDness refers to situations where devices have highly imbalanced or dissimilar data distributions, in our case, data incompleteness. Traditional federated optimization algorithms like FedProx, FedNova, and SCAFFOLD struggle to cope with such scenarios due to the significant disparities in data characteristics across devices. They may fail to capture the unique local data patterns and struggle with class and quantity skew. However, data augmentation offers a promising solution. By generating synthetic data aligned with local data characteristics of each device can address extreme non-IIdness/ data incompleteness. They can introduce diversity, enhance data distribution balance and accelerated convergence speed.

In the following sections, we will take a look at data generation techniques starting from standard methods to deep learning methods.

3.2 Standard Methods for Generating Synthetic Data

In this section, the focus is on generating synthetic data for tabular data, and it excludes methods used for image data, such as cropping, zooming, or inverting. The methods are categorized based on their algorithmic sophistication. The following standard methods are reviewed:

3.2.1 Random Oversampling (ROS)

ROS involves randomly sampling additional observations from the minority class with replacement. It is a straightforward method to expand a dataset, but it can alter the data distribution and only duplicates existing samples. Studies have shown that ROS can be more effective than random undersampling for imbalanced datasets, as it does not impact the majority class instances as much while improving the classification of the minority class. However, its effectiveness can vary across different datasets, and in some cases, it may not lead to significant changes in classification performance.

3.2.2 Synthetic Minority Oversampling Technique (SMOTE)

SMOTE [5] addresses the drawbacks of ROS by generating synthetic observations instead of duplicating existing ones. The algorithm works by selecting a data point from the minority class and its nearest neighbor from the same class. The distance between these points is calculated, and synthetic samples are generated along the line segment connecting them. SMOTE has been widely used in the literature and has shown promising results in various applications. Studies have evaluated SMOTE's performance in high-dimensional data and combined it with other classifiers, demonstrating improved recognition and classification performance for imbalanced datasets.

3.2.3 Borderline-SMOTE

Borderline-SMOTE [30] is a variant of SMOTE that focuses on the minority class instances near the decision boundary. It generates synthetic samples for these borderline instances to improve classification performance. Studies have shown that Borderline-SMOTE can outperform SMOTE and ROS in terms of true positive rate and F-value. It has been used for data augmentation in various domains, such as EEG classification and neural networks, yielding improved results.

3.2.4 Safe-Level-SMOTE

Safe-Level-SMOTE [31] improves upon SMOTE by defining safe regions to prevent oversampling in noisy or overlapping areas. Each minority class instance is assigned a safety level based on the number of minority neighbors it has. Synthetic samples are then generated closer to instances with higher safety levels, ensuring that they are created only in safe regions. Studies have shown that Safe-Level-SMOTE can outperform SMOTE and Borderline-SMOTE in terms of accuracy and classification performance, especially when certain improvements, such as moving synthetic instances away from majority class examples, are applied.

3.2.5 ADASYN

ADASYN [32] is an oversampling algorithm that addresses the issue of classifier learning difficulties for instances with fewer minority class neighbors. It uses a density distribution to determine the number of synthetic samples to generate for each instance of the minority class. ADASYN has been applied to various domains, such as telecommunications fraud and financial datasets, demonstrating its ability to improve accuracy, recall, and F1-measure compared to SMOTE.

3.2.6 K-Means SMOTE

K-Means SMOTE [33] combines K-means [34] clustering with SMOTE to avoid generating noise and effectively balance datasets. The algorithm first clusters data points using K-means and then applies SMOTE to each cluster. This approach has shown better results than standard SMOTE in terms of average recall, F1-score, and geometric mean, making it a useful method for dealing with imbalanced datasets.

3.2.7 Cluster-Based Oversampling

Cluster-Based Oversampling [35] addresses the presence of small disjuncts in the training data, which can cause a loss of classifier performance. The algorithm involves clustering the data for each class separately and then applying ROS to each cluster. It helps balance datasets and overcome the issues caused by class imbalance.

3.2.8 Gaussian Mixture Model (GMM)

GMM [36] is a probabilistic model that assumes data can be modeled as a weighted sum of Gaussian distributions. It estimates parameters using an expectation-maximization algorithm and generates synthetic data by drawing random samples from the model. GMM has shown potential in addressing the lack of data in certain domains, such as immersive virtual environments.

Standard synthetic data generation methods, including techniques like *Random Oversampling (ROS)*, *Synthetic Minority Oversampling Technique (SMOTE)*, *Borderline-SMOTE*, *Safe-Level-SMOTE*, *ADASYN*, *K-Means SMOTE*, *Cluster-Based Oversampling*, and *Gaussian Mixture Model (GMM)*, offer benefits but are also associated with limitations that underscore the necessity of incorporating deep learning techniques for more effective data augmentation. These limitations encompass challenges such as the inadequate generation of complex patterns, oversight of correlations between features, difficulties in handling multimodal data distributions, the potential for repetitive artifacts through methods like ROS and SMOTE, and their restricted applicability to non-tabular data types like images, audio, and text. In contrast, deep learning approaches provide solutions to these limitations, offering the capacity to capture intricate relationships, generate data at a feature level, handle multimodal distributions, avoid repetitive artifacts, and accommodate diverse data formats, thus making them better suited for comprehensive and sophisticated synthetic data generation.

3.3 Deep Learning Methods

In this section, the focus is on deep learning methods like Bayesian networks, Autoencoders, and Generative Adversarial Networks (GANs) for synthetic data generation

3.3.1 Bayesian Networks

Bayesian Networks (BNs) [37] represent probabilistic graphical models that utilize Bayesian inference to perform probability computations over a directed acyclic graph. These models demonstrate the dependence between variables and can express a full joint probability distribution succinctly. Each node in a Bayesian network corresponds to a random variable and contains probability information quantifying the impact of its parent nodes on itself.

As the layers in a Bayesian network increase, the model transitions into a deep Bayesian network. These models were integral in the history of deep learning but are less common in the present day. One application of Bayesian networks was shown in a study where it was used to address the problem of private data sharing, significantly outperforming other solutions in terms of accuracy.

3.3.2 Autoencoders

Autoencoders (AEs) are specific types of feedforward neural networks composed of an encoder and a decoder network. The encoder compresses high-dimensional data into a low-dimensional representation, and the decoder decompresses this representation back into the original domain [38].

Despite their usefulness, AEs have some limitations regarding synthetic sample generation. Variational autoencoders (VAEs) [39] can help overcome these issues by mapping each point in the original data to a multivariate normal distribution in the latent space and adding the Kullback–Leibler (KL) divergence to the autoencoder reconstruction function.

3.3.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) consist of two networks: the generator and the discriminator. These networks have been proven highly useful in generating synthetic samples and are widely used across numerous applications. GAN architecture was discussed in the previous section.

Few examples of the use of GANs include generating artificial EEG datasets [40], improving the accuracy of a CNN classifier in signal modulation classification, handling datasets with multiple imbalanced classes [41] , and simulating urban morphology. Another interesting development is the Quantum GAN [42], which leverages quantum circuits to overcome some limitations of traditional GANs.

Generative Adversarial Networks (GANs) stand out as a superior technique for synthetic data generation compared to methods like Bayesian networks and Autoencoders. GANs directly generate diverse and intricate synthetic samples by learning the underlying data distribution. These high-quality samples closely resemble real data, capturing complex patterns and variations effectively. Unlike Bayesian networks, GANs don't require explicit probability modeling; they implicitly learn the data distribution through adversarial training. Their versatility across data types, including images, audio, and text, makes them applicable to a wide range of tasks, while their prowess in capturing complex data relationships and reflecting real-world dataset complexities contributes to their effectiveness. With a broad spectrum of successful applications, such as image synthesis, style transfer, and data augmentation, GANs emerge as a versatile and potent solution for comprehensive synthetic data generation.

3.4 Generative Adversarial Networks

GANs have been a powerful tool that has undergone continuous improvements and modifications. Here is a chronological overview of GAN architectures showcasing their evolution over the years.

Table 3.1: Overview of GAN Architectures

GAN	Key Features	Applications
Conditional GAN (CGAN)	Generates data conditioned on class labels.	Image synthesis, data augmentation.
DCGAN	Incorporates convolutional layers for improved image quality.	High-quality image generation.
InfoGAN	Learns disentangled information in the latent space.	Attribute control, data manipulation.
CoGAN	Uses two GANs to learn joint distribution of multi-domain images.	Multi-domain image generation.
AC-GAN	Introduces an auxiliary classifier in the discriminator.	Controlled data synthesis, class-specific generation.
StackGAN	Generates high-resolution images from text descriptions.	Text-to-image synthesis.
WGAN	Addresses training instability and mode collapse with Wasserstein distance.	Stable training, improved sample quality.
CycleGAN	Translates images between domains without paired data.	Image-to-image translation, style transfer.
MuseGAN	Focuses on multi-track music generation.	Music composition.
SAGAN (Self-Attention GAN)	Incorporates self-attention mechanisms for coherent image generation.	High-quality image generation.
BigGAN	Upscales GAN models for large-scale image generation.	High-quality image synthesis.
StyleGAN	Emphasizes control over generated images using style transfer.	Controlled and customizable image generation.
GauGAN	Converts sketches into detailed images using semantic layouts.	Photorealistic scene generation.
PiiGAN	Focuses on large missing area inpainting using multiple results.	Image inpainting with varied and higher-quality results.

3.4.1 Conditional Generative Adversarial Network (CGAN)

CGAN [43], proposed in 2014, is a variant of GAN that allows users to control the class label of the generated data. It addresses the limitation of vanilla GANs, where it's challenging to generate data of a specific class since there's no control over the latent space representation. CGAN conditions both the generator and the discriminator on class labels, enabling the generation of samples belonging to a user-selected class. It finds applications in generating labeled data for tasks such as image synthesis and data augmentation.

The key advantage of CGAN is its ability to generate samples belonging to a user-selected class. For example, if the training dataset contains images of various objects, such as cars, planes, and bicycles, a CGAN can be trained with these images along with their respective class labels. Once the CGAN is trained, a user can specify a desired class label, such as "car", and the model will generate synthetic images of cars. This level of control over the generated data is crucial in tasks requiring labeled data, such as image synthesis and data augmentation.

CGAN has found applications in various domains, including computer vision and image generation. By providing class labels as input, CGANs can produce targeted data, making them valuable in scenarios where specific classes of data are required. Additionally, CGAN's ability to generate labeled data can be leveraged in tasks like image-to-image translation, style transfer, and content generation with explicit control over desired features.

3.4.2 Deep Convolutional Generative Adversarial Network (DCGAN)

Introduced in 2016, the DCGAN (Deep Convolutional Generative Adversarial Network) [44] marks a significant milestone in image generation, notably enhancing the quality of generated images when compared to previous GAN variations.

DCGAN innovatively merges convolutional layers with GANs for the purpose of image generation, leading to a remarkable enhancement in the overall quality of generated images as compared to earlier GAN iterations. This architecture harnesses the power of convolutional layers, enabling the model to identify intricate local features and structures within the data, thus rendering it especially well-suited for tasks in the realm of computer vision. The incorporation of DCGAN has fundamentally transformed image generation and laid a strong groundwork for subsequent architectures aimed at pushing the boundaries of image quality even further.

3.4.3 Information Maximizing Generative Adversarial Network (InfoGAN)

Also introduced in 2016, InfoGAN [45] extends GANs to learn disentangled information in the latent space. It assigns semantic meaning to the latent features, enabling users to control specific attributes of the generated data. For instance, in the MNIST dataset,

InfoGAN can modify latent variables to change the digit's style, rotation, or width. This architecture's focus on disentangling features provides greater interpretability and control over the generated data.

3.4.4 Coupled Generative Adversarial Networks (CoGAN)

Proposed in 2016, CoGAN [46] uses two GANs together to learn the joint distribution of multi-domain images. Unlike traditional GANs, CoGAN can generate images belonging to different domains simultaneously. By sharing weights between the two GANs, it reduces memory consumption and computational requirements, making it more efficient for multi-domain image generation tasks.

3.4.5 Auxiliary Classifier Generative Adversarial Network (AC-GAN)

Also from 2016, AC-GAN [47] modifies the GAN architecture to be class-dependent. It introduces an auxiliary model in the discriminator, which reconstructs class labels, enabling class-specific image generation. AC-GAN can produce globally coherent samples comparable to the ImageNet dataset in terms of diversity. This architecture's ability to condition the generator on class labels makes it useful for tasks requiring controlled data synthesis and class-specific image generation.

3.4.6 Stacked Generative Adversarial Network (StackGAN)

Introduced in 2016, StackGAN [48] extends GANs to generate high-resolution images from text descriptions. It divides the image generation process into two stages. The first stage generates a primitive shape and colors based on the input text, while the second stage refines the output using both the text description and the first stage's result. This two-stage approach allows for more detailed and realistic image generation based on textual descriptions.

3.4.7 Wasserstein Generative Adversarial Networks (WGAN)

Proposed in 2017, WGAN [21] modifies the training phase of traditional GANs to address issues like mode collapse and training instability. WGAN introduces a new loss function based on Wasserstein distance, which provides a meaningful metric for the generator's convergence and sample quality. This modification leads to more stable training and helps prevent mode collapse, a common problem in GANs.

3.4.8 Cycle-Consistent Generative Adversarial Network (CycleGAN)

Introduced in 2017, CycleGAN [49] is designed for image-to-image translation without paired data. It allows for style transfer between different domains by introducing two generators and two discriminators. The generators learn to translate images from one domain to another, while the discriminators verify the realism of the translated images.

Additionally, a cycle consistency loss metric ensures that translating an image back and forth between domains results in a similar image to the original.

3.4.9 Multi-track Sequential GAN (MuseGAN)

From 2019, MuseGAN [50] focuses on music generation, which poses unique challenges due to its temporal nature and multiple instrument tracks. This GAN architecture addresses these complexities and aims to generate multi-track music sequences. While the generated music may not be on par with professional musicians, MuseGAN shows promising results in generating music with some interesting properties.

3.4.10 Self-Attention Generative Adversarial Networks

Self-Attention Generative Adversarial Networks (SAGAN) [51] enhance GAN architectures by incorporating self-attention mechanisms to maintain long-range relationships within images. Zhang et al. introduced spectral normalization to stabilize training dynamics, leading to more robust and efficient generators. SAGANs demonstrate improved performance on challenging ImageNet dataset, as evidenced by higher Inception Score and Fréchet Inception Distance metrics, indicating high-quality image generation with coherent long-range structures.

3.4.11 Big Generative Adversarial Network, BigGAN

Big Generative Adversarial Network (BigGAN) [52] is a GAN architecture proposed by Brock et al. that upscales existing GAN models to generate high-quality images. It introduced techniques to handle large-scale GAN training and demonstrated superior performance compared to other state-of-the-art structures. Despite significant advancements in image quality, understanding the image synthesis process remains challenging. BigGAN represents a powerful approach to image generation, but further research is needed to gain deeper insights into GANs' inner workings.

3.4.12 Style-based Generative Adversarial Networks (StyleGAN)

Proposed in 2019, StyleGAN [53] explores an alternative generator architecture based on style transfer. It emphasizes better control over the generated image rather than realism alone. By untangling high-level features and stochastic variation in the latent space, StyleGAN improves the quality metrics over previous architectures. It enables users to control various aspects of the generated image, leading to more controlled and customizable image generation.

3.4.13 GauGAN

Introduced in 2019 by NVIDIA Research, GauGAN [54] allows users to sketch an abstract scene and convert it into a detailed image. By using a spatially-adaptive normalization layer, GauGAN achieves photorealistic image generation based on semantic layouts

provided as input. This approach finds applications in generating realistic scenes from rough sketches or guiding artistic image synthesis.

3.4.14 Pluralistic Image Inpainting GAN (PiiGAN)

Also from 2019, PiiGAN [55] focuses on filling large missing areas in an image, providing more varied and higher-quality results compared to other state-of-the-art architectures. It uses a new style extractor to extract style features from the original images and leverages multiple results rather than seeking a single optimal inpainting solution. This approach enables PiiGAN to better match the context semantics of the original image when completing incomplete parts.

3.5 GANs for tabular Data

Tabular data comes with unique challenges, including the presence of continuous and categorical features, non-Gaussian and multimodal distributions, highly imbalanced categorical variables, and sparsity in one-hot-encoded vectors. These properties make it difficult for standard GANs to effectively generate synthetic tabular data.

3.5.1 TGAN

TGAN [56] is a specialized GAN architecture designed to generate synthetic tabular data while addressing several challenges unique to this type of data. Its main goals are to achieve effective machine learning performance and maintain the correlation between columns in the generated data.

To overcome the challenges posed by tabular data, TGAN incorporates specific data transformations prior to feeding it into the GAN. For numerical features, a mode-specific normalization technique is applied, which involves fitting a Gaussian mixture model to each numerical column. This allows the model to handle non-Gaussian and multimodal distributions, ensuring accurate normalization within the range of $[-0.99, 0.99]$. The probabilities obtained from the Gaussian mixture model are also utilized to encode the original values for each row.

For categorical features, TGAN adopts one-hot encoding and introduces noise drawn from a uniform distribution to each dimension of the one-hot-encoded vector. Afterward, the vector is renormalized to ensure valid probability distributions.

The transformed data is then fed into the TGAN architecture, which consists of a generator and a discriminator. The generator, implemented as an LSTM network, generates numeric variables in two steps: first, it generates values v_i, and then it generates encoded probabilities u_i. For categorical variables, the generation is done in a single step. The discriminator, on the other hand, is a fully connected neural network.

TGAN's performance was evaluated in terms of machine learning efficacy and correlation preservation. Comparisons with other data synthesis models, using various machine

learning algorithms, were conducted on different datasets. TGAN demonstrated promising results, with only a 5.7% average performance difference between real and synthetic data, indicating its ability to preserve machine learning efficacy and effectively capture the correlation between different pairs of variables.

3.5.2 CTGAN

The CTGAN [57], proposed by Lei Xu and Kaylan Veeramachaneni et al. in 2019, is an advanced GAN architecture designed for synthesizing tabular data. It shares similar objectives with TGAN, aiming to achieve machine learning efficacy and preserve correlation between columns. However, CTGAN goes a step further by striving to preserve the joint distribution of all columns, making it more ambitious than its predecessor.

The data transformations applied by CTGAN are akin to those used in TGAN. For numerical columns, CTGAN employs a variational Gaussian mixture model (VGM) instead of a fixed Gaussian mixture model (GMM). The VGM estimates the number of modes for each numerical column, allowing more flexibility. Each continuous value is represented as a one-hot vector indicating the mode and a scalar indicating the value within that mode. This allows CTGAN to handle non-Gaussian and multimodal distributions effectively. Categorical features are still one-hot encoded, but without any additional noise.

To address unbalanced discrete columns, CTGAN incorporates a conditional generator. This generator can produce synthetic rows depending on specific discrete column values. The authors introduced a technique called "training by sampling," ensuring the CTGAN examines all possible discrete values uniformly. The integration of the conditional generator involves using a conditional vector specifying the desired categorical column values. The generator loss is modified to ensure it learns to map the conditional vector into the correct one-hot-encoded values.

The evaluation of CTGAN was performed on seven simulated datasets and eight real datasets. For simulated datasets, the likelihood fitness metric was used to evaluate performance, given the known data distribution. In the case of real datasets, machine learning efficacy served as the evaluation metric since the data distribution is unknown. CTGAN was compared with other generative models and demonstrated superior performance in terms of machine learning efficacy on real datasets. While it performed well in terms of the likelihood fitness metric on simulated datasets, it did not outperform all other models.

An ablation study was conducted to evaluate the utility of specific techniques introduced in CTGAN. The results highlighted the importance of mode-specific normalization, the conditional generator, and training by sampling. Mode-specific normalization significantly contributed to performance improvement in real datasets, while the conditional generator and training by sampling were crucial for generating high-quality tabular data.

3.5.3 TabFairGAN

TabFairGAN [58], introduced by Amirarsalan Rajabi and Ozlem Ozmen Garibay in 2021, is a WGAN with a gradient penalty designed for generating tabular data. Like TGAN and CTGAN, its main goals are to ensure effective machine learning performance and maintain correlation within the synthetic data.

To process the input data, TabFairGAN uses one-hot encoding for categorical features and a quantile transformation for numerical features. The generator architecture involves fully connected layers, utilizing Gumbel softmax for one-hot encoding of categorical variables. On the other hand, the discriminator comprises fully connected layers with leaky ReLU activation.

In the evaluation, TabFairGAN outperformed TGAN and CTGAN on various machine learning models (decision trees, logistic regression, and MLP) using metrics like F1-score and accuracy. The experiments showcased TabFairGAN's superiority in terms of machine learning efficacy, especially when applied to real-world datasets.

TabFairGAN's main strength lies in its ability to generate high-quality tabular data while preserving the joint distribution of all columns. This makes it a valuable and effective method for synthetic data generation.

However, all the above techniques are suitable only for centralised setting.

3.6 Existing Studies for Solving Non-IID/Data Incompleteness Problems in FL

3.6.1 Fed-TGAN

The paper discusses the difficulties of training a GAN on tabular data in a privacy preserving decentralized setting and therefore proposes a new method which is Fed-TGAN [59], which aims to effectively train a complex tabular GAN on non identical clients by countering data skewness and preserving privacy into consideration.

The 2 main features of Fed-TGAN are "privacy preserving multi-source Feature encoding" where for each categorical column the client computes and sends categorical frequency distribution to the server to built an aggregated global frequency distribution for the columns and also fro continuous columns the client fits and sends the parameters of a variational Gaussian mixture model to the server and then fits a new global VGM models for each column then uses it as final encoder and "Table similarity aware weighting strategies" aim to pre compute weights for each client based on the divergence of their local data from the global statistics to ensure smooth convergence in the presence of skewed data across clients.

3.6.2 HT-Fed-GAN: Federated Generative Model for Decentralized Tabular Data Synthesis

HT-Fed-GAN [60], an innovative solution to the challenge of privacy-preserving decentralized tabular data synthesis. In a landscape where digital technology has led to the accumulation of horizontally partitioned tabular data with privacy concerns, HT-Fed-GAN stands out as a pioneering approach. It aims to generate synthetic data while upholding privacy and data ownership, particularly relevant in domains like healthcare and finance. In contrast to existing methods focused on images and text, HT-Fed-GAN is uniquely tailored for tabular data, addressing specific complexities like multimodal distributions and imbalanced categorical columns. This approach's strengths lie in its pioneering concept, successful handling of multimodal distributions through Fed-VB-GMM, and its capacity to ensure balanced categorical attributes via federated conditional techniques. The proposed approach involves multiple steps, such as Fed-VB-GMM and conditional sampling, which could introduce complexity to the implementation and require careful parameter tuning.

Both Fed-TGAN and HT-Fed-GAN offer novel approaches to federated learning with tabular data, but each comes with its own set of challenges. Fed-TGAN, which uses Variational Gaussian Mixture Models (VGMM) for continuous columns, suffers from computational overhead and concerns over data privacy. Its decentralized structure also makes parameter tuning challenging and raises questions about scalability as the number of clients increases. Transformation artifacts, where the original data distribution is potentially distorted by VGMMs, are another concern. On the other hand, HT-Fed-GAN introduces additional complexity through its use of Fed-VB-GMM and federated conditional techniques, potentially slowing down deployment. This complexity also complicates parameter tuning, particularly for handling multimodal distributions effectively. Both models also share challenges in inference latency, data privacy, and dealing with data skew or variable importance in feature columns. Overall, while both methods aim to address the unique challenges of tabular data in federated learning environments, they also introduce complexities and privacy concerns that need to be carefully managed.

So it becomes evident that there is a pressing need for a new method that bypasses the use of GMMs altogether. This need serves as the foundational premise of my thesis. My research aims to develop an innovative federated learning approach for tabular data synthesis that circumvents the complexities and limitations associated with GMMs.

CHAPTER 4

Methodology

We introduce a data-driven methodology that uses Generative Adversarial Networks (GANs) to address data incompleteness in federated learning. Our approach employs the Dirichlet distribution to generate incomplete data, focusing on its impact on model performance. Special attention is given to handling categorical data by constructing a global vocabulary for uniform encoding across clients. Our federated learning models incorporate neural networks and leverage advanced GAN techniques, such as conditional and Wasserstein GANs, to improve learning efficacy. The methodology is evaluated across different levels of data incompleteness and compares synthetic data to real data in terms of its learning efficacy. Unlike existing techniques that use Gaussian Mixture Models (GMM) in GAN training, our approach does not rely on GMMs to convey data distribution information to the central server. The figure 4.1 is an overview of the methodology flow.

Algorithm 1 Federated Learning with GAN Data Augmentation

Data: Full dataset, List of clients, Number of rounds R, Epochs E, Class Labels L
Output: Final federated classification model

```
2  Dataset ← SelectDataset()
4  IncompleteData ← GenerateDataUsingDirichlet(Dataset)
6  FederatedEncodedData ← FederatedEncoding(IncompleteData)
8  InitialModel ← FederatedClassification(FederatedEncodedData)
10 for technique in [Federated GAN, Federated Class Sampling GAN, Federated Class
     Sampling and Client Grouping GAN] do
12   GeneratedData ← TrainAndGenerateData(technique, IncompleteData)
14   AugmentedData ← StepwiseAddition(GeneratedData, IncompleteData)
16   UpdatedModel ← FederatedClassification(AugmentedData)
18   if Accuracy(UpdatedModel) improves then
       /* Go back to step 15                                          */
19     Goto 15
20   end
21   else
23     Stop
24   end
25 end
```

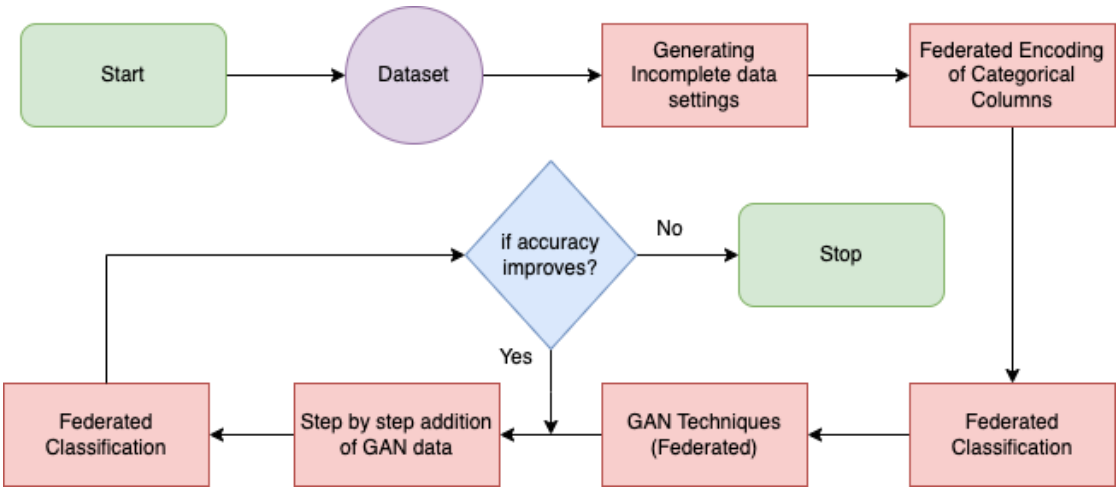

Figure 4.1: Methodology flowchart

4.1 Datasets

Adult Dataset:
The *Adult* dataset [61] is a widely-used resource for classifying income levels. Comprised of 14 attributes, this dataset features both categorical and numerical variables. Attributes like age, job type, educational background, and marital status are utilized to predict

whether a person's income is above or below $50,000 per annum. The labels for the dataset are binary: either ">50K" or "<=50K".

Intrusion Detection Dataset:
The *Intrusion Detection* dataset [62] is particularly useful for identifying cybersecurity risks. It contains a robust set of 41 attributes that describe network traffic behavior. The objective is to identify unusual activities that might signify security threats. Labels in the dataset are categorized as either "normal" or "anomaly".

Bank Marketing Dataset:
The *Bank Marketing* dataset [63] is specialized for applications in marketing analytics, focusing on the prediction of term deposit subscriptions. The dataset encompasses 17 attributes including demographic information, previous banking transactions, and economic metrics. The output variable is binary, labeled either as "yes" (indicating subscription) or "no" (indicating no subscription).

Dataset	# Features	# Categorical Features	# Continuous Features	# Instances	Target Classes	Class Distribution
Adult	14	9	5	48842	2	Class 0: 37155 Class 1: 11687
Intrusion Detection	41	4	37	25191	2	Class 0: 13449 Class 1: 11742
Bank Marketing	17	10	6	45211	2	Class 0: 39922 Class 1: 5289

Table 4.1: Dataset Characteristics and Class Distribution

4.2 Generating Incomplete data settings

In the domain of distributed data and machine learning, the consideration of non-identically and non-independently distributed (non-IID) settings holds a crucial position. This research focuses on assessing federated learning within various scenarios of incomplete data, which requires the creation of such scenarios. The Dirichlet distribution, a continuous multivariate probability distribution parametrized by a vector of positive real numbers, plays a pivotal role in this endeavor.

The Dirichlet distribution proves particularly advantageous for generating non-IID or incomplete data due to its capacity to offer controlled and adaptable measures of concentration. By adjusting the parameters, one can manipulate the equilibrium or disparity in the generated data. In this research, we harness the Dirichlet distribution by incrementing alpha values—specifically 0.05, 1.0, 1.5, 2.0, and 100—to construct a diverse range of

non-IID or incomplete data scenarios for each dataset. Each alpha value corresponds to a unique non-IID setting, resulting in a total of five distinct scenarios per dataset.

When alpha values are small, the Dirichlet distribution yields sparse distributions characterized by pronounced imbalance. This makes it probable that only a limited set of classes will be present in the local dataset. Conversely, larger alpha values lead to more balanced data distributions, where local datasets tend to encompass a more evenly spread mixture of classes. The variation in alpha values enables us to replicate a broad spectrum of incomplete data scenarios, ranging from extreme class imbalance to more representative distributions.

The importance of generating these incomplete data scenarios lies in their potential to provide a wide array of testing scenarios for the assessment of federated learning. By evaluating the performance of federated learning across these incomplete data setups, our objective is to gain insights into the influence of data distribution on learning outcomes. Additionally, we seek to identify strategies to mitigate negative impacts, thereby enhancing the robustness and sustainability of federated learning models.

It's important to note that we partitioned the dataset into training and test subsets. The incomplete settings are specifically introduced in the training dataset. Also, each client uses the same test dataset for evaluation.

4.3 Federated Encoding of Categorical Columns

Federated learning presents a unique machine learning paradigm where training occurs across multiple devices or servers, each equipped with its localized data. This decentralized approach carries advantages such as privacy preservation and data efficiency. However, it introduces challenges, particularly in the treatment of categorical variables within this federated context.

Categorical data requires transformation into numerical representations suitable for machine learning models. Traditional encoding techniques include label, one-hot, and binary encoding. Yet, these conventional methods encounter issues within federated learning, as different clients may possess dissimilar categories within a column. Such discrepancies can yield inconsistent encoding, leading to inaccuracies during model training.

Addressing this challenge necessitates the development of a federated encoding strategy. This entails the creation of a global vocabulary that amalgamates all variable categories from all clients. The dissemination of this global vocabulary among clients ensures uniform encoding methods across the entire setting.

Constructing the global vocabulary involves an initial step where each client transmits its variable categories as a dictionary to the server. The server subsequently compiles these categories to formulate the global vocabulary. This vocabulary is then transmitted back to clients, enabling them to encode their categories consistently across devices.

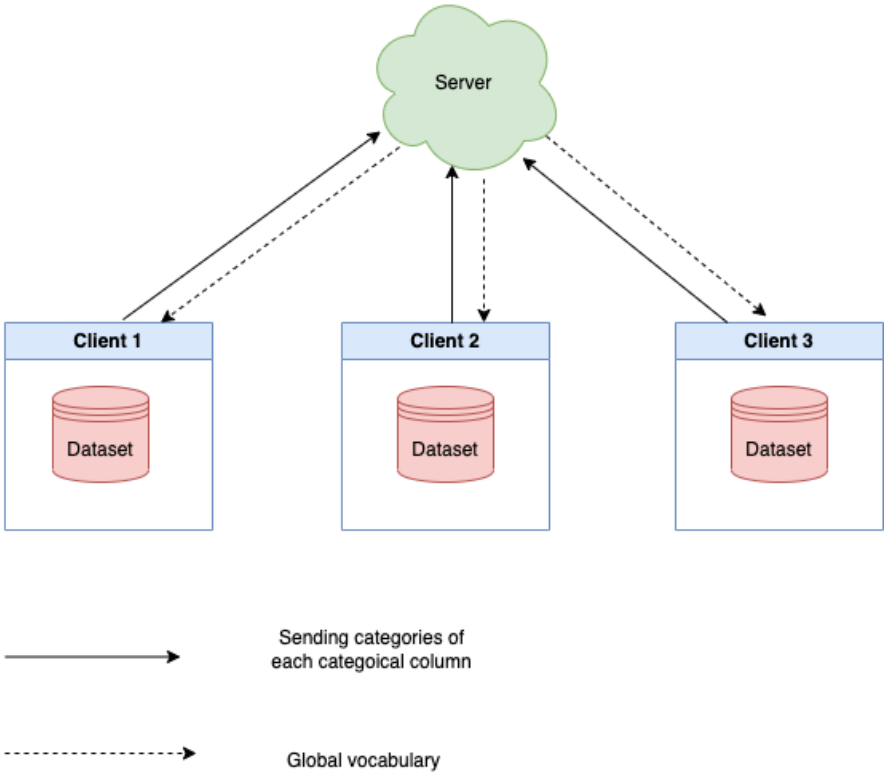

Figure 4.2: Federated encoding of categorical columns

Even with a global vocabulary, a client's data might lack certain categories, potentially leading to incomplete encoding when new data featuring unseen categories arises. This aspect should be considered in the thesis's future scope.

It is crucial to recognize that categorical variable encoding stands as a pivotal preprocessing step in machine learning, and its complexity is accentuated within the federated learning landscape. By embracing federated encoding, uniform encoding practices across all devices can be assured, thereby facilitating more efficient model training in the federated domain [59].

4.4 Federated Classification

Federated learning stands as a machine learning framework where a decentralized set of clients collaboratively trains a model using their respective local data, without the necessity of sharing raw data. This approach not only preserves privacy but also diminishes the need for centralizing data

Commencing with a predefined cluster of clients, the system initializes a central classification model denoted as M. The training unfolds over a sequence of R rounds. In

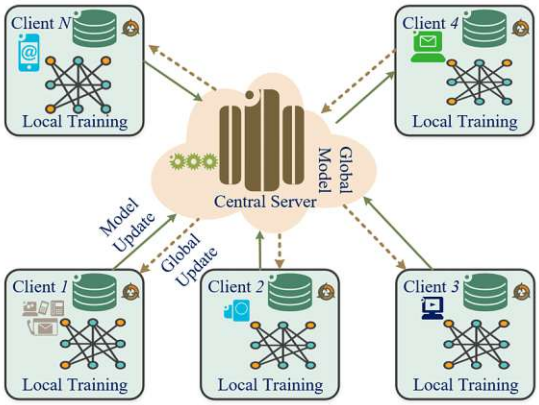

Figure 4.3: A example of general FL architecture. The same architecture is followed for our case. The image was take from the following paper [64]

Algorithm 2 Federated Learning Classification Algorithm

Data: List of clients, Number of rounds R, Epochs E
Output: Trained Classification Model M

```
SelectedClients ← SelectClients()
Initialize Classification Model M
for each round r = 1 to R do
    for each client c in Selected_C lients do
        Load local data: LocalData_c  LocalModel_c ← Copy of M
        for epoch = 1 to E do
            Train LocalModel_c on batch of LocalData_c
        end
        Send LocalModel_c weights to server
    end
    M ← AggregateWeightsFromClients()
end
```

each round, each chosen client loads its exclusive data and commences training a local version of model M for E epochs. Instead of transmitting raw data, the client transmits its local model's weight updates to a central server. This methodology safeguards privacy by confining raw data to individual clients. After collecting all weight updates, the server amalgamates them to refine and update the central model M. The ultimate result of this iterative process is a well-refined classification model, harnessing diverse data sources while upholding data privacy on an individual level.

In our thesis, the client data exhibits incompleteness, as we have previously defined. The employed machine learning model for classification is a simple feed-forward neural network, responsible for predicting binary response probabilities based on one or more predictor variables.

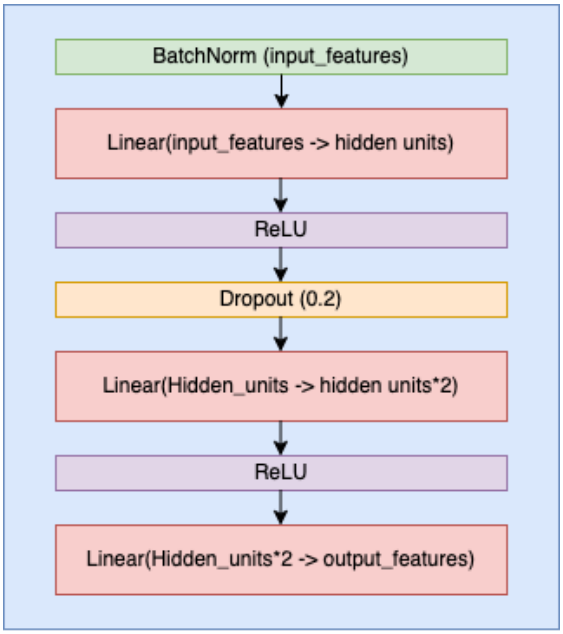

Figure 4.4: Architecture of the feedforward neural network used of classification

We utilize the Adam optimizer and cross-entropy loss to train the model, making it well-suited for tasks involving multi-class classification. The training process spans several epochs, during which the model processes data in mini-batches, generates predictions, calculates loss, and updates its parameters. Model accuracy is assessed using an accuracy metric to determine the correctness of predictions. After the training phase, the model's parameters remain unchanged, and we evaluate its performance on test data using batch processing. This evaluation yields the average loss and accuracy for the test dataset. Communication between clients and the server takes place over multiple rounds, with a maximum of 6 rounds.

4.5 GAN Data Generation Techniques

GANs consists of 2 parts namely a generator and discriminator. The generator is used to create data instances and the discriminator checks for authenticity means it decides whether each instance of data belongs to the actual training set or not. The GAN architecture used for the thesis was inspired from TabFairGAN [58]

We use three techniques to train the GANs in the federated setting.

1. Federated GAN
2. Federated class sampling GAN
3. Federated class sampling and client grouping GAN

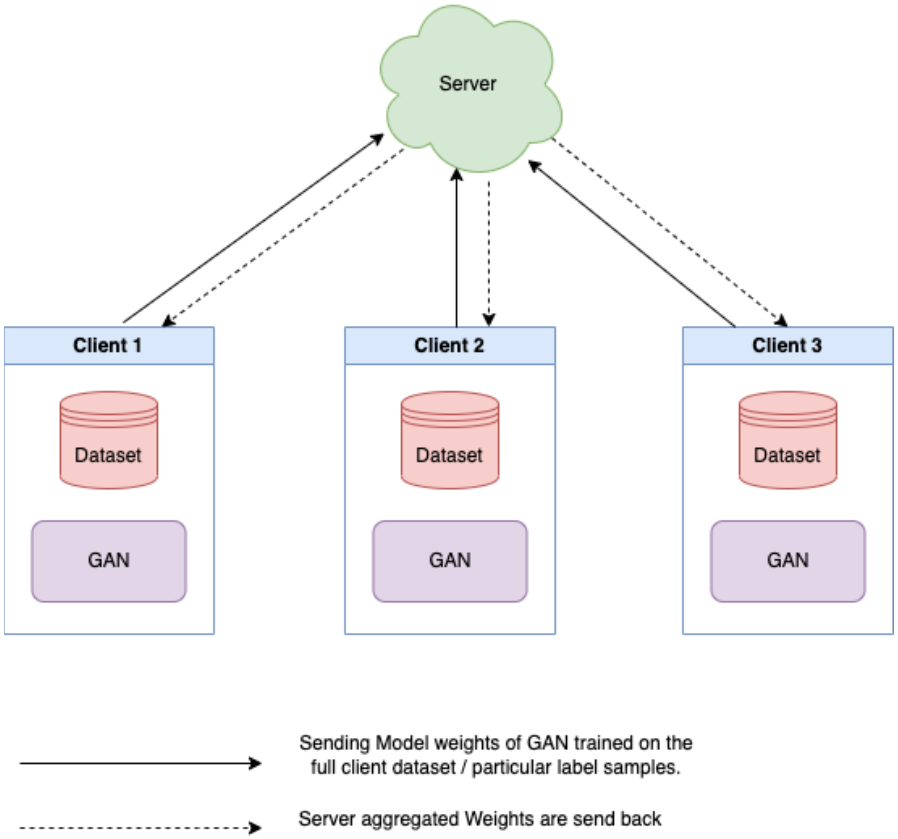

Figure 4.5: A plot for both Federated conditional GAN technique and Federated Class sampling GAN. Federated conditional GAN use full dataset available to train the GAN whereas federated Class sampling GAN use the samples of a particular class label to train the GAN in each round.

Federated GAN:

This is a direct adaptation of GAN to federated settings. Each client trains a GAN consisting of a generator and a discriminator on its local data. The parameters of the GAN models both generator and discriminator are then sent to the central server.The central server averages these parameters to produce a global model. This global model parameters are then sent back to each client for the next round of training. This same process is repeated for several communication rounds. The generator of the trained global GAN model is then used to generate synthetic data.

We use wasserstein conditional GAN with gradient penalty for training the generator and discriminator inspired from WGAN-GP [26] and CGAN [43]. A WCGAN (short form for wasserstein conditional GAN) with GP (Gradient penalty) is a GAN variant that uses conditioning information to control the generated output and the Wasserstein distance for stable gradients and a gradient penalty to enforce Lipschitz continuity and

Algorithm 3 Federated GAN Training Algorithm

Data: List of clients, Number of rounds R, Epochs E
Output: Global GAN Models: Generator G_{global}, Discriminator D_{global}

Initialize global models: G_{global}, D_{global}
for *each round* $r = 1$ *to* R **do**
  for *each client* c *in clients* **do**
    /* Train local GAN on each client */
    Load local data: LocalData_c
    Initialize local models: G_{local_c}, D_{local_c} with G_{global}, D_{global}
    for *epoch = 1 to* E **do**
      Train G_{local_c} and D_{local_c} on batch of LocalData_c
    end
    Send G_{local_c} and D_{local_c} weights to server
  end
  /* Aggregate weights at the server */
  $G_{global} \leftarrow \text{AverageWeights}(G_{local}$ from all clients) $D_{global} \leftarrow \text{AverageWeights}(D_{local}$ from all clients)
  /* Distribute the global model weights back to clients */
  Send G_{global}, D_{global} to all clients
end

prevent the critics gradients from exploding which may cause unstable training.

The significant drawback of applying traditional federated GAN techniques to tabular data arises from the inherent diversity in feature types (categorical, continuous), data distributions, and other characteristics within such datasets. The standard training approach in federated GANs may prove inadequate for effectively handling this wide array of data heterogeneity. Consequently, challenges such as convergence issues or the generation of suboptimal synthetic data can emerge. To address this limitation, the development of a novel training methodology becomes imperative that is tailored to the unique demands of incomplete tabular data scenarios. This novel approach is known as "Federated Classwise Sampling GAN."

Federated Classwise sampling GAN

This technique utilizes classwise sampling of data samples to facilitate GAN training. In a manner akin to the approach described above, we employ Wasserstein GAN with a gradient penalty for training. However, a distinction emerges: we refrain from using the complete dataset to train the local model. Instead, our focus centers on selecting samples associated with a singular class label for training.

Commencing the process, every client transmits its class distribution of local data to the server. Subsequently, the server undertakes dual tasks: first, it selects the clients for

local model training (with the aggregation transpiring on the server in each round), and second, it determines the initial class label for GAN training rounds.

Following an initial phase of GAN training for a specific class label spanning several rounds, the clients conserve the generated model for future application. Once stored, the server shifts its focus to selecting the subsequent class label for GAN training. This iterative progression repeats, encompassing the selection of both the class label and the clients containing relevant class samples. By adhering to this approach, the final outcome encompasses a set of generators equal in number to the classes present, a testament to their collective training process.

However, this technique can still be affected by data incompleteness during model training. This is because the data distribution can vary greatly across clients, even when they have the same class label. To address this challenge and enhance training stability, we developed a method to group clients based on the number of samples for each class label. This grouping ensures that clients with similar characteristics are trained together, promoting stability and ultimately improving the quality of generated data. Thus, we introduce the next technique known as 'Federated Class Sampling and Client Grouping GAN'.

Federated class sampling and client grouping GAN

This technique employs classwise sampling of data samples and client grouping to train the GAN. The concept of client grouping was inspired by the paper [65]; however, the grouping technique used in this work differs from that described in the paper. Similar to the *Federated class sampling GAN* technique, we use wasserstein GAN with gradient penalty for training. In this technique, we diverge from using the entire dataset for local model training, a characteristic differentiating it from *Federated GAN*. Instead, we opt to select samples of a single class label for training. However, an important distinction lies in our approach to grouping clients based on their sample counts. To initiate the process, each client transmits the class distribution of its local data to the server. During the initial rounds, the server determines the class label for training, subsequently clustering clients with comparable sample counts of the selected class label. This data-driven grouping is performed in a descending order based on the number of samples.

For instance, consider a scenario where the class label is '0'. Clients possessing similar quantities of class '0' samples are consolidated into groups. In the initial rounds, the first group is selected to train the GAN model, followed by the subsequent groups in successive rounds. This sequential procedure continues until all client groups complete their training. As the process advances, the number of training epochs is gradually reduced to prevent potential overfitting to a small number of samples.

After the initial training rounds for the first class label, the generated model is preserved by the clients for future utilization. Subsequently, the server selects the next class label for GAN training, along with the corresponding client group possessing samples of that

Algorithm 4 Federated class sampling GAN

Data: List of clients, Number of rounds R, Epochs E, Class Labels L
Output: Set of Generators: $\{G_{label}\}_{label \in L}$

for *each client c in clients* **do**
  Send class distribution of LocalData$_c$ to server
end
Initialize global GAN models: G_{global}, D_{global}
for *each class label label in L* **do**
  /* Server selects which clients and class labels to use */
  SelectedClients ← SelectClientsForClass(*label*)
  for *each round* $r = 1$ *to* R **do**
    for *each client c in SelectedClients* **do**
      Load local data of class *label*: LocalData$_{c,label}$
      Initialize local models: G_{local_c}, D_{local_c} with G_{global}, D_{global}
      for *epoch = 1 to E* **do**
        Train G_{local_c} and D_{local_c} using WGAN-GP on batch of LocalData$_{c,label}$
      end
      Send G_{local_c} and D_{local_c} weights to server
    end
    /* Aggregate weights at the server */
    G_{global} ← AverageWeights(G_{local} from all SelectedClients) D_{global} ← AverageWeights(D_{local} from all SelectedClients)
  end
  /* After training for the class label, save the generator model */
  $G_{label} \leftarrow G_{global}$
  Distribute G_{label} to all clients
end

class. This iterative approach results in the creation of generators for each class, providing a tailored generator for every class label by the end of the training process.

The client grouping function clusters clients based on their sample counts using the DBSCAN algorithm [66]. Input contains client identifiers and sample counts. The algorithm extracts sample counts, applies DBSCAN clustering, and groups clients accordingly, excluding noise points. It sorts the groups based on maximum sample count, resulting in a list of sublists representing clients grouped by similar sample counts. This function facilitates organized analysis of clients with similar data volume.

4.6 Step-wise Addition of GAN Data

When integrating GAN-generated data into incomplete real datasets, a cautious and gradual approach is paramount to prevent overfitting or biasing the model. Here's a high-level strategy to consider. Start by training a GAN model on your non-IID real

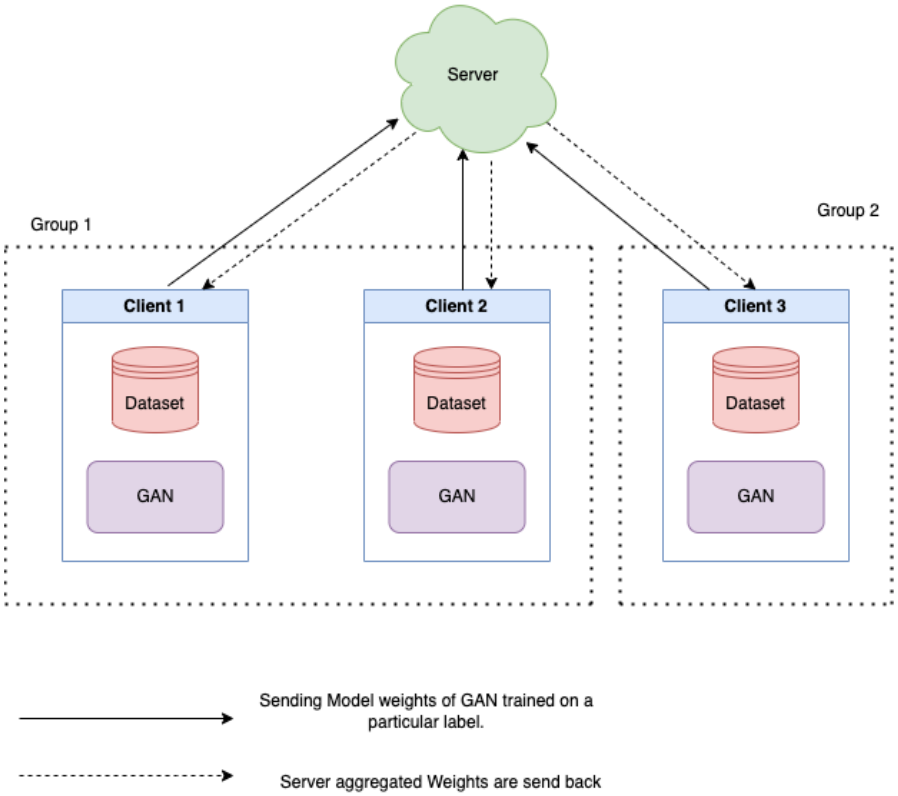

Figure 4.6: Federated class sampling and client grouping GAN

datasets, aiming to generate synthetic data that aligns closely with the distribution of your actual data. Initiate the process by introducing a small fraction of GAN-generated data to the training set, such as 10% of the size of the largest label samples across all clients. Proceed to retrain your model using this combined dataset.

Incrementally escalate the proportion of GAN data within the training set. In each training iteration, progressively increase the percentage of GAN data by an additional 10%. After each incremental step, assess the model's performance on a validation set comprising only real data. The process should halt when the performance on the validation set plateaus or even regresses. This indicates that further inclusion of synthetic data is no longer contributing to improved generalization to unseen real data.

Keep in mind that the outlined process is a general guideline. The specific percentage increments and stopping criteria should be adapted based on the unique attributes of your dataset and the problem at hand. For instance, if real data is scarce, introducing a larger proportion of synthetic data could be beneficial. Always remember that the primary objective is to enhance the model's real-world task performance, so thorough evaluation on a validation set of real data is essential.

Algorithm 5 Federated Class Sampling and Client Grouping GAN Algorithm

Data: List of clients, Number of rounds R, Epochs E, Class Labels L
Output: Set of Generators: $\{G_{label}\}_{label \in L}$

for *each client c in clients* **do**
  Send class distribution of LocalData$_c$ to server
end
Initialize global GAN models: G_{global}, D_{global}
for *each class label label in L* **do**
  /* Server selects the class label and groups clients */
  ClientGroups ← GroupClientsBySampleCount(*label*)
  for *each group in ClientGroups* **do**
    for *each round r = 1 to R* **do**
      for *each client c in group* **do**
        Load local data of class *label*: LocalData$_{c,label}$ Initialize local models: G_{local_c}, D_{local_c} with G_{global}, D_{global}
        for *epoch = 1 to E* **do**
          Train G_{local_c} and D_{local_c} using WGAN-GP on batch of LocalData$_{c,label}$
        end
        Send G_{local_c} and D_{local_c} weights to server
      end
      /* Aggregate weights at the server */
      G_{global} ← AverageWeights(G_{local} from all clients in group) D_{global} ← AverageWeights(D_{local} from all clients in group)
    end
  end
  /* After training for the class label, save the generator model */
  $G_{label} \leftarrow G_{global}$
  Distribute G_{label} to all clients
end

4.7 Research Questions

RQ1: How does the accuracy of a federated learning model change across different levels of data incompleteness??

Our methodology is explicitly designed to address this question. By simulating different levels of incompleteness/non-IIDness using a Dirichlet distribution and observing the performance of the federated learning model, we aim to understand the trade-off between data completeness/non-IIDness and model accuracy. This approach allows us to systematically investigate how increasing data incompleteness affects the accuracy of a federated learning model and how can we mitigate this drawback after GAN data imputation .

This procedure supports our research objective to improve federated learning models by

better understanding the role of data distribution in their performance. Furthermore, it provides insights on how to effectively use GAN-generated data to compensate for data incompleteness, guiding the development of more robust federated learning models.

Table 4.2: Experimental Setup for Research Question 1

Aspect	**Details**
# of Clients	8
# of Federated Settings	5 different settings (4 incomplete, 1 ideal)
α values for Dirichlet distribution	0.05, 0.1, 0.15, 0.20, 100
# of Rounds	6

RQ2: How will the accuracy of the federated learning model change with the addition of GAN-generated synthetic data to the original incomplete data

To address this question, our methodology involves a systematic investigation of the impact of GAN-generated data on the accuracy of the federated learning model. This is done through a controlled study, which includes:

In the experimental phase of our study, we undertake a systematic approach to evaluate the impact and effectiveness of GAN-generated data on enhancing federated learning models. Our experimentation process involves establishing a baseline accuracy (RQ1) by training federated models solely on original data. We then train GANs using the techniques described earlier, generating data for varying non-IIDness levels. Using an incremental approach, we integrate GAN data into the real dataset, training federated models iteratively and recording accuracy. This process continues until additional data fails to significantly improve accuracy. To analyze the influence of synthetic data (RQ2), we rigorously compare model performance against baseline on a validation set of real data. We visualize these findings by plotting model accuracies against corresponding Dirichlet distribution alpha values, providing insights into GAN-generated data's impact across non-IIDness levels. This comprehensive approach answers our research questions and validates the efficacy of our proposed methodology.

This question deepens our understanding of the utilization of GAN-generated data in federated learning contexts. By directly evaluating the impact of synthetic data on model accuracy, this approach contributes to the broader goal of enhancing federated learning model performance, especially when real data may be imbalanced or incomplete.

Table 4.3: Experimental Setup for Research Question 2

Aspect	**Details**
# of of Clients	8
# of federated fettings	4 types of incomplete settings
α values for Dirichlet distribution	0.05, 0.1, 0.15, 0.20
# of rounds after new data added	6

RQ3: What is the quality of the generated dataset compared to the original dataset??
We use the machine learning efficacy method to compare the quality of data. This technique is used in papers like [56] and [57] to compare the quality of GAN generated data with the Real data. There are 6 major steps involved here, which are described below:

1. Data Preparation: Here, we split the original dataset into two subsets: Training and Test. The Training set will be used to train the models, while the Test set will serve as an unbiased evaluation. We also split the synthetic dataset in the same ratio (or same number of instances) as the original dataset to maintain consistency in the evaluations.
2. Model Selection: This phase foucuses on specifically choosing machine learning models. We choose Decision Tree, Random Forest, and SVM. Then ensure that the chosen models are suitable for the data's nature and complexity.
3. Training and Validation: Here, we train each model on the training subset of the original dataset. Also train the same models (with the same initial configurations) on the training subset of the synthetic dataset.
4. Performance Evaluation: In this phase, we test the models trained on the original dataset and those trained on the synthetic dataset using the Test set from the original dataset. Document the performance metrics (e.g., accuracy) for both sets of models.
5. Comparison: In this phase, compare the performance metrics of models trained on the original dataset with those trained on the synthetic dataset. Also identify any significant disparities in performance.

Machine learning efficacy plays a pivotal role in validating the authenticity of the synthetic dataset, ensuring that it embodies meaningful insights rather than arbitrary information. Through the utilization of established machine learning models and the comparison of their performances on both the original and synthetic datasets, we establish an empirical and quantitative assessment of the generated data's credibility. Figure 4.7 provides a visual representation of this process.

A primary aim of our research is to establish the dependability and practicality of synthetic datasets. By evaluating the performance of models trained on synthetic data, particularly in contrast to models trained on original data, we gain valuable insights into the dataset's quality and its potential relevance in real-world applications.

Let us now delve into the results to further explore these aspects.

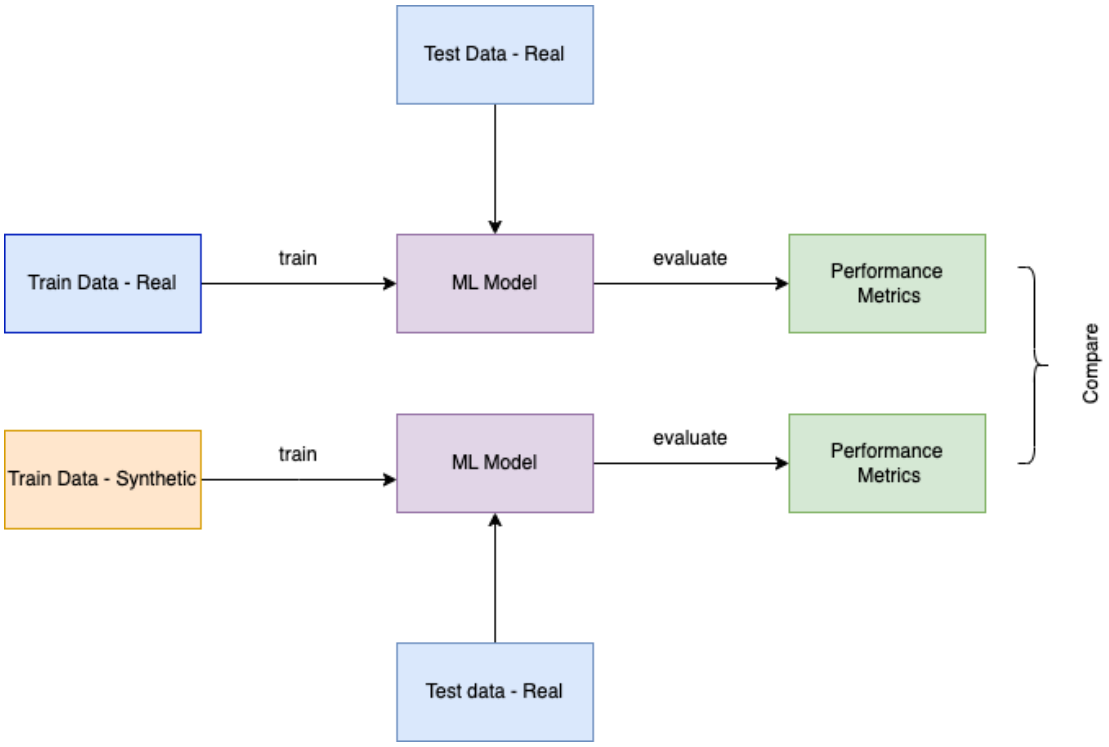

Figure 4.7: Machine learning efficacy technique.

CHAPTER 5

Results and Analysis

In this section, we offer a thorough explanation of the findings from our research, focusing on the key questions that frame our study. Our goal is to shed light on the intricate relationship between incomplete data, the inclusion of synthetic data generated by GANs, and the quality assessment of datasets. We explore three central research questions that form the foundation of our work.

For RQ1, we investigate how the accuracy of federated learning models fluctuates when faced with different levels of data incompleteness. We systematically vary these conditions by adjusting the alpha values in the Dirichlet distribution. Through careful analysis, we reveal detailed observations about how well the models adapt to various non-IID conditions.

In RQ2, we examine the effects of adding GAN-generated synthetic data to the original dataset. By stringently evaluating the subsequent changes in model accuracy, we gain valuable insights into the benefits offered by the inclusion of synthetic data. This question serves as the cornerstone of our innovative approach, highlighting its potential to improve federated learning outcomes.

Lastly, for RQ3, we compare the quality of the synthetic dataset against the original dataset. Using advanced evaluation methods in machine learning, we measure the reliability and usefulness of the GAN-generated data quantitatively. This analysis provides empirical data on the effectiveness and credibility of using synthetic data as an augmentation approach.

5.1 Experimental Setup

In this section we describe the experimetal setup including hardware, software and metrics used. The hardware table outlines key system details, such as the operating system , RAM capacity , and processor attributes. The library table introduces four vital

tools: "flwr", a Federated Learning framework; "matplotlib", a data visualization library; "torch," a deep learning framework; and "scikit-learn," a versatile machine learning library. These components collectively form the foundation for the project's development and exploration within the chosen computational environment. The evaluation metric used for all the research question is 'Accuracy'.

Table 5.1: Hardware used

Specification	Value
OS	MacOS Ventura 13.3.1
RAM	8 GB 2133 MHz LPDDR3
Processor Name	Quad-Core Intel Core i5
Processor Speed	1.4 GHz

Library	Description
flwr==1.4.0	Federated Learning framework
matplotlib==3.5.3	Data visualization library
torch==1.13.1	Deep learning framework
scikit-learn==1.0.2	Machine learning library

Table 5.2: Important Libraries

Research Question	Evaluation Metric
RQ 1	Accuracy
RQ 2	Accuracy
RQ 3	Accuracy

Table 5.3: Evaluation Metrics for Research Questions (RQs)

5.2 Research Question - 1

How does the accuracy of a federated learning model change across different levels of data incompleteness??

As elaborated in the Methodology section, we employed a simple Classification model for our Federated learning task. The model is trained using the FedAvg algorithm over six rounds.

We will use accuracy as the key performance indicator to evaluate the federated learning model's efficacy under different settings.

Our results for Research Question 1 revealed a clear impact of data incompleteness, as simulated through various alpha values in a Dirichlet distribution, on the accuracy of the federated learning model. As the alpha value decreased, indicating a higher degree of data incompleteness, the model's accuracy consistently declined. This underlines the inherent challenge posed by data distribution in federated learning contexts.

The following are our findings for each of the four datasets we examined:

5.2.1 Dataset 1: Adult dataset

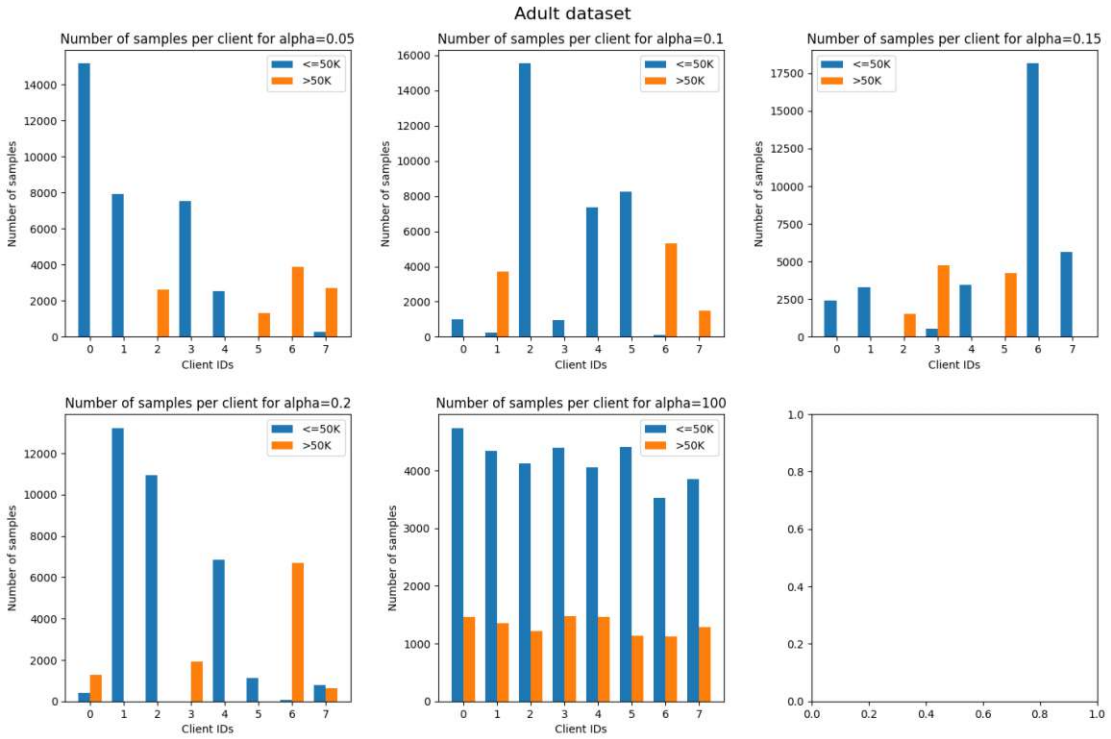

Figure 5.1: The plot displays the data distribution for each label within the Adult dataset across various incomplete data settings. Correspondingly, the alpha values for these settings are 0.05, 0.1, 0.15, 0.2, and 100

Referring to Figure 5.1, we illustrate the data distribution across various incomplete data settings generated using the Dirichlet distribution. Specifically, we generated four incomplete data settings and one ideal setting using the alpha values previously mentioned. As can be seen in Figure 5.1, the first four plots represent settings with extreme data incompleteness, while the last plot depicts a balanced, or ideal, setting.

Turning to Figure 5.2, we present the relationship between model accuracy and alpha values, which serves to showcase the performance of our Federated Classification model on the Adult dataset. For alpha values ranging from 0.05 to 0.20, the model achieves an accuracy of approximately 76%. Additionally, we display the accuracy achieved in the ideal federated setting, which is created with an alpha value of 100. Notably, there is an 8% difference in accuracy between the incomplete data settings and the ideal setting.

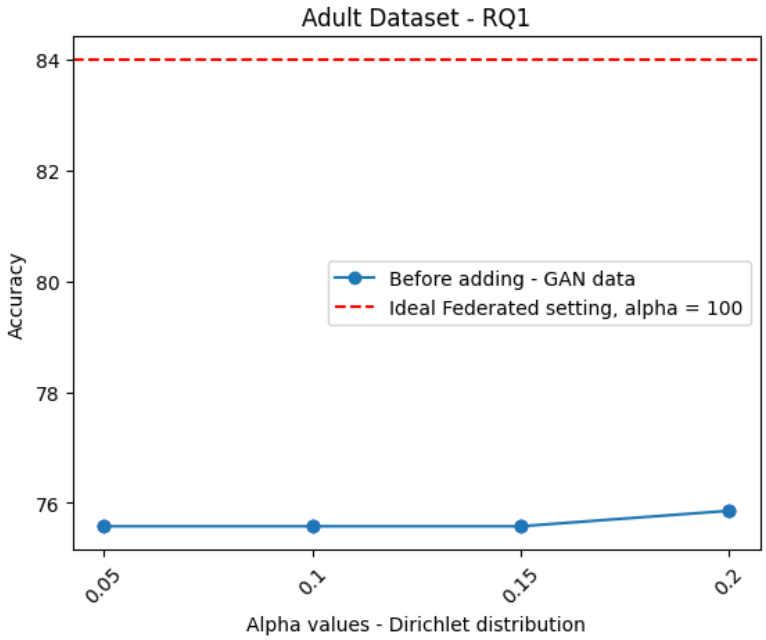

Figure 5.2: The plot depicts the accuracy of the Federated Classification model on the Adult dataset across various incomplete data settings and the ideal federated setting where alpha = 100

5.2.2 Dataset 2: Intrusion dataset

Referring to Figure 5.3, we display the data distribution across various incomplete data settings that were generated using the Dirichlet distribution. As with the Intrusion dataset, we created multiple incomplete data settings along with one ideal setting, using specified alpha values. As can be observed in Figure 5.3, the impact of varying levels of data incompleteness is visually represented.

Turning to Figure 5.4, we discuss the correlation between alpha values and the accuracy of the Federated Classification model, focusing on its performance with the Intrusion dataset. A noticeable decline in accuracy is observed specifically at an alpha value of 0.05. For the remaining alpha values, the model's accuracy closely aligns with that of a ideal setting. Notably, there is an 19% difference in accuracy between the extreme incomplete data setting and the ideal setting.

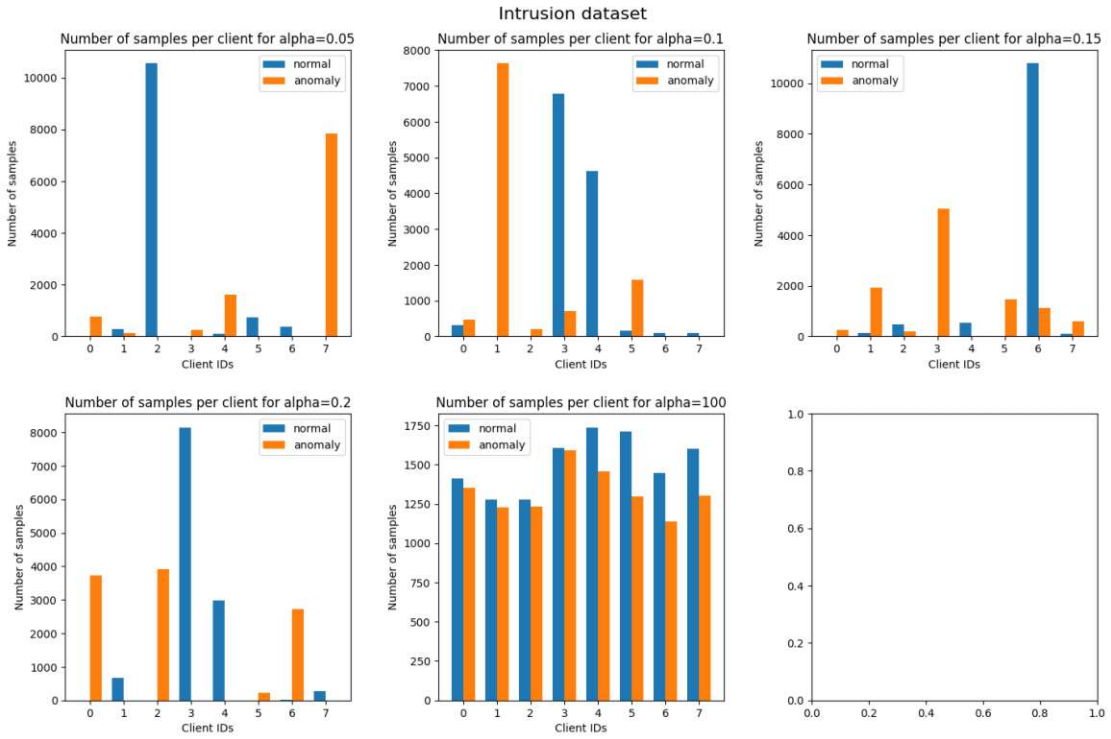

Figure 5.3: The plot displays the data distribution for each label within the Intrusion dataset across various incomplete data settings. Correspondingly, the alpha values for these settings are 0.05, 0.1, 0.15, 0.2, and 100

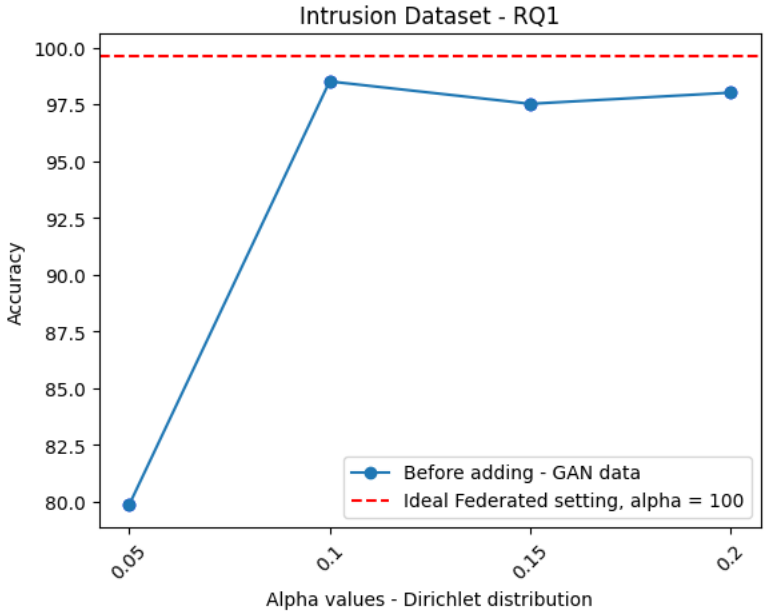

Figure 5.4: The plot depicts the accuracy of the Federated Classification model on the Intrusion dataset across various incomplete data settings and ideal federated setting where alpha = 100

5.2.3 Dataset 3: Bank marketing dataset

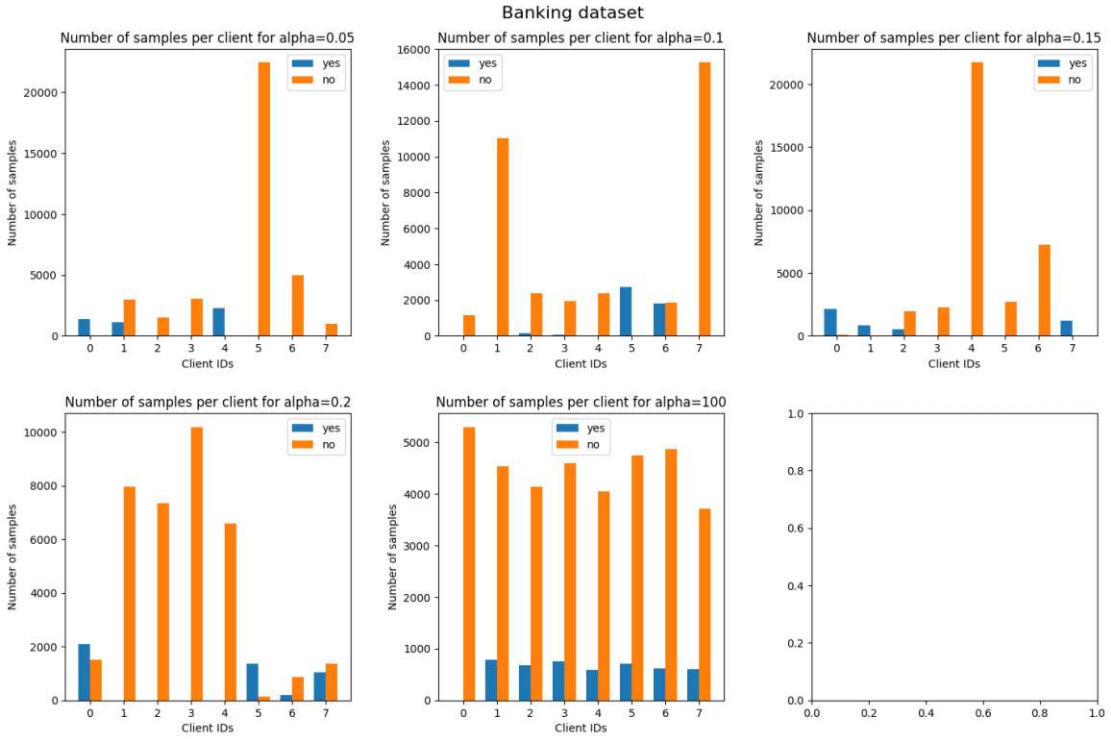

Figure 5.5: The plot displays the data distribution for each label within the Bank marketing dataset across various incomplete data settings. Correspondingly, the alpha values for these settings are 0.05, 0.1, 0.15, 0.2, and 100

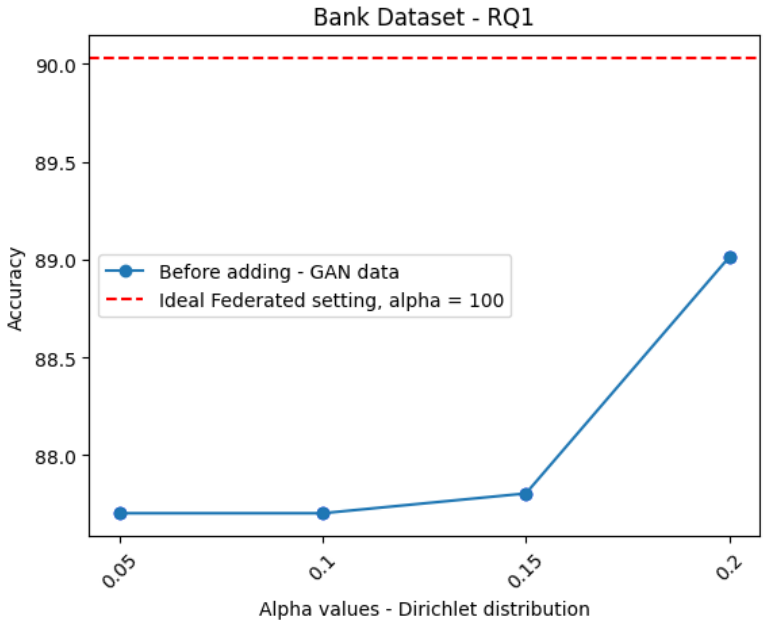

Figure 5.6: The plot depicts the accuracy of the Federated Classification model on the Bank marketing dataset across various incomplete data settings

In Figure 5.5, we exhibit how data is distributed across a range of incomplete data, generated through the Dirichlet distribution. For the bank dataset, we crafted several settings with data incompleteness, in addition to a single optimal or ideal setting, based on predetermined alpha values. As evident in Figure 5.5, the various degrees of data incompleteness are clearly illustrated.

Shifting our attention to Figure 5.6, we explore the relationship between the model's accuracy and different alpha values, with a particular focus on its efficacy on the Bank dataset. A small dip in accuracy is marked, particularly when the alpha values are set at 0.05, 0.1, 0.15 . For 0.2 alpha, the achieved accuracy closely resembles that of an ideal or balanced scenario. Importantly, there is only 2% gap in accuracy between the settings with data incompleteness and the ideal setting.

5.3 Research Question - 2

RQ2: How will the accuracy of the federated learning model change with the addition of GAN-generated synthetic data to the original incomplete data??

Our research also sought to understand how the addition of GAN-generated data can alleviate this drop in accuracy due to data incompleteness. We found that the model's accuracy significantly improved when we added GAN-generated data, thereby compensating for the initial data incompleteness. This suggests that GAN-generated data can

Table 5.4: Summary of GAN Training Techniques

Technique	Description
Federated GAN	GAN is trained in the same federated manner as the classification model.
Classwise Sampling GAN	GAN is trained using classwise sampling.
Classwise Sampling and Client Grouping GAN	GAN is trained using a combination of classwise sampling and client grouping.

be effectively used to boost the performance of federated learning models in the face of data incompleteness.

In this analysis, we use three distinct GAN training techniques to generate fake data. First, we consider the Federated GAN, where the GAN is trained in the same manner as the classification model. Next, we explore the GAN trained using Classwise sampling. Lastly, we use the GAN trained via a combination of classwise sampling and client grouping.

5.3.1 Dataset 1: Adult dataset

Figure 5.7 illustrates the relationship between model accuracy and alpha values after the incorporation of GAN-generated data. As indicated in the legend, four data lines are presented: the red line represents the findings from RQ1, while the other three lines depict the model's accuracy after the addition of data generated through three different GAN techniques. Notably, the Federated GAN displays inconsistent and poor performance, exhibiting fluctuating increases and decreases in accuracy across various incomplete settings, in contrast to the other two techniques.

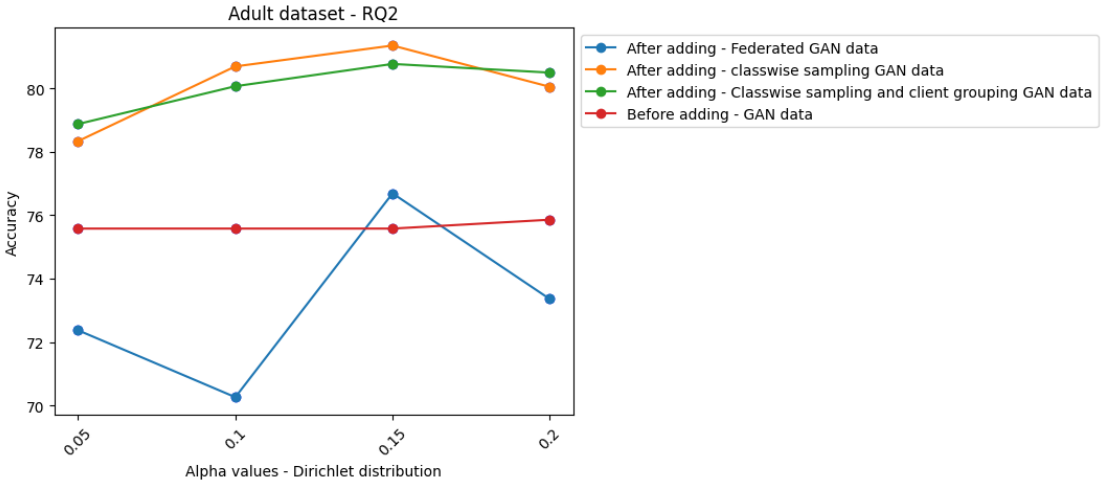

Figure 5.7: The plot illustrates the accuracy achieved by the Federated classification model after incorporating synthetic data produced by the Federated GAN (Adult dataset).

5.3.2 Dataset 2: Intrusion dataset

Figure 5.8 elucidates the relationship between model accuracy and Alpha values after the inclusion of GAN-generated data. As depicted in the figure, there is a marked improvement in accuracy specifically in the setting with an alpha value of 0.05, following the introduction of synthetic data. However, the Federated GAN demonstrates erratic and underwhelming performance, as its accuracy fluctuates inconsistently across different incomplete settings, unlike the other two techniques. It's worth noting that, as shown by the red line representing the results from RQ1, the original model had low accuracy in the setting where the alpha value was 0.05. The addition of GAN-generated data, through various techniques, leads to a significant increase in model accuracy.

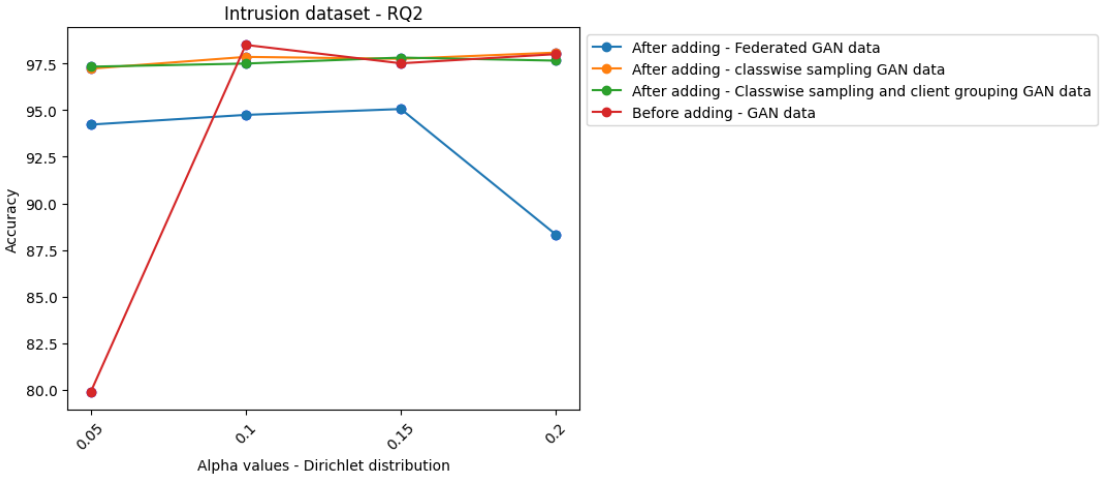

Figure 5.8: The plot illustrates the accuracy achieved by the Federated classification model after incorporating synthetic data produced by the Federated GAN (Intrusion dataset).

5.3.3 Dataset 3: Bank marketing dataset

Figure 5.9 clarifies the relationship between accuracy and Alpha values following the incorporation of GAN-generated data. As illustrated in the figure, the overall model accuracy declines after adding synthetic data from the GAN. Notably, the blue line representing the Federated GAN exhibits the poorest performance relative to the other techniques. The accuracy difference between the incomplete data settings and the ideal setting is a mere 2 percent. This drop in performance could be attributed to the poor quality of the GAN-generated data. One potential cause of this poor quality could be significant class imbalance in the centralized dataset.

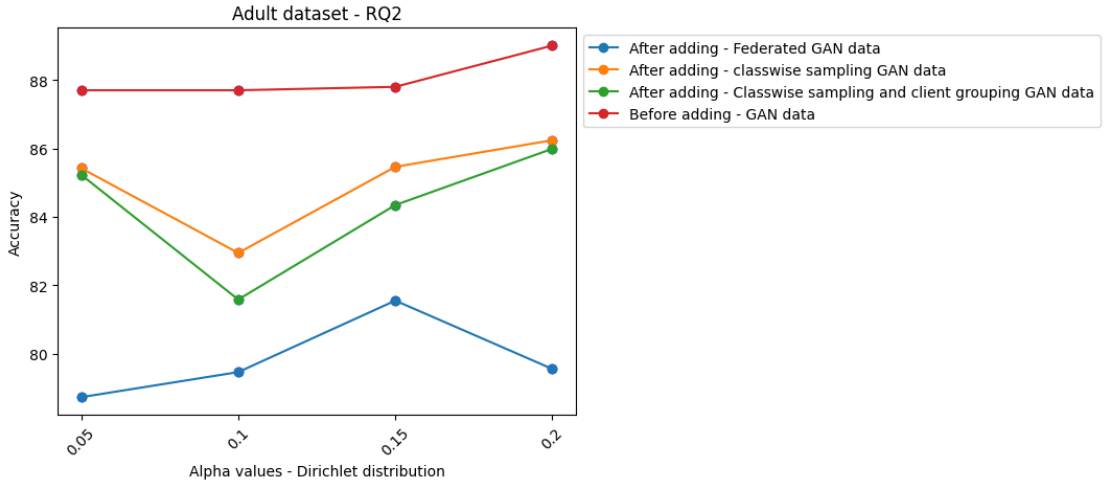

Figure 5.9: The plot illustrates the accuracy achieved by the Federated classification model after incorporating synthetic data produced by the Federated GAN (Bank marketing dataset)

5.3.4 RQ2: Time analysis

The time analysis reveals noteworthy insights into the computational efficiency of various GAN techniques. Federated GAN is the most time-efficient, with an average real-time of 407.07 seconds. This makes it approximately 1.6 times faster than Classwise Sampling and Client Grouping GAN, which has an average real-time of 627.29 seconds, and about 1.6 times faster than Classwise Sampling GAN, with an average real-time of 664.31 seconds.

The Classwise Sampling GAN and Classwise Sampling and Client Grouping GAN techniques appear to be more time-consuming, potentially due to the additional computational overhead introduced by classwise sampling and client grouping mechanisms.

Thus, if time efficiency is a crucial factor, Federated GAN could be considered the most suitable choice among these techniques. However, it's essential to weigh this against other factors like model performance and stability to make an informed decision.

Script Name	Average Real Time (seconds)
Federated GAN	407.07
Classwise Sampling GAN	664.31
Classwise Sampling and Client Grouping GAN	627.29

Table 5.5: Average Real Time for Various GAN Techniques

5.4 Research Question - 3

RQ3: What is the quality of the generated dataset compared to the orginal dataset??

We make use of three different machine learning models—Decision Tree, Random Forest, and SVM—to assess data quality through the machine learning efficacy technique. However, all three models exhibit similar trends in their performance. As such, we will present results exclusively using the Random Forest Classifier for the sake of brevity and clarity.

To assess accuracy, we train classification model(Random Forest) on multiple datasets and conduct evaluations on the real dataset. These training datasets consist of the actual data and synthetic data produced by various GAN techniques, including Centralized GAN, Federated GAN, Federated Classwise Sampling GAN, and Federated Classwise Sampling with Client Grouping GAN.

5.4.1 Dataset 1: Adult dataset

Figure 5.10 depicts the relationship between accuracy and the Alpha values generated through the Dirichlet distribution.

Our examination primarily focuses on comparing the data quality generated by the three Federated GAN techniques. It's important to note that, in general, the Federated GAN technique produces data of lesser quality compared to the other two methods, except for cases involving an Alpha value of 0.15.

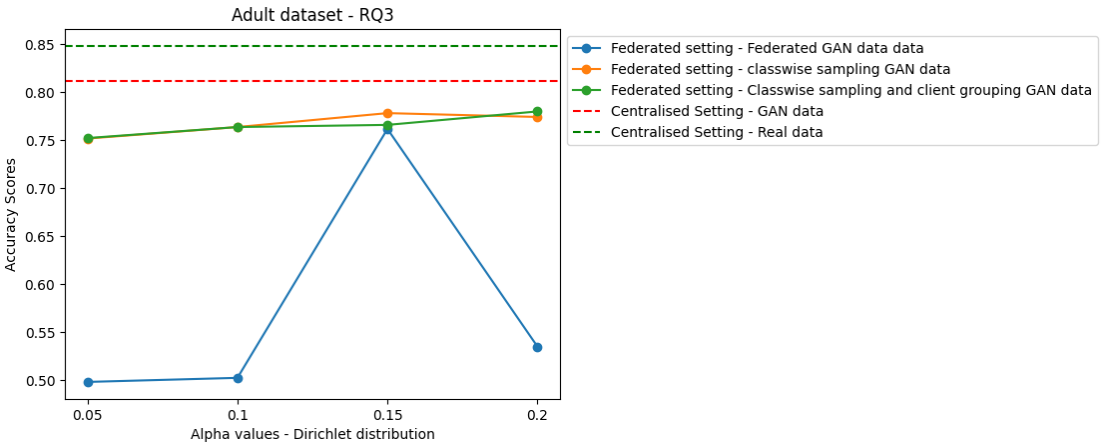

Figure 5.10: This graph illustrates the behavior of both accuracy and F1 score in relation to the alpha values from the Dirichlet distribution for the Random Forest classifier model. (Adult dataset)

5.4.2 Dataset 2: Intrusion dataset

Figures 5.11 illustrate the interplay between accuracy and F1 score in relation to the Alpha values derived from the Dirichlet distribution.

Our primary focus is the comparison of the three Federated techniques used in GAN data generation. Notably, the Federated GAN produces inferior and unstable quality data compared to the other two methods.

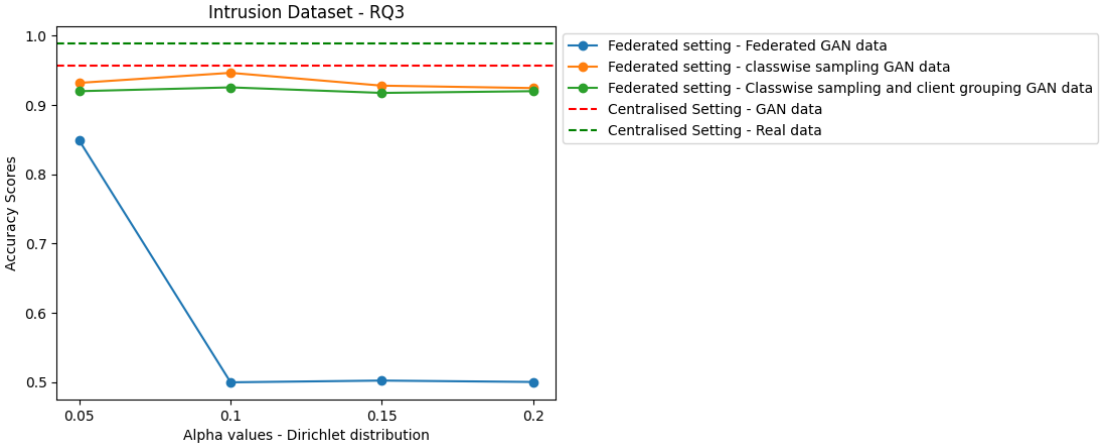

Figure 5.11: This graph illustrates the behavior of both accuracy and F1 score in relation to the alpha values from the Dirichlet distribution for the Random Forest classifier model.(intrusion dataset)

5.4.3 Dataset 3: Bank marketing dataset

Figures 5.12 illustrate the interplay between accuracy and the Alpha values of the Dirichlet distribution. As you can see from the figure 5.12, the 3 techniques almost follow the similar trend except the Federated GAN data which is unstable.

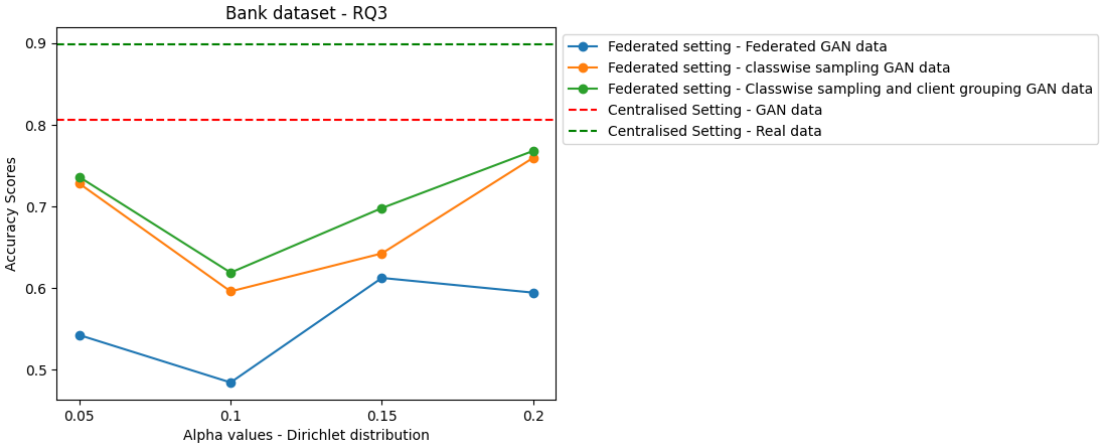

Figure 5.12: This graph illustrates the behavior of accuracy in relation to the alpha values from the Dirichlet distribution for the Random Forest classifier model.(bank marketing dataset)

5.4.4 RQ3: Stability Analysis

Federated GAN is generally the least stable across all three datasets. Class sampling and Client grouping is the most stable in the Adult and Intrusion datasets. For the Bank Marketing dataset, Federated GAN turns out to be the most stable.

In summary, if stability across different data sets is a significant concern, Classwise sampling and Client grouping would likely be the most reliable choice based on these results. Federated GAN appears to be more stable for the Bank Marketing dataset but the accuracy of generated data is low compare to other techniques. Thus, the "best stable" technique is Classwise sampling and Client grouping GAN.

Table 5.6: Standard Deviations of Accuracy of GAN Techniques - Adult dataset

Technique	Standard Deviation
Federated GAN	0.1091
Classwise samping GAN	0.0102
Classwise sampling and Client grouping	0.0098

Table 5.7: Standard Deviations of Accuracy of GAN Techniques - Intrusion dataset

Technique	Standard Deviation
Federated GAN	0.1506
Classwise samping GAN	0.0083
Classwise sampling and Client grouping	0.0029

Table 5.8: Standard Deviations of Accuracy of GAN Techniques - Bank Marketing dataset

Technique	Standard Deviation
Federated GAN	0.0498
Classwise samping GAN	0.0653
Classwise sampling and Client grouping	0.0556

CHAPTER 6

Conclusion

In this chapter, we go over our main findings and limitations of our proposed system. We finish by pointing out possible next steps for future research, especially focusing on using GAN methods for federated learning in edge computing with the incomplete data.

6.1 Summary

The primary objective of this thesis was to enhance the accuracy of Federated classification models in the context of incomplete data settings. To achieve this, we formulated three research questions.

In our first research question, we examined how the accuracy of a Federated learning model varies across different levels of data incompleteness. Our findings indicate that for the Adult and Intrusion datasets, accuracy substantially decreased in extremely incomplete data conditions. In contrast, the Bank Marketing dataset experienced only a modest 2% decline in accuracy under similar conditions.

For the second research question, we explored the changes in the accuracy of the Federated learning model after augmenting the original data with GAN-generated data. We compared three different Federated GAN techniques for this purpose. Notably, the Adult and Intrusion datasets exhibited encouraging results. The accuracy for the Adult dataset improved by approximately 4% using both the Classwise Sampling GAN and the Classwise Sampling and Client Grouping GAN methods. However, the Federated GAN technique underperformed. For the Intrusion dataset, we observed an impressive 17% increase in accuracy when using the Classwise Sampling GAN and the Classwise Sampling and Client Grouping GAN techniques. In contrast, the Bank Marketing dataset presented challenges due to the extreme imbalance in the centralized dataset. Consequently, the quality of the generated data was subpar. The disproportionately small number of samples for specific class labels may have contributed to these poor results. Moreover,

the difference in accuracy between the ideal federated settings and the incomplete data settings was a mere 2%. As a result, adding GAN-generated data to the mix actually led to a decrease in overall accuracy.

In our third research question, we assessed the quality of the generated datasets in comparison to the original datasets. For this analysis, we employed a Random Forest classifier and examined the machine learning efficacy metrics. We observed that the Classwise Sampling and Client Grouping GAN exhibited consistent stability across data points.

6.2 Limitations

Although our methodology effectively tackles numerous challenges related to data incompleteness in federated learning, it does have certain limitations. The following sub-sections provide further details on these constraints.

Sharing of categories of categorical columns and data distribution to the server

In this thesis, we address a critical limitation related to privacy in Federated Learning. Specifically, the categories of categorical columns and data distributions are shared with the server during the federated encoding step and client grouping process without the use of encryption. In traditional Federated Learning scenarios, the primary focus is on preserving data privacy by keeping sensitive data localized on the client's device. However, in our approach, certain meta-information about the data—namely the categories of categorical variables and the distribution of data—is transferred to the server in plaintext.

This disclosure could potentially expose sensitive attributes of the dataset and compromise the overall goal of data privacy inherent in Federated Learning frameworks. While our method may improve the performance and efficiency of federated encoding and client grouping, it does so at the expense of fully secure data handling.

Training time

The time required to adequately train a Generative Adversarial Network (GAN) model can fluctuate significantly based on various factors. This is especially true for specialized federated GAN techniques such as 'Classwise Sampling' and 'Classwise Sampling and Client Grouping'. In this context, two key variables that largely influence the training time are the number of classes and the number of samples within those classes. In this thesis, we only use binary class datasets for the evaluations.

Continuous Columns

In this work, we opted not to perform transformations on continuous columns, a decision that diverged from several recent papers on similar topics. Those studies often utilize

techniques such as Gaussian Mixture Models (GMMs) to transform continuous columns. The rationale for not employing GMMs in our approach was twofold. *Firstly,* using GMMs could necessitate additional processing to forward the data distribution, particularly for continuous columns. More importantly, the primary objective of this research was to develop techniques that minimally alter the original data distribution. Preserving the integrity and authenticity of the data was paramount, and for this reason, we decided against using transformation techniques like GMMs. *Secondly,* we examined the performance of our models on the Adult and Intrusion datasets, both of which, after encoding, had a large number of categorical columns. Specifically, there were around 100 encoded categorical columns as opposed to just 4 or 5 continuous columns. This imbalance led to interesting observations.

We found that the Federated Classwise Sampling GAN and Federated Classwise Sampling with Client Grouping GAN exhibited superior performance on these datasets. The large number of encoded categorical columns seemed to mitigate the challenge of modeling the distribution of the fewer continuous columns. This suggests that our approach was effective even without transformations for continuous variables, particularly when the dataset is predominantly categorical in nature.

6.3 Future work

In future work, we plan to use Gaussian Mixture Models (GMM) to make transformations to continuous columns, allowing for a comparative analysis of performance metrics. Techniques such as differential privacy or homomorphic encryption can also be leveraged to preserve the privacy-centric focus of Federated Learning, especially during the federated encoding of categorical columns and the sharing of data distribution.

List of Figures

List of Tables

List of Algorithms

Safe-Level-SMOTE

Bibliography

[1] Wikipedia contributors. Edge computing – Wikipedia, the free encyclopedia, 2023.

[2] Wikipedia contributors. Internet of things – Wikipedia, the free encyclopedia, 2023.

[3] Wikipedia contributors. Federated learning – Wikipedia, the free encyclopedia, 2023.

[4] Jason Brownlee. Random oversampling and undersampling for imbalanced classification, 2019.

[5] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[7] Léon Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer, 2012.

[8] Bernhard Schölkopf and A.J. Smola. *Smola, A.: Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge, MA*, volume 98. 01 2001.

[9] J. MacQueen. Some methods for classification and analysis of multivariate observations. 1967.

[10] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 1996.

[11] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[12] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1986.

[13] Leo Breiman. Random forests. *Machine Learning*, 2001.

[14] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, 1992.

[15] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 1991.

[16] Li Fei-Fei, Rob Fergus, and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[17] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 1986.

[18] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 2001.

[19] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, may 2020.

[20] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[21] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[22] Wikipedia contributors. Jensen–shannon divergence, 2023.

[23] Wikipedia contributors. Kullback–leibler divergence, 2023.

[24] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[25] Wikipedia contributors. Dirichlet distribution, 2023.

[26] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[27] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization, 2020.

[28] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning, 2021.

[29] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning, 2021.

[30] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[31] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*, pages 475–482. Springer, 2009.

[32] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. Ieee, 2008.

[33] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1–20, 2018.

[34] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[35] Taeho Jo and Nathalie Japkowicz. Class imbalances versus small disjuncts. *ACM Sigkdd Explorations Newsletter*, 6(1):40–49, 2004.

[36] Chanachok Chokwitthaya, Yimin Zhu, Supratik Mukhopadhyay, and Amirhosein Jafari. Applying the gaussian mixture model to generate large synthetic data from a small data set. In *Construction Research Congress 2020*, pages 1251–1260. American Society of Civil Engineers Reston, VA, 2020.

[37] Devin Soni. Introduction to bayesian networks. `https://towardsdatascience.com/introduction-to-bayesian-networks-81031eeed94e`.

[38] David Foster. Chapter 3: Variational autoencoders. *Generative deep learning: teaching machines to paint, write, compose, and play*, pages 61–96, 2019.

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[40] Fatemeh Fahimi, Zhuo Zhang, Wooi Boon Goh, Kai Keng Ang, and Cuntai Guan. Towards eeg generation using gans for bci applications. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 1–4. IEEE, 2019.

[41] Adamu Ali-Gombe and Eyad Elyan. Mfc-gan: Class-imbalanced dataset classification using multiple fake class generative adversarial network. *Neurocomputing*, 361:212–221, 2019.

[42] Murphy Yuezhen Niu, Alexander Zlokapa, Michael Broughton, Sergio Boixo, Masoud Mohseni, Vadim Smelyanskyi, and Hartmut Neven. Entangling quantum generative adversarial networks. *Physical Review Letters*, 128(22):220505, 2022.

[43] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[44] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[45] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

[46] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. *Advances in neural information processing systems*, 29, 2016.

[47] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[48] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[49] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[50] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[51] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[52] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[53] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[54] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019.

[55] Weiwei Cai and Zhanguo Wei. Piigan: generative adversarial networks for pluralistic image inpainting. *IEEE Access*, 8:48451–48463, 2020.

[56] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*, 2018.

[57] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

[58] Amirarsalan Rajabi and Ozlem Ozmen Garibay. Tabfairgan: Fair tabular data generation with generative adversarial networks. *Machine Learning and Knowledge Extraction*, 4(2):488–501, 2022.

[59] Hui Wen, Yue Wu, Chenming Yang, Hancong Duan, and Shui Yu. A unified federated learning framework for wireless communications: towards privacy, efficiency, and security. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 653–658, 2020.

[60] Shaoming Duan, Chuanyi Liu, Peiyi Han, Xiaopeng Jin, Xinyi Zhang, Tianyu He, Hezhong Pan, and Xiayu Xiang. Ht-fed-gan: Federated generative model for decentralized tabular data synthesis. *Entropy*, 25:88, 12 2022.

[61] Adult data set. `https://archive.ics.uci.edu/dataset/2/adult`. Retrieved August 29, 2023.

[62] Network intrusion detection dataset. `https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection?select=Train_data.csv`. Retrieved August 29, 2023.

[63] Bank marketing dataset. `https://archive.ics.uci.edu/dataset/222/bank+marketing`. Retrieved August 29, 2023.

[64] Shaashwat Agrawal, Sagnik Sarkar, Ons Aouedi, Gokul Yenduri, Kandaraj Piamrat, Sweta Bhattacharya, Praveen Reddy, and Thippa Gadekallu. Federated learning for intrusion detection system: Concepts, challenges and future directions, 06 2021.

[65] Mei Cao, Yujie Zhang, Zezhong Ma, and Mengying Zhao. C2s: Class-aware client selection for effective aggregation in federated learning. *High-Confidence Computing*, 2(3):100068, 2022.

[66] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd*, 96:226–231, 1996.