Federated Learning for Anomaly Detection in Healthcare Network Traffic: A Snowflake Integrated Approach

Atshaya Srinivasan

201774445

A DISSERTATION

Submitted to

The University of Liverpool

in partial fulfilment of the requirements
for the degree of

MASTER OF SCIENCE

DATA SCIENCE AND ARTIFICIAL INTELLIGENCE

Student Declaration

- I confirm that I have read and understood the University's Academic Integrity Policy.
- I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.
- I confirm that I have not copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work.
- I confirm that I have not previously presented the work or part thereof for assessment for another University of Liverpool module.
- I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.
- I confirm that I have not incorporated into this assignment material that has been submitted by me or any other person in support of a successful application for a degree of this or any other university or degree-awarding body.
- I confirm that I used ChatGPT solely for proofreading spelling and grammar correction in this assessment, and all content and ideas are my own.

SIGNATURE `Atshaya Srinivasan`
DATE September 20, 2024

This dissertation contains material that is confidential. It is included here on the understanding that this will not be revealed to any person not involved in the assessment process.

Acknowledgments

I would like to express my sincere gratitude to Angelos Koufonikos (a.koufonikos2@liverpool.ac.uk), my supervisor, for his invaluable support throughout this journey. A heartfelt thank you to Dr. Alexandar Vincent-Paulraj (alexandar.vincent-paulraj@liverpool.ac.uk) for being a supportive mentor. His encouragement and advice have been greatly appreciated along the way. Lastly, I am deeply thankful to my parents for their unconditional love, support, and belief in me, which have been a constant source of motivation.

Federated Learning for Anomaly Detection in Healthcare Network Traffic: A Snowflake Integrated Approach

Abstract

This project explores the use of **Federated Learning (FL)** to enhance cybersecurity in **Internet of Healthcare Things (IoHT)**, a network of connected medical devices that collect and share sensitive health data. The primary goal was to create a framework that is both scalable and privacy preserving, capable of identifying cyber threats (anomalies) and guaranteeing adherence to healthcare data rules like GDPR and HIPAA. FL protects patient privacy by allowing each device to locally train machine learning models, particularly multi classifiers, sharing only model parameters rather than raw data. **Snowflake**, cloud based data platform, was used to manage model aggregation, store encrypted model parameters, and provide a secure infrastructure. This research specifically investigated how differences in client models, such as varying input dimensions, impact the performance of the global model at the first iteration of FL. Results showed that combining client models with similar input dimensions yielded better results, while models with different dimensions led to challenges in accuracy and generalization. A streamlit application integrated with Snowflake was developed to enable near real time interaction, allowing authorised users to visualize results, assess model performance, predict attack types.

Keywords: Federated Learning, Snowflake, Streamlit, IoHT, Healthcare, cyberattacks, GDPR, model heterogeneity

Statement of Ethical Compliance

This research was conducted in full compliance with the ethical guidelines and standards of the University of Liverpool. The data used in this study falls under Category A0, with no direct involvement of human participants. Publicly available dataset was utilized throughout the project, and all ethical guidelines and data protection regulations were strictly adhered to. Throughout the research, privacy and data security measures were consistently applied to ensure full compliance with ethical standards. No sensitive or personal data was involved at any stage of the research.

Contents

List of Figures

List of Tables

Abbreviation and Nomenclature

V1 – Version 1
V2 – Version 2
V3 – Version 3
V4 – Version 4
V5 – Version 5
UI – User Interface
FL – Federated Learning
IoT – Internet of Things
ReLU – Rectified Linear Unit
NHS – National Health Service
CSV – Comma-Separated Values
SQL – Structured Query Language
RBAC – Role-Based Access Control
RPA – Robotic Process Automation
IoHT – Internet of Healthcare Things
JSON – JavaScript Object Notation
API – Application Programming Interface
HDF5 – Hierarchical Data Format version 5
GDPR – General Data Protection Regulation
y – Target variable in machine learning models
HIPAA – Health Insurance Portability and Accountability Act
df – DataFrame (a data structure in Python's pandas library)
ROC-AUC – Receiver Operating Characteristic - Area Under Curve
ECU-IoHT – Electronic Control Unit - Internet of Healthcare Things

Chapter 1

Introduction

The increasing integration of IoHT devices into healthcare systems is transforming the industry by enabling real time patient monitoring, data driven diagnostics, and more personalized care. These devices, including wearable health monitors, smart medical equipment, and diagnostic tools, generate vast amounts of data that can improve patient outcomes. However, the same interconnected nature of these systems also makes them vulnerable to cyberattacks. As the number of connected devices grows, so do the opportunities for breaches, service disruptions, and compromised patient safety.

This research addresses these challenges by developing a scalable and privacy preserving framework that uses **Federated Learning(FL)** and **Snowflake's** cloud platform to detect and mitigate cyber threats in IoHT environments.

1.1 Real World Example - Ransomware Attack on Synnovis

The ransomware attack on **Synnovis**, a major diagnostics provider in the UK, is a recent example of the danger posed by cyberattacks in healthcare. Synnovis delivers essential laboratory services to the **National Health Services (NHS)** and process large amounts of medical data. The attack targeted its IT infrastructure, causing delays in processing test result, which directly affected patient care. This incident led to postponed diagnoses, delayed treatments, and, in some cases, rescheduled surgeries due to unavailable medical data [1]

This example highlights the serious consequences cyberattacks can have on healthcare systems. Beyond financial losses, such attacks risks patient safety and trust. With the rise in these attacks, healthcare organizations are under pressure to strengthen their cybersecurity defenses to protect critical infrastructure and patient data.

1.2 Scope

This project focuses on addressing the cybersecurity risks within IoHT systems [2], particularly in safeguarding the integrity, confidentiality and availability of healthcare data.

It emphasizes the unique characteristics of IoHT networks, including their decentralised nature, the sensitivity of the data involved, and the wide range of devices with varying computational capacities.

The research develops a framework combining FL, enabling decentralized model training without sharing sensitive data, and the scalable and secure architecture of Snowflake [3], [4] , a cloud based data platform. The scope includes developing machine learning models to detect threats, using FL to maintain patient privacy, and evaluating the framework using the ECU-IoHT network traffic dataset [5], which simulates real world attack scenarios.

1.3 Problem Statement

The rapid adoption of IoHT devices in healthcare has improved patient care through continuous data collection and real time monitoring. However, the highly decentralised nature of these networks presents multiple vulnerabilities to cyberattacks. Traditional centralised security measures are ill-suited for protecting such systems, and centralizing data collection increases the risk of breaches. IoHT and edge devices often lack the processing power to implement conventional centralized security methods.

Attacks like the one on **Synnovis** demonstrate the vulnerability of healthcare infrastructure [1]. Beyond ransomware, healthcare organizations face risks from **DDoS attacks, malware and unauthorized access**, all of which can disrupt services and endanger patient safety. Healthcare data is also subject to strict privacy regulations such as **GDPR**, which require careful handling of patient data. As such, there is a need for a scalable, privacy preserving solution that monitors and protects IoHT networks while complying with regulatory requirements.

1.4 Approach

To address these challenges, this research proposes a novel **anomaly detection framework** that integrates FL with Snowflake's cloud platform. The framework enables decentralized model training on IoHT devices [6], ensuring that sensitive patient data remains localized. Only model updates are shared, which are then aggregated on Snowflake's infrastructure to form a global model capable of detecting cyber threats.

The key component of this approach is FL, which allows decentralized machine learning across multiple healthcare devices without sharing raw data. **Snowflake** acts as the central hub for aggregating model updates and managing network traffic data. Its scalable architecture and advanced security features [4] ensure that the framework can handle large volumes of data while maintaining high security standards and regulatory compliance.

The framework is developed and tested using the **ECU-IoHT network traffic dataset**, providing realistic data on IoHT network operations and cyberattack scenarios [5].

1.5 Outcome

The primary outcome of this project phase is to evaluate how **model heterogeneity** and **local model performance** affect the global model's overall performance within a single iteration of FL, integrated with Snowflake. Additionally, the research will develop a **Snowflake integrated streamlit application** that allows users to Interact with the FL system, predict different types of cyberattacks, and visualize and analyze prediction results.

By the end of this phase, the project will have laid the foundation for understanding key factors influencing FL model performance and demonstrated an initial interface for practical interaction with the system.

Chapter 2

Background

The rise of the **IoT** and **IoHT** has transformed healthcare by enabling real time monitoring and faster diagnostics. However, with the increase in connected devices [7], new challenges around **data privacy, security, and resource management** have emerged. FL has shown potential in addressing these issues, especially in the areas of **cybersecurity and healthcare applications**. FL allows healthcare systems to utilize **machine learning** while preserving the privacy of sensitive medical data [6]. It enables multiple institutions or devices to collaboratively train models without sharing raw data, ensuring compliance with regulations like **GDPR**. For example, [2] proposed a deep neural network based FL system for intrusion detection in IoHT environments, which achieved a high detection accuracy of 91.4% while preserving patient data privacy by training models locally on distributed datasets.

Similarly, [8] discusses a FL self-learning anomaly detection system for IoT, identifying compromised devices based on their communication patterns. The system further highlighted FL's role in securing IoT networks, achieving a 95.6% detection accuracy without the need for labeled data.

A key gap in [2], [8] is understanding how **client performance variability**, especially from clients with limited computation power or lower quality data impacts the global model. In real world deployments, clients with poor performance can introduce noise or skew aggregated model updates leading to a decline in the global model's accuracy. Although some methods have been proposed [9] to handle this issue, such as weighting model updates based on data quality or volume, and model compression and adaptive update strategies, and studies like [10] [11] have addressed some aspects of data heterogeneity using federated aggregation techniques, they have primarily focused on scenarios where feature sets among clients are consistent and more research is needed to fully mitigate the impact of client heterogeneity on FL systems. In response to these gaps, this project will explore client selection strategy, padding techniques to better handle data heterogeneity and understand the performance of the global model.

Research on attack detection in medical cyber-physical systems (MCPS) has faced similar challenges. A systematic review by [6] noted that the heterogeneity of devices and lack

of standardization in healthcare environments complicates efforts to secure these systems, especially when using machine learning techniques and it addresses the lack of suitable medical-specific datasets for testing and validating intrusion detection systems in healthcare. However, [5] provides ECU-IoHT dataset specifically designed for healthcare network traffic which includes protocols, packet lengths, source, destinations, etc,.

As highlighted in [12], FL introduces concerns around the security of model updates and potential data leakage. Cloud computing plays a critical role in supporting FL, as it provides the infrastructure needed for distributed model updates and aggregation. For example, [13] proposed a secure **fog cloud architecture** for cyberattack detection in IoHT environments, ensuring that data processed at different nodes remains secure throughout the model update process.

Additionally, the application of **RBAC** in managing global model updates within FL systems has not extensively explored in [7] [6] [11]. RBAC could help secure the system by controlling which users or devices can contribute to the global model, ensuring that only trusted sources are involved in the aggregation process. This would help reduce the risk of malicious contributions and further enhance the security of FL systems.

Moreover, while some studies focus on monitoring model performance, there has been limited development of real-time dashboards capable of retrieving data from cloud warehouses and predicting cyberattacks in near real time [7]. Creating such a tool would significantly improve the usability of FL systems by providing immediate feedback to administrators, enabling faster detection and response to potential threats. This project will also focus on this open area of research.

To address the challenges of managing global model updates, monitoring model performance, and predicting cyberattacks in real-time, Snowflake, a cloud data warehouse platform, was integrated as a solution. By utilizing Snowflake's elastic computing capabilities [3] [4], the platform allows for scalable and efficient handling of the vast amounts of data required for the federated learning (FL) model update process. This ensures that data from multiple devices and institutions is processed without overburdening local resources, which is especially crucial in heterogeneous environments such as healthcare.

Furthermore, a versioning system [7] was introduced within Snowflake to track the history of model updates, offering a clear audit trail. This feature enhances transparency by allowing administrators to monitor changes, identify issues, and revert to previous versions when necessary, ensuring the integrity of the global model. Leveraging Snowflake's powerful processing infrastructure, FL systems can more effectively manage real-time updates, monitor model accuracy, and predict potential cyberattacks, thereby improving overall system security and performance.

Incorporating Snowflake into the FL workflow not only optimizes the model aggregation process but also facilitates real-time dashboards for administrators to track model performance and potential threats, significantly enhancing both usability and response time in FL-based healthcare applications. This novel approach ensures more reliable and secure management of federated learning models in IoT and IoHT environments.

Chapter 3

Project Ethics

This project was carried out in line with the ethical guidelines of the University of Liverpool. The data used was publicly available, classified as Category A, and no human participants were involved, falling under Category 0. Great care was taken to follow all rules around data privacy and security, making sure no personal or sensitive information was included. By using publicly available datasets, the project stayed within all the required legal and institutional guidelines.

Throughout the research, the focus was on being transparent, accurate, and maintaining integrity at every step, ensuring that ethical standards were always upheld.

Chapter 4

Design

This section provides an overview of the system architecture and design decisions that were made to develop a **FL framework** for anomaly detection in IoHT networks. The goal was to design a system that could not only detect cyber threats in real-time but also maintain data privacy, comply with healthcare regulations, and remain scalable for future needs.

4.1 System Architecture

The overall architecture consists of three main components. See Figure 4.1

1. Hospitals edge device (clients)
2. FL Framework
3. Snowflake cloud platform

At the core of the system is FL, which allows edge devices to locally train machine learning models.Instead of sharing raw data, these devices send only the learned model parameters (weights and biases) to a central server [9]. The server aggregates these updates into a **global model** and then sends it back to the devices. This decentralized approach helps protect sensitive patient data and ensures compliance with HIPAA and GDPR.

The Snowflake cloud platform serves as the central hub for aggregating model updates. It is also responsible for managing client data, storing model parameters, and providing infrastructure for real time analysis and visualisation of results. The system was designed to scale easily, allowing additional IoHT devices to join the network without requiring significant changes to the underlying architecture.

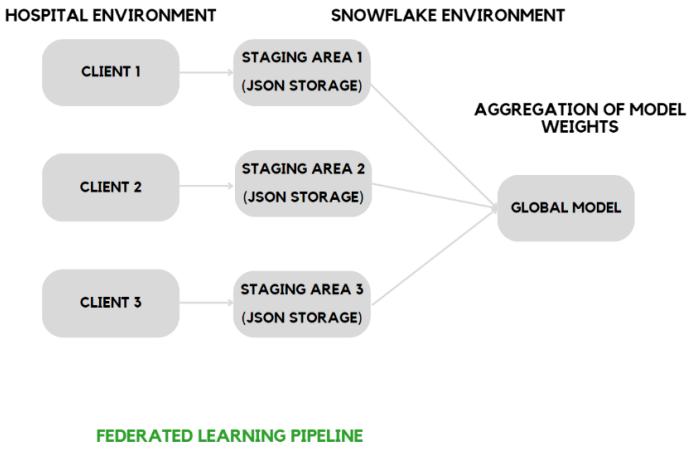


Figure 4.1: System Architecture

4.2 Key Design Considerations

4.2.1 Privacy and Security

One of the most important considerations during the design phase was ensuring privacy and security. With sensitive medical data involved, it was critical that no raw data be shared between devices [7] or uploaded to a central location. By using FL, the system keeps all patient data on the local server and shares only anonymized model parameters. This reduces the risk of data breaches or exposure.

Additionally, the Snowflake platform was selected because of its advanced security features, including end to end encryption, role based access control [3], [4]. These features ensure that only authorized personnel can access sensitive information, further protecting patient data.

4.2.2 Scalability

The system was designed to be scalable, accommodating a large number of edge devices without sacrificing performance. As more devices join the network, the **Snowflake infrastructure** efficiently manages the increased volume of data and model updates using its elastic compute power. The use of cloud resources ensures that the system can handle growing data loads while maintaining responsiveness and reliability.

4.2.3 Federated Learning Framework

The FL process is broken down into several key steps

Local Model Training

This project used Snowflake's virtual machine as a local server. Each hospital environment trains its local machine learning model using its own data. In this case, a neural network was used, with each model designed to classify network traffic into five categories: normal traffic and four types of attacks. The training process is local to each device, ensuring that patient data stays private.

Model Updates

After training, each device sends only its model parameters to the central server, which in this case is managed by Snowflake.

Global Model Aggregation

Once the central server receives the model parameters from all the devices, it aggregates them using a padding and average technique. This technique combines the weights and biases from all local models to create a global model. The aggregated global model is then sent back to each edge devices. This allows the devices to update their local models. However, this phase of the project only focusing on a single iteration, without redistribution.

Versioning for Model Updates

Each saved model and its weight file are versioned (e.g., v1, v2) to keep a clear record of changes over time. This helps track updates [7], like retraining or tuning, and makes it easy to compare or retrieve specific versions later. By versioning, this project ensure every step in the model's development is documented and accessible.

4.2.4 User Interface and Visualization

A key part of the design was creating a user friendly interface to interact with the FL framework. A streamlit based web application was developed to make the system accessible to administrators and healthcare professionals. The application is only available to authorised user access to Snowflake. Figure A.1 shows the application UI.

Chapter 5

Implementation

5.1 Overview

This chapter explains the implementation of **FL framework** used for anomaly detection in the IoHT environment. The system integrates decentralized machine learning models, a central aggregation mechanism, and an interactive user interface built using streamlit. Data management and model storage are supported by Snowflake.

The implementation process involved the following steps

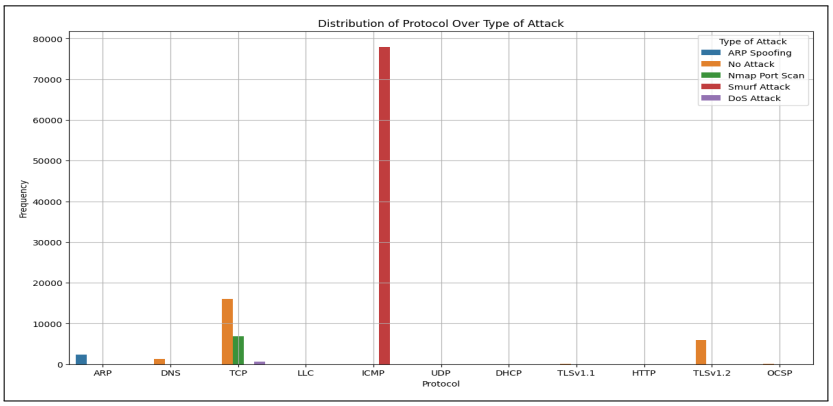
5.2 Data Preparation and Partitioning

The **ECU-IoHT dataset** [5] was first processed to make it suitable for FL. Total number of records in the dataset was 111207. Before splitting the dataset among clients, 10% of the dataset was set aside as a test holdout to evaluate the model's final performance. The remaining dataset was further divided into three stratified subsets to prevent class imbalance, ensuring each client received a representative distribution of attack types (Figure 5.2). This partitioning prevented data skew, which could negatively impact the local model performance (Figure 5.1). After the split, the subsets were stored in any local database server. For this study, I used the Snowflake database, allowing each client to access its dataset for local training. See Figure A.1(c)

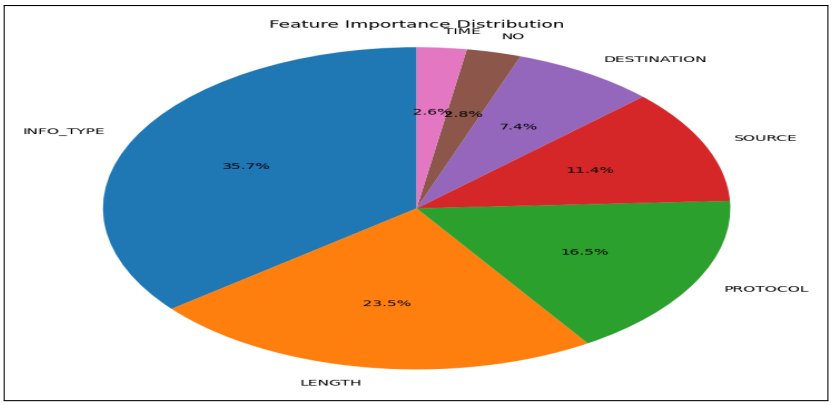
5.3 Federated Learning Setup

5.3.1 Data Preprocessing and Feature Engineering

This section describes the steps taken to preprocess the data and prepare it for model training. The dataset used in this research contains network traffic data. Among nine features [5] such as `'NO.'`, `'INFO'`, `'PROTOCOL'`, `'TIME'`, `'TYPE'`, `'LENGTH'`, `'SOURCE'`, `'DESTINATION'`,`'TYPE`, `'TYPE OF ATTACK'`, based on the feature importance (Figure 5.1



(a) Distribution of protocol over type of attack



(b) Feature importance

Figure 5.1: ECU- IoHT dataset analysis

(b)), only `INFO, PROTOCOL, LENGTH` features are considered crucial in determining the type of attack. Below is the detailed description of the preprocessing steps.

Data Fetching and Conversion

The initial step involved fetching data from a local server. In this project, Snowflake was used as local virtual machine. The data retrieval was achieved using a custom function `fetch_from_table_data(session, table_name)`, which retrieves the data from the session and load it into memory. The fetched data was then converted into a pandas DataFrame format for easy manipulation and analysis using the function `create_dataframe(data)`.

Train-Test Split with Stratification

To ensure that the distribution of the categorical feature `PROTOCOL` was maintained across both the training and testing sets, a stratified shuffle splitting technique [14] was em-

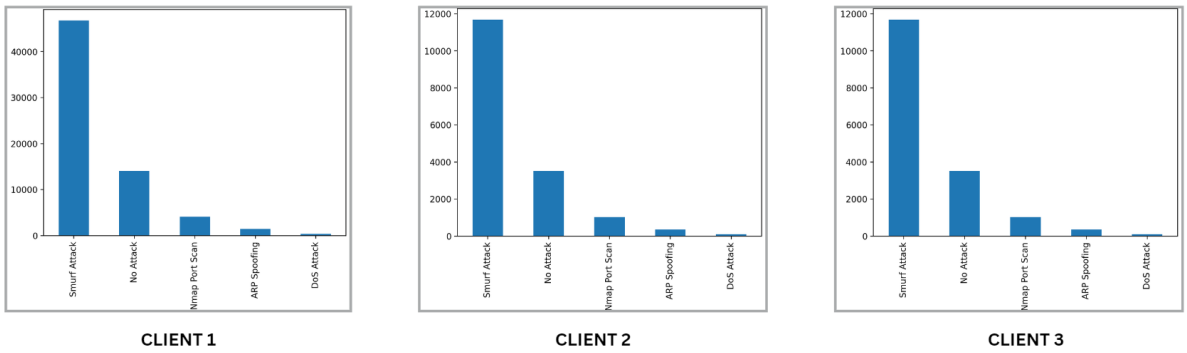


Figure 5.2: Distribution of target variable among each client

ployed.The function `data_stratified_split_train_test(df)` was developed for this purpose. It removes rare occurrences of protocols(fewer than two instances) to ensure balanced feature distributions. A `StratifiedShuffleSplit` method was used to split the dataset into training and testing sets while maintaining the same `PROTOCOL` distribution across both sets.

Feature Preprocessing

The raw feature data required cleaning and transformation before being used by the machine learning models. The function `preprocess_x(train_df,test_df)` handled this process in two key steps.

1. Fields such as `'NO','TIME','TYPE','SOURCE', and 'DESTINATION'` were removed, as they do not directly contribute to the detection of network attacks.

2. The `INFO` variable was analyzed to create a new feature based on predefined patterns related to network attack signatures. For instance, specific patterns in the `INFO` column combined with the `PROTOCOL` value were used to label entries as potential attacks as `Nmap, DoS, SMURF, ARP`. This new feature, `INFO_TYPE`, was added to the dataset for further processing.

3. Both `PROTOCOL` and `INFO_TYPE` were one-hot encoded using `OneHotEncoder`. This conversion was necessary to transform the categorical data into numerical format for machine learning algorithms.

The transformed features were then combined with the original dataset, and the redundant columns were dropped. The result was a fully preprocessed set of training and testing features.

Clients	Input dimension
client 1	16
client 2	15
client 3	13

Table 5.1: Input dimension of each client feature set after preprocessing

Label Preprocessing

The target variable `TYPE_OF_ATTACK` required preprocessing to compatible with machine learning models. This was achieved through one-hot encoding using the `preprocess_y(y)` function. Each class was encoded as a vector, and the corresponding class names were retained for future analysis.

The same steps would be followed for each client in their own dedicated Snowflake Virtual machine(local server). Although careful consideration was given to distributing features evenly among the three clients, after preprocessing, the clients still became heterogeneous in terms of their feature sets. See Table 5.1

5.3.2 Local Model Training

This section describes the process of training a local neural network model using the preprocessed data. This includes defining the architecture of the model, training it on the preprocessed dataset, saving the model and its weights, and versioning both for future updates and historical tracking.

Model Architecture and Configuration

The architecture of the model was implemented using the keras sequential API and consists of three layers.

1. Input layer takes in the preprocessed feature vectors, with the input dimenisons equal to the number of features.
2. Two dense layers with ReLU activation. The first hidden layer has 64 units. The second hidden layer has 32 units.
3. A final dense layer with 5 units and a softmax activation function to output the probabilities for five different classes representing various types of network attacks.

The model was compiled with Adam optimizer (with a learning rate of 0.01) and the categorical cross entropy loss function, which is suitable for multi class classification problems. The accuracy metric was used to evaluate the model's performance during training.

Model Training

The training function `client_training(x_train,y_train, input_dim)` handled the training process, where the model was fitted to the data over 10 epochs with a batch size of 32. The training process iteratively updated the model weights based on the categorical cross entropy loss, with the goal of minimizing the loss and improving accuracy.

After the training was complete, the trained model was saved in an HDF5 format for future use. To maintain a historical record of the model's evolution, the model file was versioned with a version identifier. This versioning allows for tracking changes as the model is updated or retrained.

Extracting and Saving Model Parameters

The weights of the trained model were extracted and saved in JSON format. The weight extraction process involved iterating through each layer of the model and retrieving its weights. These weights were converted into lists and stored in a dictionary, which was then saved to a versioned JSON file.

Uploading Model and Weights to Snowflake

The trained model HDF5 file and its corresponding weights JSON files were uploaded to Snowflake to ensure that they are securely stored as a compressed `.gz files` and accessible for future use.

5.3.3 Global Model Aggregation

This section describes the process of aggregating local model updates from three clients to create a global model. This method was part of a FL approach, where data stays with each client, but model updates were shared and combined to improve the global model.

Collecting Client Model Updates

The stored compressed model updates were downloaded from Snowflake internal stage, decompressed, and then loaded into memory for processing.

Aggregating Model Weights

To create the global model, the weights from each client were combined. To handle heterogeneity among client's input dimensions, the process involved Padding or trimming weights that it adjusted to the shape of the weights to match across clients.

Once the weights were aligned, calculated average across all clients for each layer in the model. This includes both the weights and the bias. This ensures that the knowledge learned by each client is effectively merged into a single global model.

Updating and Saving the Global Model

After the weights were aggregated, they were set into the global model. The model was then compiled and saved with a version number, ensuring that each update is tracked. This allowed to keep a record of all our experiment's model updates and compare performance over time.

Both the model file and a JSON file containing the model parameters were saved and uploaded to Snowflake internal stage, where they could be accessed for further training or deployment. Each version was clearly labeled for easy retrieval and future updates.

5.4 Development of the Streamlit Application

To make the FL system easy to use, a streamlit application was developed which connects directly to Snowflake for data management and model storage. This app allows users to load datasets, run predictions, and view results all in one place. It provides an intuitive interface where users can interact with the system without needing to deal with the underlying technical details.

The app connects to Snowflake using **Snowpark session** [3], which makes it easy to access the Snowflake warehouse. This connection setup, which includes credentials and warehouse settings, allows the app to run SQL queries, fetch datasets in real-time, and interact with stored models seamlessly.

Through the app, users can select a dataset using a simple dropdown menu and load it into the application with the "Load Dataset" button. See figure A.1 (a). Once the selected dataset is retrieved from Snowflake, users can start prediction process by clicking the "Predict Attack" button. The app fetches the global model stored in Snowflake (figure A.1 (c)), applied it to the loaded dataset and predicts the types of attacks present in the data by undergoing proper data preprocessing step. See Figure A.1 (b)

The results are then visualized through easy to understand bar/pie charts, showing the distribution of predicted attack types. Users also have the option to download the prediction results as a csv file for further analysis or reporting. See Appendix for application code and UI screenshot.

Chapter 6

Experimental Results

This section presents the results of the FL framework implemented for anomaly detection in the IoHT environment. The system's performance was evaluated based on key metrics such as **accuracy, precision, recall, F1-Score and ROC-AUC score**. These metrics provide insight into how well the local and global model performs. The results were obtained from applying the model to the test holdout set.

A more detailed visualisation of true positives, false positives, true negatives, and false negatives is available in Appendix B, where confusion matrices shows the model's classification performance for different types of attacks. These matrices provide misclassification patterns, supporting the analysis of performance trends and improvements across different versions.

Client selection	Accuracy	Precision	Recall	F1-Score	ROC-AUC score
Client 1	0.992	0.986	0.992	0.989	0.999
Client 2	0.993	0.987	0.993	0.990	0.998
Client 3	0.992	0.986	0.992	0.989	0.995

Table 6.1: Local clients model performance result - Before fine tuning

Client selection	Accuracy	Precision	Recall	F1-Score	ROC-AUC score
Client 1	1.000	1.000	1.000	1.000	1.000
Client 2	0.993	0.987	0.993	0.990	0.998
Client 3	0.994	0.988	0.994	0.991	0.999

Table 6.2: Local clients model performance result - After fine tuning

6.1 Local Model Performance Before and After Fine-Tuning

The tables below demonstrate the performance of each client's local model both before and after fine-tuning. The aim was to assess whether fine-tuning could enhance the models' ability to accurately detect anomalies in network traffic.

1. Table 6.1 illustrates the results prior to fine-tuning. All three clients show strong initial performance, with accuracy rates of around 99.2% for Client 1 and Client 3, and 99.3% for Client 2. These high accuracy values indicate that the models were already effective at classifying network traffic. In addition, the precision, recall, and F1-scores were also high, signifying that the models were not only detecting anomalies but doing so with a high level of reliability. The ROC-AUC scores, which measure how well the models can distinguish between normal traffic and cyberattacks, were close to 1.0, suggesting robust performance.

2. Table 6.2 presents the performance after fine-tuning the local models. Client 1 showed the most significant improvement, reaching a perfect score in all metrics—accuracy, precision, recall, F1-score, and ROC-AUC—indicating that its model had become highly optimized for its local dataset. In contrast, Client 2 and Client 3 saw only slight improvements after fine-tuning. Although their accuracy and other metrics increased slightly, the changes were less dramatic compared to Client 1.

3. These results suggest that while fine-tuning can lead to substantial improvements in some cases, such as for Client 1, the impact may be minimal for other clients, depending on their initial performance and dataset characteristics. The perfect score achieved by Client 1, although impressive, raises the possibility of overfitting, where the model performs exceptionally well on local data but may struggle with generalization when aggregated into a global model. This highlights the need for careful consideration when fine-tuning models in a federated learning environment.

Client selection	Version	Accuracy	Precision	Recall	F1-Score	ROC-AUC score
All clients (Before fine tuning)	v1	0.909	0.848	0.909	0.873	0.994
All client (After fine tuning)	v2	0.029	0.068	0.029	0.041	0.862
client 1 and 2	v3	0.723	0.703	0.723	0.712	0.865
client 1	v4	1.000	1.000	1.000	1.000	1.000
client 1 and 3	v5	0.227	0.244	0.227	0.202	0.711

Table 6.3: Global model performance - Experimental result

6.2 Global Model Performance

6.2.1 Experiment on all three clients model updates before fine tuning - Global Model v1

In this initial experiment, the global model was created by aggregating the model parameters from all three clients before any fine tuning. The results in Table 6.3 indicate that the initial aggregation of client models was effective, resulting in a global model with strong generalization capabilities. The high ROC-AUC score suggests that the model was able to distinguish between positive and negative instances effectively. However, the slight discrepancy between precision and recall highlights a potential issue with imbalanced contributions from clients with varying input dimensions.

6.2.2 Experiment on all three clients after fine tuning - Global Model v2

After fine tuning the local models on each client, the model parameters were aggregated into a second global model. The performance on the test dataset was mentioned in Table 6.3

The sharp drop in performance highlights a significant issue such as fine tuning caused the local models to overfit their individual datasets. When the fine-tuned models were aggregated, the resulting global model was unable to generalize well to the test data. The performance degradation also suggests that the padding technique, especially for Client 3 (with the smallest input dimensions), diluted its contribution to the global model, leading to poor results.

6.2.3 Experiment on client 1 and 2 fine-tuned - Global Model v3

To mitigate the effects of padding and fine-tuning, a third experiment aggregated only the model parameters from Client 1 and Client 2, which had more similar input dimensions (16 and 15, respectively). The results for Global Model v3 were mentioned in Table 6.3

By excluding Client 3 from the aggregation, the global model's performance improved significantly. The reduced need for padding likely contributed to better generalization, as the local models were more compatible with one another. This experiment demonstrates the importance of considering input dimension similarity when aggregating client models.

6.2.4 Experiment on client 1 fine-tuned - Global Model v4

In the next experiment, the model parameters from only Client 1 were aggregated into the global model (Global Model v4). The results showed perfect scores mentioned in table 6.3 While these results may appear impressive, they indicate overfitting. The global model essentially replicated the performance of Client 1's model, which had been fine-tuned to perfection on its local dataset. Although the test set results were flawless, the global model would likely fail when applied to data from other clients. This use case will be tested in

the next phase while performing redistribution step. The current result highlights the risk of overfitting in FL when relying too heavily on a single client's contributions.

6.2.5 Experiment on client 1 and 3 fine-tuned - Global Model v5

In the final experiment, the model parameters from Client 1 (16 dimensions) and Client 3 (13 dimensions) were aggregated into Global Model v5. The performance results were mentioned in Table 6.3. The significant performance drop highlights the challenges of aggregating model parameters from clients with heterogeneous input dimensions. The padding technique diluted the contribution of Client 3, making it difficult for the global model to generalize effectively. This experiment underscores the importance of aligning input dimensions or exploring more sophisticated aggregation methods when dealing with heterogeneous data.

Chapter 7

Discussion

7.1 Insights from experiments

The experiments demonstrated that when clients in FL have consistent input dimensions, the global model performs significantly better. In contrast, mismatched input dimensions and local fine-tuning led to overfitting and weaker results. This underscores the importance of data consistency among clients, especially in IoHT environments where devices often have varying capabilities and data quality.

However, the padding technique used to handle different input sizes introduced challenges like diluted contributions from smaller models and increased the risk of overfitting. For instance, smaller models were overshadowed during aggregation, resulting in an imbalance where larger models had a greater influence. Additionally, fine-tuning at the client level often made overfitting worse, making it harder to create a global model that generalized well. Ultimately, aggregating clients with similar input dimensions produced much stronger performance, while mismatched dimensions caused a drop due to parameter incompatibility.

Although many papers have discussed federated learning (FL), most [15] [12] [7] have approached it in simulated environments by randomly splitting datasets and running FL over multiple epochs. However, they often fail to clearly explain how the framework performs at each iteration and how models are managed in real hospital settings. This project investigated how different client selection strategies impact global model performance in a single iteration, particularly in the context of the healthcare industry.

7.2 Challenges Addressed

Handling Heterogeneous Data and Client Model Input Dimensions

One of the significant challenges in this project was managing the heterogeneity of data and client models. The ECU-IoHT dataset presented a variety of network traffic data [5], which required careful handling to avoid issues such as class imbalance and data skew. To address this, a stratified splitting technique was employed, ensuring that each subset of

data provided to the clients had a representative distribution of attack types. This approach helped maintain consistency in model performance across different clients.

Additionally, the clients model in this FL system had varying input dimensions, which could have severely impacted the compilation of the global model during aggregation. To manage this, a padding technique was introduced to align the input dimensions across all clients. While this method ensured that models with different input sizes could contribute to the global model, it also posed challenges in balancing the contributions from clients, as smaller models tended to have less influence. Despite these hurdles, this approach allowed the system to maintain scalability and improve model performance across diverse environments.

Versioning of Model Updates in Snowflake

Another key challenge was tracking the different iterations [7] of model updates, especially as new versions of the local and global models were continuously generated and fine-tuned. Versioning these models in Snowflake was essential to maintain a clear history of changes, making it easier to compare different versions and track improvements. By saving each version of the model and its corresponding weights with a distinct identifier (e.g., v1, v2), and ensured transparency and traceability throughout the development process. This approach also allowed for efficient retrieval of specific versions for further analysis, ensuring that the experimentation process remained well-organized and manageable.

Access to Snowflake and Utilization of a Free Trial

A practical challenge during the project was gaining access to a suitable cloud platform for performing the experiments. Snowflake, with its advanced data handling and security features, was the ideal choice for this study. However, access to Snowflake's infrastructure was initially a limitation. To overcome this, I utilized Snowflake's one-month free trial, which provided ample resources and time to carry out the experiments. During this period, I was able to run multiple FL experiments, aggregate model updates, and perform data preprocessing at scale. The free trial offered the necessary computational power and secure environment to successfully complete the project, ensuring that all the tests and analyses were conducted without interruption.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this project, explored how FL can be used to improve security in healthcare networks, specifically focusing on the IoHT. As more medical devices are connected to each other, the risk of cyberattacks increases. Our goal was to create a system that could detect potential security threats while keeping sensitive patient data safe.

Using FL to train machine learning models directly on these devices, without needing to share raw data between them. This helps protect patient privacy, which is crucial in healthcare. Combined this with Snowflake, a cloud platform, to manage the model updates and provide a secure environment for data processing.

One key finding was that the success of FL depends on how similar the data and models from different devices are. When the models had similar input dimensions, the system performed well. But when the devices had different types of data, the system struggled, showing that the differences can lead to less accurate results. Other finding was that while fine-tuning models locally helped individual devices, it sometimes led to overfitting when combining them, making the overall system less effective at generalizing.

Snowflake played a vital role by providing the infrastructure to handle large amounts of data and keeping everything secure. The Streamlit application we developed allows users to easily interact with the system, view predictions, and monitor potential cyberattacks.

8.2 Real World Application

This research goes beyond just healthcare. As more industries rely on connected devices, the need for secure, privacy-preserving solutions grows. The framework developed for anomaly detection can be adapted to other areas, such as smart cities, industrial systems, and finance, where security is a top concern. The combination of FL and cloud platforms like Snowflake offers a scalable and flexible solution for organizations that need to protect their data while complying with privacy regulations.

8.3 Future Development and Research Frontiers

Looking ahead, there are numerous ways in which this project could be further developed and refined.

1. Improve how the models from different devices are aggregated by exploring more advanced techniques [10]. Analyse how the global model performs through multiple iteration is the next phase of the project.

2. Also, to make the Streamlit application more powerful by adding features like real-time alerts to notify administrators when suspicious activity is detected. Automating some of the processes using RPA, such as updating the models regularly, would make the system more efficient and responsive.

Chapter 9

BCS Project Criteria and Self Reflection

9.1 BCS Project Criteria

Application of Practical and Analytical Skills

This project applied federated learning and anomaly detection techniques within a healthcare setting to improve network security. The model was integrated with Snowflake. This project ensured secure data handling and real-time analysis, crucial for protecting sensitive healthcare information.

Innovation and Creativity

The main innovation of this project was the development of a privacy-preserving anomaly detection system using federated learning and understanding of how client selection impacts global model performance. Additionally, real world examples were incorporated to make the model more practical, aligning with real healthcare challenges.

Synthesis of Information

The project combined insights from various research papers and technical resources to create a well-rounded solution. This included drawing from studies on federated learning, anomaly detection, healthcare cybersecurity, snowflake's analytical and computing resources, combining these ideas to develop a model that addresses current industry needs.

Meeting a Real Need

There is a growing demand for systems that can detect cybersecurity threats in healthcare while preserving patient privacy. This project addressed that need by developing a system

capable of anomaly detection without compromising data security. The solution contributes to making healthcare networks more secure and reliable.

9.2 Self-Reflection

This project provided a great opportunity to learn both technical and project management skills. The ability to integrate federated learning and anomaly detection methods with a potent cloud-based platform such as Snowflake and use them in a realistic situation was a key strength. The real-world examples, such as the ransomware attack on Synnovis, helped ground the project in reality and made the solution more relevant to current issues in healthcare cybersecurity.

Working with federated learning and making sure that the models from many clients could be successfully merged was one of the challenges faced. This proved to be more challenging than anticipated, especially when handling the heterogeneity of the data from several clients.

With hindsight, more effective time management might have made handling these unforeseen obstacles go more smoothly. Even if the project's main objectives were fulfilled, more time and effort could have improved the global model even more. The study proved successful in spite of these challenges, offering a strong basis for federated learning applications in healthcare cybersecurity.

Both technical proficiency and self-management of a sizable project were enhanced by this experience. It allowed for the development of problem-solving and self-management skills and emphasized the importance of preparing for unforeseen technological difficulties.

Additionally, this project introduced me to the use of Overleaf and LaTeX for writing the dissertation and improved my document organization and formatting skills. I also learnt how to properly cite academic papers. This further improved my academic writing skill.

Bibliography

[1] D. Munroe, "Synnovis and data security - information for patients - king's college hospital nhs foundation trust," Aug. 2024. [Online]. Available: https://www.kch.nhs.uk/news/synnovis-and-data-security-information-for-patients/

[2] F. Mosaiyebzadeh, S. Pouriyeh, R. M. Parizi, M. Han, and D. M. Batista, "Intrusion detection system for ioht devices using federated learning," in *IEEE INFOCOM 2023 - Conference on Computer Communications Workshops, INFOCOM WKSHPS 2023*. Institute of Electrical and Electronics Engineers Inc., 2023.

[3] R. Sudeep, "Snowflake: a game-changer for healthcare industry - appstek corp," Aug. 2023. [Online]. Available: https://appstekcorp.com/blog/snowflake-a-game-changer-for-healthcare-industry/

[4] R. L. . D. D. Norms, "Healthcare data analytics — snowflake cloud - randy levine — disrupting data norms - medium," *Medium*, Apr. 2024. [Online]. Available: https://medium.com/@rjlhhi/healthcare-data-analytics-snowflake-cloud-d332d436177d

[5] M. Ahmed, S. Byreddy, A. Nutakki, L. F. Sikos, and P. Haskell-Dowland, "Ecu-ioht: A dataset for analyzing cyberattacks in internet of health things," *Ad Hoc Networks*, vol. 122, nov 2021.

[6] S. B. Weber, S. Stein, M. Pilgermann, and T. Schrader, "Attack detection for medical cyber-physical systems-a systematic literature review," *IEEE Access*, vol. 11, pp. 41 796–41 815, 2023.

[7] F. Zhang, D. Kreuter, Y. Chen, S. Dittmer, S. Tull, T. Shadbahr, M. Schut, F. Asselbergs, S. Kar, S. Sivapalaratnam, S. Williams, M. Koh, Y. Henskens, B. de Wit, U. D'Alessandro, B. Bah, O. Secka, P. Nachev, R. Gupta, S. Trompeter, N. Boeckx, C. van Laer, G. A. Awandare, K. Sarpong, L. Amenga-Etego, M. Leers, M. Huijskens, S. McDermott, W. H. Ouwehand, J. Rudd, C.-B. Schnlieb, N. Gleadall, M. Roberts, J. Preller, J. H. Rudd, J. A. Aston, C.-B. Schönlieb, N. Gleadall, and M. Roberts, "Recent methodological advances in federated learning for healthcare," *Patterns*, vol. 5, no. 6, p. 101006, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666389924001314

[8] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. R. Sadeghi, "DÏot: A federated self-learning anomaly detection system for iot," in *Proceedings - International Conference on Distributed Computing Systems*. Institute of Electrical and Electronics Engineers Inc., jul 2019, pp. 756–767.

[9] P. Nakayama, Kiyoshi and G. Jeno, "Federated learning with python." [Online]. Available: https://www.oreilly.com/library/view/federated-learning-with/9781803247106/B18369_02.xhtml

[10] Y. Zeng *et al.*, "Adaptive federated learning with non-iid data," *Computer Journal*, vol. 66, no. 11, 2023.

[11] D. Chu, W. Jaafar, and H. Yanikomeroglu, "On the design of communication-efficient federated learning for health monitoring," *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, Dec. 2022. [Online]. Available: https://doi.org/10.1109/globecom48099.2022.10001077

[12] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J Healthc Inform Res*, vol. 5, no. 1, 2021.

[13] U. Zukaib, X. Cui, C. Zheng, D. Liang, and S. U. Din, "Meta-fed ids: Meta-learning and federated learning based fog-cloud approach to detect known and zero-day cyber attacks in iomt networks," *Journal of Parallel and Distributed Computing*, vol. 192, p. 104934, Oct. 2024. [Online]. Available: https://doi.org/10.1016/j.jpdc.2024.104934

[14] S. Panthi, "Good train-test split: An approach to better accuracy," *Medium*, Feb. 2022. [Online]. Available: https://code.likeagirl.io/good-train-test-split-an-approach-to-better-accuracy-91427584b614

[15] S. Tijani, "Federated learning: a step by step implementation in tensorflow," *Medium*, Dec. 2021. [Online]. Available: https://towardsdatascience.com/federated-learning-a-step-by-step-implementation-in-tensorflow-aac568283399

Appendix A

A.1 Python Packages Used for FL Framework and UI Interface Development

Below are the Python packages used in the code:

- **pandas** - Used for data manipulation and analysis, particularly to work with DataFrame objects.
- **numpy** - Provides support for arrays and numerical computations.
- **matplotlib** - A plotting library for creating interactive visualizations.
- **scikit-learn** - A machine learning library for using OneHotEncoder methods,etc.
- **tensorflow** - A deep learning library used for building neural networks.
- **streamlit** - Used to build web applications for data science and machine learning models.
- **snowpark** - A Snowflake library to activate the session
- **gzip** - provides a command line interface to decompress files.

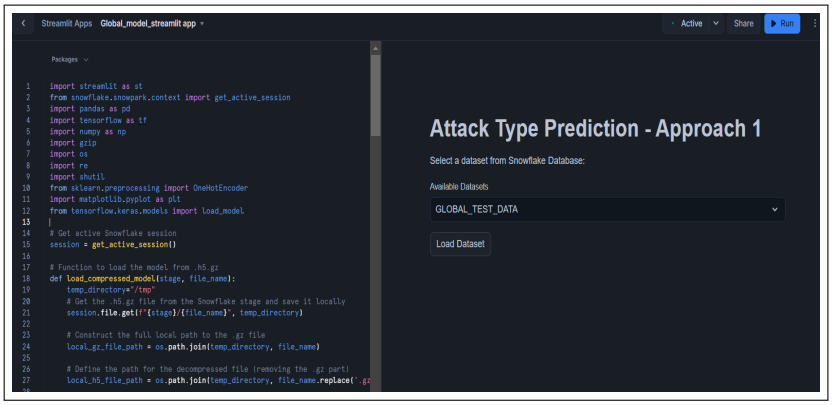
A.2 Code Snippets for Streamlit App

```
# Get active Snowflake session
session = get_active_session()

# Function to load the model from .h5.gz
def load_compressed_model(stage, file_name):
    temp_directory="/tmp"
    # Get the .h5.gz file from the Snowflake stage and save it locally
    session.file.get(f"{stage}/{file_name}", temp_directory)

    # Construct the full local path to the .gz file
```

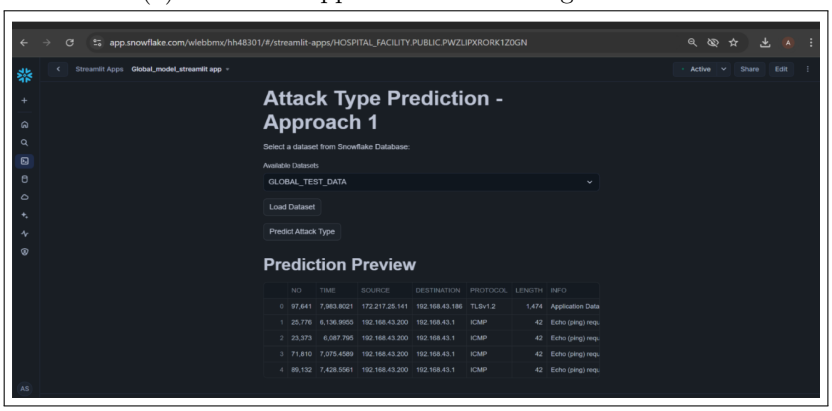
```
    local_gz_file_path = os.path.join(temp_directory, file_name)

    # Define the path for the decompressed file (removing the .gz part)
    local_h5_file_path = os.path.join(temp_directory, file_name.replace('.gz', ''))

    # Decompress the .gz file
    with gzip.open(local_gz_file_path, 'rb') as gz_file:
        with open(local_h5_file_path, 'wb') as h5_file:
            shutil.copyfileobj(gz_file, h5_file)

    # Load the .h5 file as a Keras model
    model = load_model(local_h5_file_path)

    # Optionally, clean up the files after loading into memory
    os.remove(local_gz_file_path)
    os.remove(local_h5_file_path)
    return model


def predict_attacks(df):

    # Preprocess the data
    df_x = preprocess_x(df)

    # # Ensure all features are numeric and have no missing values
    # df_x = df_x.apply(pd.to_numeric, errors='coerce')
    # df_x = df_x.fillna(0) # Example: fill NaNs with 0s

    # Convert the DataFrame to a NumPy array
    df_values = df_x.values.astype(np.float32) # Ensure the data type is float32 for
        TensorFlow

    # Define the stage and file name for the single file
    stage = "GLOBAL_MODEL_STAGE/v4"
    file_name = "global_model_v4.h5.gz"

    # Load the model
    model = load_compressed_model(stage,file_name)

    # Make predictions
    predictions = model.predict(df_x)
    y_pred = np.argmax(predictions, axis=1)
    predicted_labels = [class_labels[i] for i in y_pred]

    # Add predictions to the DataFrame
    df['Predicted Attack Type'] = predicted_labels

    return df

# Streamlit App
st.title("Attack Type Prediction - Approach 1")
```

```
# Fetch available datasets from Snowflake
st.write("Select a dataset from Snowflake Database:")

dataset_options = ['GLOBAL_TEST_DATA', 'STREAMLIT_APP_NEAR_REAL_TIME_TEST_HOLD_OUT']
selected_dataset = st.selectbox("Available Datasets", dataset_options)
if 'data_loaded' not in st.session_state:
    st.session_state['data_loaded'] = False

if st.button("Load Dataset"):
    with st.spinner("Loading data from Snowflake..."):
        # Replace with actual query to load the dataset from Snowflake
        query = f"SELECT * FROM {selected_dataset}" # query to fetch data from the
            Snowflake table
        df = session.sql(query).to_pandas()
        st.session_state['data'] = df
        st.session_state['data_loaded'] = True
        st.write("## Input Data Preview")
        st.dataframe(df.head(5))

if st.session_state['data_loaded']:
    if st.button("Predict Attack Type"):
        with st.spinner("Predicting..."):
            st.session_state['result_df'] = predict_attacks(st.session_state['data'])

        if 'result_df' in st.session_state:
            result_df=st.session_state['result_df']
            st.write("## Prediction Preview")
            st.dataframe(result_df.head(5))

            # Visualization of attack types as a pie chart
            st.write("## Attack Type Distribution")
            attack_type_counts = result_df['Predicted Attack Type'].value_counts()
            attack_type_labels = attack_type_counts.index
            attack_type_sizes = attack_type_counts.values

            # Customize colors
            colors = plt.cm.Paired(np.linspace(0, 1, len(attack_type_labels)))

            # Create the pie chart with enhanced visuals
            fig, ax = plt.subplots()
            ax.pie(attack_type_sizes, labels=attack_type_labels, autopct='%1.1f%%',
                startangle=45,
                  colors=colors, shadow=True, textprops={'fontsize': 10})

            # Draw circle to make it a donut chart (optional)
            centre_circle = plt.Circle((0, 0), 0, fc='white')
            fig.gca().add_artist(centre_circle)

            # Equal aspect ratio ensures that pie is drawn as a circle.
            ax.axis('equal')

            st.pyplot(fig) # Display the pie chart in Streamlit
```

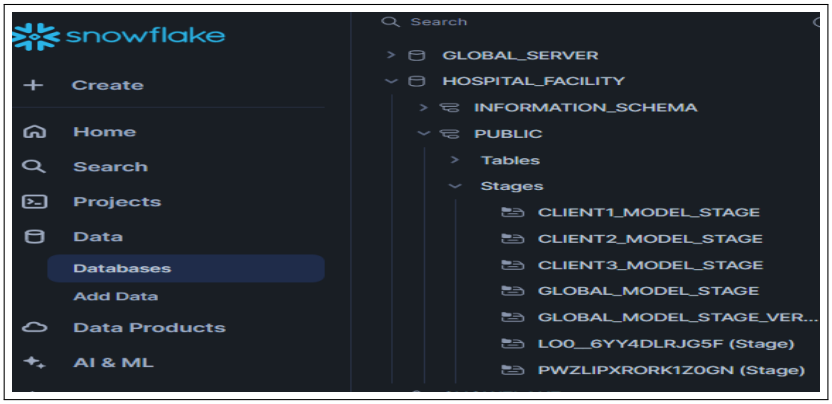
```
        st.write("## Download Predictions")
        output_csv = result_df.to_csv(index=False).encode('utf-8')
        st.download_button(
            label="Download CSV",
            data=output_csv,
            file_name='predictions.csv',
            mime='text/csv'
        )
```



(a) Streamlit App UI Before Loading Dataset



(b) Streamlit App UI after clicking Prediction



(c) Files at Internal Stage in Snowflake

Figure A.1: Snowflake Streamlit Application

Appendix B

B.1 Confusion Matrix Analysis

The confusion matrices offer a visual breakdown of how well the models performed in identifying different types of network traffic, including both normal and attack scenarios.

Each matrix, shown in Figure B.1, B.2 and B.3, displays the number of correct and incorrect predictions made by the models, highlighting true positives (correctly identified attacks), false positives (normal traffic misclassified as attacks), true negatives (correctly identified normal traffic), and false negatives (missed attacks). The comparison across model versions shows how adjustments to the models helped reduce errors and improve accuracy.

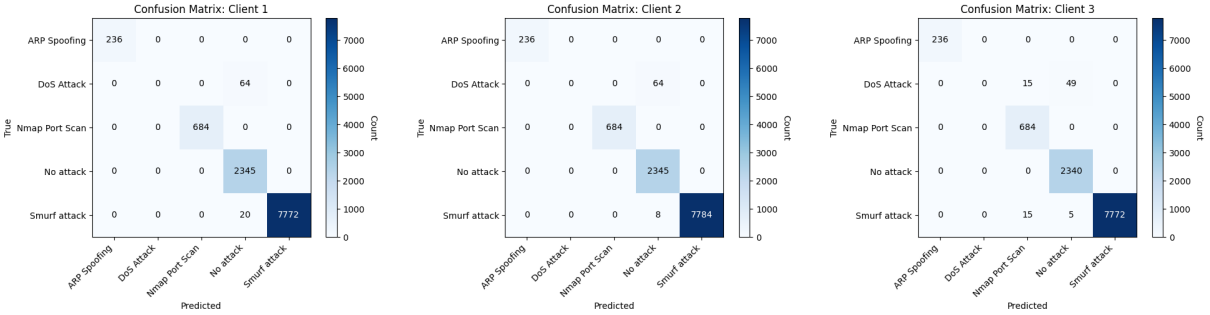


Figure B.1: Confusion Matrix - Client local Model Before Fine-Tuning

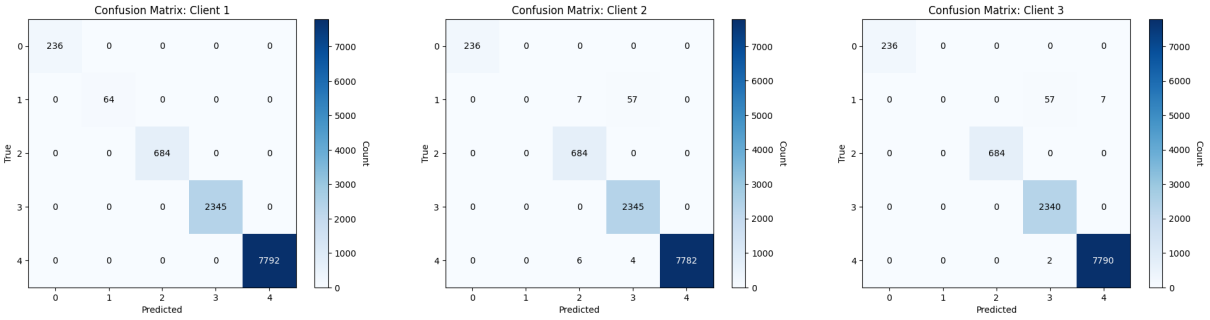


Figure B.2: Confusion Matrix - Client local Model After Fine-Tuning

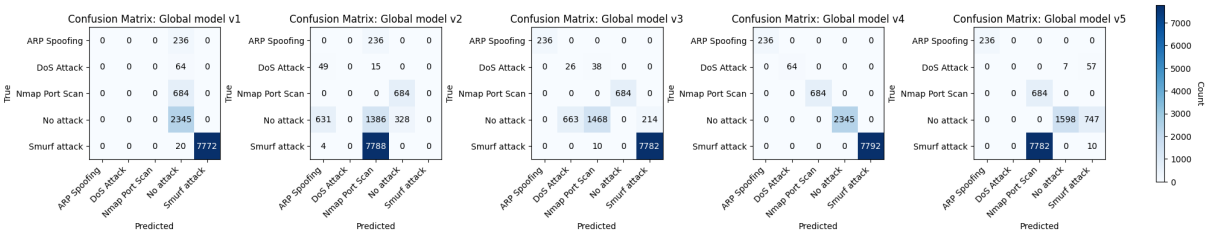


Figure B.3: Confusion Matrix - Global Model all versions