FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING
DEGREE PROGRAMME IN WIRELESS COMMUNICATIONS ENGINEERING

MASTER'S THESIS

Continual Federated Learning For Network Anomaly Detection in 5G Open-RAN

Author	Fahim Muhtasim Hossain
Supervisor	Professor Tarik Taleb
Advisor	Dr. Chafika Benzaid

June 2023

Fahim Muhtasim Hossain. (2023) Federated Continual Learning For Network Anomaly Detection in 5G. Faculty of Information Technology and Electrical Engineering, Degree Programme in Wireless Communications Engineering

ABSTRACT

This dissertation offers a unique federated continual learning setup for anomaly detection in the fast growing 5G Open Radio Access Network (O-RAN) environment. Conventional AI techniques frequently fall short of meeting the security automation needs of 5G networks, owing to their outstanding latency, dependability, and bandwidth demands. As a result, the thesis provides an anomaly detection system that does not only use federated learning (FL) to solve inherent privacy problems and resource constraints but also incorporates replay buffer concept in the training phase of the model to eradicate catastrophic forgetting. To allow the intended federated learning architecture, anomaly detectors are incorporated into the Near-real time RIC, while aggregation servers are installed within the Non-real time RIC. The configuration was carefully tested using the 5G NIDD Dataset, revealing a considerable boost in detection accuracy by reaching close to 99% for almost all datasets after including the continual learning process. The thesis also investigates the notion of transfer learning, in which pre-trained local models are evaluated against a hybrid Application layer DDoS dataset that includes benign samples from the CICIDS 2017 dataset and attack flows generated in proprietary SDN environment. The captured results show almost over 99% of accuracy, confirming the suggested system's efficacy and flexibility. The study represents a significant step forward in the development of a more secure, efficient, and privacy-protecting 5G network architecture.

Keywords: 5G, Network automation, Security, Federated learning, Continual learning, Catastrophic forgetting, and DDoS

CONTENTS

PREFACE

This master's thesis, produced for the Master's Program in Wireless Communications Engineering at the Mobile Network Softwarization and Service Customization (MOSAIC) research group, Centre for Wireless Communications, University of Oulu, proposes a federated continual learning-based anomaly detector for 5G O-RAN. It was created as part of the Master of Science in Wireless Communications Engineering program. This research work is partially supported by the European Union's Horizon 2020 Research and Innovation Program through the 6GSandbox project under Grant No. 101096328 and the Business Finland 6Bridge 6Core project under Grant No. 8410/31/2022. This dissertation goes into extensive detail on the O-RAN architecture, federated learning, continual learning, and anomaly detection. I would like to convey my profound appreciation to Dr. Chafika Benzaid for her invaluable support, direction, and insightful recommendations during the thesis process. Furthermore, I would like to thank Prof. Tarik Taleb for his collaboration and advice during the duration of my master's thesis. I would also like to thank my parents for their support.

Oulu, June, 2023
Fahim Muhtasim Hossain

LIST OF SYMBOLS AND ABBREVIATIONS

5G	Fifth Generation
AI	Artificial Intelligence
NR	New Radio
IoT	Internet of Things
DDoS	Distributed Denial of Service
ML	Machine Learning
DL	Deep Learning
CF	Catastrophic Forgetting
CL	Continual Learning
FL	Federated Learning
O-RAN	Open Radio Access Network
RAN	Radio Access Network
O-C U-CP	O-RAN Central Unit Control Plane
O-CU-UP	O-RAN Central Unit User Plane
O-D U	O-RAN Distribution Unit
O-RU	O-RAN Radio Unit
SMO	Service Management and Orchestration
O-eNB	O-RAN evolved Nobe B
NIDD	Network Intrusion Detection Datatet
5GTN	5G test bed
RIC	RAN Intelligent Controller
NN	Neural Networks
ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
MLP	Multi Layer Perceptron
SVM	Support Vector Machines
XSS	Cross site scripting
MITM	Man In The Middle Attack
SQL	Structured Query Language
NMAP	Network Mapper
NIDS	Network Intrusion Detection Systems
EWC	Elastic Weight Consolidation
GEM	Gradient Episodic Memory
HTTPS	Hyper Text Transfer Protocol Secure
SSH	Secure Shell
SFTP	Simple File Transfer Protocol
ICMP	Internet Control Message Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
ReLU	Rectified Linear Unit
BCE	Binary Crossentropy
VM	Virtual Machine
FedAvg	Federated Averaging
FedSGD	Federated Stochastic Gradient Descent
FedAM	Federated Averaging Momentum
REST	Representational State Transfer
API	Application Programming Interface
KPI	Key Performance Indicator

DNN	Deep Neural Network
PNN	Progressive Neural Network
DEN	Dynamically Expandable Network
SON	Self-Organizing Networks
MLB	Mobility Load Balancing
CFeD	Continual Federated Learning with Distillation
EWC	Elastic Weight Consolidation
LwF	Learning Without Forgetting
VNE	Virtual Network Embedding
RL	Reinforcement Learning
SIRL	Slice Isolation-based Reinforcement Learning
LSTM	long short-term memory
ZSM	zero-touch network and service management
3GPP	3rd Generation Partnership Project
NFV	Network Functions Virtualization

1 INTRODUCTION

The most recent development in wireless communication technology is the fifth-generation (5G) network, which promises revolutionary reductions in latency, connection density, and data speeds. These developments will spur innovation in many other fields, including autonomous driving, smart cities, and the Internet of Things (IoT). The advantages of 5G are, however, offset by an increased attack surface and vulnerability to cyber-threats, mostly because of the network's dispersed architecture. To protect these next-generation networks, it is necessary to have strong and effective security procedures.

System for detecting network anomalies is crucial to protecting emerging 5G networks. These systems work by encountering variations from conventional network traffic patterns, which gives them a reliable method for spotting cutting-edge cyber attacks that are both innovative and complex. The capacity to identify Distributed Denial of Service (DDoS) assaults, one of the most pervasive and crippling dangers to network security in the connected world of today, is especially crucial [1]. DDoS attacks have the capacity to quickly overload a network, leading to substantial service disruptions and possible data loss. The rapid innovation and increasing complexity of DDoS attack techniques, however, make it difficult to sustain the effectiveness of traditional attack detection systems. In fact, traditional rule-based detection systems struggle to keep up with the complexity of such threats as attack vectors become more varied. This has led to the use of machine learning (ML) and deep learning (DL) approaches to improve the generalization and prediction accuracy of these systems. They help in spotting tiny changes or trends in network traffic that might point to a DDoS assault, even in its early stages.

ML and DL-based anomaly detection systems nevertheless face a number of difficulties in spite of the aforementioned advantages. Catastrophic forgetting (CF) [2], a situation where the learning model tends to forget or ignore previously acquired patterns when exposed to fresh data or attack patterns, is an important problem. Given that attackers constantly innovate and new attack variations routinely appear, this problem is especially harmful in the context of DDoS detection. Therefore, a CF issue solution is essential to maintaining resilience in identifying different forms of DDoS attacks.

A viable remedy to lessen the CF issue has been found as continual learning (CL) [3]. Models can adapt to new tasks while preserving information from earlier ones because to CL, which enables continuous learning from a stream of data. Although CL has demonstrated success in a number of fields, including computer vision, its use in network security, especially in the creation of cyber attack detector, is still unexplored. A number of approaches have been developed in CL to combat CF, with the usage of replay buffers being one of the most promising [4]. Replay buffers preserve a balance between old and new information by storing a portion of the historical data that the model may ”replay” during training. In numerous contexts, this method has proven very effective in preventing catastrophic forgetfulness.

However to provide effective, immediate reactions to any incursions, decentralized learning mechanisms must be used due to the distributed design of 5G networks. Federated Learning (FL) stands out as a possible approach in this regard. Distributed devices, as those in a 5G network, can develop a shared prediction model cooperatively using FL while keeping all of the training data on their original devices. This decentralized strategy greatly improves data privacy, a crucial issue in modern cybersecurity measures, while also optimizing learning processes. For network intrusion detection, the fusion of FL with CL [5] offers great potential. A powerful approach for improving the performance of intrusion detection in 5G networks may be developed by combining CL's capability to handle continually developing tasks, which addresses the problem of CF, with FL's distributed and privacy-preserving nature. This combination provides a thorough learning framework that not only successfully identifies new threats but also preserves the privacy of local data, thereby addressing the key issues with 5G security. This might open

the door for the creation of sophisticated DDoS assault defense systems and the development of enhanced network anomaly detection systems that protect user data and privacy in the vast 5G network environment.

1.1 Background and Motivation

A new age of significant decreases in latency, connection density, and data rates has been ushered in by the introduction of 5G wireless communication technology. A wide range of industries, including autonomous vehicles, smart cities, and IoT, might benefit from this technical advancement. The benefits of the 5G network, principally because of its decentralized design, also open the door for an extended attack surface and greater susceptibility to cyberthreats. It highlights the urgent requirement for strong and efficient security solutions to protect these future networks.

DDoS assaults, which may quickly overwhelm a network and cause substantial service disruption and possibly data loss, are one of the most pervasive and disruptive dangers in today's linked world. Traditional rule-based detection systems face major difficulties due to the DDoS attack tactics' fast growth and increasing complexity. There is a pressing need to improve these systems' resilience and forecast accuracy as attack vectors become more varied. The use of ML and DL technologies has increased as a result, making it easier to spot minute changes or patterns in network traffic that might be signs of a DDoS assault.

However, because of always changing nature of DDoS attack variants, problems with catastrophic forgetting (CF), which occurs when a model forgets prior patterns when exposed to fresh data, plague ML and DL systems. To address CF, strategies like CL have been developed. Although effective in fields like computer vision, CL's use in network security has not yet been fully investigated. Integrating FL, a decentralized approach that protects data privacy, with CL might significantly improve network intrusion detection. This combination may enhance network anomaly detection and DDoS protection systems while preserving data privacy in 5G networks.

CL has shown promise in preventing catastrophic forgetting, especially when replay buffers are used to maintain a balance between new and old information. Despite CL's shown effectiveness in areas like computer vision, its use in network security, particularly in the creation of a cyberattack detector, is mostly unexplored. Network intrusion detection may greatly benefit from the combination of CL and FL, which offers a decentralized method for collaboratively creating a shared prediction model while protecting data privacy. A complete learning framework that can successfully identify new risks while protecting the privacy of local data might be created by integrating the skills of CL and FL.

1.2 Research Problem

The security and integrity of network systems are crucially maintained by their ability to identify anomalies. This is especially true for 5G's Open Radio Access Networks (O-RAN), which are expansive heterogeneous systems with a range of latency and privacy needs. Internal problems and malicious assaults are only two examples of the many things that might cause anomalies in a RAN. Only a few ML-based research papers have particularly focused on O-RAN, despite the fact that there are many studies on anomaly detection in the RAN that employ ML. This is a significant lacuna in the body of knowledge, especially in light of the particular difficulties and dangers that O-RAN in 5G networks could provide.

When using AI/ML for operations, protecting data privacy is essential. By protecting data privacy and improving communication effectiveness, FL provides potential answers. Models are trained locally, and just the parameters required for aggregation are shared. FL is especially helpful for handling sensitive data in intricate, extensive environments like RAN. However, further study is required on the use of FL for RAN anomaly detection to stop assaults from reaching the core network. FL implementation at the RAN level might greatly increase communication effectiveness and data privacy protection. As a result of abnormalities found using FL, it could also make it easier to carry out appropriate control activities, including resource management or the transfer of User Equipment (UE).

Additionally, in an FL system, local detection models have a propensity to forget previously acquired information over time. Given how quickly cyber dangers are growing, this is an urgent problem to tackle. Even while federated learning has been mentioned in many research projects, very few have looked at how it may be combined with continuous learning. This is an important topic of study that needs to be addressed right away.

Determining a comprehensive, FL-based anomaly detection framework for O-RAN in 5G networks that not only protects data privacy and improves communication efficiency but also handles the problem of learnt knowledge gradually ”forgetting” over time is the main research challenge as a result. The proposed study will look into how FL may be combined with ongoing learning to offer a reliable and efficient method for O-RAN anomaly detection. This research problem’s core will also include the investigation of appropriate control measures in response to observed abnormalities.

1.3 Selected scope

In this research, the use of continual federated learning has been explored for the detection of various forms of DDoS attacks in 5G networks with a focus on replay buffers. The main goal is to create a powerful DDoS attack detection system that can skillfully handle the particular difficulties of identifying various DDoS kinds that a 5G network may experience. Hypothesis has been put forward that continual federated learning strategies, specifically the addition of replay buffers, may substantially decrease the problem of catastrophic forgetting in DDoS detection models while identifying DDoS attack types striking different nodes of distributed 5G network that are not encountered by other hops in the network.

1.4 Methodology

As the proposed anomaly detector is based on an FL model, the methodology of this thesis is primarily comprised of steps required to generate a federated learning setup. It consists of following steps: data preprocessing, model training, communication between clients and aggregation server, federated averaging and model testing. 5G NIDD dataset [6] which has been populated with both real world attack and genuine traffic flows generated in the 5GTN test bed built in University of Oulu has been chosen to utilize. The dataset covers a variety of DDoS attack types mixed with real world legitimate network traffic. As the local models work with only numerical values, therefore, various data processing steps have been conducted to organize the dataset and making the data samples eligible to feed to the models. At the end of the data preparation segment, normalization took place which sets the feature values of all data samples within a predetermined range of values. Anomaly types are only one example of superfluous values that might exist. It is sufficient to establish whether or not an abnormality exists, hence

this data column was eliminated. There are 10 datasets categorized by different types of DDoS attacks amalgamated with benign traffic for training and evaluation purposes.

Two local models, one for each client were developed for each training data set during the training step. The FL-based training model's cycle begins with the model being trained in each client for 10 rounds with each dataset, which are subsequently transmitted back to the main server. Before being transmitted back to the end devices for training, the model parameters are combined at the central server. After updating the local models in all clients with aggregated weights, another dataset comes into place for training process. In the thesis, this is seen as one complete cycle. The models were trained in this manner for 3 rounds. The updated model after conducting mentioned rounds of training is then used to test the model's performance with test datasets.

Two scenarios which are traditional FL and FL integrated with continual learning has been adopted to conduct training and evaluation. At the first stage, only standalone FL setup has been run and CF has been spotted while feeding new datasets one by one in the system. At the later stage, reservoir sampling buffer [7] has been added to preserve the portion of representative samples from previous datasets which is then mixed up with current datasets before feeding to the local models for training. After completing training and federated aggregation process for all datasets, evaluation has been done which is same as with traditional federated learning to observe the difference in performance metrics values.

1.5 Contribution

This study's major contribution is a FL based network anomaly detector integrated with CL to eradicate CF from local model for the sake of achieving sustainability in efficiently detecting continually evolving DDoS attacks over the time. O-RAN is relatively new, and there are only a few applications available. Utilizing ML or FL for security in O-RAN is currently in its early phases of research. Anomaly detection has been studied for the networked systems such as the IoT, and Zero-touch Network and Service Management systems mostly by using centralized single node machine learning models but leveraging FL based detection has not been studied much in O-RAN. Moreover, addressing the critical problem of forgetting old knowledge of attack patterns with time has also not been highlighted much in federated setup-based detection techniques. Therefore, this study provides an initial step for research into using privacy along with knowledge preserving FL for anomaly detection in O-RAN.

Due to the hierarchical positions of Near real-time RIC (Near-RT RIC) and Non real-time RIC (Non-RT RIC) and their various closed loops, FL-based solutions may be deployed in O-RAN with ease. As a result, the continual learning based federated setup could be easily implemented in an O-RAN architecture as a service in Near-RT RIC segment. Additionally, a reference design for deployment has been proposed for more research.

Performance was evaluated based on accuracy, precision, recall and the F1 score. Furthermore, the cost of the implementation of the system was considered in terms of training and evaluation time and size of the space buffer takes to preserve representative samples. After 10 rounds of training with each datasets and 3 rounds of federated averaging, it was possible to find anomalies in the test dataset with all the mentioned metrics value of above around. Therefore, it can be concluded that the proposed models are effective against network attacks specifically DoS attack in the O-RAN architecture.

1.6 Organization of the Thesis

The thesis is structured as follows.

Chapter 2 describes general details, objectives and application of continual FL.

Chapter 3 provides a comprehensive review of related works and corresponding limitations in the fields of anomaly and intrusion detection in computer networks.

Chapter 4 provides comprehensive details about the dataset and traffic patterns that have been adopted for this thesis along with the proposed network anomaly detection framework and implementation of the system.

In chapter 5, granular and elaborate analysis of the performance of the framework has been conducted with substantial graphs and tables. The performance has been measured in terms of accuracy, precision, recall and f1 score both in traditional FL and integrated continual FL setup scenarios. Last part of the chapter compares both scenarios and shows the difference in terms of resource utilization cost and performance.

Chapter 6 summarizes the comparative analysis of thesis project with similar research papers, how much the objectives of the research has been achieved along with future directions.

Chapter 7 presents an overall summary of the thesis.

2 Background and Preliminaries

This chapter explores the evolution of diverse environment of cyber threats in 5G networks. Furthermore, investigation of the structure of O-RAN architecture, different types of 5G anomalies emphasizing DDoS attack vectors, and highlighting the transformational potential of FL in safeguarding 5G networks has been conducted. Lastly, continual FL, a unique strategy for mitigating future threats has been discussed elaborately.

2.1 Cyber threats in 5G and Beyond

5G and beyond networks promise to transform the digital world by offering game-changing improvements including high data speeds, low latency, and huge interconnectedness. However, growing complexity and interconnectedness broaden the threat landscape and pose new security issues.

2.1.1 Evolution of Cyber Threats

Cyber risks have progressed from isolated attacks on computers to sophisticated, distributed, and ubiquitous threats affecting all aspects of the digital realm. The growth of connected devices in the 5G and beyond age, along with an increased dependence on machine learning and artificial intelligence, broadens the attack surface and makes networks more vulnerable to cyber assaults [8]. In what follows, we provide an overview of some existing cyber threats against 5G and beyond networks.

2.1.2 Security Threats against 5G RAN

- **Latency anomalies:** One of the key performance indicators in 5G RAN network is latency. There can be considerable delay in delivering services or in any particular interface of 5G network. Latency in 5G network can be defined or categorized by two ways. First one is by measuring the time to deliver a service and the second one is time taken to send or receive one packet. This latency issue can occur due to several factors such as software faults, application overload or conflicts, scheduling problems, difficulties with network routing, overloaded servers, high memory usage or hardware failure. By collecting and analyzing performance metrics or system logs through ML algorithms, granular view and understanding of the complex network like 5G New Radio (NR) can be possible as well as detection of latency and root cause can be identified. In one research paper [9], authors collected the system logs from the base station containing all call records and information. Then the log stream has been filtered and split into streams of individual user equipment. Then those split data have been fed to ML algorithm to know deeply the reason behind subtle latency in the network.
- **Jamming attacks:** [10] Jammers placed on purpose by adversaries constitute a substantial danger in 5G networks. They disrupt wireless networks in a variety of ways. Regular jammers, which need a lot of power, continually block legitimate signals without monitoring user behavior. Delusive jammers fool by sending legitimate bit sequences, making them more difficult to detect. Random jammers save energy by alternating between active and idle modes. Responsive jammers monitor the communication channel and transmit

only when necessary, using less power. Go-next jammers are selective, focusing on one frequency channel at a time, and if a jammer is detected, it will follow the transmitter to the next frequency. Control channel jammers interfere with information exchange between transmitters and receivers, potentially resulting in service denial or network access denial.

2.1.3 DDoS attack in 5G

The transition to 5G networks opens up previously unimagined possibilities for improving connection speeds, latency, and overall connection. This transformation, however, offers major issues in terms of network security. The Distributed Denial of Service (DDoS) assault, which is a significant danger in the 5G context, is one of the most common and disruptive types of cyberattack. DDoS assaults in 5G networks can have disastrous consequences, such as considerable downtime, loss of crucial data, and damaged services, all of which can result in major financial losses and reputational harm. Because of the highly scattered and dynamic nature of 5G networks, as well as the vast number of connected devices and increasing data rates, DDoS assaults have a larger attack surface.This makes detecting and mitigating such risks difficult. Added to that, because of developments in attack plans and methodologies, DDoS attacks on 5G networks may be more complicated and sophisticated than their predecessors. As a result, building DDoS defensive methods that are resilient, scalable, and economical is a critical necessity for safeguarding 5G networks. Various form of DDoS attacks in mobile network has been described below.

Existing DDoS attacks can be classified into [11]:

- **Volume based attacks:** This sort of DDoS attack seeks to flood a network's bandwidth with massive amounts of data. UDP (User Datagram Protocol) floods, ICMP (Internet Control Message Protocol) floods, and other spoofed-packet floods are examples. The primary purpose is to overwhelm the target site's bandwidth, bringing its services to a standstill. These assaults are often quantified in bits per second (Bps).

- **Protocol based attacks:** These attacks take use of flaws in a server's resources or in intermediary communication devices such as firewalls and load balancers. SYN (Synchronize) floods, fragmented packet assaults, the Ping of Death, Smurf DDoS, and other attacks are examples. The attack operates by consuming server resources until the server is unable to perform the targeted services. The frequency of these assaults is expressed in packets per second (Pps).

- **Application layer attacks:** These are some of the most devious forms of DDoS assaults since they use requests that appear to be genuine and benign. The purpose is to crash the web server, which is frequently accomplished by exploiting vulnerabilities in popular systems such as Apache, Windows, or OpenBSD. Low-and-slow assaults and GET/POST floods, in which attackers overwhelm a server with HTTP requests, are examples of this kind. These assaults are often quantified in requests per second (Rps).

Each form of DDoS assault poses a substantial risk to network security and necessitates various defenses. Understanding these threats and establishing powerful defensive measures against them will be crucial as we go into the 5G era and beyond. Given the sophistication of these assaults and the sheer volume of network data that must be monitored, traditional detection and mitigation approaches may no longer be enough. This is where Artificial Intelligence (AI) can come in handy.

With their capacity to learn, adapt, and reply in real-time, AI-based solutions are particularly well-suited to solving the issues posed by DDoS assaults. AI can analyze trends, detect abnormalities, and identify possible risks faster and more accurately than traditional approaches. Furthermore, AI can respond to threats independently, allowing for faster response times and decreasing the window in which assaults might do damage.

Deploying these AI technologies at the network edge provides further benefits. Edge computing moves processing and data storage closer to the point of use, improving reaction times and reducing bandwidth use. In terms of DDoS mitigation, this implies being able to respond to an assault almost immediately and as near to the source as feasible. On top of that, a coordinated approach to DDoS detection can improve the efficacy of these tactics. Networks can work together more effectively to identify and shut down DDoS assaults by exchanging threat information and responses in real-time. A cooperative paradigm based on FL, in which ML models are taught across numerous decentralized edge devices, offering a global perspective of the threat landscape while protecting data privacy, might be used.

2.1.4 Cyber Threat Prevention Leveraging Federated Learning and O-RAN

As we have discussed in previous section that AI-based solutions, edge computing along with collaborative approach represent a promising future in our DDoS defenses in the 5G era and beyond. These novel approaches reflect the future of cybersecurity, in which proactive and intelligent systems collaborate to ensure the integrity and dependability of our networks.

To combat the growing threat of cyber attacks in 5G and beyond, security paradigms must evolve. Integrating FL based detection mechanism with O-RAN is one interesting technique. [12] FL enables decentralized machine learning, inetwork edge (i.e., on the devices themselves), improving privacy and avoiding some of the hazards associated with data centralization. FL integration n which models are trained at the with O-RAN near real-time RAN Intelligent Controller (RIC) services may enable effective threat detection and response mechanisms. It might help identify and mitigate DDoS assaults, for example, by learning from network traffic patterns and making localized judgments at the network edge.

Moreover, Open-RAN encourages interoperability and reduces vendor lock-in, potentially lowering the danger of supply chain assaults. It enables operators to select various suppliers for different RAN components, enhancing resistance against specific vulnerabilities. The combination of these technologies offers a viable path ahead in terms of securing 5G and beyond networks. However, it is a challenging endeavor that will need continual study and collaboration from academia, business, and government.

2.2 Open RAN Architecture

The 5G Open O-RAN represents a fundamental shift in the design of cellular networks. The RAN, which comprises base stations and associated digital signal processors, has traditionally been the component of a mobile telecommunications system that links individual devices to other sections of the network via radio connections. The hardware and software in these classic designs are tightly integrated and frequently come from the same manufacturer.

Open RAN changes this by disconnecting the RAN's hardware and software components. This makes the RAN ecosystem more adaptable, interoperable, and competitive. Networks may be designed with software and hardware from a variety of suppliers, enabling for more innovation, faster deployment, and potentially reduced prices. Furthermore, Open RAN is

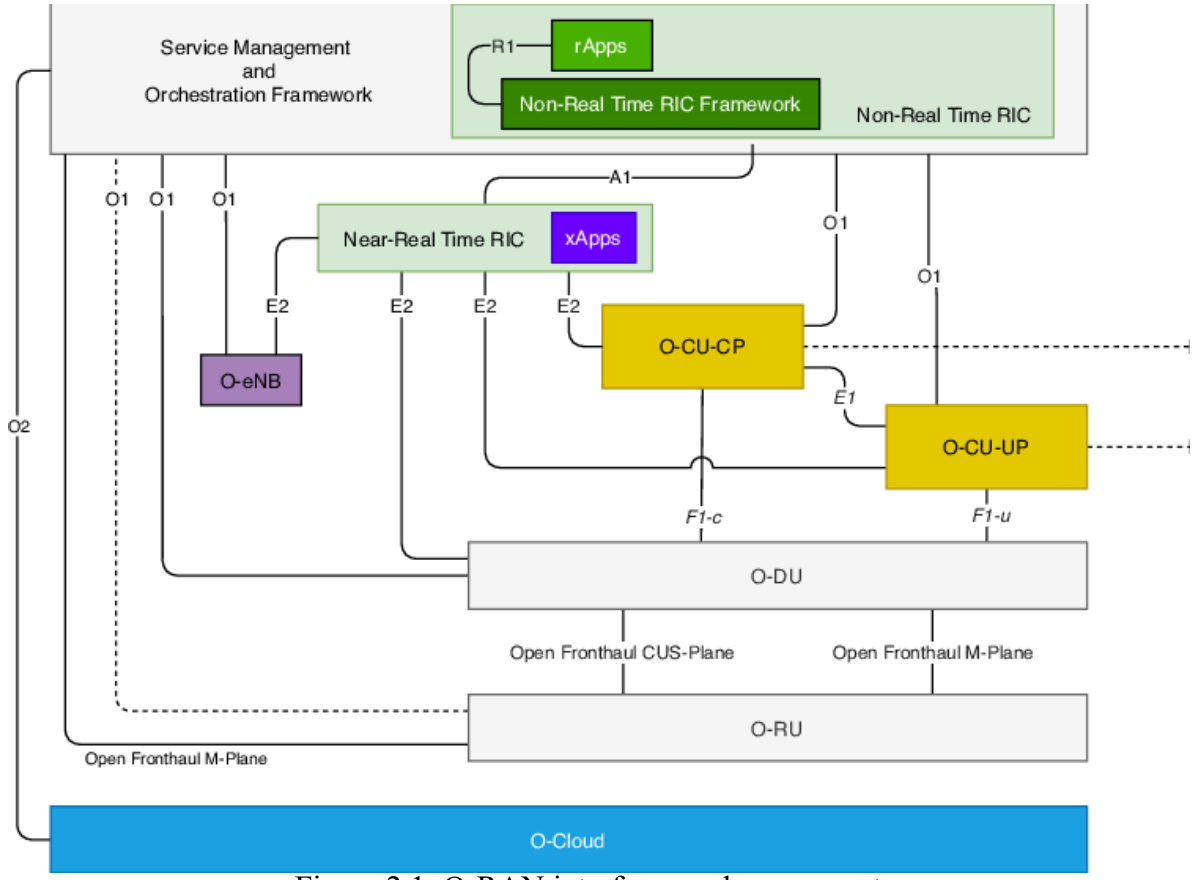


Figure 2.1: O-RAN interfaces and components.
[13]

concerned with the disaggregation of the Baseband Unit (BBU) into discrete pieces that may be scaled independently. The BBU is made up of three major parts: the Central Unit (CU), the Distributed Unit (DU), and the Radio Unit (RU).

RIC is a critical component of Open RAN in the context of 5G. The RIC is further subdivided into two essential components: non-real-time (Non-RT RIC) and near-real-time (Near-RT RIC).The Non-RT RIC focuses on non-time-sensitive policy, optimization, and lifecycle management functions. It is in charge of higher-level network-wide optimization and hosts the service management platform as well as non-real-time apps.The Near-RT RIC, on the other hand, is concerned with time-sensitive operations, including features such as real-time radio resource management, admission control, and scheduling. It allows near-real-time applications and houses RAN functions that demand a faster reaction time.

5G Open RAN is a vital part of the 5G network revolution because it provides advanced features such as network slicing, which allows various virtual networks to be established on the same infrastructure to serve different types of services or applications.

2.2.1 O-RAN Interfaces

In the O-RAN architecture, there are a number of open interfaces with various network responsibilities that enable more interoperability and increased functionality [13] [14]. Figure 2.1 depicts the entire connectivity of different interfaces with corresponding building blocks of 5G O-RAN architecture.

- **O1 Interface:** The framework for service management and orchestration can leverage network capabilities through this interface. It connects managed elements such as Open Central Unit Control Plane (O-CU-CP), Open Central Unit User Plane (O-CU-UP), O-RAN evolved Nobe B (O-eNB), O-RAN Distribution Unit (O-DU), O-RAN Radio Unit (O-RU) to Service Mangement and Orchestration (SMO) framework. This interface provides FCAPS services which are respectively fault, configuration, accounting, performance and security management.

- **O2 Interface:** It refers to the interface between O-cloud and SMO framework for providing workload management and resource allocation. The functions this interface includes are learning about O-cloud infrastructure, administration, scale in/out, platform software management. Besides, through this interface, O-cloud resource management like creation, deletion, and assigning O-cloud infrastructure are also performed.

- **R1 Interface:** Through this interface, RAN applications (rApps) communicates to leverage non-real time RIC functions. R1 interface is also used for ML models training as well as data analysis purposes.

- **A1 Interface:** It is the interface that establishes connection between Non-RT RIC functions in SMO framework and Near-RT RIC functions. A1 interface provides policies and ML model management as well as information enrichment services.

- **E2 Interface:** This interface connects near real time RIC to the nodes such as O-CU, O-DU and O eNB. The protocols here in this interface are basically control plane protocols. The functions this interface provide are near real time RIC services, control the policies in E2 nodes, E2 nodes configuration updates, E2 set up and reset and report general error situations.

- **Open Front-haul M-Plane:** The components of the O-RU are handled using the Open Front-haul M-Plane interface. It is used to initialize and configure operational settings or for performance reporting. The risk factor here is that, software up-gradation in the components occur through this interface.

- **Open Front-haul CUS -Plane:** This interface has three parts which are control, user and synchronization plane. This interface is responsible for time synchronization between O-DU and O-RU units. Apart from that, downlink and uplink IQ data along with scheduling and beam-forming commands are transmitted through this interface.

2.2.2 Security Opportunities of O-RAN

O-RAN architecture, a critical component of the 5G ecosystem, offers various options to improve network security. O-RAN can provide the flexibility required to handle the security concerns of 5G and beyond by shifting away from a proprietary and monolithic model of network function provision and toward a more open and disaggregated approach. The O-RAN design not only transforms 5G infrastructure, but also includes new features such as the RIC, as well as rApps and xApps. Together, these components add to the network's substantial security potential.

RIC is a revolutionary RAN design feature that incorporates AI/ML capabilities into the RAN, enabling intelligent control and optimization of network resources. Software applications that operate on the RIC platform are known as rApps and xApps [15]. rApps are responsible for

radio resource management and radio layer control. xApps on the other hand, provide higher-layer network management and control functions, giving a wide range of services ranging from traffic management to advanced security features.

Security rApps and xApps have the ability to significantly improve network security. These systems may use machine learning and AI to analyze network traffic in real time, detect irregularities, and rapidly mitigate possible risks such as DDoS assaults. A security-oriented xApp, for example, may monitor data flow to discover patterns associated with known risks and then take quick automatic action to neutralize these threats, considerably decreasing possible harm. Furthermore, these applications can employ FL models for joint threat detection and mitigation across various network nodes while protecting individual user data privacy. It is conceivable to implement near real-time, AI-driven security services directly within the RAN by integrating these models within rApps and xApps.

in a nutshell, rApps and xApps provide an incredible opportunity to strengthen the security protections in the O-RAN architecture. They might serve as the first line of defense against increasingly complex and persistent attacks to 5G networks, making them an essential component of future-proof and resilient network security policies.

2.3 Federated Learning

It is a ML approach that enables models to be trained on decentralized data, which is data that is dispersed over several devices or places as opposed to being centralized in one place. As per shown in Figure 2.2, training regional ML models on regional datasets is how federated learning operates. Then, these nearby data centers regularly swap the model's parameters and create global parameters by applying various federated aggregation techniques. The local models are then update their parameters to global ones served by centralized data centers. FL is especially helpful when the data is confidential or it is not feasible or desired to send the data to a central site for training via the internet.

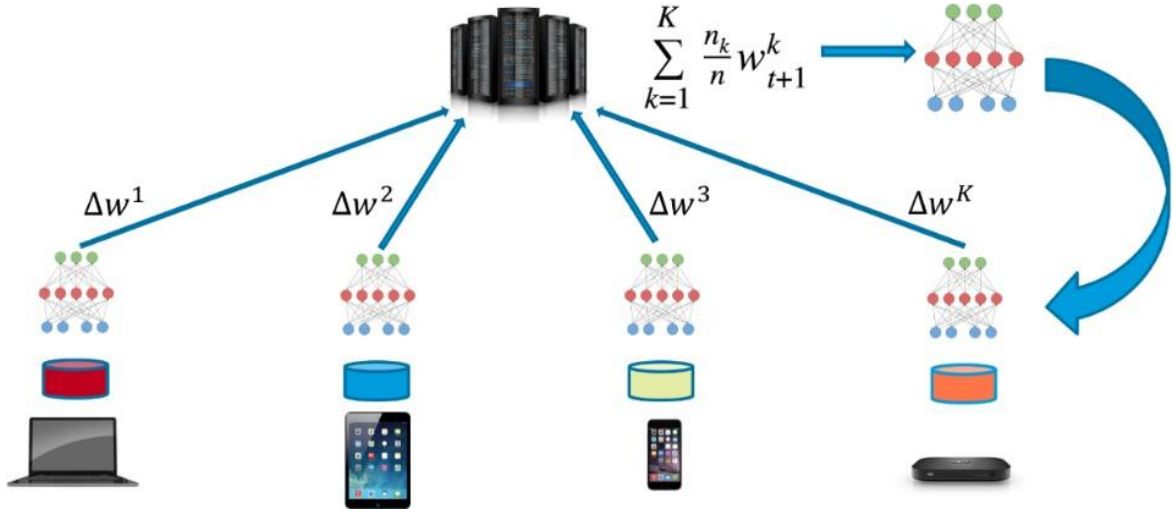


Figure 2.2: Federated Learning Architecture.
[16]

2.3.1 Benefits of Federated Learning

- The ability to train ML models on data that would otherwise be challenging or impossible to utilize for training is one of the key advantages of FL. Consider a healthcare firm that has patient medical record data dispersed across many hospitals and clinics, for instance. For the purpose of training a central ML model, this data may be sensitive and subject to privacy rules, making its transmission via the internet challenging or impossible. Without ever sending the data over the internet, federated learning makes it feasible to train a ML model on this decentralized data.

- The ability to train machine learning models on data that is continuously updated or changing is another advantage of federated learning. Consider a social networking site with information on user interactions and postings as an illustration. As new posts are created and exchanges take place, this information is continuously updated. Without having to continually transfer the data to a central site for training, federated learning makes it feasible to train a machine learning model on this dynamic data.

- In situations when the data is dispersed throughout several organizations or groups that might not wish to share their data with one another, federated learning might be helpful. In these situations, FL enables ML models to be trained on distributed data without the requirement for data exchange. This implies that companies may utilize AI to improve choices without compromising data privacy and running the risk of compromising customer information.

- Using FL, ML models may be trained on data produced by mobile devices, such as behavioral patterns of users, location and sensor data. This may be very beneficial when creating tailored suggestions or anticipating user demands. The ability to train the model on a lot of data without having to send it over the internet or spend a lot of battery capacity is one advantage of utilizing federated learning in this situation.

- ML models may be trained upon data produced by industrial control systems, such as data from sensors or actuators, using federated learning. Using this, control parameter optimization or equipment failure prediction may be possible. The ability to train the model on data from several systems without sending the data over the internet or interfering with the systems' normal functioning is one advantage of utilizing federated learning in this situation.

- ML models may be trained using data produced by financial institutions, such as transaction data or credit ratings, using federated setup. For the purpose of identifying fraud or estimating credit risk, this is beneficial. In this situation, one advantage of adopting FL is that it permits the model to be trained on data from many financial institutions without requiring them to exchange sensitive information.

- FL can be adapted and implemented on student-generated data, such as assessing performance information or learning preferences. This can help with individualized instruction or making adjustments for certain students' requirements. Overall, by enabling the training of models on sensitive and decentralized data, FL has the potential to significantly broaden the scope and application of ML. By enabling the training of models on continuously changing data without the requirement for constant data transfer, it can also increase the effectiveness of machine learning.

2.3.2 Adopting Federated learning over traditional machine learning approach

- **Data privacy and security:** FL makes it possible to train a machine learning model without sending the data over the internet or storing it in a single location, which is enormously beneficial when the data is sensitive or subject to privacy laws. It may not be practical or desirable in certain situations to transport the data to a central site for training as is typical for traditional ML methodologies.
- **Large data volume:** Federated learning enables the training of models without the requirement for data to be sent over the internet or stored in a centralized place, making it suitable for training machine learning models on enormous quantities of decentralized data. This can be especially helpful when the amount of data is massive to transfer or keep in one place.
- **Data updating:** Federated learning can be effective for training machine learning models on data that is continually evolving or being updated since it eliminates the need to send training data continuously to a centralized place. When data is continually changing and frequent data transfers to a central location are needed for training, this might be especially helpful.

2.4 Continual Federated Learning

In the developing subject of ML known as continual FL, the problems of FL and constant learning are combined. The capacity of a model to learn from a stream of data, continuously adapt to new tasks, and retain information from past ones is known as continual learning, sometimes known as lifelong learning. It has been a ML aim for a while, and there has been substantial recent development [17] [18]. However, 'catastrophic forgetting'—the phenomena where the model forgets previous tasks while it learns new ones—remains a challenge for many ongoing learning techniques. FL, on the other hand, is a distributed method to ML that enables model training on a vast corpus of decentralized data. Without sharing data samples, the model is trained across a number of distributed edge devices or servers retaining local data samples [19]. The protection of privacy and decrease in communication costs are two key advantages. Combining these two ideas, continual federated learning aims to continuously develop a model on dispersed data streams. The FL paradigm is used to collaboratively enhance the global model while allowing models to learn from a series of challenges on each local device [20]. It is a particularly active study subject because it has the potential to be used in fields including health informatics, IoT, and mobile devices, where data privacy and learning continuity are crucial.

As Continual Federated Learning emerges as a powerful paradigm for improving machine learning models, tactics like experience replay and buffer management become critical to its successful implementation. Experience Replay is a mechanism that saves the model's experiences (data samples) in a replay buffer or memory. Instead than using only the most recent data during training, the model randomly picks a batch of earlier events from the buffer to learn. This approach enables the model to break correlations in the observation sequence, flattening the data distribution and making better use of experiences, hence assisting in the mitigation of catastrophic forgetting (CF). This method becomes much more essential in Continual FL, as models continuously learn from a sequence of tasks on each local device [21]. There are several buffer techniques that can be adopted to enhance continual learning output.

- **Naive Buffer:** A naive buffer is the most basic type of replay buffer. It randomly gathers

samples from the training data and then randomly picks from these samples during replay. The advantage of this strategy is its simplicity and effectiveness in breaking correlations in the observation sequence. It may, however, neglect essential but unusual events on occasion.

- **Reservoir Sampling Buffer:** This form of buffer, which is an example of reservoir sampling, keeps a fixed-size sample of its items regardless of the amount of elements it encounters. When the buffer is full, it uses a probability to determine whether to replace an existing element in the buffer with a new one. The reservoir buffer guarantees that each viewed sample has an identical probability of appearing in the buffer, regardless of when it was encountered. It is especially beneficial when the data size is unknown or unlimited. [7]

- **Generative Replay Buffer:** Instead of keeping prior samples in a replay buffer, a generative model is employed in a generative replay buffer to generate them. The benefit of this strategy is that the buffer may produce as many" past" samples as needed, conserving memory space.[22]

- **Hindsight Replay Buffer:** The agent learns from a strategy it did not really adopt during its investigation in retrospect replay buffers, treating it as if it were the original plan. Because the agent may learn from its errors, it can lead to a more efficient learning process.[23]

These buffer solutions provide greater effectiveness and efficiency for continuous learning, allowing for better handling of catastrophic forgetting as well as more efficient memory utilization. As continual learning progresses and more research is undertaken, increasingly more complex and effective buffer methods are expected to be employed.

3 RELATED WORK AND LIMITATIONS

In this chapter, I will discuss about established research work related to the field of detecting anomalies and intrusions in networked systems using machine learning and their limitations. Additionally, I will highlight the differentiators of this thesis work with the mentioned related research papers.

In [24], authors proposed a Hierarchical FL architecture consisting of three layers which are clients, edge servers and global servers. Fully connected Artificial Neural Networks (ANNs) with 2 hidden layers populating 256 perceptrons in each layer has been used as local and global models. One of the significant findings of this research is the reduced training time advantage of FL in comparison to centralized model. The trade-offs between communication expense and training time based on frequency of contact with the servers were underlined by the authors in this paper. To conduct experiments researchers have used UNSW-NB15 dataset [25] to feed to their model, which is basically constructed out of synthetic environment along with major issues such as class imbalance and class overlap [26].

To fullfill the purpose of detecting IoT network intrusions, another proposition [27] has been made by the researchers by adopting both centralized and federated setup based on convolutional neural network (CNN) and recurrent neural network (RNN) models. To observe the outcome of the training and evaluation process of the federated setup, Edge-IIoTset [28], which consists of real-world traffic flows mixed up with various attack types such as DDoS, XSS, MITM, SQL Injection, and backdoor, has been split and distributed among a range of 3 to 15 participating clients in this work.

Another research work [29] related to application layer DDoS Protection has proposed a framework with four building blocks of Network Flow Collector, Features Extractor, Detector, and Security Policy Manager. Here Multi-Layer Perceptron (MLP) model has been used as the detector which has been fed app layer traffic collected and extracted by network flow collector and feature extractor, respectively. The authors have used intrusion detection dataset CICIDS- 2017 [30] to train the model. Being the traditional single node centralized anomaly detection system is the limitation of the proposed architecture which does not cover unknown attacks and sufficient attack surfaces of the network as well. Moreover, Neural Networks (NNs) have a tendency to forget previously learned information due to a condition known as CF which can be another drawback that can happen over time with the proposed model in this paper.

In [6], authors have investigated the performance of various ML approaches such as Decision Tree [31], Random Forest [32], K-Nearest Neighbor [33], Naive Bayes [34] and MLP [35] in terms of anomaly detection in 5G. To conduct training, authors have constructed 5G NIDD dataset [36] populating real world network traffic collected from 5GTN implemented at University of Oulu, Finland. Researchers have covered all types of DDoS attacks by generating attack flows using various attack tools like Hping3, goldeneye, slowloris, torshammer and NMAP along with legitimate traffic flows to construct the dataset. However, in this paper, instead of leveraging federated learning process, traditional single node machine learning approach has been taken into account.

To address the challenge of continuously evolving cyber-attacks, authors of the paper [37] proposed Anomaly-based Network Intrusion Detection Systems (A-NIDS) by applying continual learning to eradicate catastrophic forgetting. Researchers have chosen CICIDS 2017, CICIDS 2018 and KDD Cup'99 datasets to investigate the performance of the proposed system. To cope up with class imbalance problem of the datasets, class incremental and domain incremental learning settings have been adopted with the framework. Researchers have chosen simple MLP and CNN [38] for detection purpose. Furthermore, two continual learning algorithms, which are Elastic Weight Consolidation (EWC) and Gradiaent Episodic Memory (GEM), have been implemented to reduce CF.

The authors of the survey paper [39] presented a thorough investigation of FL in the context of intrusion detection systems. By training models locally on devices and communicating just the model parameters to a central server, FL, as a decentralized learning approach, protects security and improves privacy. The authors investigated numerous technologies and methods that use ML, DL, and FL for intrusion detection, highlighting their advantages and synergies. Additionally, they emphasized current difficulties in the use of FL in intrusion detection and suggest potential future study routes, successfully laying the groundwork for greater research in this field. This thorough study offers a useful insight of FL's function and possibilities for enhancing intrusion detection systems. With a motivation to leverage immense capabilities of FL described in this research paper[39], I have employed FL as a part of my solution to tackle DDoS attack and protect decentralized 5G network in this dissertation,

Another research work regarding anomaly detection in RAN environment is presented in [40], where researchers investigated the increasing complexity of mobile radio access networks as well as the rising of customer service quality demands. They emphasized the necessity for automated network management and maintenance solutions to meet high service level expectations, evaluating existing solutions such as Self-Organizing Networks (SON), Mobility Load Balancing (MLB), and automatic antenna tilting, which frequently overlook service KPIs, which directly influence user experience. They discovered a gap in anomaly detection accuracy as well as a lack of telecommunication data-specific functions in existing R libraries, noting that most current algorithms, designed for specialized datasets such as fraud detection or earthquake signals, fail when applied to telecommunication network performance data. To address this, they examined and tweaked R's" changepoint" package[41], which can identify not just local abnormalities but also anomalous time series—an important aspect of radio access network (RAN) performance monitoring. They improved the method to reduce false positive anomalies and compared its performance to that of other prominent R libraries, confirming its higher accuracy for radio network performance data. Their research proposes a scalable, general anomaly detection system for high-dimensionality data that avoids the need for further tuning for different KPI groups or technologies, hence aiding automated network performance monitoring significantly.

The authors in [42] addressed two key issues in the realm of Deep Learning-based IDS in this study: CF and covariate shift. To identify and quantify these alterations in data distribution, the authors presented an eight-stage statistics and machine learning guided implementation approach. They also presented a unique feature importance-based approach for assessing the influence of individual feature drift on IDS performance. The NSL KDD[43] and CICIDS 2017[44] datasets were analyzed using this methodology. The study found that continuous learning-based approaches outperformed classic statistical techniques and cutting-edge boosting and DNN models in terms of accuracy and false-positive rates.

Here[45], the authors offer a unique solution to network slicing in 5G networks, which they describe as a Virtual Network Embedding (VNE) problem in their research study. Their purpose is to map slice requests onto the core network as efficiently as possible. They found two important hurdles in this area: guaranteeing slice isolation for protection against DDoS attacks and obtaining high request coverage. To overcome these issues, the authors provide Slice Isolation-based RL (SIRL), an actor-critic Reinforcement Learning (RL) paradigm. This methodology creates the issue environment using five ideal graph characteristics, which are subsequently controlled using a ranking mechanism. This ranking system simplifies the characteristics while also improving learning performance.

In this paper[46], researchers presented an autonomic and cognitive security management architecture developed for 5G and beyond networks in this study. This unique architecture enables fine-grained zero-touch security control at several levels, including network functions,

sub-slices, and slices, as well as across multiple administrative and technological domains. They proved the framework's interoperability with current standards including zero-touch network and service management (ZSM), 3rd Generation Partnership Project (3GPP), and Network Functions Virtualization (NFV). This was done to demonstrate that their technique is compliant with current standards and so acceptable for real-world use. Authors created a testbed to enable distributed and totally autonomous detection of abnormalities inside a network slice and to serve as a Proof-of-Concept (PoC) for monitoring and analytics functionalities. An anomaly detection model was incorporated into the analytics service utilizing multivariate time series and the unsupervised deep learning method LSTM (long short-term memory) AutoEncoder. They used a dataset of 2361 samples to train the LSTM-based AutoEncoder model to reconstruct time-series for typical behavior, with 20% of these samples being saved for validation. They carried out tests where abnormalities were effectively identified, such as application-layer DDoS assaults on the video streamer CNF.

In response to the problem of CF in FL, in which the global model forgets past learned knowledge while adjusting to new tasks, the researchers [47] developed Continual Federated Learning with Distillation (CFeD). The team set up a testing environment with 100 clients, 10 percent of whom were chosen at random to participate in each training cycle. For text and picture classification tasks, they employed datasets such as THUCNews, SogouCS, Sina2019, NLPIR Weibo Corpus, CIFAR10, CIFAR-100, and Caltech-256. They created task sequences for two different scenarios: Domain-IL and Class-IL. Domain-IL represented circumstances in which input distributions changed continuously during the series, whereas Class-IL indicated instances in which new classes arose progressively across the run. They employed Finetuning, FedAvg, MultiHead, EWC, LwF, DMC, and their own developed approach, CFeD, for comparative evaluation. The study includes successively training models on various tasks and assessing the performance of their methods against these proven methodologies.

The below mentioned table (Table 3.1) depicts comparative overview of this thesis work with related research works described in this chapter.

Table 3.1: Comparison with related research work.

Reference	Year	Techniques used	Contribution
[23]	2022	Conventional FL setup	The research work presented traditional Federated Learning architecture, demonstrating its reduced training time advantage compared to centralized models. The work also provides insight into the trade-offs between communication costs and training time in a FL context, based on the frequency of contact with servers
[26]	2023	Single node ML models and conventional FL setup	The research approached detecting IoT network intrusions, using both centralized and federated setups with convolutional neural network (CNN) and recurrent neural network (RNN) models, and demonstrating their effectiveness with real-world traffic data containing various attack types.
[28]	2020	Single node ML models	This research proposed a four-block framework for application layer DDoS Protection using a Multi Layer Perceptron (MLP) model as single node model, demonstrating its effectiveness through the use of the CICIDS-2017 dataset.
[6]	2022	Single node ML models	The authors in this study evaluated the effectiveness of various machine learning models, including Decision Trees, Random Forest, K-Nearest Neighbor, Naive Bayes, and MLP, for anomaly detection in 5G, utilizing a custom-built 5G NIDD dataset composed of real-world network traffic and a comprehensive range of DDoS attack types.
[36]	2022	Single node ANN with EWC and GEM	The authors proposed an Anomaly-based Network Intrusion Detection System (A-NIDS) that employs continual learning to mitigate catastrophic forgetting, effectively addressing evolving cyber threats, utilizing a combination of different datasets and continual learning algorithms like Elastic Weight Consolidation (EWC) and Gradient Episodic Memory (GEM)
[38]	2022	Conventional FL setup	The authors conducted a comprehensive exploration of Federated Learning (FL) in intrusion detection systems, highlighting its privacy and security benefits, investigating related technologies and methodologies, addressing existing challenges, and paving the way for future research avenues, thus providing valuable insight into the potentials of FL in enhancing intrusion detection.
[39]	2018	changepoint (R library)	The researchers addressed the complexity of mobile radio access networks and customer service expectations by adapting the "changepoint" package in R to develop a scalable, generic anomaly detection algorithm for high-dimensionality data, proving its superior accuracy in handling telecom network performance data and aiding automated network performance monitoring.
[41]	2022	Single node DNN models with LwF, experience replay, and dark experience replay	The researchers addressed the challenges of CF and covariate shift in deep learning-based IDS, employing eight-stage framework and continual learning models in large-scale systems security.
[45]	2023	Reinforcement Learning (RL)	Slice requests are efficiently mapped into the core network using the author's suggested Slice Isolation-based Reinforcement Learning (SIRL) model, which makes use of five ideal graph characteristics and a ranking mechanism to provide high request coverage and slice isolation for DDoS prevention.
[46]	2022	LSTM (long short-term memory) AutoEncoder	In order to provide fine-grained zero-touch security control across many levels and domains, the researchers designed an autonomic and cognitive security management architecture for 5G and beyond networks.
[44]	2022	Continual FL with Distillation technique	Continual Federated Learning with Distillation (CFeD) approach has been made by the researchers to prevent catastrophic forgetting in Federated Learning
	2023	Continual FL with reservior sampling buffer technique	To detect DDoS attacks in 5G, continual FL based setup has been proposed with reservoir sampling buffer to maintain effeciency in performance over time by preserving old knowledge during continuous learning of new traffic pattern while detecting attacks targeting various nodes in distributed network.

4 CONTINUAL FEDERATED LEARNING FRAMEWORK

In this section, I will describe the complete continual FL framework and work process that I am proposing towards making a sustainable network attack detection system.

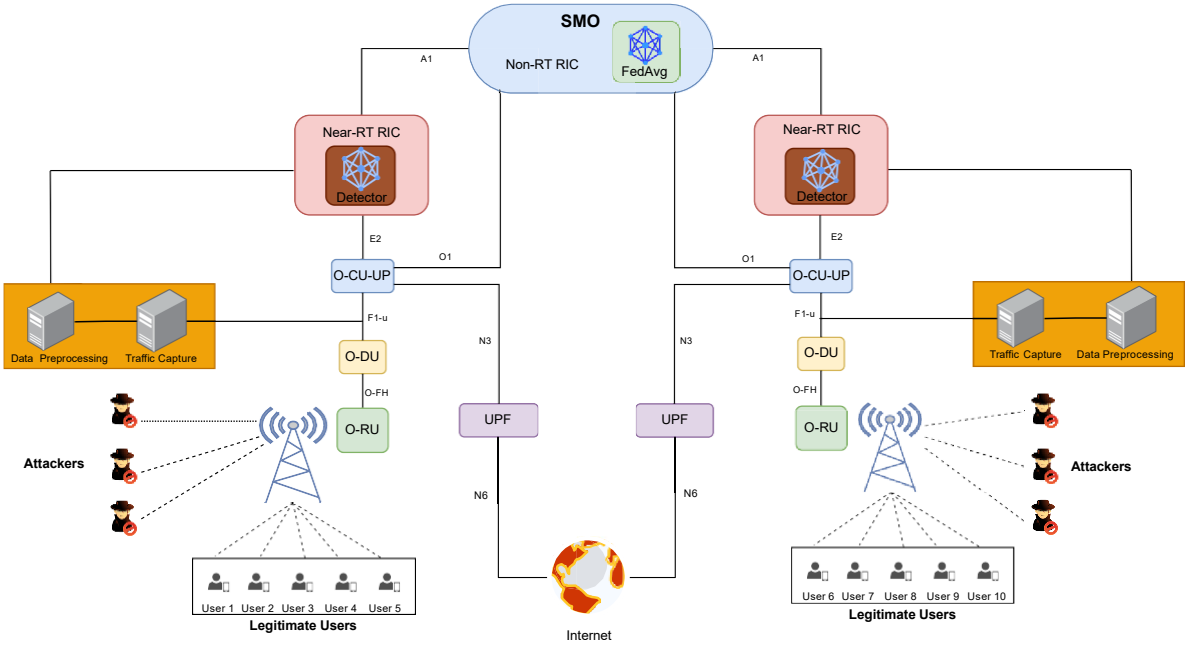


Figure 4.1: Integration of the Proposed Model in O-RAN.

Integration and placement of proposed DDoS detection framework in O-RAN architecture has been shown in Figure 4.1. The local detector for each basestation is deployed in Near-RT RIC as xAPPs to protect corresponding basestation from incoming anomalous traffics. Federated aggregation process is placed in Non-RT RIC of the segment of SMO, where global parameters for all basestations is generated along with updating local models accordingly. As shown in Figure 4.1, incoming traffic to each basestation is captured from the F1 interface between O-DU and O-CU-UP and redirect towards local detector(xApp) after conducting data preprocessing. Then after finishing necessary training process in clients, local weights are sent for federated aggregation in SMO section through A1 interface. The entire dataprocessing steps with the detailed description of used dataset, model framework and deployment procedure has been described in below sections.

4.1 Description of Dataset

The 5G-NIDD dataset is unique since it produces benign traffic using live traffic from actual mobile devices in the network in contrast to other datasets that often use simulated traffic. The use of many protocols, including HTTP, HTTPS, SSH, and SFTP, provides more realistic traffic behavior and adds to the depth and variety of the dataset.[6]

4.1.1 Attack types and tools

A prominent class of cyber risks is known as DoS attacks, the primary objective of which is to prevent authorized users from accessing a system or network resource [48]. This can be done by temporarily or permanently interrupting the functions of a host that is connected to the Internet, frequently by flooding it with an overwhelming amount of Internet traffic [49]. In a DoS attack, the attacker specifically seeks to impede authorized users from using the targeted service [50]. The use of network ”flooding”, which blocks lawful network traffic, or machine interference, which limits access to a certain service, are common techniques [50]. There are two classes of DoS attacks. The first category consists of assaults intended to bring down a server. The second, often referred to as flood assaults, happen when the server is overloaded with too much traffic to buffer, which slows down the system until it ultimately stops working [49]. Researchers [6] examined three common forms of DoS/DDoS attacks: application layer, protocol-based, and volume-based assaults. Using tools like Hping3, volume-based assaults utilized methods like ICMP Flood and UDP Flood to deliver an excessive amount of data and overload network resources. The SYN Flood, SYN Scan, and TCP Connect Scan attacks, which manipulate the TCP three-way handshake protocol to drain network resources or use Nmap to search for open ports, are examples of protocol-based attacks that took use of weaknesses in the current protocol. Application layer assaults, like the HTTP Flood, target application layer services by impersonating normal human activity in order to avoid detection and are carried out utilizing tools like the Goldeneye. The Slowrate DoS was also investigated for its distinctive, slowly moving assaults, which are often more challenging to spot. Furthermore, they used Nmap to carry out a UDP Scan to assess port statuses. All of these methods reflect several DoS/DDoS attack subcategories that are included in the dataset.

- **ICMP Flood:** To overrun the target network in this investigation, ICMP echo requests were sent at high frequency. This assault, which made use of the Hping3 program, rendered the service unavailable to normal users.
- **UDP Flood:** The researcher used the Hping3 program to send a large number of UDP packets in an attempt to simulate a UDP flood assault. As a result, the system stopped responding since there were so many ”Destination Unreachable” answers.
- **SYN Flood:** The researchers launched a SYN flood assault by taking advantage of the TCP three-way handshake protocol. This resulted in a large number of partially open connections, eventually draining the receiver and blocking access to authorized users.
- **HTTP Flood:** An HTTP flood attack, which is good at imitating human behavior and evading detection, was also used in the research to target the application layer. This web server assault was conducted using the Python-based Goldeneye tool.
- **Slowrate DoS:** The researchers also looked at slow rate DoS assaults, which are more difficult to identify since they move at a slower pace and send out fewer packets. They used several Python scripts and tools to carry out slow POST requests and slowloris DoS attacks.
- **SYN Scan:** The researchers used a partial three-way handshake during the setup of a TCP connection to find available ports using the SYN scan technique. The Nmap open-source program was used to carry out this quick and well-liked scanning methodology.

- **TCP Connect Scan:** Unlike SYN scan, the TCP connect scan successfully completed the three-way handshake, which made it slower but necessitated less privileges. The Nmap program was also used to do this scan.
- **UDP Scan:** As a last step, the researchers sent UDP datagrams to certain ports to perform a UDP scan. The target's response, or lack thereof, was used to establish a port's state. The Nmap tool was also used to carry out this scan.

4.2 Data Preprocessing

- **Handling Missing values:** I have removed all missing values and infinite values from the datasets, as they contain substantial number of missing and infinite values.
- **Feature and Target Seperation:** Here I have seperated features from recorded network flows in the datasets and their labels in the form of independent and dependent variables.
- **Label Encoding:** Moving forward with binary classification approach, I have categorized the input traffic into legitimate and malicious traffic. I have taken all DDoS classes as part of the attack category along with regular flow as benign traffic. Therefore, I have encoded the legitimate and attack traffic patterns to binary values of 0 and 1 correspondingly.
- **One-Hot Encoding:** The process to convert categorical features into a format that can be feed to machine learning model is named as one-hot encoding. In my experiments, I have conducted one-hot encoding transformation for certain categorical features, where I have created separate binary feature for each and every unique category contained by a particular categorical feature. These newly created dummy features basically represent the presence with a binary 1 or the absence with a binary 0 of a specific feature in the datasets.
- **Feature Scaling** Since many machine learning algorithms struggle when the input numerical properties have variable scales, standardization of the features is crucial. In this research, I have performed standardization of the feature set prior to feeding data into model using standard scaler technique which standardizes the features by removing the mean and scaling to unit variance [51].
- **Sampling Buffer** I have kept the size of the buffer reservoir as 5000. This means that to maintain a broad and representative sample of experiences for the training process, the buffer will keep track of a subset of 5000 data samples from the incoming data stream. The reservoir sampling technique decides whether to keep or reject a new experience based on a given probability when fresh experiences come in and the buffer fills up [7]. In order to ensure that the model can learn from a representative collection of events without stressing the memory resources, the reservoir sampling approach is implemented in this way, which is a crucial step in memory management for tasks involving continuous learning.

4.3 Proposed model

I have used MLP approach for training and detection of network anomalies in our research. The detection model which is a feedforward neural network is constructed with one input layer

of 82 input features coming from 5G NIDD dataset [6], three hidden layers with 64 neurons each and one output layer containing only one node for binary classification detection. I have used ReLU activation function in hidden layers and sigmoid in the final layer of our model. To go with the nature of detection scenario, I have used binary crossentropy (BCE) as criterion to minimize the loss during training process. Moreover, as optimizer of the model, adaptive learning rate optimization algorithm, Adam has been utilized. Carrying out investigation with various values, learning rate has been kept as 0.01 for optimum detection accuracy. From feature selection perspective, I have deleted the features such as source and destination IP addresses as well as ports to keep the anomaly detection system as generic as possible. I also eliminated features with null values and consistent values across all flows since they have no influence on model correctness. After removing unnecessary features, I have considered most other features critical and kept them as input of the model by considering the fact that, network traffics are continuously changing over time with the evolution of zero day attacks.

4.4 Experimental Setup

To conduct experiments, I have used three virtual machines (VMs) as shown in Figure 4.2, two of which served as clients and one of which served as the central server for federated averaging of the weights coming from clients after determined rounds of training. Each VM had the configuration of two CPU cores, 5 GB of RAM, and the Ubuntu 22.04.2 LTS (Jammy Jellyfish) operating system installed.

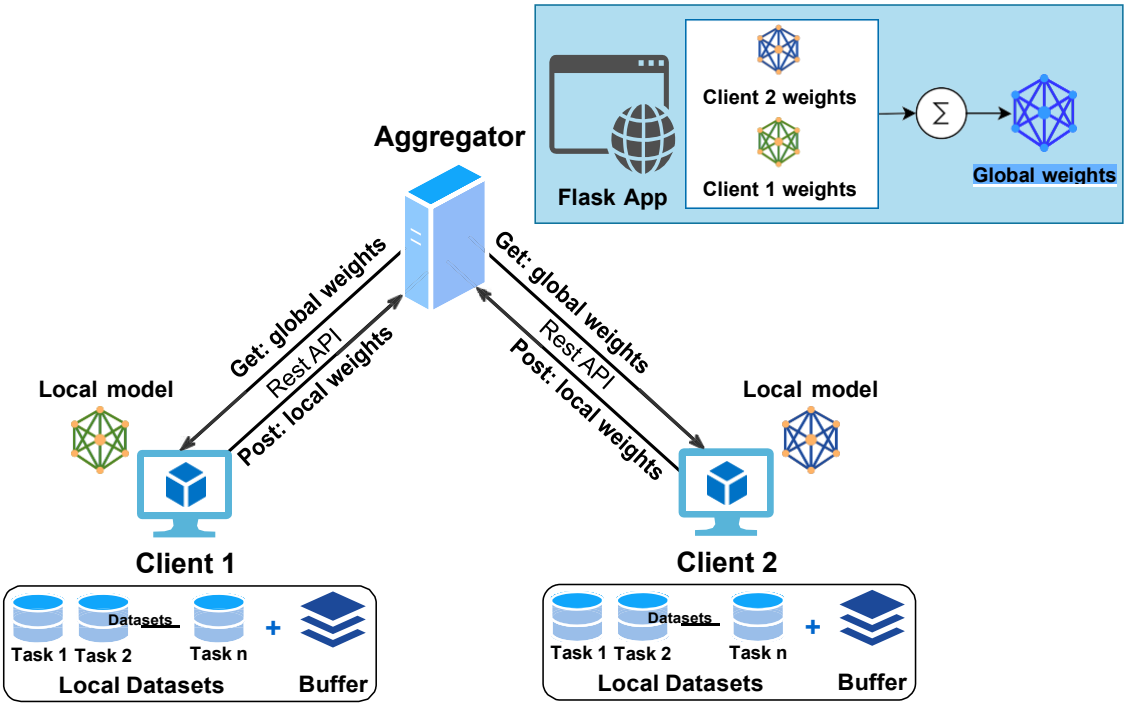


Figure 4.2: Experimental Setup.

Each of the remote clients sends the weights of the model using HTTP post request after the training phase with local data and checks HTTP response to ensure successful transaction of weights. The central server performs standard federated averaging (FedAVg)[19] after ensuring

arrival of weights from all the remote clients and serves the updated weights to each of the client using HTTP GET method. Besides enabling scalability in terms of handling large number of clients, FedAvg can reduce communication cost and handles Non-IID data effectively which has motivated in applying this technique as aggregation process in this thesis work.

4.5 Training Strategy and Federated Aggregation Process

4.5.1 Local Training Process

After preprocessing and preparing all datasets to fit for the proposed model, I have trained the model by considering two different scenarios. The first scenario goes with conventional FL skipping CL integration where I have fed all the datasets categorized by different attack types one by one to the local models in two clients and train the models with the batch size 128 along with learning rate 0.01. During the training process of one batch, output of one layer has been fed forward to other layer and at the end of the training, average loss has been calculated and fed backward to the model for optimization. Training has been conducted in 10 epochs for each dataset. When the training is finished, corresponding client sends the weights of the model to aggregation server for federated averaging [19] to be conducted. After federated averaging process, the clients get the updated weights coming from aggregation server and updates their models accordingly. By following 3 rounds of the same process of federated averaging with one training dataset, evaluation process for all the test datasets takes place to see how the effectively the local models are identifying attacks of other client. Here to test the efficiency of the local models, I have chosen training datasets of one client as the test dataset for other client which has not been encountered by the local model before. Evaluation for all training dataset have also been performed to see how CF is happening in the model, as the model proceeds further with the training of new datasets everytime.

At the second scenario, I have chosen Reservoir Sampling Buffer [7] to preserve a portion of previously learned samples. The algorithm of entire local training and evaluation process has been mentioned in Figure 4.3. As per the algorithm in Figure 4.3, I have employed techniques to eradicate over-fitting problem, where the the model checks for maximum 3 epochs of accuracy and compares with highest accuracy point got so far. If the accuracy doesn't improve in 3 epochs it stops the training process and moves forward with the highest accuracy achieved. After completing training with one dataset, buffer is filled with representative samples from that dataset and when new dataset comes into place for training, the buffer is concatenated and shuffled with the current dataset before feeding to the model. By following the above processes of training and federated aggregation, evaluation for all train and test datasets happen the same way as first scenario and metrics results between two scenarios have been compared with each other to observe how the local models are detecting anomalies along with eradicating CF in parallel.

Algorithm 1 Client-Side Training and Evaluation

```
procedure TRAIN_AND_EVAL_ON_MULTIPLE_DATASETS(train_datasets, eval_datasets, num_epochs)
    for i = 0 to len(train_datasets) do
        best_accuracy ← 0.0
        patience ← 3
        wait ← 0
        for epoch = 0 to num_epochs do
            avg_loss, avg_accuracy ← TRAIN(train_datasets[i])
            if avg_accuracy > best_accuracy then
                best_accuracy ← avg_accuracy
                wait ← 0
            else
                wait ← wait + 1
                if wait ≥ patience then
                    break
                end if
            end if
        end for
        Send updated model weights and current data size to central server
        Retrieve averaged weights from central server
        Update local model with averaged weights
        Add the current training sample to the replay buffer
        for j = 0 to len(eval_datasets) do
            EVAL(eval_datasets[j])
        end for
        for m = 0 to len(train_datasets) do
            EVAL(train_datasets[m])
        end for
    end for
end procedure
procedure TRAIN(train_dataset)
    if len(storage.buffer) = 0 then
        Standard training loop
    else
        Training loop with replay buffer
    end if
    return avg_loss, avg_accuracy
end procedure
procedure EVAL(eval_dataset)
    Evaluation loop
    return avg_loss, avg_accuracy
    return precision, recall, f1score
end procedure
```

Figure 4.3: Algorithm of Training and Evaluation in Client

4.5.2 Federated Averaging Process

After performing local training process, all clients send their weights and corresponding data size which is used by the central server to perform standard federated averaging (FedAvg) [19] by following below formula:

$$w_{\text{avg}} = \frac{\sum_{k=1}^{K} n_k \cdot w_k}{\sum_{k=1}^{K} n} \tag{4.1}$$

Here, w_{avg} denotes the averaged weights, K is the total number of clients, n_k is the number of samples at client k, and w_k are the weights received from client k. After conducting federated averaging, updated weights are then distributed to all clients before starting next training and evaluation process locally. The pseudo code of entire aggregation process of the central serverhas been shown in Figure 4.4

Algorithm 2 Federated Averaging Process in Central Server

Initialization:
Initialize an empty list *weights_received*
Initialize an empty list *data_sizes_received*
Initialize *averaged_weights* to None
procedure FEDERATED_AVERAGING(weights1, weights2, data_size1, data_size2)
    $averaged\_weights \leftarrow \frac{\sum_{k=1}^{K} n_k \cdot w_k}{\sum_{k=1}^{K} n_k}$
    return *averaged_weights*
end procedure
procedure RECEIVE_WEIGHTS
    Get weights and the number of samples from the client request
    Append weights and the number of samples to the *weights_received* and *data_sizes_received* list
    if length of *weights_received* equals to 2 **then**
        averaged_weights ← FEDERATED_AVERAGING(*weights_received[0], weights_received[1], data_sizes_received[0], data_sizes_received[1]*)
        Empty *weights_received*
        Empty *data_sizes_received*
    end if
    return success status
end procedure
procedure GET_WEIGHTS
    if *averaged_weights* is not None **then**
        return *averaged_weights*
    else
        return no weights status
    end if
end procedure

Figure 4.4: Algorithm of Federated Aggregation Process

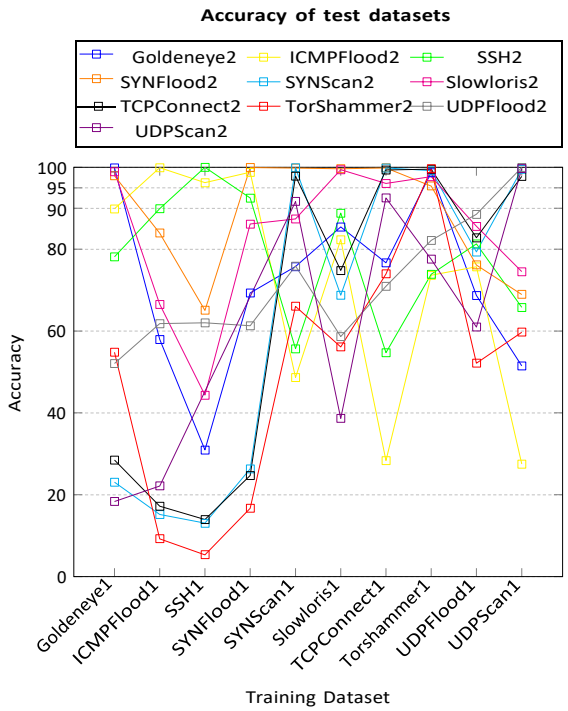
5 EVALUATION

In this chapter, the entire evaluation process of the local models in all remote clients will be described elaborately. We know that, with the distributed nature of 5G O-RAN, possibility and susceptibility to cyber attacks have gone up high. Moreover, day by day network traffic patterns along with the way of performing cyber attacks are changing along with increasing attack surface. That is why although federated setup can efficiently reduce cyber attack from various vulnerable nodes, the consistency in detecting anomalies decrease with time due to forgetful nature of the local models. As the models in various network end points continue to learn new traffic patterns, they tend to forget the knowledge preserved from old samples which is called CF. To observe the performance of the proposed model in detecting cyber threats specifically DDoS attacks, I made two strategies, one of which are conventional FL where the performance metrics along with the sign of CF has been observed and another one comes with knowledge preserving nature which is the strategy of FL integrated with CL.

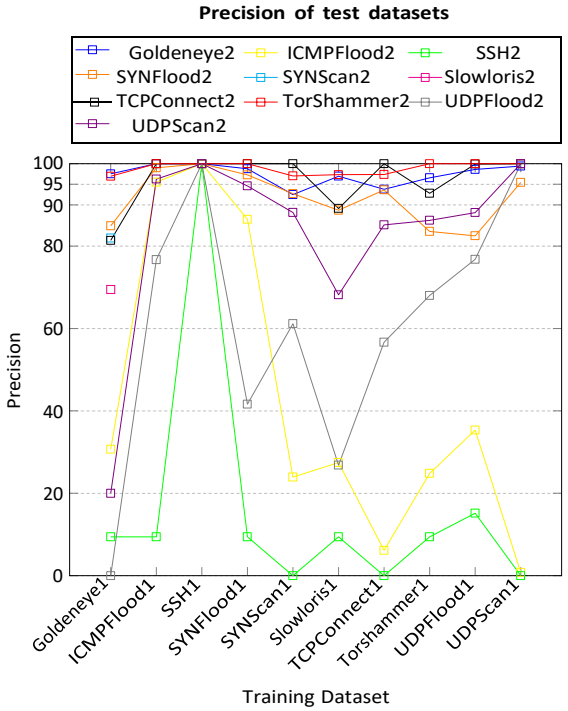
5.1 Conventional Federated Learning Strategy

To evaluate the designed model, at the first place, I have started by making a strategy which will perform and improve the detection efficiency of the local models by standard federated learning concept. one loop of the training process include 10 epochs of training and immediately after the training process, each of the two remote clients sends their local model's weights to central server for the calculation of standard federated averaging process. Upon completion of federated averaging, central server updates all local models with updated weights. The above mentioned process has been considered as one round of training. By careful observation of evaluation metrics results, I had to conduct 3 rounds of training process with each training dataset to get satisfactory performance from the local models.

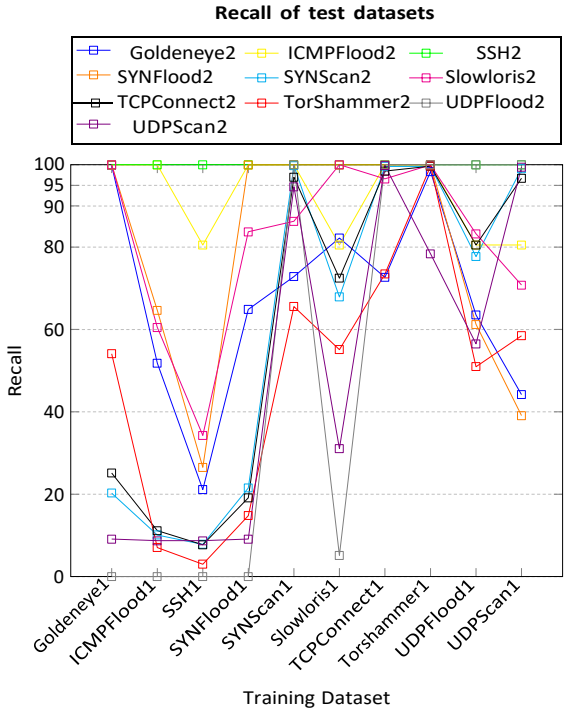
After completing training with each dataset, I have swapped current training dataset of one client to other client which has been considered as test dataset for that client so that each client can be tested with flows that their corresponding local model has not experienced during training. Moreover, to investigate the existence of CF, I have tested local models with all datasets right after conducting training process with each separate attack dataset.



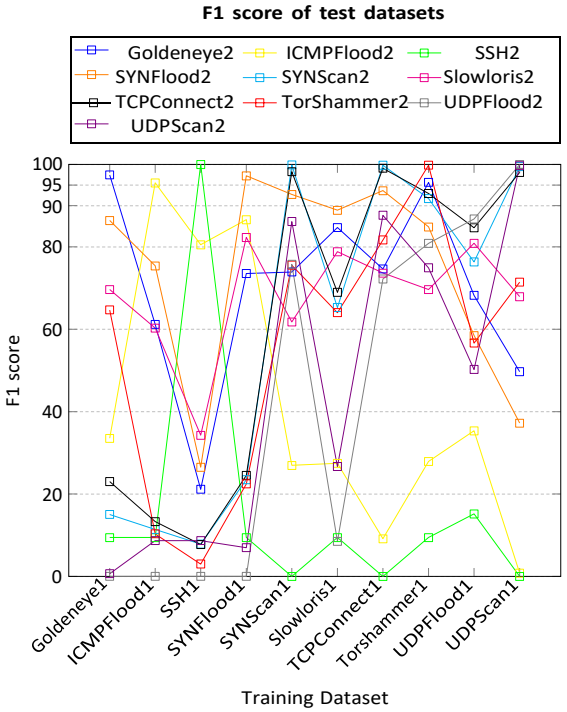
(a) Accuracy of test dataset in client 1.



(b) Precision of test dataset in client 1.

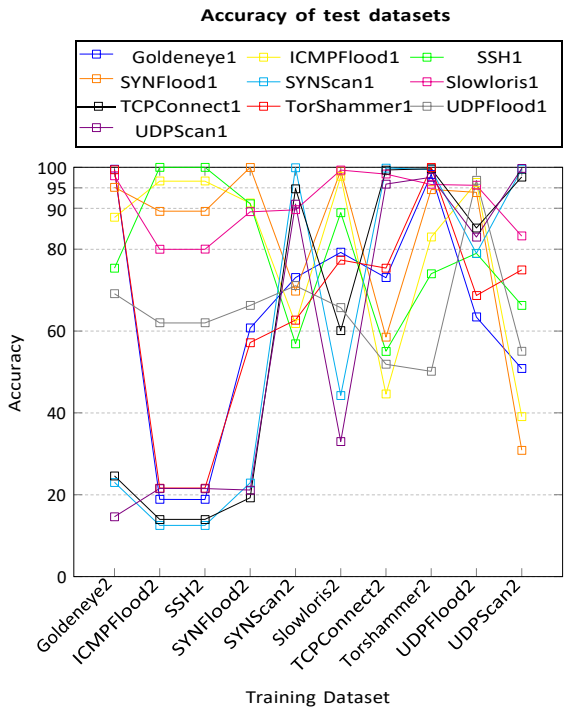


(c) Recall of test dataset in client 1.

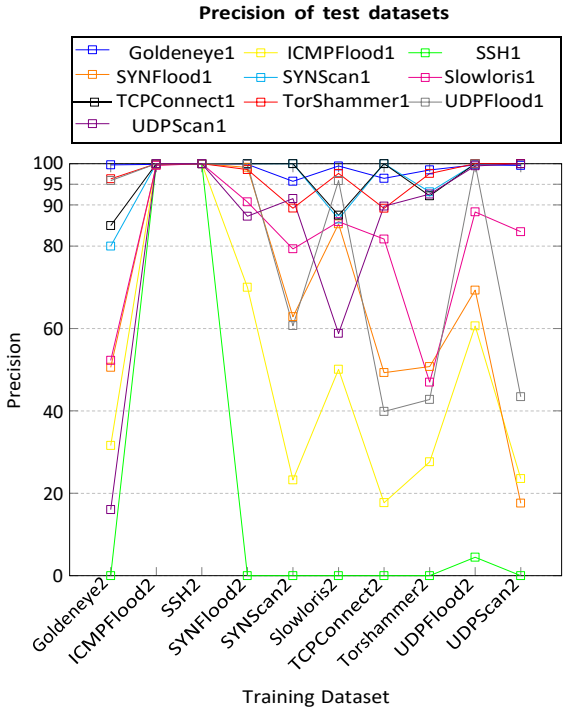


(d) F1 score of test dataset in client 1.

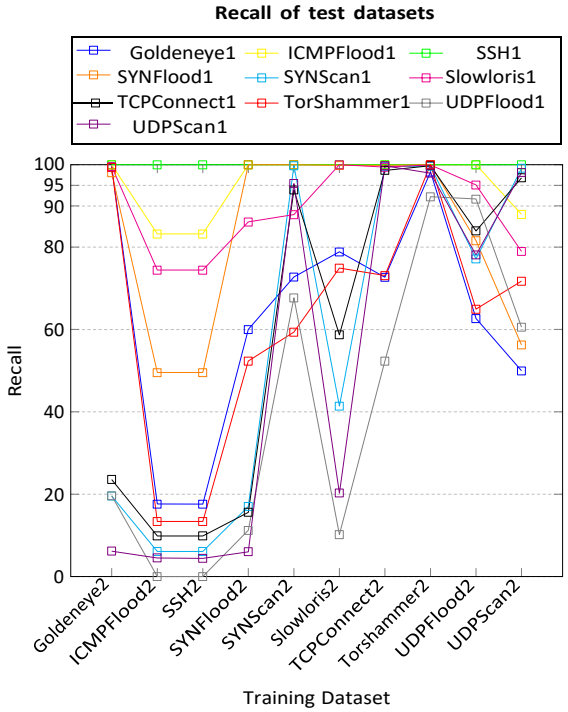
Figure 5.1: Performance evaluation metrics of the test dataset in client 1.



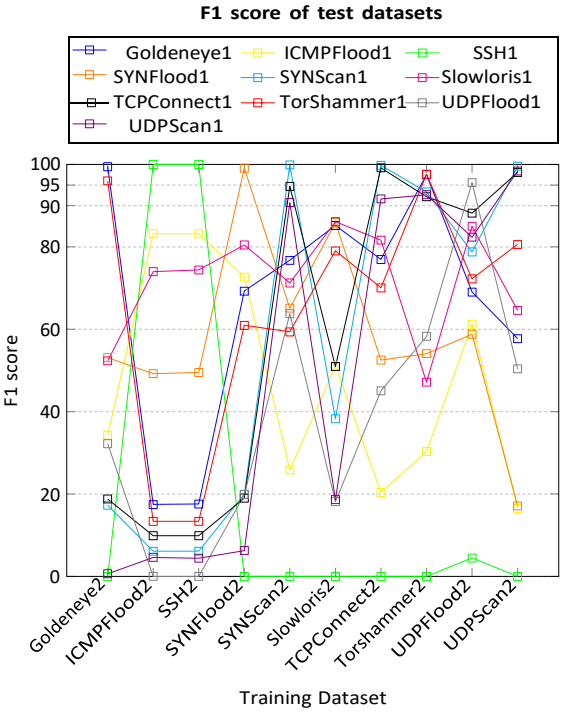
(a) Accuracy of test dataset in client 2.



(b) Precision of test dataset in client 2.



(c) Recall of test dataset in client 2.

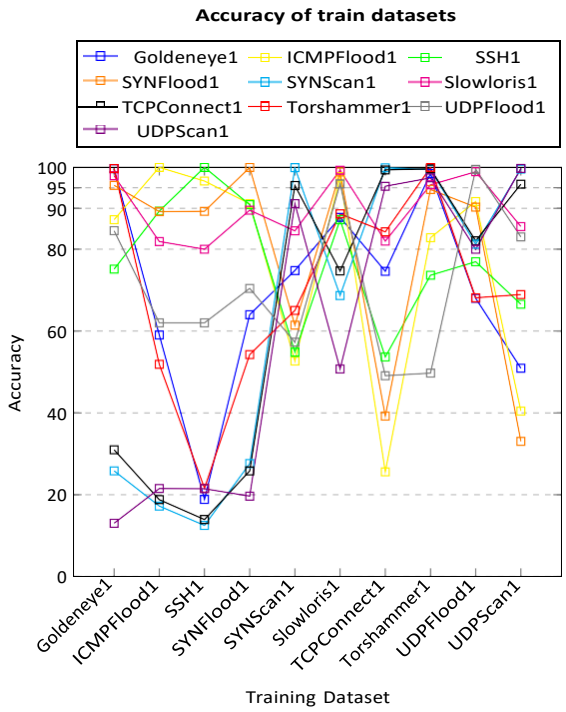


(d) F1 score of test dataset in client 2.

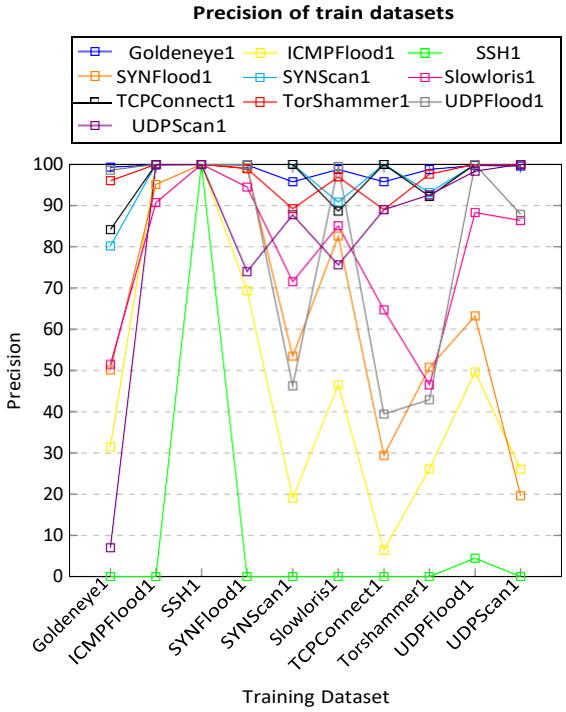
Figure 5.2: Performance evaluation metrics of the test dataset in client 2.

I have drawn dedicated line graphs for accuracy, precision, recall and F1 score of both test data and train data separately for each client. From the line graphs of accuracy, precision, recall and F1 score for both clients, a significant fluctuation can be seen in results, which means that, as the recorded network flows are different in different dataset, moving forward with each dataset,

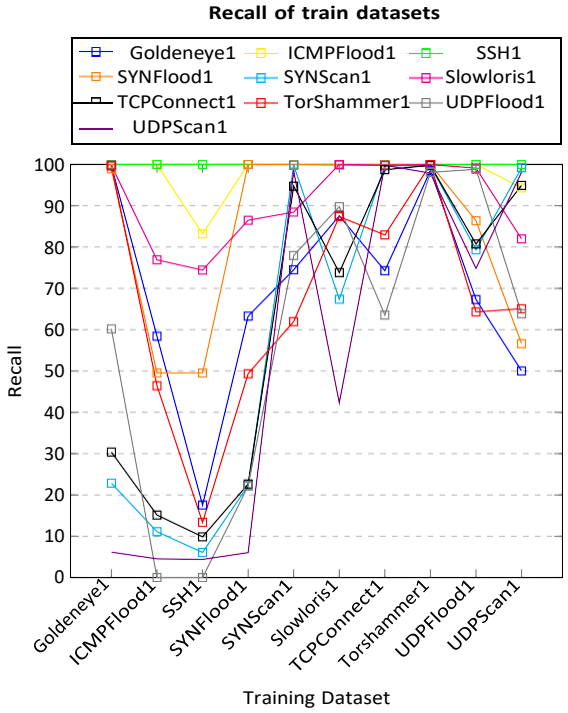
associated local models in clients tend to forget what it has learned from the previous dataset. That is why when any local model of any client encounters old samples, it is not being able to detect accurately. It is a clear sign of forgetful nature of the ML models.



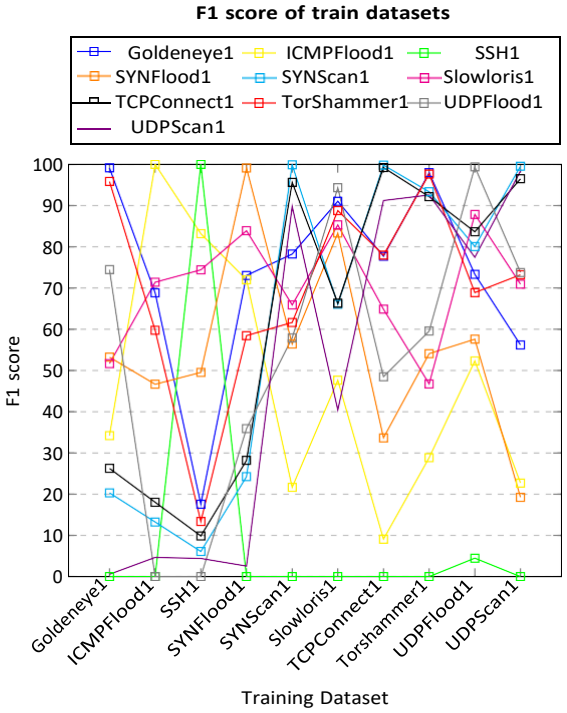
(a) Accuracy of train dataset in client 1.



(b) Precision of train dataset in client 1.

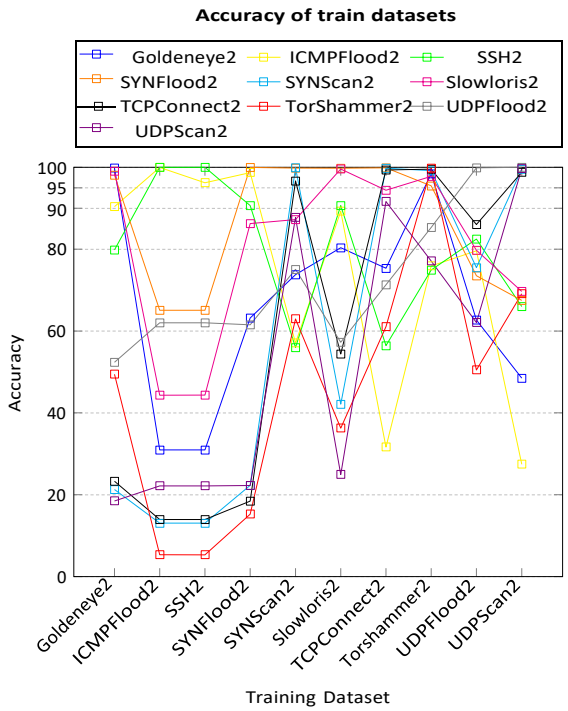


(c) Recall of train dataset in client 1.

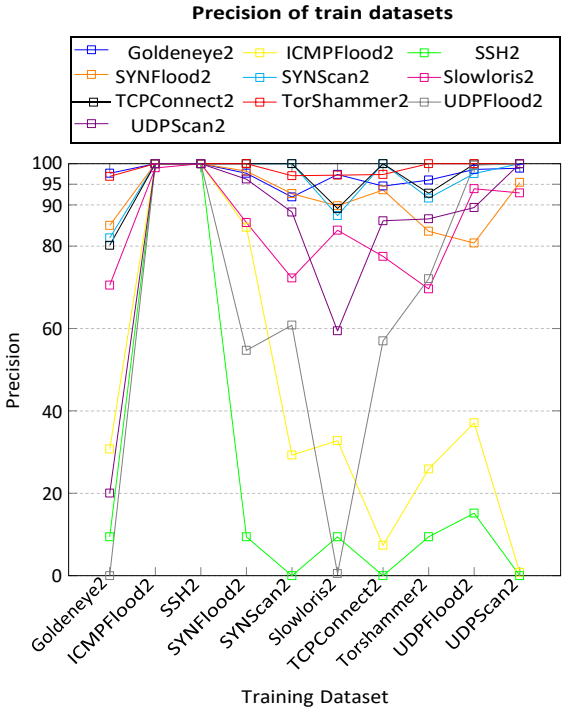


(d) F1 score of train dataset in client 1.

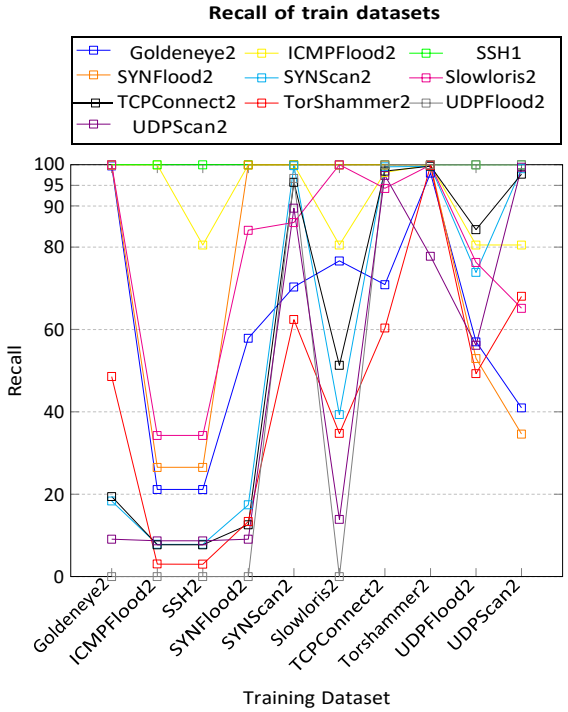
Figure 5.3: Performance evaluation metrics of the train dataset in client 1.



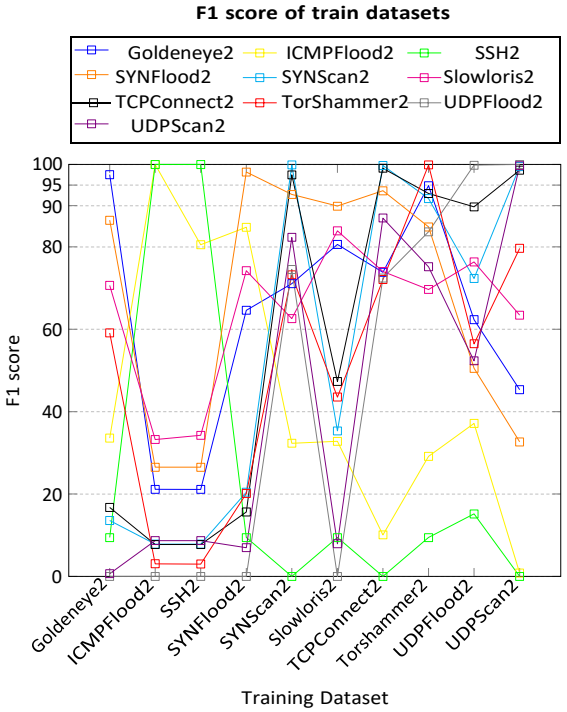
(a) Accuracy of train dataset in client 2.



(b) Precision of train dataset in client 2.



(c) Recall of train dataset in client 2.



(d) F1 score of train dataset in client 2.

Figure 5.4: Performance evaluation metrics of the train dataset in client 2.

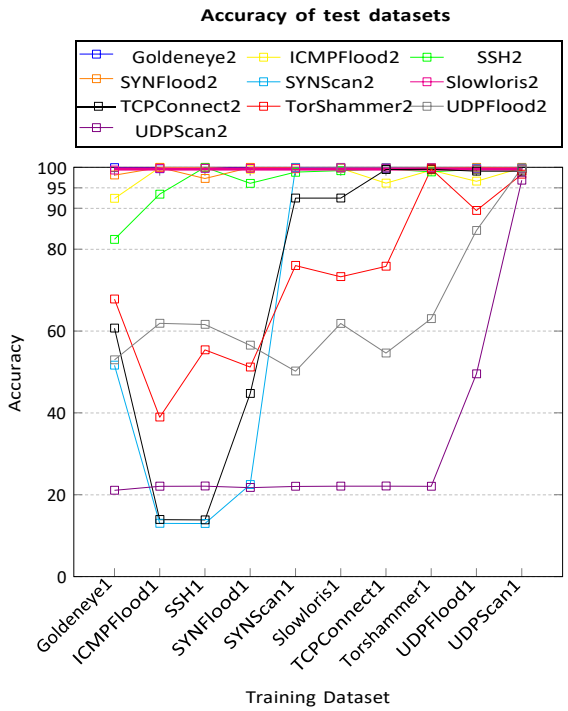
Evaluation metrics table 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6 mentioned at the end of this chapter for both clients offer more clear perception about how CF is occurring in traditional FL setup. For example, after training with first dataset named Goldeneye 1 and Goldeneye 2, respectively, in client 1 and client 2, I tested both clients against both datasets. The evaluation accuracy

for client 1 came out as 99.67% and 99.88% for train and test data, respectively. On the other hand,in client 2, an accuracy of 99.85% and 99.54% sequentially for train and test data has been achieved. However, when the second training dataset (i.e., ICMPFlood 1 and ICMPFlood 2) is fed to the local models, weights start to change by the knowledge gathered around the current dataset. That is why accuracy drops drastically for previous train and test data to 59.04% and 57.94% in client 1 along with 30.93% and 18,86% percent in client 2. Same problem has been experienced in the values of precision, recall and F1 score also. After finishing entire training process with 10 datasets in each client, all the metrics output stays quite low and inconsiderable, except for the last dataset.

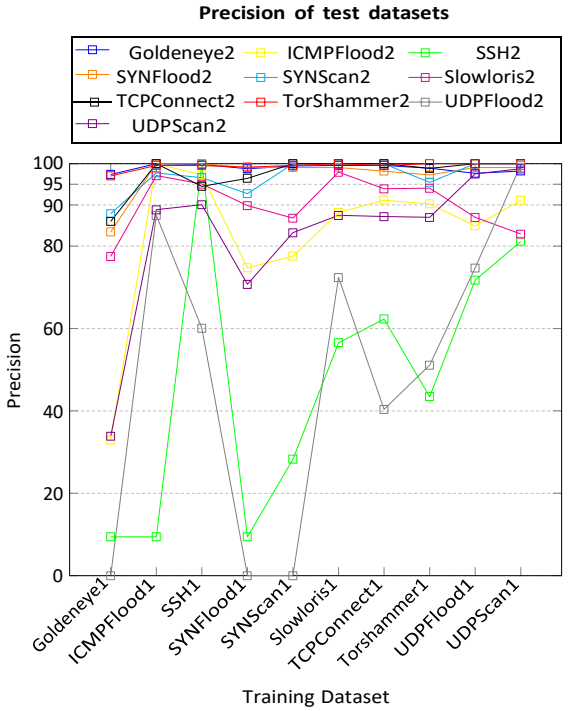
5.2 Continual Federated Learning Strategy

To reduce the effects of CF in local models, I have made another strategy, where I have included reservoir sampling buffer from the avalanche library.[7] The difference between one training cycle with previously mentioned conventional strategy is that, after training with first dataset, buffer preserves 5000 representative samples from that dataset. Then, before feeding second dataset to client for training, the samples in buffer is concatenated and shuffled with current dataset so that a replay of previous experience can occur during training period. By following this way, the weights are reforming in all layers of the model in such a way that the model doesn't forget what it learned in previous training round.

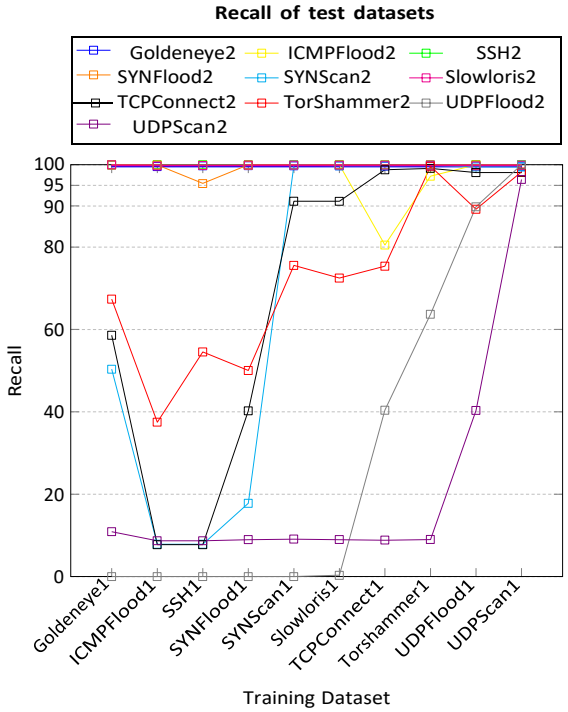
Substantial understanding can be achieved by meticulous observation of line graphs of continual FL strategy. According to the line graphs in Figure 5.7 and 5.8, the metrics values for each dataset stays low and fluctuating until the dataset comes into training phase, but when a particular dataset is fed to the model, the detection efficiency of that dataset goes approximately above 95% and from that phase, it continues to maintain high evaluation values even after the local models being trained on other datasets. Besides satisfactory evaluation results in train data, the models in both clients perform efficiently in detecting test data which are unknown to corresponding local models as depicted in Figure 5.5 and 5.6. Eventually, when the local models are trained with all types of DDoS attack datasets, due to the beneficiary effect of experience replay strategy, the weights in the models of both clients are set in such a way to make the local models competent enough to detect any type of DDoS attack along with genuine traffic with significantly high metrics values.



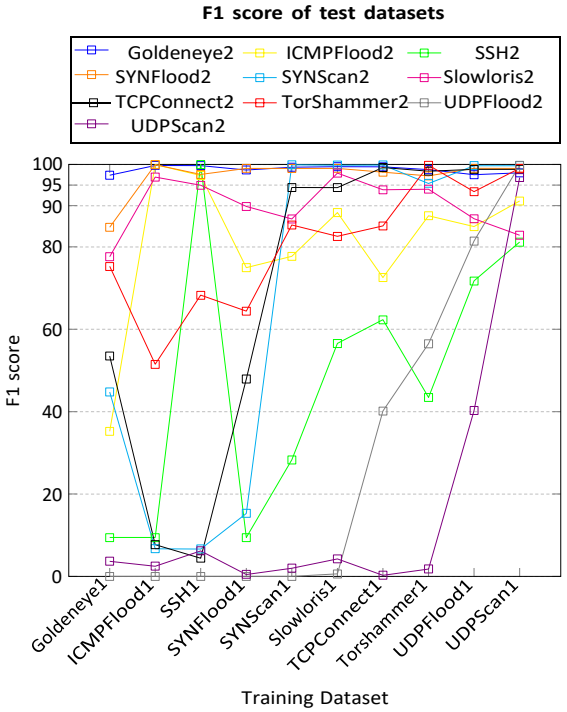
(a) Accuracy of test dataset in client 1.



(b) Precision of test dataset in client 1.

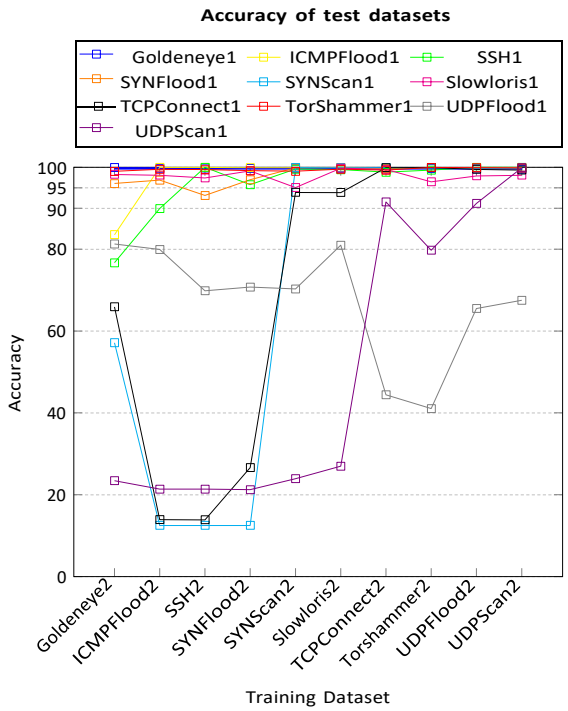


(c) Recall of test dataset in client 1.

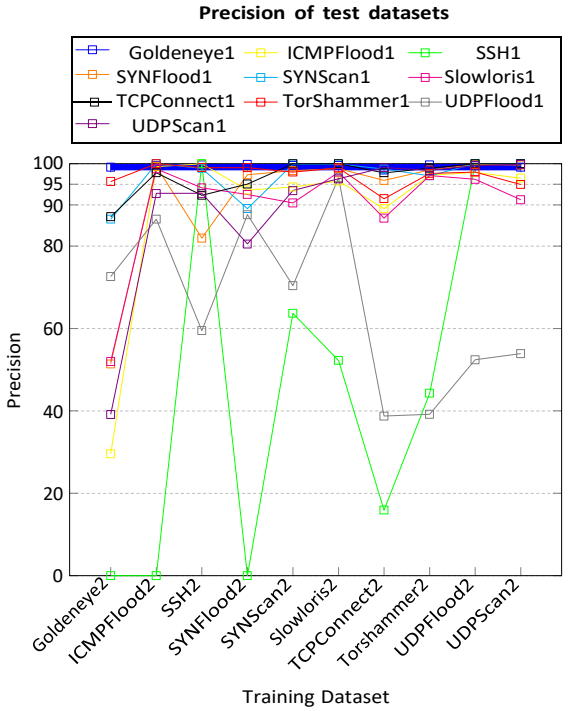


(d) F1 score of test dataset in client 1.

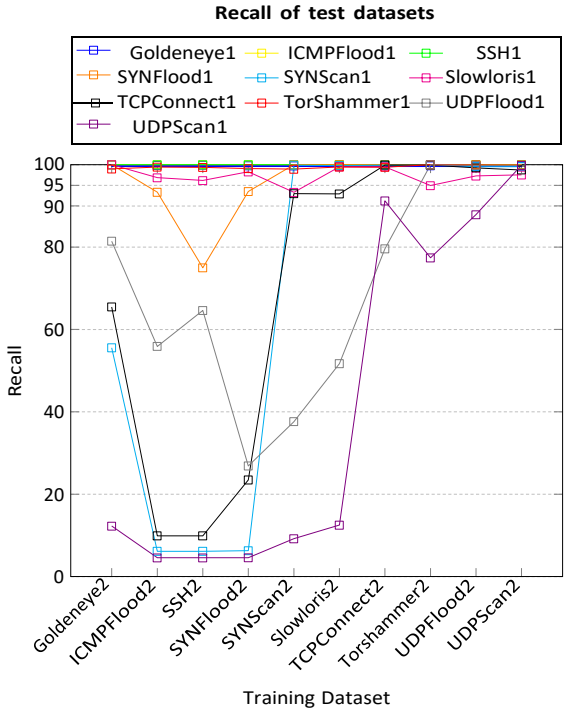
Figure 5.5: Performance evaluation metrics of the test dataset in client 1 (With Buffer).



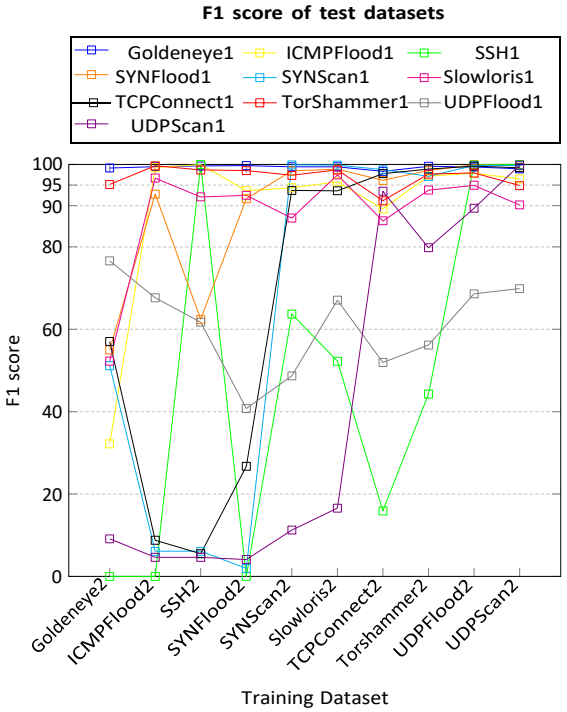
(a) Accuracy of test dataset in client 2.



(b) Precision of test dataset in client 2.

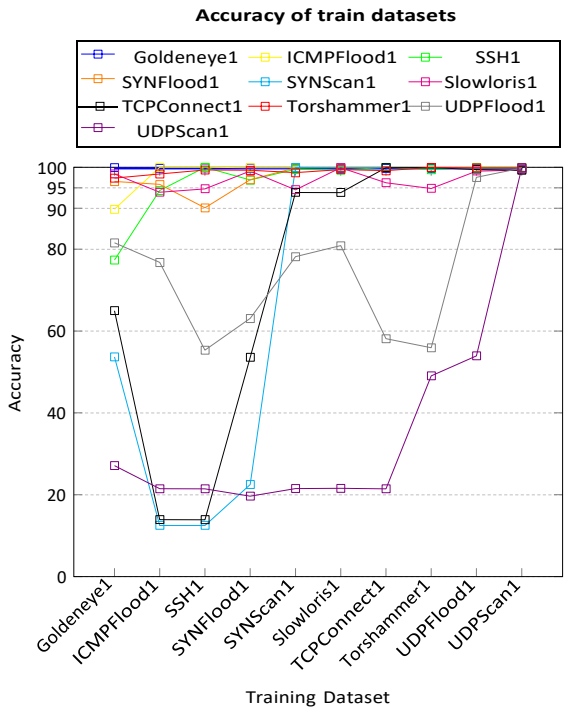


(c) Recall of test dataset in client 2.

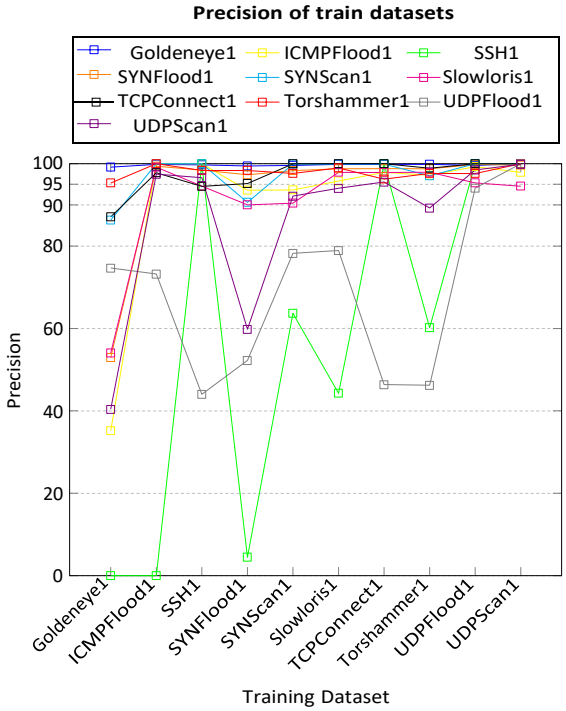


(d) F1 score of test dataset in client 2.

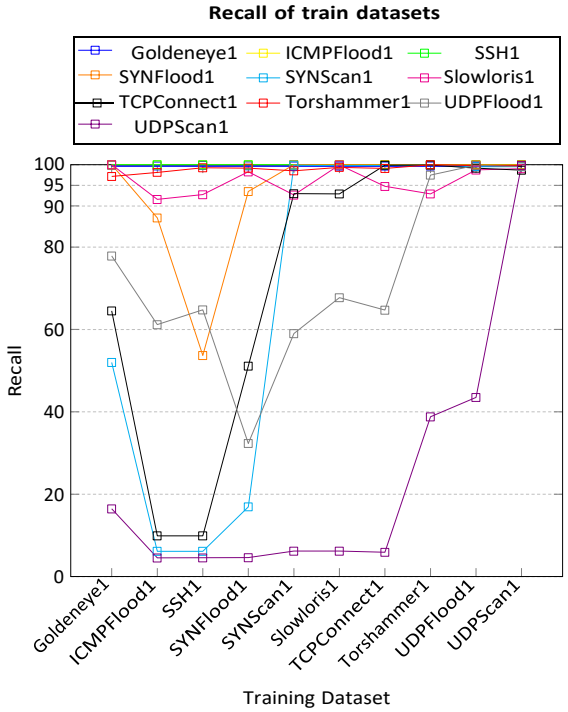
Figure 5.6: Performance evaluation metrics of the test dataset in client 2 (With Buffer).



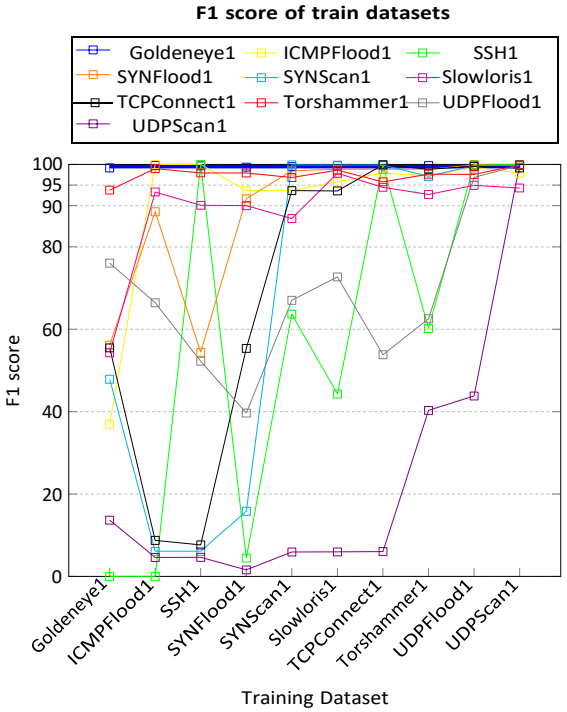
(a) Accuracy of train dataset in client 1.



(b) Precision of train dataset in client 1.

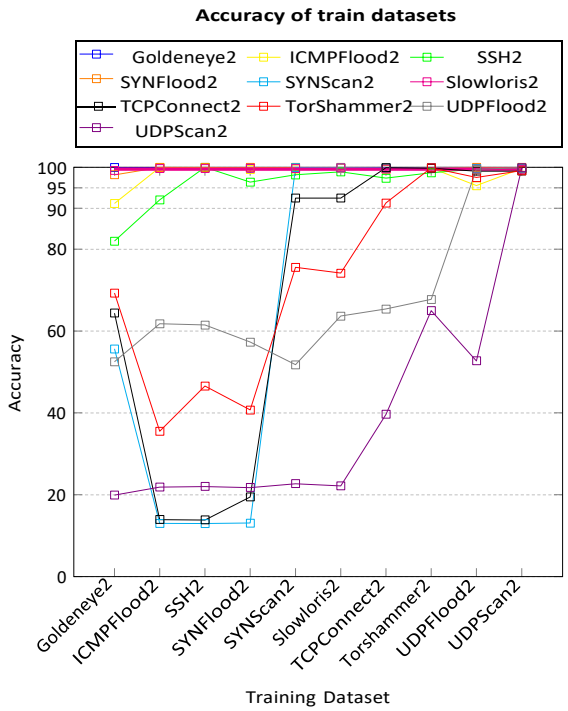


(c) Recall of train dataset in client 1.

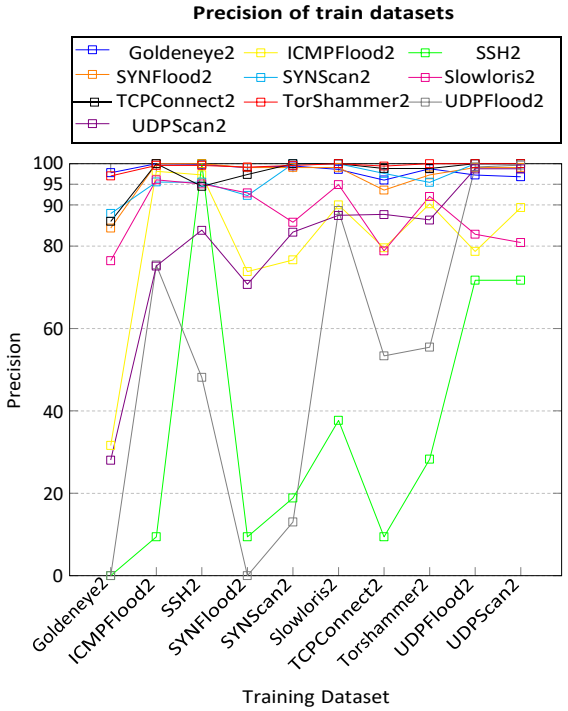


(d) F1 score of train dataset in client 1.

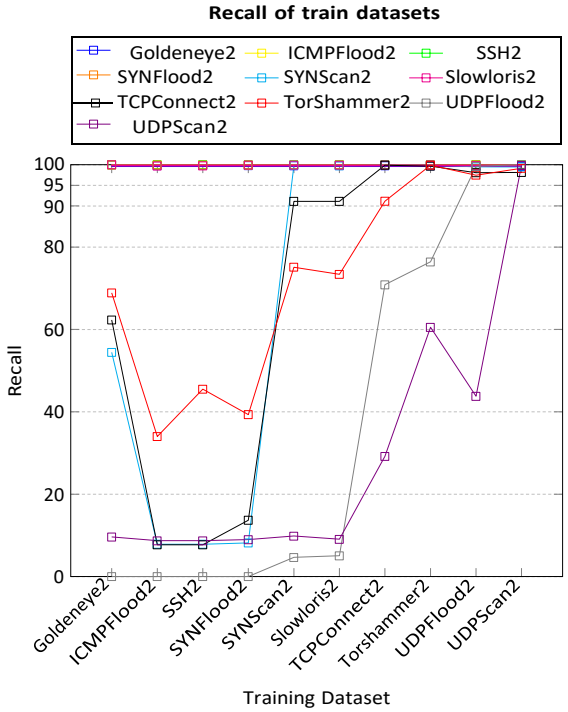
Figure 5.7: Performance evaluation metrics of the train dataset in client 1 (With Buffer).



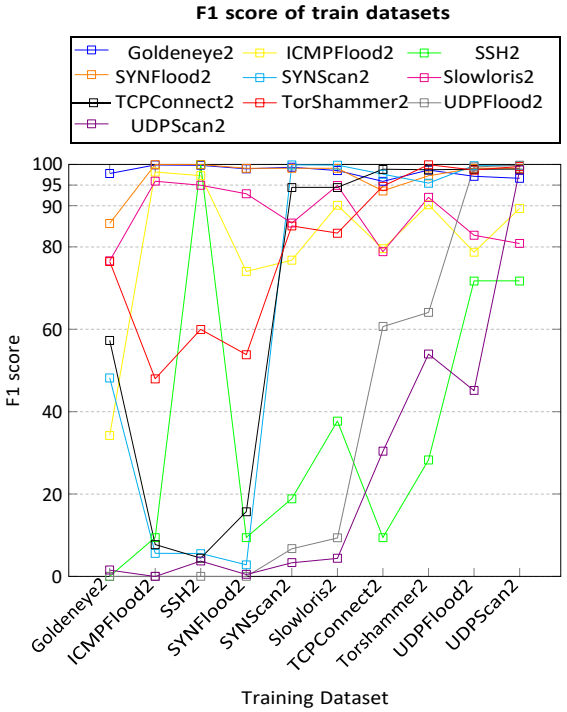
(a) Accuracy of train dataset in client 2.



(b) Precision of train dataset in client 2.



(c) Recall of train dataset in client 2.



(d) F1 score of train dataset in client 2.

Figure 5.8: Performance evaluation metrics of the train dataset in client 2 (With Buffer).

The essential benefit of this replay buffer strategy can be seen from the line graphs shown in Figure 5.5, 5.6, 5.6 and 5.8 and table 5.4, 5.5 and 5.6. According to the tables 5.4, 5.5 and 5.6, after having data samples in buffer, the accuracy in evaluation stage does not go significantly low which means the model retains considerable amount of knowledge from the previous data

points with the help of buffer itself. For instance, accuracy of Goldeneye 1, ICMPFlood 1, SSH1 in client 1 can be observed which are maintaining the mark of 99% with a negligible decimal value. Same goes for other datasets also. At the end of entire training process, both clients are detecting almost all test datasets with above 98% accuracy. However, there is an overhead cost in terms of training time and memory buffer size that comes with this solution. The buffer with 5000 samples posses 273.4375 MB in memory, If the buffer size is increased to 10000 and 20000 samples, it will take 546.875 and 1093.75 MB respectively. Moreover, from Table 5.4 and 5.6, it can be clearly seen that for comparatively large datasets, the solution with the buffer takes extra 5 to 7 seconds and for smaller datasets the system takes 2 to 3 seconds extra to process training phase and smaller datasets 2 to 3 seconds. Also for client 2, it takes even longer time than client 1. By giving dedicated hardware with increased resource configuration and schedule cleaning of the operating system resources, training time can be significantly reduced.

Table 5.1: Accuracy and Precision Tables for Client 1.

ACCURACY TABLE OF CLIENT 1

Evaluation Dataset		Training Dataset																			
		Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
		without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Train Dataset	Goldeneye1	99.67	99.92	59.04	99.74	18.85	99.86	64	99.77	74.79	99.7	87.79	99.71	74.59	99.71	98.51	99.72	67.99	99.63	50.9	99.6
	ICMPFlood1	87.2	89.76	100	100	96.65	100	91.11	99.91	52.67	99.91	97.68	99.95	25.6	99.97	82.8	99.94	91.64	99.98	40.44	99.97
	SSH1	75.12	77.36	89.3	94.34	100	100	90.86	96.89	54.85	99.5	87.19	99.32	53.67	100	73.63	99.44	76.93	100	66.54	100
	SYNFlood1	95.61	96.53	89.18	95.82	89.26	90.1	99.97	96.97	61.42	99.87	98.87	99.96	39.23	99.97	94.59	99.96	90.27	99.88	33.03	99.97
	SYNScan1	25.84	53.68	17.24	12.5	12.5	12.5	27.62	22.48	99.91	99.95	68.68	99.9	99.87	99.92	99.45	99.87	81.21	99.77	99.57	99.79
	Slowloris1	97.94	98.39	81.88	93.92	79.99	94.76	89.53	99.11	84.47	94.53	99.31	99.92	81.96	96.24	95.73	94.85	98.92	99.09	85.53	99.36
	TCPConnect1	30.99	64.97	18.8	13.91	13.94	13.89	25.83	53.56	95.56	93.88	74.67	93.85	99.36	99.93	99.65	99.95	82.03	99.49	95.89	99.3
	Torshammer1	99.7	97.37	51.86	98.37	21.6	99.36	54.25	99.28	65.04	98.68	88.63	99.45	84.23	99.18	99.97	99.97	68.12	99.84	68.92	99.84
	UDPFlood1	84.52	81.55	62	76.76	62	55.32	70.42	63.07	57.27	78.17	95.94	80.86	49.09	58.12	49.72	55.91	99.53	97.55	82.98	99.81
	UDPScan1	13.03	27.13	21.51	21.48	21.47	21.46	19.67	19.67	91.12	21.52	50.74	21.57	95.38	21.46	97.42	49.06	79.98	53.95	99.75	99.9
Test Dataset	Goldeneye2	99.88	99.93	57.94	99.66	30.92	99.8	69.31	99.79	75.74	99.77	85.43	99.77	76.67	99.71	98.72	99.71	68.68	99.71	51.47	99.67
	ICMPFlood2	89.86	92.41	99.91	99.99	96.25	99.97	98.79	99.54	48.64	99.63	82.33	99.76	28.34	96.14	73.62	99.32	75.7	96.61	27.48	99.91
	SSH2	78.13	82.4	89.91	93.45	100	100	92.49	96.1	55.67	98.82	88.81	99.19	54.71	99.56	73.86	98.9	81.3	99.48	65.76	99.63
	SYNFlood2	97.94	98.11	83.94	99.98	65.06	97.29	99.96	99.98	99.83	99.97	99.66	99.98	99.87	99.96	95.45	99.96	76.15	99.99	68.94	99.98
	SYNScan2	23.06	51.64	15.17	13.01	13.03	12.97	26.31	22.48	99.91	99.95	68.75	99.9	99.84	99.9	99.18	99.84	79.33	99.75	99.52	99.78
	Slowloris2	98.94	99.32	66.52	99.76	44.32	99.79	86.13	99.76	87.4	99.74	99.42	99.89	96.06	99.83	97.72	99.89	85.59	99.56	74.5	99.58
	TCPConnect2	28.46	60.7	17.19	13.93	13.93	13.85	24.69	44.73	97.86	92.5	74.79	92.49	99.35	99.47	99.55	99.54	82.79	99.07	97.79	99.08
	Torsahmmer2	54.82	67.8	9.23	38.96	5.32	55.41	16.69	51.2	66.05	76.01	56.12	73.28	73.96	75.82	99.67	99.63	52.15	89.46	59.73	98.31
	UDPFlood2	52.06	52.96	61.8	61.89	62	61.61	61.29	56.53	75.87	50.2	58.64	61.85	70.93	54.62	82.14	63.03	88.52	84.58	99.98	99.86
	UDPScan2	18.34	21.08	22.15	22.07	22.18	22.11	22.23	21.76	91.66	22.05	38.65	22.1	92.52	22.11	77.55	22.06	60.97	49.53	99.69	96.86

PRECISION TABLE OF CLIENT 1

Evaluation Dataset		Training Dataset																			
		Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
		without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Train Dataset	Goldeneye1	99.32	97.4	99.86	100	100	99.86	99.86	98.77	95.76	99.45	98.77	99.59	95.77	99.59	98.77	98.91	99.73	97.68	99.59	98.22
	ICMPFlood1	31.49	32.9	100	100	100	97.31	69.33	74.73	19.04	77.54	46.57	88.16	6.41	91.14	26.15	90.26	49.64	84.91	26.06	91.11
	SSH1	0	9.43	0	9.43	100	100	0	9.43	0	28.28	0	56.55	0	62.3	0	43.45	4.48	71.72	0	81.15
	SYNFlood1	50.17	83.44	95.13	100	100	100	99.1	99.09	53.45	99.08	82.58	99.09	29.36	98.19	50.75	97.28	63.21	99.09	19.63	99.09
	SYNScan1	80.18	87.86	100	97.78	100	96.62	100	92.72	100	99.97	90.84	99.97	100	99.97	93.16	95.48	99.96	100	100	100
	Slowloris1	51.47	77.49	90.71	97.09	100	95.05	94.49	89.83	71.55	86.79	85.09	97.95	64.7	93.89	46.53	94.06	88.31	86.96	86.38	82.91
	TCPConnect1	84.16	86	100	100	100	94.51	100	96.46	100	100	88.65	100	100	100	92.24	98.89	100	100	100	100
	Torshammer1	96.01	97.04	100	99.6	100	99.57	98.93	99.19	89.24	99.6	96.95	100	89.02	99.6	97.63	100	100	100	100	100
	UDPFlood1	98.59	0	100	87.46	100	60.06	99.95	0	46.29	0	99.51	72.37	39.45	40.36	42.92	51.09	99.96	74.69	87.9	99.63
	UDPScan1	7	33.85	100	88.82	100	90.06	73.98	70.71	87.74	83.25	75.6	87.45	88.99	87.16	92.58	86.96	98.34	97.52	100	98.76
Test Dataset	Goldeneye2	97.54	99.18	100	99.86	100	99.73	98.77	99.45	92.51	99.59	96.98	99.86	93.81	99.86	96.58	99.86	98.63	99.59	99.44	100
	ICMPFlood2	30.68	35.18	95.51	100	100	100	86.5	93.57	23.93	93.65	27.46	95.8	6.15	97.9	24.84	97.2	35.34	99.3	0.8	97.9
	SSH2	9.43	0	9.43	0	100	100	9.43	4.48	0	63.68	9.43	44.28	0	100	9.43	60.2	15.17	100	0	100
	SYNFlood2	84.93	52.97	99.01	99.41	100	98.46	97.28	97.32	92.73	98.34	88.71	98.8	93.65	98.81	83.57	98.8	82.51	99.82	95.46	99.68
	SYNScan2	81.96	86.31	100	100	100	100	100	90.62	100	99.93	90.37	99.93	100	99.93	91.65	97.07	97.62	100	100	100
	Slowloris2	69.49	54.05	92.75	99.16	100	94.57	93.88	90.01	71.28	90.44	78.75	97.92	75.43	97.89	69.57	97.93	92.91	95.35	92.92	94.56
	TCPConnect2	81.39	87.1	100	97.8	100	94.5	100	95.19	100	99.98	89.15	100	100	100	92.83	98.9	99.93	100	100	100
	Torsahmmer2	96.96	95.33	100	100	100	98.37	100	98.38	97.05	97.64	97.35	98.99	97.4	96.27	100	97.64	100	97.63	100	100
	UDPFlood2	0	74.65	76.7	73.21	100	44.02	41.63	52.23	61.18	78.23	26.83	78.92	56.67	46.36	68.02	46.21	76.79	94.07	99.95	99.99
	UDPScan2	19.99	40.37	96.27	97.45	100	96.6	94.62	59.79	88.17	92.11	68.18	94.02	85.15	95.54	86.27	89.23	88.15	98.47	100	99.86

Table 5.2: Recall and F1 score Tables for Client 1.

RECALL TABLE OF CLIENT 1

Evaluation Dataset			Training Dataset																			
			Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
			without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Train Dataset		Goldeneye1	99.66	99.95	58.4	99.54	17.58	99.73	63.26	99.76	74.5	99.73	87.53	99.73	74.27	99.66	98.49	99.65	67.3	99.68	50.01	99.49
		ICMPFlood1	100	99.97	100	99.97	83.19	99.97	100	99.97	100	99.97	100	99.97	100	80.51	100	97.22	100	99.97	94.29	99.97
		SSH1	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
		SYNFlood1	98.91	100	49.53	99.93	49.53	95.45	99.99	99.95	99.99	99.95	99.82	99.95	99.82	99.93	100	99.95	86.36	99.97	56.62	99.95
		SYNScan1	22.85	50.34	11.11	7.83	6.1	7.83	22.27	17.77	99.77	99.88	67.36	99.74	99.67	99.75	99.65	99.78	79.32	99.47	99.17	99.46
		Slowloris1	99.76	100	76.89	99.73	74.41	99.83	86.48	99.93	88.45	99.91	100	99.92	99.76	99.92	99.93	99.93	99.1	99.76	81.95	99.84
		TCPConnect1	30.39	58.58	15.15	7.73	9.87	7.73	22.66	40.22	94.72	91.12	73.82	91.11	98.7	98.77	99.76	99.1	80.73	98.09	94.92	98.1
		Torshammer1	99.7	67.35	46.44	37.47	13.37	54.53	49.35	50.02	61.96	75.54	87.37	72.47	82.91	75.35	100	99.62	64.28	89.18	65.07	98.24
		UDPFlood1	60.18	0	0	0	0	0	22.19	0	77.96	0	89.76	0.31	63.52	40.41	98.06	63.7	98.81	89.81	63.87	100
		UDPScan1	6.17	10.89	4.54	8.69	4.41	8.69	6.05	8.95	98.33	9.09	42.29	8.97	99.77	8.84	97.88	8.99	74.84	40.33	98.11	96.42
Test Dataset		Goldeneye2	99.83	99.93	51.83	99.73	21.14	99.86	64.89	99.76	72.87	99.7	82.23	99.7	72.67	99.71	98.33	99.71	63.54	99.62	44.21	99.59
		ICMPFlood2	99.97	100	99.97	100	80.51	100	99.97	100	100	100	80.51	100	100	100	99.97	100	80.51	100	80.51	100
		SSH2	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
		SYNFlood2	99.97	100	64.64	87.09	26.51	53.68	99.95	93.5	99.97	99.99	99.95	99.99	99.97	99.99	100	99.99	61.22	99.99	39.08	99.99
		SYNScan2	20.27	51.96	10.09	6.1	7.83	6.1	21.51	16.9	99.76	99.88	67.9	99.77	99.62	99.8	99.62	99.88	77.7	99.52	99.03	99.52
		Slowloris2	100	100	60.49	91.58	34.26	92.73	83.71	98.28	86.26	92.65	100	99.93	96.55	94.74	99.93	92.92	83.22	98.72	70.77	99.04
		TCPConnect2	25.16	64.49	11.15	9.87	7.73	9.87	19.13	51.08	96.99	92.96	72.48	92.89	98.48	99.86	99.67	99.91	80.52	99.16	96.72	98.66
		Torsahmmer2	54.11	97.17	7.04	98.16	3	99.28	14.8	99.19	65.57	98.54	55.11	99.38	73.47	99.12	99.66	99.99	50.97	99.85	58.48	99.77
		UDPFlood2	0	77.85	0	61.16	0	64.76	0	32.32	100	59	5.14	67.72	100	64.7	100	97.48	100	99.87	100	99.52
		UDPScan2	9.09	16.44	8.69	4.52	8.69	4.54	9.09	4.59	94.66	6.17	31.08	6.17	99.81	5.89	78.37	38.79	56.5	43.49	99.41	99.87

F1 SCORE TABLE OF CLIENT 1

Evaluation Dataset			Training Dataset																			
			Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
			without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Train Dataset		Goldeneye1	99.14	97.38	68.83	99.75	17.58	99.72	73.06	98.65	78.24	99.31	91.05	99.45	77.71	99.41	97.96	98.73	73.33	97.51	56.19	97.95
		ICMPFlood1	34.2	35.2	100	99.98	83.19	97.31	72	74.94	21.66	77.68	47.64	88.31	9.09	72.54	28.85	87.54	52.32	84.91	22.7	91.11
		SSH1	0	9.43	0	9.43	100	100	0	9.43	0	28.28	0	56.55	0	62.3	0	43.45	4.48	71.72	0	81.15
		SYNFlood1	53.23	84.77	46.66	99.96	49.53	97.6	99.11	99.07	56.41	99.06	83.33	99.07	33.66	98.15	54.09	97.25	57.59	99.08	19.22	99.07
		SYNScan1	20.32	44.77	13.25	6.72	6.1	6.66	24.25	15.31	99.88	99.92	66.06	99.85	99.83	99.86	93.34	95.41	80.07	99.72	99.56	99.72
		Slowloris1	51.67	77.62	71.41	96.95	74.41	94.97	83.91	89.82	65.9	86.78	85.31	97.92	64.87	93.87	46.74	94.02	87.87	86.84	70.93	82.84
		TCPConnect1	26.26	53.52	18.04	7.73	9.87	4.44	28.2	47.93	95.64	94.42	66.33	94.41	99.22	99.28	92.16	98.39	83.64	98.82	96.5	98.83
		Torshammer1	95.87	75.25	59.75	51.49	13.37	68.26	58.46	64.4	61.65	85.31	88.78	82.55	77.99	85.08	97.64	99.81	68.9	93.4	73.22	99.07
		UDPFlood1	74.49	0	0	0	0	0	35.9	0	57.91	0	94.32	0.61	48.49	40.15	59.56	56.48	99.38	81.41	73.77	99.81
		UDPScan1	0.57	3.68	4.66	2.48	4.41	6.21	2.56	0.49	89.83	1.99	40.42	4.25	91.23	0.28	92.57	1.8	77.46	40.27	98.4	96.81
Test Dataset		Goldeneye2	97.45	99.15	61.17	99.73	21.14	99.66	73.52	99.33	73.88	99.44	84.67	99.71	74.63	99.71	95.64	99.72	68.22	99.4	49.69	99.79
		ICMPFlood2	33.52	36.96	95.53	100	80.51	100	86.59	93.63	26.95	93.67	27.46	95.8	9.14	97.9	27.91	97.2	35.34	99.3	0.8	97.9
		SSH2	9.43	0	9.43	0	100	100	9.43	4.48	0	63.68	9.43	44.28	0	100	9.43	60.2	15.17	100	0	100
		SYNFlood2	86.4	56.18	75.38	88.54	26.51	54.41	97.25	91.69	92.72	98.44	88.88	98.82	93.64	98.83	84.83	98.82	58.48	99.9	37.21	99.69
		SYNScan2	15.02	47.87	11.37	6.1	7.83	6.1	23.46	15.82	99.88	99.9	65.26	99.84	99.8	99.86	91.71	97.1	76.33	99.75	99.48	99.75
		Slowloris2	69.64	54.31	60.28	93.34	34.26	90.11	82.29	90.02	61.77	86.89	78.82	97.89	73.63	94.47	69.64	92.73	80.87	94.95	67.92	94.31
		TCPConnect2	23.01	55.5	13.28	8.77	7.73	7.67	24.51	55.37	98.23	93.67	68.95	93.6	99.07	99.93	92.94	98.86	84.62	99.54	98.05	99.17
		Torsahmmer2	64.69	93.76	10.33	98.99	3	97.94	22.46	97.91	75.55	96.83	64.03	98.61	81.67	95.77	99.82	97.64	56.65	97.56	71.43	99.88
		UDPFlood2	0	76.04	0	66.41	0	52.22	0	39.64	75.76	67	8.49	72.69	72.19	53.8	80.83	62.54	86.76	96.85	99.98	99.75
		UDPScan2	0.66	13.67	8.69	4.62	8.69	4.65	6.97	1.6	86.14	5.94	26.66	5.97	87.66	6.06	74.9	40.32	50.23	43.81	99.69	99.86

Table 5.3: Training and Evaluation Time Tables for Client 1.

EVALUATION TIME TABLE OF CLIENT 1

			Training Dataset																			
			Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
			without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Evaluation Dataset	Train Dataset	**Goldeneye1**	492.42	483.73	463.44	504.7	462	474.53	549.85	478.19	515.25	477.02	508.84	622	516.73	478.48	520.76	504.33	480.17	493.95	495.17	520.3
		ICMPFlood1	99.8	84.13	94.08	76.61	89.51	79.27	116.07	76.03	108.97	78.26	107.97	82.87	142.04	79.05	91.72	79.47	126.93	78.07	103.18	79.27
		SSH1	24.07	14.86	11.21	11.58	11.18	10.75	14.14	11.17	12.47	12.39	11.86	10.86	12.77	16.36	11.74	12.09	11.56	15.77	15.26	11.21
		SYNFlood1	251.02	71.55	228.94	75.74	226.18	74.12	276.14	74.56	269.24	74.52	249.03	75.14	243.04	77.19	234.6	79.29	235.81	78.91	264.38	76.48
		SYNScan1	58.9	63.51	58.14	62.76	58.45	61.96	66.12	61.49	82.8	61.61	57.85	62.04	77.95	64.72	59.85	63.8	59.15	63.94	71.75	62.84
		Slowloris1	167.63	66.78	160.8	69.34	163.95	67.18	185.83	71.58	197.49	66.24	163.53	69.08	190.07	65.97	171.04	68.3	177.48	70.06	198.3	67.74
		TCPConnect1	63.87	60.48	58.91	67.32	59	62.03	66.84	69	73.07	61.34	58.92	63.62	67.93	66.81	70.28	63.18	74.81	65.59	95.65	62.76
		Torshammer1	221.95	167.34	193.89	176.46	197.47	170.7	228.58	183.86	223.8	166.24	197.43	179.18	225.73	182.05	261.28	181.39	238.04	175.87	208.41	177.49
		UDPFlood1	264.04	299.67	252.22	276.46	317.69	267.19	314.02	265.66	277.78	256.97	306.94	385.31	253.55	262.62	270.23	301.6	272.68	265.75	260.33	281.32
		UDPScan1	58.45	78.43	50.01	56.87	55.65	59.15	64.55	55.78	57.68	56.75	56.64	62.72	75.1	60.78	53.89	63.9	51.87	55.29	53.2	60.01
	Test Dataset	**Goldeneye2**	473.69	504.82	585.53	480.36	453.27	478.76	532.9	478.69	493.45	469.2	491.43	526.95	484.24	480.25	564.72	499.04	488.21	485.16	496.82	499.38
		ICMPFlood2	79.19	97.68	74.83	94.67	71.97	100.84	72.95	94.96	78.54	95.04	79.58	110.51	75.91	101.57	82.96	98.65	74.43	95.99	76.48	99
		SSH2	12.19	14.66	10.14	18.09	9.98	15.64	19.12	12.47	11.44	11.55	10.39	16.36	11.36	16.26	16.13	16.31	12.87	12.45	10.42	12.85
		SYNFlood2	78.09	236.69	69.08	241.6	70.59	246.7	79.26	236.41	74.22	230.67	73.86	277.82	77.82	262.67	101.48	244.75	84.34	237.44	81.7	249
		SYNScan2	61.68	60.36	63	67.91	58.06	61.99	64.55	63.4	60.75	60.22	62.54	101.41	59.93	65.55	70.21	62.66	79.35	60.9	69.38	62.37
		Slowloris2	69.06	173.59	66.58	175.17	62.66	169.2	77.38	171.88	70.67	165.11	65.47	224.29	67.8	192.22	93.36	171.38	85.69	173.26	72.34	173.46
		TCPConnect2	60.98	90.81	63.11	64.49	58.91	63.91	66.01	68.67	64.72	61.47	61.63	73.9	63.65	68.92	96.72	61.7	57.34	62.1	81.54	62.89
		Torsahmmer2	176.24	198.06	172.61	209.58	166.49	214.08	189.29	208.46	187.18	207.19	191.57	206.89	175.27	225.41	185.01	200.56	174.2	220.66	185.36	221.4
		UDPFlood2	366.75	256.55	273.4	271.42	251.12	282.19	314.11	262.72	275.12	262.22	321.63	292.37	276.38	281.22	293.17	271.57	255.9	297.59	278.36	270.01
		UDPScan2	68.52	73.64	59.72	56.42	53.29	55.27	69.41	55.1	59.51	54.82	66.78	68.82	57.85	58.38	66.18	56.44	72.02	55.76	61.94	56.57

TRAINING TIME TABLE OF CLIENT 1

Training Dataset	without buffer	with buffer
Goldeneye1	34.93	41.4
ICMPFlood1	6.54	11.7
SSH1	0.39	2.36
SYNFlood1	14.71	19.62
SYNScan1	2.93	7.13
Slowloris1	13.9	15.9
TCPConnect1	3.3	8.66
Torshammer1	13.2	17.19
UDPFlood1	17.99	29.59
UDPScan1	2.52	7.95

Table 5.4: Accuracy and Precision Tables for Client 2.

ACCURACY TABLE OF CLIENT 2

Evaluation Dataset		Training Dataset: Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
		without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Train Dataset	Goldeneye2	99.85	99.95	30.93	99.82	30.92	99.87	63.18	99.8	73.76	99.79	80.34	99.78	75.32	99.72	98.41	99.77	62.69	99.74	48.46	99.68
	ICMPFlood2	90.4	91.18	99.99	99.97	96.25	99.97	98.74	99.5	57.09	99.63	89.29	99.79	31.66	99.52	75.92	99.83	79.64	95.52	27.46	99.72
	SSH2	79.82	81.96	100	92.05	100	100	90.65	96.39	55.96	98.16	90.65	98.9	56.41	97.35	74.89	98.67	82.47	99.56	65.98	99.63
	SYNFlood2	98.08	98.18	65.06	99.99	65.06	99.97	99.97	99.98	99.81	99.98	99.77	99.98	99.87	99.74	95.46	99.96	73.5	99.99	67.48	99.98
	SYNScan2	21.26	55.63	13.03	12.98	13.03	12.96	22.25	13.06	99.91	99.94	42.1	99.9	99.77	99.86	99.18	99.87	75.48	99.75	99.52	99.78
	Slowloris2	99.03	99.26	44.3	99.73	44.32	99.73	86.24	99.83	87.21	99.72	99.59	99.87	94.39	99.59	97.77	99.86	79.72	99.57	69.67	99.51
	TCPConnect2	23.28	64.38	13.93	13.93	13.93	13.83	18.43	19.45	96.67	92.5	54.38	92.5	99.35	99.93	99.55	99.79	86.03	99.07	98.8	99.1
	Torshammer2	49.46	69.27	5.34	35.5	5.32	46.55	15.27	40.7	62.98	75.56	36.29	74.15	61.08	91.28	99.8	99.88	50.5	97.51	69.12	99.17
	UDPFlood2	52.36	52.48	62	61.78	62	61.46	61.51	57.29	75.06	51.76	57.24	63.65	71.3	65.38	85.3	67.73	99.87	99.54	100	99.87
	UDPScan2	18.49	19.91	22.18	21.86	22.18	22.02	22.25	21.76	87.75	22.7	24.94	22.16	91.66	39.65	77.13	64.98	62.09	52.72	99.69	99.89
Test Dataset	Goldeneye1	99.54	99.98	18.86	99.8	18.85	99.86	60.73	99.75	73.07	99.7	79.25	99.7	73.06	99.7	98.26	99.72	63.43	99.64	50.83	99.57
	ICMPFlood1	87.78	83.54	96.65	99.99	96.65	99.99	91.27	99.92	61.81	99.92	97.81	99.95	44.6	99.7	82.96	99.94	96.31	99.96	39.07	99.92
	SSH1	75.37	76.68	100	89.93	100	100	91.11	95.77	56.9	99.56	88.93	99.44	55.04	98.82	74	99.32	78.98	100	66.29	100
	SYNFlood1	95.11	96.05	89.26	96.88	89.26	93.13	99.98	96.97	69.77	99.87	99.14	99.97	58.52	99.7	94.58	99.96	93.86	99.88	30.86	99.97
	SYNScan1	23	57.14	12.5	12.5	12.5	12.49	22.87	12.49	99.91	99.95	44.25	99.9	99.82	99.92	99.45	99.87	78.97	99.76	99.57	99.79
	Slowloris1	97.92	98.28	79.98	98.08	79.99	97.42	89.16	99.16	89.64	95.06	99.33	99.79	98.36	99.5	95.84	96.47	95.62	97.92	83.22	98.09
	TCPConnect1	24.59	65.92	13.94	13.91	13.94	13.87	19.24	26.62	94.78	93.88	60.11	93.85	99.33	99.92	99.64	99.95	85.18	99.51	97.65	99.34
	Torsahmmer1	99.42	99.01	21.61	99.51	21.6	99.45	57.16	99.17	62.68	99.04	77.33	99.53	75.36	99.3	99.97	99.98	68.69	99.91	74.95	99.82
	UDPFlood1	69.12	81.29	62	79.94	62	69.86	66.25	70.76	71.09	70.3	65.71	80.99	51.88	44.44	50.16	41.06	96.82	65.53	55.04	67.52
	UDPScan1	14.61	23.44	21.5	21.38	21.47	21.38	21.11	21.25	91.02	23.94	32.99	26.99	95.9	91.56	97.56	79.76	82.96	91.21	99.75	99.9

PRECISION TABLE OF CLIENT 2

Evaluation Dataset		Training Dataset: Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
		without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Train Dataset	Goldeneye2	97.68	97.81	100	100	100	99.86	97.68	99.04	91.93	99.45	97.4	98.63	94.59	96.03	96.03	98.77	98.63	97.26	98.9	96.85
	ICMPFlood2	30.74	31.6	100	98.16	100	97.31	84.58	73.81	29.29	76.65	32.77	89.99	7.37	79.57	25.9	90.26	37.12	78.71	0.8	89.34
	SSH2	9.43	0	100	9.43	100	100	9.43	9.43	0	18.85	9.43	37.7	0	9.43	9.43	28.28	15.17	71.72	0	71.72
	SYNFlood2	85.03	84.37	100	100	100	100	98.19	99.09	92.73	99.09	89.8	99.09	93.65	93.59	83.6	97.28	80.75	99.09	95.46	99.09
	SYNScan2	81.99	87.91	100	95.56	100	95.51	100	92.29	100	99.97	87.41	99.97	100	97.64	91.65	95.48	97.62	100	100	100
	Slowloris2	70.52	76.44	98.99	96.08	100	95.07	85.71	92.89	72.25	85.77	83.86	94.91	77.5	78.82	69.62	92.04	93.93	82.87	92.91	80.85
	TCPConnect2	80.19	86.01	100	100	100	94.51	100	97.4	100	100	89.14	100	100	98.84	92.83	98.84	99.93	100	100	100
	Torshammer2	96.93	97.04	100	99.6	100	99.6	100	99.19	97.09	99.6	97.23	100	97.4	99.42	100	100	100	100	100	100
	UDPFlood2	0	0	100	75.42	100	48.13	54.69	0	60.82	13.02	0.51	88.63	56.99	53.35	72.1	55.47	99.65	99.2	100	99.65
	UDPScan2	20.03	28.05	100	75.16	100	83.85	96.27	70.71	88.26	83.39	59.43	87.45	86.15	87.67	86.62	86.34	89.37	98.76	100	98.76
Test Dataset	Goldeneye1	99.73	99.18	99.86	99.59	100	99.73	99.86	99.86	95.74	99.59	99.45	99.59	96.46	98.5	98.5	99.73	99.73	99.59	99.59	99.18
	ICMPFlood1	31.61	29.57	100	99.3	100	99.97	70.01	93.6	23.24	94.35	50.09	95.8	17.72	89.12	27.64	97.2	60.63	97.9	23.56	96.5
	SSH1	0	0	100	0	100	100	0	0	0	63.68	0	52.24	0	15.92	0	44.28	4.48	100	0	100
	SYNFlood1	50.58	51.41	99.71	99.41	100	81.95	99.12	97.32	62.86	98.34	85.4	98.81	49.3	95.96	50.76	98.5	69.31	99.82	17.6	99.68
	SYNScan1	80	86.53	100	100	100	98.88	100	89.1	100	99.93	86.58	99.93	100	98.71	93.16	97.07	99.96	100	100	100
	Slowloris1	52.26	51.97	99.59	98.73	100	94.13	90.74	92.52	79.34	90.47	85.92	97.92	81.71	86.77	46.97	97.1	88.31	96.18	83.49	91.24
	TCPConnect1	84.98	87.13	100	97.8	100	92.33	100	95.06	100	99.98	87.5	100	100	97.79	92.2	98.9	100	100	100	100
	Torsahmmer1	96.34	95.67	100	100	100	99.06	98.59	99.05	89.21	97.98	97.64	98.99	89.18	91.51	97.63	97.64	100	97.97	100	94.95
	UDPFlood1	95.92	72.63	100	86.56	100	59.52	99.84	87.64	60.69	70.42	95.91	96.61	39.86	38.75	42.77	39.19	99.95	52.43	43.44	53.92
	UDPScan1	16.03	39.14	100	92.78	100	92.78	87.25	80.51	91.53	93.47	58.81	96.34	89.7	99.06	92.64	98.37	99.48	99.51	100	99.86

Table 5.5: Recall and F1 score Tables for Client 2.

RECALL TABLE OF CLIENT 2

Evaluation Dataset		Training Dataset																			
		Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
		without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Train Dataset	Goldeneye2	99.74	99.98	21.15	99.77	21.14	99.83	57.87	99.77	70.33	99.74	76.64	99.75	70.88	99.74	97.97	99.73	57.01	99.72	40.97	99.56
	ICMPFlood2	99.97	99.97	99.97	99.97	80.51	99.97	99.97	99.97	100	99.97	80.51	99.97	98.13	99.97	99.97	99.97	80.51	99.97	80.51	99.97
	SSH2	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	SYNFlood2	99.97	100	26.51	99.95	26.51	99.92	99.95	99.95	99.97	99.95	99.95	99.95	99.95	99.95	100	99.95	52.96	99.97	34.61	99.95
	SYNScan2	18.34	54.42	7.83	7.83	7.83	7.83	17.41	8.15	99.76	99.84	39.32	99.74	99.5	99.87	99.62	99.88	73.85	99.47	99.03	99.45
	Slowloris2	100	100	34.26	99.7	34.26	99.74	84.09	99.93	85.97	99.91	100	99.92	94.26	99.93	99.93	99.93	76.29	99.8	65.15	99.85
	TCPConnect2	19.44	62.32	7.73	7.73	7.73	7.73	12.58	13.69	95.76	91.12	51.32	91.12	98.48	99.9	99.67	99.68	84.25	98.09	97.73	98.15
	Torshammer2	48.57	68.86	3.04	33.98	3	45.48	13.36	39.3	62.41	75.1	34.75	73.36	60.33	91.13	99.79	99.88	49.27	97.45	68.04	99.11
	UDPFlood2	0	0	0	0	0	0	0	0	96.69	4.64	0	5.04	100	70.85	100	76.41	100	99.62	100	100
	UDPScan2	9.09	9.6	8.69	8.69	8.69	8.69	9.09	8.95	89.47	9.82	13.89	9.04	97.43	29.18	77.8	60.52	56.17	43.77	99.41	99.81
Test Dataset	Goldeneye1	99.52	99.98	17.59	99.8	17.58	99.86	59.99	99.74	72.74	99.7	78.84	99.69	72.67	99.72	98.24	99.72	62.66	99.63	49.95	99.56
	ICMPFlood1	100	100	83.19	100	83.19	100	100	100	100	100	100	100	100	100	100	100	100	100	87.95	100
	SSH1	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	SYNFlood1	98.07	100	49.53	93.33	49.53	74.97	99.99	93.5	99.99	99.99	99.82	99.99	99.82	99.99	100	99.99	81.57	99.99	56.25	99.99
	SYNScan1	19.57	55.53	6.1	6.1	6.1	6.1	17.03	6.26	99.77	99.88	41.4	99.77	99.59	99.88	99.65	99.88	77.15	99.51	99.17	99.52
	Slowloris1	99.53	100	74.41	96.84	74.41	96.16	86.11	98.28	87.88	93.29	100	99.48	99.4	99.48	99.93	94.93	95.09	97.28	78.96	97.52
	TCPConnect1	23.56	65.45	9.87	9.87	9.87	9.87	15.6	23.43	93.86	92.96	58.69	92.89	98.64	99.91	99.76	99.91	83.93	99.23	96.82	98.71
	Torshammer1	99.37	98.98	13.37	99.43	13.37	99.36	52.33	99.07	59.35	98.93	74.89	99.46	73.09	99.39	100	100	64.92	99.91	71.72	99.84
	UDPFlood1	19.62	81.45	0	55.9	0	64.62	11.21	26.86	67.68	37.63	10.19	51.71	52.32	79.59	92.23	99.99	91.66	99.87	60.56	99.87
	UDPScan1	6.17	12.25	4.52	4.54	4.41	4.54	6.02	4.56	95.44	9.18	20.27	12.49	99.76	91.22	97.96	77.39	78.09	87.87	98.11	99.87

F1 SCORE TABLE OF CLIENT 2

Evaluation Dataset		Training Dataset																			
		Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
		without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Train Dataset	Goldeneye2	97.53	97.8	21.15	99.88	21.14	99.78	64.62	98.93	71.08	99.32	80.61	98.5	73.84	95.9	94.87	98.63	62.35	97.12	45.35	96.63
	ICMPFlood2	33.56	34.23	99.98	98.18	80.51	97.31	84.74	74.04	32.31	76.79	32.77	90.11	10.07	79.58	29.13	90.24	37.12	78.71	0.8	89.34
	SSH2	9.43	0	100	9.43	100	100	9.43	9.43	0	18.85	9.43	37.7	0	9.43	9.43	28.28	15.17	71.72	0	71.72
	SYNFlood2	86.46	85.69	26.51	99.97	26.51	99.96	98.16	99.07	92.72	99.07	89.88	99.07	93.62	93.59	84.85	97.25	50.52	99.08	32.66	99.07
	SYNScan2	13.55	48.19	7.83	5.6	7.83	5.55	20.43	2.82	99.88	99.9	35.3	99.85	99.74	97.64	91.71	95.46	72.32	99.72	99.48	99.71
	Slowloris2	70.66	76.59	33.25	95.93	34.26	94.94	74.24	92.87	62.6	85.76	83.9	94.89	74.03	78.82	69.67	92	76.36	82.8	63.43	80.81
	TCPConnect2	16.73	57.3	7.73	7.73	7.73	4.44	15.63	15.71	97.46	94.42	47.3	94.42	99.07	98.82	92.94	98.7	89.69	98.82	98.61	98.87
	Torshammer2	59.11	76.44	3.07	47.97	3	59.95	20.12	53.83	73.2	85.08	43.53	83.32	72.06	94.71	99.89	99.94	56.46	98.64	79.63	99.52
	UDPFlood2	0	0	0	0	0	0	0	0	74.51	6.74	0	9.36	72.44	60.66	83.66	64.08	99.82	99.41	100	99.82
	UDPScan2	0.66	1.55	8.69	0	8.69	3.73	6.97	0.49	82.3	3.34	7.92	4.4	86.99	30.39	75.16	53.98	52.33	45.13	99.69	98.66
Test Dataset	Goldeneye1	99.48	99.17	17.46	99.49	17.58	99.66	69.26	99.73	76.74	99.44	85.2	99.44	76.94	98.35	97.55	99.58	69.03	99.4	57.73	98.96
	ICMPFlood1	34.31	32.21	83.19	99.3	83.19	99.99	72.69	93.65	25.87	94.37	51.15	95.8	20.42	89.25	30.34	97.2	61.24	97.9	16.52	96.5
	SSH1	0	0	100	0	100	100	0	0	0	63.68	0	52.24	0	15.92	0	44.28	4.48	100	0	100
	SYNFlood1	53.11	55	49.24	92.83	49.53	62.41	99.12	91.69	65.14	98.44	85.91	98.82	52.54	96.13	54.1	98.53	58.82	99.9	17.13	99.69
	SYNScan1	17.27	51.18	6.1	6.1	6.1	6.1	19.93	1.94	99.88	99.9	38.25	99.84	99.78	98.74	93.34	97.1	78.76	99.74	99.56	99.75
	Slowloris1	52.35	52.24	74	96.69	74.41	92.13	80.48	92.51	71.25	86.98	86.14	97.46	81.65	86.31	47.17	93.78	85.02	94.95	64.59	90.17
	TCPConnect1	18.78	57.01	9.87	8.77	9.87	5.5	19	26.71	94.66	93.67	50.94	93.6	99.17	97.77	92.14	98.86	88.17	99.59	98.06	99.2
	Torshammer1	96.03	95.15	13.37	99.68	13.37	98.67	60.95	98.52	59.37	97.36	79.04	98.67	70.05	91.19	97.64	97.65	72.25	97.93	80.59	94.87
	UDPFlood1	32.21	76.62	0	67.66	0	61.74	19.83	40.77	63.77	48.7	18.15	67.08	45.04	51.95	58.28	56.18	95.58	68.61	50.38	69.88
	UDPScan1	0.64	9.14	4.62	4.65	4.41	4.65	6.28	4.1	90.74	11.27	18.64	16.59	91.63	93.52	92.65	79.81	82.34	89.39	98.4	99.86

Table 5.6: Training and Evaluation Time Tables for Client 2.

EVALUATION TIME TABLE OF CLIENT 2																					
		Training Dataset																			
		Goldeneye1		ICMPFlood1		SSH1		SYNFlood1		SYNScan1		Slowloris1		TCPConnect1		Torshammer1		UDPFlood1		UDPScan1	
		without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer	without buffer	with buffer
Evaluation Dataset / Train Dataset	Goldeneye1	505.03	480.67	467.22	490.91	453.1	513.48	523.94	476.72	559.88	480.63	505.6	488.94	454.46	490.87	488.77	511.54	481.72	598.06	466.12	520.71
	ICMPFlood1	82.76	78.06	72.93	79.71	75.49	81.63	79.04	76.52	84.22	79.41	74.95	81.98	73.71	81.1	78.52	88.32	77.95	101.81	72.95	81.67
	SSH1	15.62	11.23	10.72	21.39	10.34	16.5	11	12.8	11.54	14.54	10.4	11.79	14.34	11.99	11.1	16.74	21.27	14.34	11.28	12.83
	SYNFlood1	88.29	76.59	72.84	80.07	80.25	78.46	105.99	82.4	80.59	76.39	81.73	76.69	86.81	76.35	83.84	82.85	74.78	85.41	86.22	88.76
	SYNScan1	72.43	65.55	59.52	66.93	58.67	63.85	69.22	65.75	64.96	63.57	59.24	62.59	75.58	63.07	63.42	69.13	71.16	81.31	58.88	64.29
	Slowloris1	79.9	68.06	63.73	72.11	79.35	88.6	77.48	70.87	65.35	69.06	65.54	69.46	82.3	67.98	66.5	71.86	73.18	81.03	75.37	76.81
	TCPConnect1	63.31	61.58	68.98	65.45	71.94	77.51	67.8	69.67	63.73	62.91	62.34	62.64	70.18	63.13	68.59	65.56	60.3	68.78	57.67	68.57
	Torshammer1	175.6	213.67	180.45	178.32	208.44	205.85	172.64	176.2	181.67	172.72	172.53	191.03	175.65	174.21	160.16	184.93	173.65	193.03	166.99	187.23
	UDPFlood1	258.7	300.62	257.96	275.34	265.26	308.51	273.63	273.96	296.32	261.67	278.79	287.88	260.72	284.9	253.25	272.47	255.39	301.47	241.79	282.6
	UDPScan1	55.81	54.9	56.59	57.77	51.96	77.24	55.11	62.35	64.89	57.8	59.54	59.61	55.88	57.68	54.42	61	53.32	72.78	64.9	58.21
Evaluation Dataset / Test Dataset	Goldeneye2	515.73	492.95	459.58	490.01	501.38	553.97	488.25	458.51	463.3	479.86	466.39	540.55	465.78	547.88	485.37	540.34	460.9	532.76	476.44	512.84
	ICMPFlood2	109.5	98.12	86.16	98.57	96.9	122.19	95.38	95.67	95.43	101.57	91.52	106.35	94.89	111.2	93.47	117.74	94.95	111.36	93.97	104.13
	SSH2	14.83	16.27	12.34	13.16	12.03	14.95	12.86	12.19	17.73	12.7	12.68	14.7	11.82	13.69	12.06	14.12	12.69	14.07	15.72	21.3
	SYNFlood2	278.2	253.67	225.42	234.21	244.15	291.03	259.86	233.25	232.61	243.48	256.07	257.94	231.9	256.18	240.69	266.56	254.57	254.23	253.26	256.8
	SYNScan2	64.15	60.59	56.38	63.52	67.99	78.95	65.55	61.07	63.58	65.2	86.04	68.04	64.26	65.09	67	71.79	58.36	73.16	64.72	64.49
	Slowloris2	178.67	172.79	161.22	172.91	180.12	225.63	171.31	167.45	168.21	172.79	175.55	194.85	166.71	189.66	165.52	179.64	166.4	184.48	174.85	181.86
	TCPConnect2	61.33	66.18	58.52	64.57	64.13	72.71	65.11	62.42	64.3	63.97	71.8	68.12	59.68	67.59	56.78	71.58	58.08	67.66	86.57	68.35
	Torsahmmer2	204.08	202.49	189.52	206.87	209.01	227.35	223.32	201.24	212.16	199.67	201.86	226.2	193.9	250.18	190.95	223.34	212.89	223.53	217.47	230.39
	UDPFlood2	260.26	264.16	255.9	263.99	252.57	301.21	290.54	266.67	279.21	262.91	264.09	276.61	256.04	286.52	255.34	283.57	249.38	276.36	267.7	294.05
	UDPScan2	55.21	53.63	53.88	58.84	61.71	61.67	53.8	54.92	61.9	55	67.39	56.91	51.7	60.2	54.06	60.5	53.59	57.84	52.33	61.06

TRAINING TIME TABLE OF CLIENT 2		
Training Dataset	without buffer	with buffer
Goldeneye1	34.93	60.3
ICMPFlood1	6.53	9.91
SSH1	0.39	3.03
SYNFlood1	14.71	20.8
SYNScan1	2.93	6.5
Slowloris1	13.86	7.78
TCPConnect1	3.3	10.14
Torshammer1	13.2	21.1
UDPFlood1	17.99	25.42
UDPScan1	2.52	7.45

I have evaluated the pretrained local model against other dataset, named 'Application layer DDoS dataset' [29], consisting of DDoS attacks generated in a SND-based testbed using Hulk and Slowloris tools and amalgamated with genuine traffic flow. 70% of the dataset has been taken for training the pretrained models and 30% samples have been taken for testing purpose. The evaluation table of this experiment has been shown in Table 5.7 also at the end of this chapter. According to Table 5.7 for application layer DDoS dataset, both clients have reached either over or close to 99% in terms of accuracy, precision, recall and F1 score which validates the trustworthiness of the proposed continual FL setup for DDoS attack detection.

Table 5.7: Evaluation of Application layer DDoS dataset.

EVALUATION OF APPLICATION LAYER DDoS DATASET				
	Accuracy	**Precision**	**Recall**	**F1 score**
Client1	99.17	99.94	98.52	99.22
Client2	99.73	99.88	99.62	99.75

6 DISCUSSION

6.1 Comparative Analysis with Related Research

This thesis proposes a unique technique to anomaly detection within the context of 5G O-RAN architecture, based on FL and CL principles. Because the 5G O-RAN architecture is still in its early stages, there is a distinct lack of particularly built security algorithms that adapt to its unique requirements and constraints. Despite the fact that the technology is still in its infancy, the quickly expanding environment of 5G networks needs urgent and efficient solutions to protect these networks from any attacks. To fill this need, this thesis work made a proposition to create an anomaly detection system adapted to the 5G Open RAN architecture, harnessing the strong capabilities of FL. This innovative method, when paired with the highly sophisticated 5G architecture, has the potential to greatly improve network security, marking a watershed moment on the path to secure, robust, and dependable 5G networks.

Single node models could be a two-edged sword. On the one hand, they may be hacked, which poses serious security threats. However, they might not be able to recognize sophisticated attack patterns that target several nodes in a networked system. The adoption of more thorough and reliable models is required as a result to guarantee the highest level of network security. Additionally, these models' inherent amnesia might pose a serious danger to 5G networks. This problem may be resolved by combining FL with continual learning, opening the door for a more developed and reliable security layer. Recent research has not given this novel strategy much attention.

The thesis goes beyond the conventional use of machine learning algorithms in anomaly detection systems by incorporating FL and CL in a realistic networked configuration, in contrast to the works described previously in 'related works and limitations' chapter. Unlike synthetic datasets and simulation-based setups, this dissertation is based on real-world data, which increases the usefulness and application of the conclusions as well as the actual deployment method. Furthermore, this study employs strong machine learning models but goes a step further by including continual learning, minimizing the CF commonly observed in NN, an element that was not well addressed in the preceding research.

6.2 Assessment of Thesis Objectives

The main objective of the thesis is to develop a FL-based anomaly detector for 5G O-RAN architecture. The proposed model can be deployed as a security service in RIC unit of O-RAN architecture. The thesis also proposes how CL can be integrated with the anomaly detector that can preserve important weights of the model by replaying old samples. The performance of the proposed system has been measured considering three scenarios: the evaluation values of accuracy, precision, recall and f1 score metrics in normal federated setup, continual learning applied on top of FL setup in next stage and lastly by evaluating the pretrained model from the second stage for other DDoS datasets.

The number stays over 90% in the majority of situations, while the greatest recorded system accuracy is above 99%. Though the majority of the suggested algorithms shown in the literature study were successful in achieving acceptable accuracy, they skipped or downplayed the essential issue of CF. As a result, when compared to earlier studies, the suggested system accuracy can be deemed satisfactory. The improvement can clearly be seen with the implementation of second strategy where buffer has been added to training process of local models to preserve knowledge compared to traditional FL strategy. There is a positive improvement in all cases except the case

where the anomaly happens in the form of UDPFlood. Moreover, the proposed model gained reliability while evaluating with over 99% accuracy, precision, recall and F1 score on other datasets containing DDoS flow.

Therefore, it can be said that the proposed system performance is within the expected range. According to the observations, the detector becomes more efficient if training data adequately cover all possible attack scenarios encountered by the detector. With the increasing variety of flow patterns in a traditional FL setup, dependability suffers as it frequently fails to accurately identify a sufficient number of flow categories. This problem is addressed substantially in the second step, when previous data are replayed throughout the training phase, resulting in considerable metric improvements. Thus, it can be asserted that the suggested detection system has successfully achieved sustainability in reliability while adding an extra layer of security in the networked systems.

6.3 Future Research Directions

To proceed with further investigation of the proposed thesis work, below research directions can be chosen to make the threat detection system more secure, matured and trustworthy.

- **Examining Various Aggregation Methods:** FedAvg was mostly used in this study to aggregate the weights from local models. Other aggregation techniques, such as Federated Stochastic Gradient Descent (FedSGD), Federated Adam, and Federated Averaging Momentum (FedAM), might, however, produce results that are different. Future research may look into investigating these and other techniques within the framework of federated learning for 5G Open-RAN and evaluating their results against FedAvg.

- **Utilization of Various Machine Learning Models:** For the purpose of identifying DDoS assaults, the work detailed in this thesis made use of a MLP model. Other machine learning models that may be used for this purpose include Support Vector Machines (SVM), Random Forests, and sophisticated deep learning architectures like CNN or RNN. Future studies should concentrate on implementing and contrasting these various models in terms of how well they identify DDoS attacks within a FL environment.

- **Working with Different Communication Systems:** In the current study, clients and the aggregating server communicated via the RESTful API technology. Other communication techniques, such as gRPC, MQTT, or even peer-to-peer communication systems, might, however, improve the effectiveness of communication in FL systems. In the framework of 5G O-RAN, future research could take these other communication protocols into consideration.

- **Investigation of Prospective Other Datasets:** Although the present study concentrated on the 5G NIDD and CICIDS2017 datasets, more datasets from other domains or with various attack types may be explored in further research. This would put the robustness and generalizability of the federated learning models to the test once more. Additionally, artificial datasets might be created to test certain situations or uncommon attack kinds, enhancing the model's capacity to counter a variety of assaults.

- **Adoption of Privacy-Enhancing Techniques in System Communication:** A possible weakness in federated learning systems is the communication of weights between clients and the aggregation site. In the future, studies could incorporate privacy-enhancing techniques (PETs) into the weight exchange procedure to protect the confidentiality and

privacy of the weights to protect from sniffing. To assure safe and private model training and inference in FL frameworks, PTEs methods such as homomorphic encryption, secure multi-party computing, or differential privacy might be researched. FL would become more privacy-preserving as a result, making it more suited for use in delicate or privacy-sensitive applications.

- **Integration of Various CL Methods:**

  Another significant research direction would be to investigate different CF techniques to enhance the overall cost in terms of resource utilization and latency of the system. Either applying EWC or LwF strategy standalone or combining these strategies with different kinds of replay buffer techniques would be a subject of exploration. To reduce memory space while keeping samples in buffer, compression method can be applied in preserved samples which also adds the subject of investigating the effects of compression in model's performance. Moreover, Progressive Neural Network (PNN)[52] or Dynamically Expandable Network (DEN)[53] can be adopted and investigated to reduce the effects of CF.

7 SUMMARY

The junction of security automation, 5G O-RAN architecture, and FL are examined in this thesis, with an emphasis on how they might be used to reduce DDoS assaults. The necessity for advanced and automated security solutions within the framework of 5G O-RAN, an architecture that is becoming more and more crucial in contemporary telecommunication networks, is what motivates the research.

The thesis starts off by examining security automation and describing the vital function it plays in the 5G O-RAN. O-RAN, which has the potential to spur flexibility and innovation, has emerged as a crucial area of attention as 5G networks continue to develop. However, this also creates additional security difficulties, especially with regard to DDoS assaults. As a result, robust and automated security measures are necessary to guarantee the dependability and resilience of these networks.

The study offers FL and CL as viable remedies to solve these problems. These learning methodologies allow for collaborative ML while protecting privacy, with model training occurring locally and just aggregated updates being shared. This lowers the possibility of data leakage and maintains network effectiveness.

After then, the study is divided into two sections. A model is created in the first stage using standard FL. In order to address the problem of CF, the technique is progressed in the second stage with the development of a reservoir sampling buffer replay in conjunction with FL. 5000 samples from an earlier training dataset are buffered and mixed with the current training dataset at this step. Then, using a widely used federated averaging method called FedAvg, this dataset is utilized to update the weights of the local model.

Three VMs make up the actual configuration, with the third serving as a central server for aggregation and the other two functioning as distant clients. Total 20 datasets of real-world traffic collected from 2 base stations (10 from each base station) of 5GTN built in University of Oulu consisting of various DDoS attack patterns from the 5G NIDD dataset repository have been used for training and assessment of this method. The detection effectiveness of the pre-trained model on data from other dataset named 'Application layer DDoS dataset' [29] is also examined that is a combination of the DDoS attack flows produced in proprietary testbed and genuine samples from CICIDS2017 datasets.

The results of the thesis show that even when training datasets are provided sequentially, the suggested technique may retain adequate accuracy. Importantly, the technique successfully detects novel data patterns, highlighting its usefulness in actual 5G O-RAN situations. These findings highlight the promise of FL for creating automated, flexible, and reliable security solutions for 5G O-RAN architecture.

8 REFERENCES

[1] buffer. https://www.netscout.com/threatreport/.

[2] French R.M. (1999) Catastrophic forgetting in connectionist networks. Trends in cognitive sciences 3, pp. 128–135.

[3] Rolnick D., Ahuja A., Schwarz J., Lillicrap T. & Wayne G. (2019) Experience replay for continual learning. Advances in Neural Information Processing Systems 32.

[4] buffer. https://avalanche-api.continualai.org/en/v0.3.1/training.htmlreplay-buffers-and-selection-strategies.

[5] Yoon J., Jeong W., Lee G., Yang E. & Hwang S.J. (2021) Federated continual learning with weighted inter-client transfer. In: *International Conference on Machine Learning*, PMLR, pp. 12073–12086.

[6] Samarakoon S., Siriwardhana Y., Porambage P., Liyanage M., Chang S.Y., Kim J., Kim J. & Ylianttila M. (2022) 5g-nidd: A comprehensive network intrusion detection dataset generated over 5g wireless network. arXiv preprint arXiv:2212.01298 .

[7] avalanche.training.reservoirsamplingbuffer. https://avalanche-api.continualai.org/en/v0.1.0/generated/avalanche.training.ReservoirSamplingBuffer.html.

[8] buffer. https://www.5gamericas.org/wp-content/uploads/2019/08/5G-Security-White-Paper8.15.*pdf*.

[9] Sundqvist T., Bhuyan M. & Elmroth E. (2022) Uncovering latency anomalies in 5g ran-a combination learner approach. In: *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, IEEE, pp. 621–629.

[10] Arjoune Y. & Faruque S. (2020) Smart jamming attacks in 5g new radio: A review. In: *2020 10th annual computing and communication workshop and conference (CCWC)*, IEEE, pp. 1010–1015.

[11] buffer. https://www.imperva.com/learn/ddos/ddos-attacks/.

[12] Doriguzzi-Corin R. & Siracusa D. (2022) Flad: adaptive federated learning for ddos attack detection. arXiv preprint arXiv:2205.06661 .

[13] Klement F., Katzenbeisser S., Ulitzsch V., Krämer J., Stanczak S., Utkovski Z., Bjelakovic I. & Wunder G. (2022) Open or not open: Are conventional radio access networks more secure and trustworthy than open-ran? arXiv preprint arXiv:2204.12227 .

[14] O-ran. https://www.o-ran.org/.

[15] buffer. https://rimedolabs.com/blog/ran-intelligent-controller-ric-overview-xapps-and-rapps/.

[16] buffer. https://www.kdnuggets.com/2020/08/breaking-privacy-federated-learning.html.

[17] Parisi G.I., Kemker R., Part J.L., Kanan C. & Wermter S. (2019) Continual lifelong learning with neural networks: A review. Neural networks 113, pp. 54–71.

[18] Aljundi R., Babiloni F., Elhoseiny M., Rohrbach M. & Tuytelaars T. (2018) Memory aware synapses: Learning what (not) to forget. In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154.

[19] Brendan McMahan H., Moore E., Ramage D., Hampson S. & Agüera y Arcas B. (2016) Communication-efficient learning of deep networks from decentralized data. arXiv e-prints pp. arXiv–1602.

[20] Lopez-Paz D. & Ranzato M. (2017) Gradient episodic memory for continual learning. Advances in neural information processing systems 30.

[21] buffer. https://avalanche.continualai.org/how-tos/dataloading*buffersreplay.*

[22] Shin H., Lee J.K., Kim J. & Kim J. (2017) Continual learning with deep generative replay. Advances in neural information processing systems 30.

[23] Andrychowicz M., Wolski F., Ray A., Schneider J., Fong R., Welinder P., McGrew B., Tobin J., Pieter Abbeel O. & Zaremba W. (2017) Hindsight experience replay. Advances in neural information processing systems 30.

[24] Marfo W., Tosh D.K. & Moore S.V. (2022) Network anomaly detection using federated learning. In: *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, IEEE, pp. 484–489.

[25] Moustafa N. & Slay J. (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: *2015 military communications and information systems conference (MilCIS)*, IEEE, pp. 1–6.

[26] Zoghi Z. & Serpen G. (2021) Unsw-nb15 computer security dataset: Analysis through visualization. arXiv preprint arXiv:2101.05067 .

[27] Rashid M.M., Khan S.U., Eusufzai F., Redwan M.A., Sabuj S.R. & Elsharief M. (2023) A federated learning-based approach for improving intrusion detection in industrial internet of things networks. Network 3, pp. 158–179.

[28] Ferrag M.A., Friha O., Hamouda D., Maglaras L. & Janicke H. (2022) Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning. IEEE Access 10, pp. 40281–40306.

[29] Benza¨ıd C., Boukhalfa M. & Taleb T. (2020) Robust self-protection against application-layer (d) dos attacks in sdn environment. In: *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, pp. 1–6.

[30] Sharafaldin I., Lashkari A.H. & Ghorbani A.A. (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp 1, pp. 108–116.

[31] Ye N. & Li X. (2000) Application of decision tree classifiers to computer intrusion detection. WIT Transactions on Information and Communication Technologies 25.

[32] Tan X., Su S., Huang Z., Guo X., Zuo Z., Sun X. & Li L. (2019) Wireless sensor networks intrusion detection based on smote and the random forest algorithm. Sensors 19, pp. 203.

[33] Liao Y. & Vemuri V.R. (2002) Use of k-nearest neighbor classifier for intrusion detection. Computers & security 21, pp. 439–448.

[34] Li W. & Li Q. (2010) Using naive bayes with adaboost to enhance network anomaly intrusion detection. In: *2010 Third International Conference on Intelligent Networks and Intelligent Systems*, IEEE, pp. 486–489.

[35] Esmaily J., Moradinezhad R. & Ghasemi J. (2015) Intrusion detection system based on multi-layer perceptron neural networks and decision tree. In: *2015 7th Conference on Information and Knowledge Technology (IKT)*, IEEE, pp. 1–5.

[36] Samarakoon S., Siriwardhana Y., Porambage P., Liyanage M., Chang S.Y., Kim J., Kim J. & Ylianttila M. (2022), 5g-nidd: A comprehensive network intrusion detection dataset generated over 5g wireless network.

[37] Amalapuram S.K., Tadwai A., Vinta R., Channappayya S.S. & Tamma B.R. (2022) Continual learning for anomaly based network intrusion detection. In: *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, IEEE, pp. 497–505.

[38] Mohammadpour L., Ling T.C., Liew C.S. & Chong C.Y. (2018) A convolutional neural network for network intrusion detection system. Proceedings of the Asia-Pacific Advanced Network 46, pp. 50–55.

[39] Agrawal S., Sarkar S., Aouedi O., Yenduri G., Piamrat K., Alazab M., Bhattacharya S., Maddikunta P.K.R. & Gadekallu T.R. (2022) Federated learning for intrusion detection system: Concepts, challenges and future directions. Computer Communications .

[40] Momkute D., Žvinys K. & Barzdėnas V. (2018) Adapted anomaly detection for ran performance. In: *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, IEEE, pp. 1–4.

[41] Killick R. & Eckley I. (2014) changepoint: An r package for changepoint analysis. Journal of statistical software 58, pp. 1–19.

[42] Prasath S., Sethi K., Mohanty D., Bera P. & Samantaray S.R. (2022) Analysis of continual learning models for intrusion detection system. IEEE Access 10, pp. 121444–121464.

[43] Revathi S. & Malathi A. (2013) A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. International Journal of Engineering Research & Technology (IJERT) 2, pp. 1848–1853.

[44] Stiawan D., Idris M.Y.B., Bamhdi A.M., Budiarto R. et al. (2020) Cicids-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access 8, pp. 132911–132921.

[45] Javadpour A., Ja'fari F., Taleb T. & Benza¨ıd C. (2023) Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks. IEEE Transactions on Network and Service Management .

[46] Benza¨ıd C., Taleb T. & Song J. (2022) Ai-based autonomic and scalable security management architecture for secure network slicing in b5g. IEEE Network 36, pp. 165–174.

[47] Ma Y., Xie Z., Wang J., Chen K. & Shou L. (2022) Continual federated learning based on knowledge distillation. In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, volume 3.

[48] What is a denial-of-service (dos) attack? https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/.

[49] Denial-of-service attack - wikipedia. https://en.wikipedia.org/wiki/Denial-of-service$_{a}ttack$.

[50] Understanding denial-of-service attacks. https://www.cisa.gov/news-events/news/understanding-denial-service-attacks.

[51] Hale J., Scale, standardize, or normalize with scikit-learn. https://towardsdatascience.com/scale-standardize-ornormalize-with-scikit-learn-6ccc7d176a02.

[52] Rusu A.A., Rabinowitz N.C., Desjardins G., Soyer H., Kirkpatrick J., Kavukcuoglu K., Pascanu R. & Hadsell R. (2016) Progressive neural networks. arXiv preprint arXiv:1606.04671 .

[53] Yoon J., Yang E., Lee J. & Hwang S.J. (2017) Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 .

9 APPENDICES

Table 9.1: List of Packages.

Package	Version
Python	3.10.6
Flask	2.2.3
ipykernel	6.12.2
Keras	2.11.0
Jupyter Notebook	6.5.2
Numpy	1.24.2
Pandas	1.5.3
Scikit-learn	1.2.1
tensorflow	2.11.0
torch	1.13.1
Avalanche-lib	0.3.1