8-22-2022 11:00 AM

Exploring Artificial Intelligence (AI) Techniques for Forecasting Network Traffic: Network QoS and Security Perspectives

Ibrahim Mohammed Sayem, *The University of Western Ontario*

Supervisor: Haque, Anwar, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

This thesis identifies the research gaps in the field of network intrusion detection and network QoS prediction, and proposes novel solutions to address these challenges. Our first topic presents a novel network intrusion detection system using a stacking ensemble technique using UNSW-15 and CICIDS-2017 datasets. In contrast to earlier research, our proposed novel network intrusion detection techniques not only determine if the network traffic is benign or normal, but also reveal the type of assault in the flow. Our proposed stacking ensemble model provides a more effective detection capability than the existing works. Our proposed stacking ensemble technique can detect 90.4% and 98.7% cyberattacks with an f1-score of 90.0% and 98.5%, respectively. Our second topic proposes a novel QoS prediction model tested in a live 5G network environment. Compared to the existing work in this domain, our study is the first approach to conduct a large-scale field test in a 5G network to measure and forecast the network QoS metrics. More than 50 days of continuous data have been collected, cleaned, and used for training the deep sequence models to predict the 5G network QoS metrics such as throughput, latency, jitter, and packet loss. Our experiments demonstrate the effectiveness of predicting the QoS metrics using LSTM and LSTM Encoder-Decoder models, providing lower prediction errors of 14.57% and 13.75%, respectively.

Keywords: Intrusion Detection System, Cybersecurity, 5G, QoS metrics, Time-Series analysis, QoS Prediction, Artificial Intelligence.

Summary for Lay Audience

The recent advancement of communication technology, such as 5G, has opened up opportunities for next-generation digital services and applications in various sectors such as autonomous and connected vehicles, autonomous drones, smart grid, e-health, and many other smart-city applications. To ensure the uninterrupted operations of such networks, adequate cyber security measures against network intrusions and forecasting network health status are important to the Internet Service Provider (ISP) / network operators. In this thesis, a novel AI-based cyberattack detection methodology and a network quality of service (network health status) prediction framework have been developed and validated. These contributions are presented under two related topics: the first article introduces a network intrusion detection system that uses different deep learning models to detect cyberattacks. The second topic comprises two parts: the first part proposes a network QoS analyzer tool to collect 5G network QoS data, including throughput, latency, jitter, and packet loss, from a live 5G network; and the second part presents a novel network QoS prediction strategy by utilizing deep sequence models. Our proposed models are expected to assist the network planners and operations team in ensuring their committed service level agreement (SLA) to their customers.

Acknowledgments

Without the support of kind and helpful people, it is not always easy to continue research while transitioning to a new country with a diverse culture. In this quotation, I want to show my thanks to those kind-hearted individuals who helped me greatly throughout my master's journey.

I want to start by thanking ALLAH (SWT) for guiding me and allowing me to overcome all the challenges.

I would like to acknowledge and express my gratitude to my supervisor, Dr. Anwar Haque, who made this work possible. Thank you for your continuous help and support, which works as a catalyst for me to grow as a researcher in the exciting field of intelligent 5G networks and the network security field. Throughout this journey, I found him eager to assist and listen to my problems and advise me with any questions I had.

My research partner and a PhD student of Dr. Haque, Sadia Yeasmin, deserve my sincere appreciation for supporting me throughout my research journey. I would also like to thank Amreen Anbar, an MSc student from Dr. Haque's lab, for her participation in the 5G data collection process.

I want to thank my parents, elder brother, and younger sister for their support. You always helped me out when I felt down, and without your support, it would not have been possible for me.

I would also like to express my appreciation to all the professors with whom I worked as a Teaching Assistant (TA), from whom I learned much in undergraduate teaching.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Next-generation networks, such as 5G, is intended to provide high throughput, ultra-low latency, and high availability to ensure a better service experience for its end users. To ensure the uninterrupted operations of such networks, adequate cyber security measures against network intrusions; and insight into the current and future network QoS status /metrics are essential for the Internet Service Provider (ISP) / network operators as the network security and network Quality of Service (QoS) are directly related to the overall performance of the network. Recent advancements in Artificial Intelligence (AI) techniques and tools can be leveraged to analyze, classify, and detect/predict cyber security threats and network QoS status in real-time. In this thesis, a novel AI-based cyberattack detection methodology and a network QoS prediction framework have been developed to equip the network with such capability. In this regard, we proposed an intelligent network intrusion detection system (NIDS) and a QoS prediction system. These contributions are presented under two related topics: the first topic introduces a network intrusion detection system that employs a novel stacking ensemble technique comprised of different deep learning models to detect cyberattacks, including DoS, DDoS, shellcode, web attacks, etc., utilizing two well-known network traffic datasets, namely, UNSW-15 and CICIDS-2017. The second topic comprises two parts: the first part proposes a network QoS analyzer tool to collect 5G network QoS data, including throughput, latency, jitter, and packet loss, from a live 5G network; and the second part presents a novel network QoS prediction framework by utilizing deep sequence models Our proposed novel methodologies, techniques, and tools directly contributes in network intrusion and QoS prediction. Our developed framework is expected to assist the network planners and operations team in ensuring their committed service level agreement (SLA) to their customers.

1.1 Research Contributions

The overall research contributions are summarized in the following chapters:

- **Chapter 3:**

1. Proposed a multi-class network intrusion detection system (NIDS) by utilizing two network traffic datasets, namely, UNSW-15 and CICIDS-2017;
2. Employed several data preprocessing strategies, including feature selection and data resampling techniques for imbalanced classes;
3. Employed a stacking ensemble technique comprised of distinct deep learning models in both layers of the stacking model;
4. Evaluate and compare the performance of the proposed stacking ensemble technique with other state-of-the-art single deep learning models.

- **Chapter 4:**

1. To collect network QoS data from a live 5G network platform, a 5G network QoS data collection tool is developed in a controlled environment;
2. Generated a dataset that contains different QoS metrics of 5G networks, including throughput, latency, packet loss, and jitter;
3. Performed autocorrelation and data windowing techniques to reveal the hidden patterns of the data and employed several deep sequence models to predict the throughput;
4. Evaluate the performance of the deep sequence models in different time frames in terms of MSE, RMSE, MAE, and WMAPE errors.

Chapter 2

Background

This section introduces background concepts that are necessary for comprehending the various technical concepts in the field of network intrusion detection, 5G network QoS prediction and associated AI techniques.

2.1 Basics of Intrusion Detection in Network Security

A network intrusion detection system (NIDS) is a form of network protection management system that detects potential security breaches in computers, or any devices connected with networks. While certain NIDS can respond to security threats once they are identified, attackers constantly devise new ways to steal or obliterate data. Moreover, it is essential to discover security-related issues before they cause data loss or service disruption.

2.1.1 How Network Intrusion Detection System (NIDS) Work?

The Network Intrusion Detection System (NIDS) is responsible for detecting any abnormalities that may represent a network threat. An IDS looks for attack signatures or deviations from regular network activity by analyzing the forwarded network traffic. Then it looks at the attack patterns to determine if there's anything harmful going on. Examining traffic patterns can be done in a variety of ways, including address matching, TCP/UDP port matching, packet anomaly detection, and so on [1]. Finally, all activities are reviewed at the protocol and application layers to assess the presence of an attack. A NIDS is different from an Intusion Prevention System (IPS), which diagnoses network traffic flows and searches for possible known attacks by comparing traffic flows to its internal signature database, dropping adversary packets, and preventing the attacker's IP address or port from being exposed.

2.1.2 History of Cyber-Attacks

Cyberattacks have a long history and are growing in number and becoming sophisticated every day. The first attack was a worm designed by Bob Thomas that infected DEC PD-10 machines using the Tenex operating system in 1971 when most people didn't even own a computer (OS) [2]. In 1988, a graduate student of Cornell University attempted to quantify the scale of the internet users by creating a software that could crawl the web, install itself, and report the total number of copies it produced on users' computers. Unfortunately, it was installed on the user's computer many times till it crashed [3]. In 1995, a radio station called LA KIIS FM hosted a radio contest that was rigged by contestants who used the phone network to cheat [4]. In 1996, Panix, a New York-based internet service provider (ISP), was the first victim of a Distributed Denial of Service (DDoS) attack that was initiated by an SYN flood attack. However, a major cyberattacks occurred in 1999 on a university's private network that completely paralyzed the system for over two days with a massive UDP flood [5]. The biggest cyberattack ever reported was in 2002, which targeted 13 Domain Name System (DNS) root servers and all of these servers were the primary connection for almost all internet communications [6]. In 2013, Yahoo announced that 1 million users' profiles had been compromised due to the “state-sponsored” attack [7]. During the previous two decades, network intrusion along with many other network security concerns has become a more common disruption of the worldwide Internet. With some of the most enormous attacks ever reported, especially in Covid-19 lockdown. With peak congestion of 2.3 Tbps in June 2020, Amazon Cloud Platforms safeguarded off the world's largest-ever Cyberattack in history.

2.1.3 History of Intrusion Detection System

The need for IDS increased as business network access created a new issue of granting access to the user while monitoring their actions for safe and secure operations. Anomaly detection and exploitation detection are two methods for detecting intrusion, where anomaly detection inspects users' activities and tries to differentiate them from normal behavior, the exploitation detection examines the attack signature to identify the known attacks [9]. In 1967, Bernard Peters and Willis Ware of the National Security Agency (NSA) and RAND Corporation recognized the necessity for intrusion detection systems while dealing with the federal government's classified papers housed in time-sharing settings [10]. James Anderson initially proposed the concept of intrusion detection in 1980, when he proposed the idea of monitoring user activity and tracking usage [11]. The major adaption of intrusion detection systems started in the late 1980s, but they can only alert to known attacks; however, scanning and updating the attack signature lists needed a lot of resources. Dr. Dorothy Denning, a scientist at SRI International, proposed the Intrusion Detection Expert System or IDES in 1983 as an expert system for detecting intrusion. The IDES system's second component was a rule-based method for encoding known attack signatures [10]. By the 1990s, her research had paved the ground for the development of a real-time, fully functioning intrusion detection system that consisted of four components: a Realm Interface, an Anomaly Detector based on a statistical approach, an Anomaly Detector with Intelligence, and a graphical user interface [10]. The IDS system gained popularity among users from various backgrounds in 1997, and the commercial development of the system was launched by the end of that year. Later, researchers used statistical methods for detecting intrusions in the network. Nowadays, various machine and deep learning techniques are introduced to detect network intrusion on a large scale.

2.1.4 Types of Network Intrusion Detection systems

By considering the use case and budget, a NIDS may be classified under four distinct types, ranging from security software to a hierarchical system that continually monitors network traffic. Furthermore, there are two ways of detecting intrusion detection based on the detection strategy. Figure 2.1 depicts the various forms of NIDS based on the data source and detection mechanism.

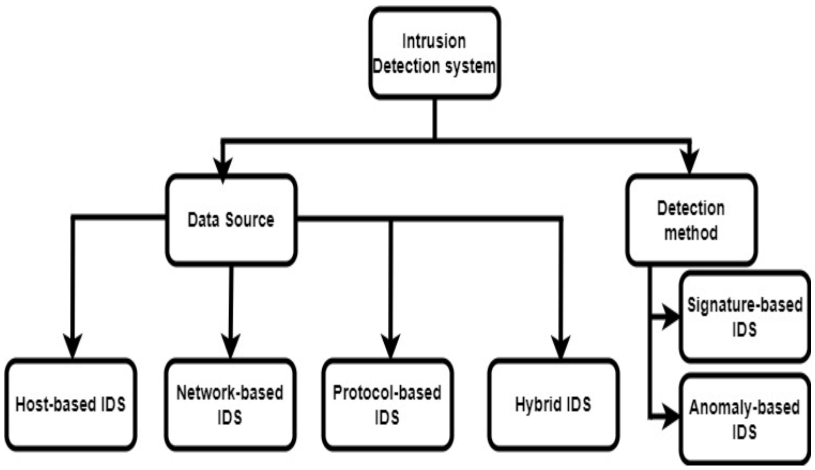


Figure 2.1: IDS Taxonomy

2.1.5 Network-Based Intrusion Detection System (NIDS)

The most prevalent genre of IDS is the intrusion detection system developed based on the network flow, which is implemented at tactical places throughout the network. The basic task of NIDS is to monitor subnets and compare traffic to discover illegal, restricted, and unusual network activities. It works passively while analyzing the incoming traffic and uses a network tap, span port, or hub to collect the packets, and when it finds any abnormality in the traffic, an alert is sent to the administrator. It passively examines incoming traffic and collects packets using a network tap, port span, or hub. If it can detect any abnormalities in the traffic, it sends an alert to the administrator. IDS functionality is passive, in that it does not immediately restrict network traffic; instead, it maintains track of the traffic's analytical data. However, maintaining an IDS is expensive.

2.1.6 Host-Based Intrusion Detection System (HIDS)

Host-based intrusion detection systems or HIDS are device-centric, they monitor the device's both incoming and outgoing traffic and warn the administrator if any adversary packet activity is noticed. HIDS employs a variety of heuristics, rules, and signatures to detect unwanted network traffic behaviour. It also keeps an eye on the system log files, taking snapshots and comparing them to the prior file system to examine file location. The main disadvantage of this system is that it is useless for large businesses since it only supports one device and is more susceptible than NIDS.

2.1.7 Protocol-Based Intrusion Detection System (PIDS)

The PIDS stands for protocol-based intrusion detection systems, which are configured in the server's front end, is used to monitor the HTTPS and HTTP servers. PIDS is responsible for supervising and analyzing user-device communications and delivering a secure web server. When a PIDS is deployed on a group of servers, it becomes an application-centric system that is capable of giving service to a specific set of devices.

2.1.8 Hybrid Intrusion Detection system

The hybrid intrusion detection system or HIDS is the combination of multiple IDS, to acquire a complete overview of any system's network and its traffic flow, it integrates the host or system with network traffic information. Its effectiveness surpasses that of other IDS.

2.1.9 Signature-Based Intrusion Detection System (SIDS)

In computer network terminology, a signature is referred to as a pattern or footprint which contains information about system activity. SIDS stands for signature-based network intrusion detection system, and it searches for signatures, patterns, or any other form of identity linked with a certain event. Misuse detection is another term for it. For example, to detect an intrusion, it compares network activity to log data and matches the log file for any known patterns. It's also known as a knowledge-based system since it draws inferences from network traffic data and then takes action based on those decisions [13]. On the other hand, the signature-based intrusion detection system can only detect existing attacks; new forms of attacks are impossible to detect because no signatures exist for them.

2.1.10 Anomaly-Based Intrusion Detection System (AIDS)

AIDS or anomaly-based intrusion detection system detects unknown threats in network traffic. Rather than looking for a specific pattern in the malicious packet, it monitors and analyses current traffic. The backbone of AIDS is machine learning (ML) and knowledge-based and statistical techniques, and it works as a baseline for predicting system behaviour [14]. ML-based methods performed exceptionally well because these models are trained so that, when they find anything that does not reside in the model, they declare it as a malicious pattern. AIDS protects the system from traffic and protocol anomalies. In terms of reliance, attack detection AIDS is better than SIDS and frequent database update is not required. However, managing AIDS is a complex task and the false positive rate is significant in this approach.

Because attack types and scenarios are changed so frequently, detecting attack patterns is always a challenge for IDS. Furthermore, IDS is unable to alter attack signatures or develop new signatures to predict the new variant. In this scenario, ML and DL-based algorithms can be effective in developing IDS; nevertheless, the number of false alarms, low recognition rates, imbalanced datasets, and long reaction times pose challenges.

In this thesis, we attempt to develop a system that focuses on the limitations and flaws of existing IDS. To detect anomalies and their signatures, we proposed a model that is trained with the most recent network traffic and a variety of attack types in orto Our model will employ new datasets to address flaws such as false-positive rates and low detection rates.

2.2 Cyber-Attacks: What They Are and How They Work

A cyberattack, in network security terms, is an intentional attempt by a person or organization to disable a victim's network and steal data in order to profit from the disruption of the victim's network. Cybercriminals are those who engage in cyberattacks. They first investigate computer system weaknesses, and if any are identified, they initiate cyberattacks for financial or personal benefit. Nowadays, cyber warfare between opposing countries is quite widespread, with hackers from one country attempting to damage financial, military, and other government functions. More people are using cellular networks since the launch of 5G, giving attackers a larger attack surface. According to the CSIS (Center for Strategic and International Studies), some of the largest cyberattacks is reported in several huge IT companies such as SolarWinds, Amazon, Twitter, and Microsoft occurred in recent years [15].

2.2.1 Common Types of Cyber-Attacks

1. **DoS (Denial of Service):** DoS attacks deplete the resources and bandwidth of systems, servers, and networks by sending hundreds of illegitimate requests, preventing legitimate users from accessing the servers, and resulting in service interruptions. A DoS attack does not result in data theft, and hackers do not demand monetary compensation from organizations; instead, it costs time and resources to restore service. DoS attacks can be carried out in two ways: flooding the target system or service and exploiting flaws that force the target system or facility to fail. ICMP floods, SYN floods, and buffer overflow assaults are all common floods [16]. A Distributed Denial of Services (DDoS) assault is related to a DoS attack in that it utilizes several compromised devices to launch the attack rather than a single device. Proxy servers, firewalls and switches protect a system or server against a

small DoS and DDoS attack assault. DDoS attack can have many variations in different attack scenarios.

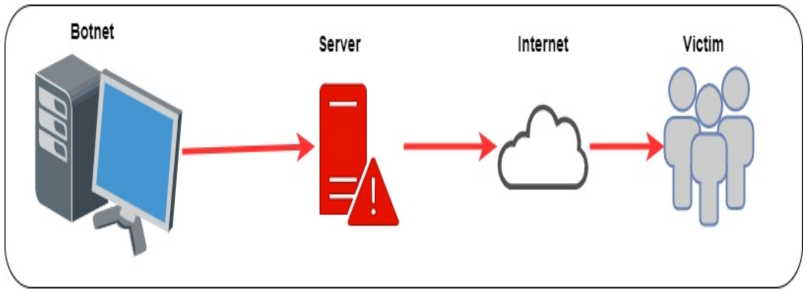


Figure 2.2: DoS Attack Strategy

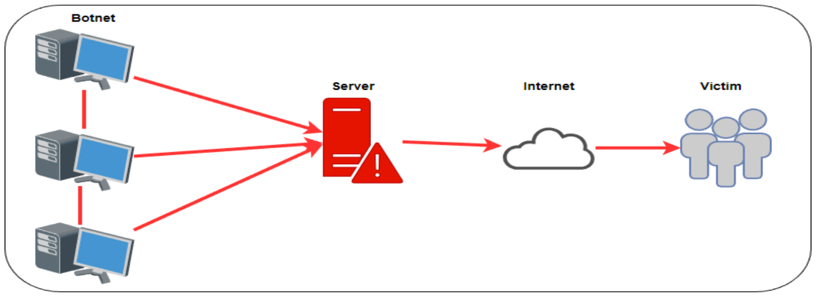


Figure 2.3: DDoS Attack Strategy

2. **Fuzzers:** In this attack, attackers put the system under stress by supplying arbitrary data as inputs that cause unexpected behaviour, resource leakage, and a variety of error codes. Fuzzers are typically used by system development and cyber security teams to detect system bugs, but hackers have used them to obtain system information, which leads them to retrieve important information from the system to do malicious activity [42]. Machine learning is now a popular method for cybercriminals to discover vulnerabilities in a system. Dumb fuzzers, smart fuzzers, evolutionary fuzzing attacks, and mutation-based fuzzing are the commonly used fuzzing attacks deployed by the attacker.

3. **Shellcode:** Shellcode uses a collection of commands and delivers them to a remote shell to take control of a compromised machine, which is essentially identical to exploits. Shellcode uses a collection of commands and delivers them to a remote shell to take control of a compromised machine, which is essentially identical to exploits [17]. Shellcode attacks can be mitigated with security solutions that have

a machine learning method in the back that can learn and predict the behaviour of a shell command.

4. **Reconnaissance:** The term "reconnaissance" is frequently used in the military to describe an assignment to obtain intelligence from a target to help uncover a security flaw. It's typically used for penetration tests, but pirates have used it to gather data from email or servers, as well as root privileges and user profile information from a variety of target sources. Active and passive reconnaissance are two popular methods of reconnaissance in which information is gathered using OS fingerprinting and Wireshark methodologies [18].
5. **Computer Worms:** The terminology "computer worms" was originally used in 1975 by John Brunner in his novel "The Shockwave Rider," and the very first computer worm was "Morris Worm," which was released in 1988 [19]. Computer worms affect other computers linked to this system by duplicating themselves or spreading across the entire network, rather than infecting the file system. However, many worms have payloads planned to rob vital information. It can infect the target machine in a variety of ways, such as spam emails or instant messaging with attachments; when the user clicks the attachments, the malware is installed without the user being notified. Computer worms can be detected and removed with high-quality anti-virus software.
6. **Port Scanning:** It's a typical strategy employed by cybercrooks while looking for weak spots in networks. Hackers first scan for target hosts and send packets to an open port, then determine whether the port is being utilized by obtaining a response [41]. They then collect user and system data such as services, authentication systems, and other system-related data. Hackers utilize another method to detect vulnerabilities in ports called port mapping.
7. **Brute Force Attack:** It's an old but effective method of obtaining users' personal information or sometimes, hackers spread malware in the entire system of the users. Hackers use all feasible permutations to guess login information and encryption keys in an attempt to acquire access to the target system. Brute force cyberattacks come in many forms, including dictionary, hybrid, reverse, and credential [41].

In this thesis, we aim to cover practically all types of cyberattacks, and we train and evaluate our unique IDS Ensemble stacking model using two large publicly available cyber-attack datasets, UNSW-15 and CICIDS2017.

2.3 Communication Networks

A computer network is a set of devices such as laptops, servers, smartphones, etc., connected through cables, optical fibers, and wireless radio-frequency methods to share resources. Moreover, network devices communicate with each other via modem, which turns data into signals and allows for a two-way network connection. Since the advent of the 5G network, streaming services have grown in popularity, and users are more interested in sharing massive multimedia files across many platforms. However, as the number of subscribers on cellular networks expands exponentially, ISP providers face a significant challenge in guaranteeing the network's Quality of Service (QoS) to its end users. The network building blocks and principles used throughout this thesis are briefly discussed in the following sections.

2.3.1 5G Network Basics

5G stands for 5th generation mobile cellular network that is significantly faster compared to its predecessor, 4G/LTE, and the commercial deployment of 5G started in 2020. The need for high-speed connectivity is growing every second as the world's mobile data traffic doubles every 18 months. 5G was created to effectively supply tremendous data expansion to meet this need. According to Global System for Mobile Communications Associations (GSMA) 25% of the world population will be connected to 5G within 2025 [21]. Along with multi-Gbps data speed 5G also provides massive network capacity, ultra-low latency, more availability, and high reliability. Another 5G feature is network slicing, which allows each connected service to have a separate, unique portion of the network that is guaranteed for use only for that reason. By 2025 the number of IoT devices globally will be increased to 30.9 billion and only 5G can fulfill the staggering network bandwidth demand to connect all the devices [22].

2.3.2 5G Network History

There are no specific organizations or person is working behind to bring 5G into the market instead, it is a collaboration of several telecom consortiums and companies within the mobile ecosystem. The term "Internet of Things" (IoT), proposed by Kevin Ashton in 1990, is closely associated with the idea of high-speed internet [23]. IoT is a concept of connecting billions of devices to share real-time information from home appliances across the globe. To share data, users need high-speed Internet connections. By early 2004 developers concluded that it was impossible to connect billions of devices for real-time responses by 4G or 3G with a latency of 40ms or higher.

NASA initiated the official development of 5G in 2008, when NASA funded a company named Machine-to-Machine Intelligence (M2Mi) to develop IoT and M2M technology and the development of 5G technology to support the IoT devices. Meanwhile, South Korea started an IT R&D program that same year to design a beam-division multiple access-based 5G mobile communication system. While the University of Surrey obtained funding from the British government for a 5G research center where they partnered with Telefonica Europe, Huawei, Fujitsu Laboratories, and Aircom International, New York University established a research facility titled NYU WIRELESS in 2012 for 5G research. A 5G research project called Mobile and Wireless Communications Enablers for the Twenty-Twenty Information Society was introduced by the European Union (EU) later in 2012 [26]. Moreover, European Union (EU) also established a public-private partnership to accelerate research and innovation on 5G in 2013.

South Korea became the first nation to deploy 5G in 2019 following several stages of development carried out by various research laboratories. South Korean telecoms firms SK Telecom, LG Uplus, and KT serve as the key contributors to the debut of 5G. Later Verizon from the United States rolled out its first 5G services publicly by connecting a few thousands of customers. Moreover, at the beginning of 2020, the number of live 5G network providers increased significantly in Europe. In 2020, Rogers, Bell, TELUS launched 5G networks in some major cities in Canada. For using 5G the main requirements have a 5G support phone.

2.3.3 How 5G Network Works

All 5G compatible devices in a cell are connected to the Internet by radio waves through a local antenna located in the cell, and 5G services are separated into these tiny geographic regions known as cells. 5G utilized the radio frequencies to send and receive data like other available cellular networks. 5G also provides a different band of the spectrum – low band, high band, and mid band, where low band is for long distance coverage, but not fast, high band can carry a large amount of data with less delay, but coverage area is limited, and the mid band is also known as C band provides faster Internet with wider coverage area.

To fulfill the customer needs, the architecture of 5G is conjugated with Radio Access Networks (RANs) like Orthogonal Frequency Division Multiplexing (OFDM) based wave signals, frame technology and multiple access technologies [24]. OFDM is very popular system for transmitting signals over wireless channels, it was also broadly used in Wi-Fi and 4G LTE. OFDM is completely different from its predecessor, Frequency Division Multiplexing (FDM) where it modulates the digital signals received from different channels to reduce the inference. 5G utilized a 5G-New-Radio (5G-NR) air interface along with OFDM technologies [24]. The advantages of using OFDM are it has low noise, wider spectral efficiency by utilizing Multi-Input and Multi-Output (MIMO) technology, and data transmission efficiency. However, for heterogenous network traffic MIMO and OFDM is not well enough because of their fixed transmission time interval (TTI) during transmission. To overcome this issue, 5G incorporated multiple access technology to achieve both downward or upward expansion of different TTI with constant frequency, and it helps 5G reduce the latency. Integrating TDD frame technology is another approach for reducing delay in 5G where a single frame is a combination of transmitted data along with acknowledgement.

2.3.4 Quality of Service (QoS) Metrics of the Networks

The measurement of service quality that the network providers offer to the users is known as Quality of Service (QoS) [27]. Measurement of QoS metrics can be used to justify the impacts of 5G internet traffic on network performance. The following are descriptions of several QoS parameters:

Packet Loss: A packet is a compact bit of data that is routed between a source and a destination through a network protocol. A packet loss happens when a delivered packet fails to reach its intended destination. It is measured as a proportion of lost packets compared to total messages delivered. The main reasons behind packet loss are weak network signal at the destination, system noise, network congestion, bounded memory at nodes, etc. Eq. 2.1 shows how Packet loss is formulated.

$$packet\ loss = \frac{packets\ transmitted - packets\ received}{packets\ transmitted} \times 100 \qquad (2.1)$$

Latency: The term latency is used to define the delays in network communication. Latency is the total proportion of transmission time taken by a packet to reach from source to destination. When transmission time is small it is referred as low latency and higher transmission time is referred as high latency. Eq. 2.2 shows how the latency is calculated.

$$Latency = \frac{packets\ size}{Link\ Bandwidth} \qquad (2.2)$$

The following are the four main factors that influence the network latency:

a) **Propagation Delay:** A packet's propagation delay is the time it takes for it to travel via a transmission channel like copper, coax, or fiber optics from source to destination.

b) **Queuing Delay:** It happens when one packet is waiting for the transmission to the destination while another is using the router's services.

c) **Transmission Delay:** The duration of time it takes to transfer all the bits in a packet from the host to the transmission medium is also known as packetization delay.

d) **Processing Delay:** It is the time takes for a router to examine all incoming packets based on their packet header and routing table to identify which node to forward the packet to next.

Jitter: When a packet is sent from one source to another, it is sent in regular intervals over a set period of time. Eq. 2.3 depicts the calculation of jitter. When there is a variation in time delay in sending packets, it is called jitter. The main causes of jitter are network congestion and route changes.

$$Jitter = \frac{sum\ of\ variation\ of\ delay}{sum\ of\ total\ packet\ received} \tag{2.3}$$

Throughput: The amount of data transported from a source to a destination in a unit of time is referred to as throughput. Bits per second, or bps, is a common unit of measurement for throughput. Throughput is determined by the available bandwidth, which resembles a network's maximum capacity at any one time. Throughput increases as bandwidth increases. The formulation of throughput is given in Eq. 2.4.

$$Throughput = \frac{Total\ data\ sent}{Delivery\ time} \tag{2.4}$$

2.4 Impact of 5G on QoS

5G networks significantly outperform its predecessor 4G in terms of Quality of Service (QoS). MIMO, mm waves, and OFDM technology were employed in 5G to enhance the number of clients served. Additionally, compared to 4G, 5G can deliver higher throughput, expanded capacity, lower latency, reliability, availability, and dynamic bandwidth distribution. The main features of 5G are briefly discussed below:

a) **eMBB (Enhanced Mobile Broadband):** Since the network is being densified day by day, to solve this, 5G is delivering seamless coverage with higher data speeds and dynamically distributing bandwidth. Video streaming, virtual reality, augmented reality, and other services benefit from high throughputs.

b) **mMTC (Massive Machine Type Communication):** To deliver various services, many heterogeneous embedded sensors are connected. This huge network might be accommodated through 5G by scaling down data rates, power, and mobility to connect one million devices per square kilometre.

c) **uRLLC (ultra-reliable and low-latency communications):** Machine-to-machine (M2M) connections, such as robotic manufacturing, remote surgery, and expensive simulation, require a network with low latency, high reliability, and high availability. When the network state changes, 5G provides ultra-reliable, low-latency services that will not disrupt any services.

A class identifier (QCI) technique known as 5QI is employed in 5G to measure quality-of-service. The improvement of overall QoS with 5G is given below:

I. The **throughput** of 5G is up to 20 Gbps in downlink peak data rates.
II. The average **latency** is reduced to 1 to 10 milliseconds for 5G which was 50 milliseconds for 4G.
III. In 5G, the block error ratio, a network **reliability** benchmark, is expected to be 0.00001 in 1 millisecond, down from 0.01 in 4G.
IV. The **availability** of 5G is expected to be 100%.
V. **Jitter** was reduced to 10–100 microseconds which is less than 4G LTE.

2.5 Machine Learning Concepts

For the past two decades, researchers have attempted to automate intrusion detection in various contexts. Additionally, they attempt to use machine learning approaches to solve drawbacks in earlier methodologies such as false-negative rates and detection accuracy. Earlier machine learning systems relied on trial and error, with models attempting to discover the pattern of attacks and then making predictions based on the matched pattern information. Later, signature-based detection was used, in which ML models compared normal and attacked traffic and identified an attack if the signature fell below a particular threshold value. Instead of detecting a specific attack, researchers can only distinguish between attack traffic and regular traffic using this method. Machine learning methods get more powerful in computing over time, providing better results in managing normal and abnormal data in a variety of real-world scenarios. Researchers combined machine learning approaches to produce an IDS that is generalizable enough to identify attack victim's variants and unique attacks. Moreover, machine learning models are not domain-specific;

rather, to develop a basic IDS model, it combines various statistical techniques with a mathematical analysis.

2.5.1 What is Machine Learning and How Does it Work?

Machine Learning (ML) is an application of Artificial Intelligence (AI) that attempts to emulate the human brain by achieving knowledge from a given dataset and learning from it to increase the accuracy. According to Jordan and Mitchell, machine learning (ML) tackles two interconnected research questions: firstly, how a system can use its knowledge to continuously achieve a better performance, and secondly, to achieve that knowledge, what should be the core learning system for constructing a model [20]. Data is the most important part of building an ML model, and data can take numerous forms, including text, image, video, audio, time series, and many more. In its learning process, an ML model usually goes through three steps: it gives a pattern based on input data in the decision-making process, it uses an error function to compare the model to known instances, and finally, it updates its weight values to minimize the error gap between known examples and model estimation for optimization. ML model data is divided into two parts during the learning process: Training and Testing set. Figure 2.4 shows the learning process of machine learning.

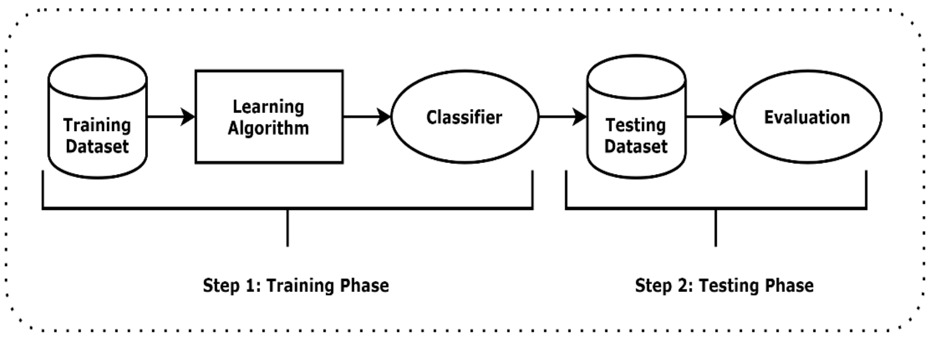


Figure 2.4: Learning Process of Machine Learning

1. **Training Set:** This portion of the data is typically utilized to train the machine learning model, and data from the training sample is excluded from the test sample and total percentage of training data is higher than the percentage of testing data, which is between 70% and 80%. The data in the training set sometimes have labels, therefore during learning ML algorithm can correlate data with the label.

2. **Testing Set:** A chunk of data that is not used in the training process. After training the model with a train dataset, the test set is used to validate our model and observer how the model is performing for unseen data. Test set needs labels to calculate the different metrics to evaluate the performance of the models.

Aside from the train and test sets, a component of data called the validation set is employed during training by dividing the training set, which is used to estimate the performance of our model.

There are mainly three different kinds of ML classifiers: supervised ML, unsupervised ML, and reinforcement machine learning. We will briefly discuss supervised machine learning and its various algorithms in the following section because it is a crucial component of this thesis. We'll aim to cover a variety of supervised learning algorithms, from the most basic to the most advanced, such as deep learning.

2.5.2 Supervised Machine Learning Algorithms

Supervised ML strategies try to create a model which learns about the characteristics of previous data with class labels and can make predictions about new data without labels [20]. The algorithms learned through training data can be thought of as a supervisor, monitoring the learning approach until it discovers sequences and input label and out labels are linked together.

Supervised learning algorithm maps from input x to output y, where the learning phase of the algorithm starts with a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots\ldots, (x_n, y_n)\}$. Here, xi is input vector or matrix, which are also referred as features and yi is the label we will predict and it could predict one or more than one thing. All of these data are from an unidentified distribution M, so $(x_i, y_i) \sim$ M, where (x_i, y_i) are independent. The formula of supervised learning is presented in Eq. 2.5.

$$D = \{(x_1, y_1), (x_2, y_2), \ldots\ldots, (x_n, y_n)\} \subset = N^d x\, L \tag{2.5}$$

Here, n is the dataset size, N^d is the feature space with d-dimension and L is the labels of the data. Labels L can be binary, where label space lies between {0, 1}, for example for IDS dataset binary label can be attack traffic and normal traffic. For multi-class classification problem, we need to specify multiple class, like L = {1 = DoS, 2 = DDoS, 3 = Worm, etc.}.

Classification and Regression are two types of problems handled by Supervised ML algorithms. Models for regression problems attempt to predict numerical representation between the input and output data. Classification algorithms use input data to try to find patterns and relationships among different categories and classes. Binary classifiers are used to predict whether network traffic is being attacked or not and multi-classifiers can predict different cyberattack variants. In the following few sections, we will briefly describe some important types of classification-based supervised ML algorithms that are related to this thesis. Figures 2.5 shows the taxonomy of supervised machine learning.

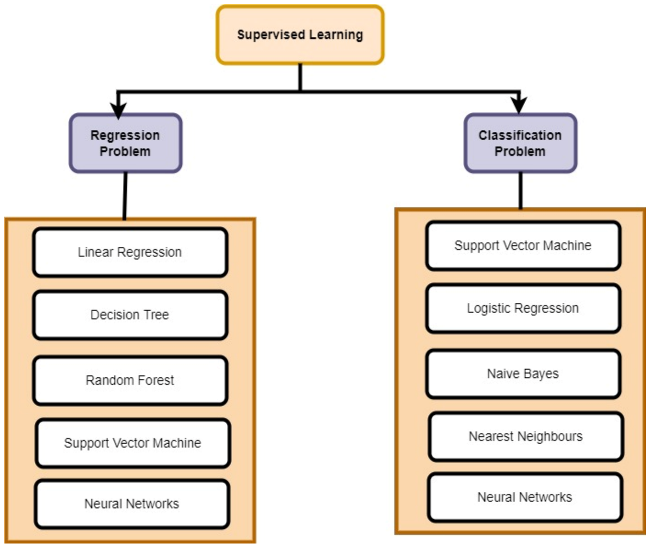


Figure 2.5: Taxonomy of Supervised Machine Learning

2.5.3 Linear Regression

The most commonly used supervised learning approach is linear regression, which analyses the strength and link between two or more variables and depicts the relationship with a straight line. When two variables are compared, one is termed the independent variable, while more than others could be use as dependent variable, also known as the goal. The linear regression is formulated in Eq. 2.6.

$$y = \beta_0 + \beta_1 X + \varepsilon \tag{2.6}$$

Where, the predicted value of the dependent variable is y, β_0 is considered as the intercept of y, β_1 is the coefficient of regression, X is the independent variable and the error rate is ε. Three sorts of relationships that can exist in a regression line: no relationship, positive relationship, and negative relationship. The coefficients are generated to reduce the error rate between the real data and the predictive model to acquire the best predictive model. The error might be positive or negative, and the mean squared error (MSE) is determined as the sum of the squared distance between model predictions and actual data [43].

$$MSE = sum((Actual - prediction)^2) \tag{2.7}$$

The key benefit of linear regression is that it is easy to use and less difficult than other techniques. Furthermore, regularization and dimensionality reduction keep the linear regression model from overfitting. Linear regression, on the other hand, can only handle single dependent and independent variables.

2.5.4 Decision Tree

A Decision tree, or DT, is a ML classifier with a tree structure that learns decision rules from features to forecast a target. It can contain three nodes: parent, child, and leaf nodes. The decision tree's learning process begins at the root node, which holds the complete dataset, and proceeds along the tree, answering yes or false questions until it reaches the leaf node, where it obtains knowledge about a particular sample. The decision tree provides final predictions for a given data point by taking the average value of the dependent variable from the leaf node. Pruning is used to overcome an overfitting problem that might occur with a deep decision tree design.

A decision tree is simple to understand, uses little processing power, and requires less data. Its overall performance is unaffected by pretreatment non-linearity. However, when only a tiny percentage of the data changes, the decision tree becomes unstable.

2.5.5 Random Forest

Random Forest was created to address the decision tree's shortcomings and is used for non-linear regression. It is also known as an ensemble decision tree since it is a mix of numerous decision trees. The training data is sent through numerous decision trees, and after processing, various outputs from all of the trees are combined to provide outputs that are either majority voted or averaged.

Random forest overcomes the overfitting problem of the decision tree and provides more accurate results than the decision tree. In a decision tree, only one tree structure is used while a random forest is a combination of multiple decision trees. That is why its computation complexity and memory requirements are higher compared to the decision tree.

2.5.6 Support Vector Machine

Support vector machine or SVM, attempts to categorize data points in N-dimensional feature space by locating a hyperplane or decision border. Hyperplanes are decision boundaries that separate data, with data points on opposite sides classified as belonging to distinct classes [30]. The primary purpose is to identify the highest margin that produces a distance between two classes, with a greater margin distance indicating the best SVM model.

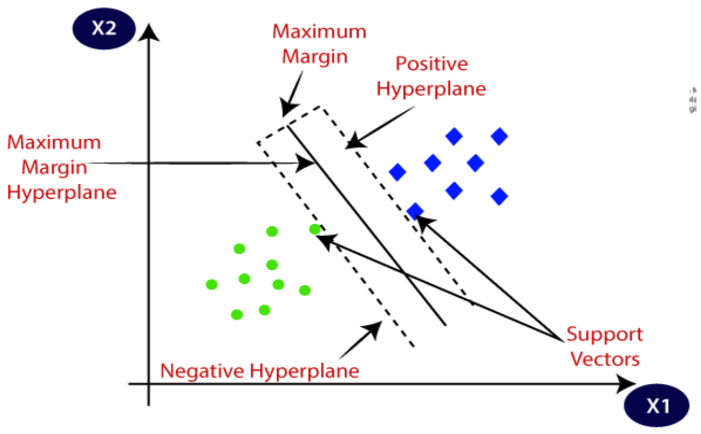


Figure 2.6: Support Vector Machine [30]

Figure 2.6 depicts an SVM with a maximum margin hyperplane, with a solid black line representing the maximum margin between two separate classes; moreover, positive as well as negative hyperplanes separate the classes, and decision boundaries called hyperplanes are utilized to predict the classes as output.

2.5.7 Neural Networks

In machine learning concepts, a neural network is a mathematical model that can simulate the capabilities of neurons in the human brain. A neural network's basic structure is a mathematical model, often referred to as a function, used to address computational problems. The three sets of laws in a neural network are multiplication, summation, and addition, where each input value in the network is multiplied by a weight. Every weighted input is summed with a bias, and finally, weighted input and bias are pass-through functions called activation functions [20].

The most common form of the neural network is the perceptron, which consists of input and output layers, with the input layer containing many inputs x_n multiplied by a weight w_i and output after summing them. Fig 2.7 shows the basic architecture of the neural network.

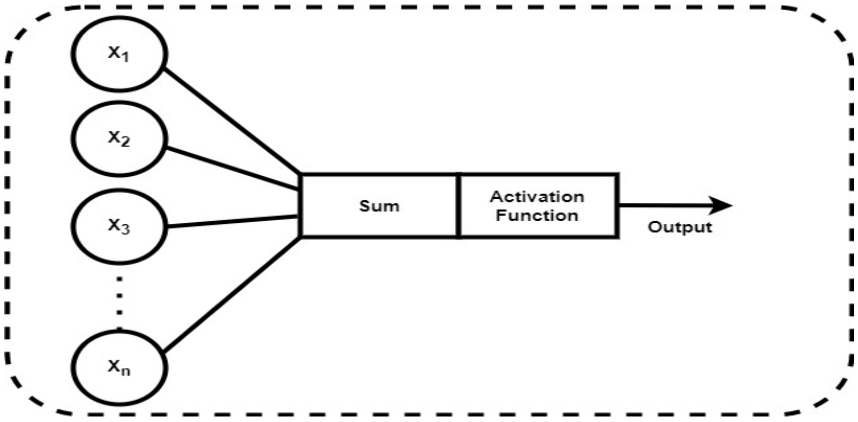


Figure 2.7: Perceptron Architecture

The perceptron's activation function can only be used with linearly separable classes; otherwise, it will not classify properly. Moreover, because of their binary label transfer function, the activation function can only provide two outputs (0 or 1). Perceptrons are also incapable of learning from the logical function "XOR". Because of the aforementioned reason, the perceptron has no real-world utility. However, the hidden layers, are added to the perceptron, it becomes a multilayered perceptron capable of solving complicated problems and non-linear activation function is used to achieve the non-linearity.

2.5.8 Multi-layered Perceptron

A multilayer perceptron (MLP) is a perceptron that has more than one one-liner layer, also referred to as a hidden layer. It's also used interchangeably as the feed-forward neural network because the outputs of one layer become the inputs of the next layer, and each layer is connected to the next layers, resulting in a fully connected layer. Input units, output units, and one or more hidden layer units are the major MLP layers. The depth of the MLP is defined as the number of layers, while the width is defined as the number of units. The input layer extracts feature from input data, forwards them to the hidden units, and performs computation before passing them on to the output units [20]. Fig 2.8 shows a multilayer perceptron comprised of one input layer, one hidden layer and one output layer.

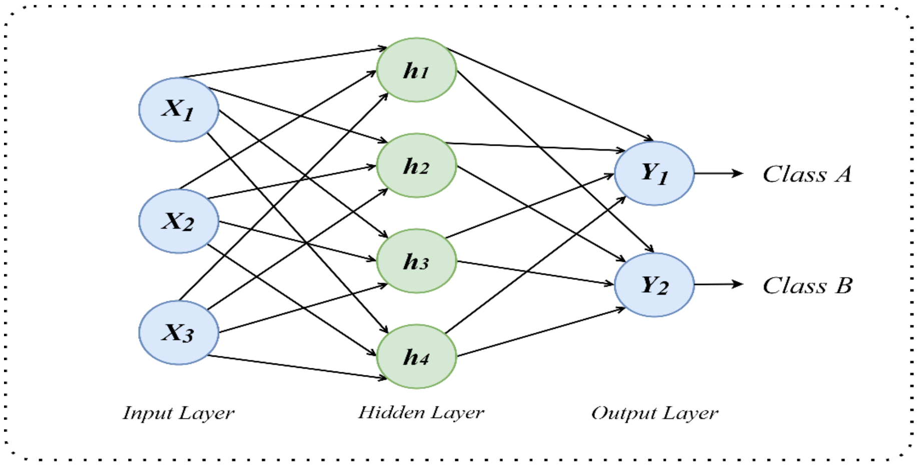


Figure 2.8: Multi Layered Perceptron Architecture

The input layer has x_n units, associated with different weights and same bias. Because the network is fully connected, the output of the input layers is forwarded as inputs to the hidden units. The units of the hidden layer are denoted as h_i and Eq. 2.8 and 2.9 is used to calculate the hidden layer.

$$h_1 = \Phi^1 (w_1 x + b_1) \tag{2.8}$$

$$h_2 = \Phi^2 (w_2 h_1 + b_2) \tag{2.9}$$

The function Φ represents the activation function, we provide two activation functions since the activation functions of different levels differ. The bias value is denoted by b, while the weights are represented by w. The weights w influence the strength of connections between neurons. The weight vector w_{i_j} which is a combination of ith input layer and jth hidden layer combined into a single matrix called w by using dot product multiplication. The result of the hidden unit then becomes the input data of the output unit, which is denoted by z. Eq. 2.10 shows the formulation of the multilayer perceptron.

$$z = \Phi^1 (w_3)\, h_2 + b_3 \tag{2.10}$$

We must normalize the output vector, which comprises real number values, in order to forecast the class labels. To estimate the class labels, we'll encode to a probability distribution y and use the SoftMax function to normalize the output layer in neural networks and the calculation is given in Eq. 2.11.

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{2.11}$$

The SoftMax function translates the raw outcomes of a function into probability distributions, with a range of 0 to 1. For multi-class classification problems, SoftMax functions are sometimes employed as the activation function in the output layer of a neural network, where it displays the probability of each class. It probabilistically maps the output so that the total sum of the output classes equals 1.

The multilayer perceptron (MLP), which has numerous input layers, hidden layers, and output layers, is sometimes referred to as an artificial neural network (ANN). A deep neural network (DNN) is a type of artificial neural network that contains several hidden layers in between the input and output layers and a specific degree of complexity. It can handle both linear and non-linear connections and performs complex mathematical operations. Data goes from the input to the output layer without looping back in this feedforward network. DNN multiplies and adds weights with an input value and outputs a value between 0 and 1, and it also modifies the weights if the method doesn't work correctly.

Neural networks can extract features from imperfect or complex data to identify solutions that are too difficult and time-demanding for the human brain to solve. They can also be considered a collection of algorithms for detecting hidden patterns and storing them for future use once they learn the patterns. When the relationships between the inputs and outputs are nonlinear and difficult to interpret, Neural networks draw assumptions and generalizations, as well as expose relationships and generate predictions. Neural networks outperform other machine learning models when it comes to making predictions from unknown data. In this part, we covered a simple version of a neural network with one hidden layer, often known as shallow neural networks. We will look at several deep neural networks with multiple hidden layers in the sections ahead.

2.6 Deep Learning Concepts

In this section, we'll go through the various deep learning (DL) models that are relevant to our thesis. DL is a subset of ML which allows multilayer mathematical models to learn data representation by emulating the behaviour of the human brain. DL can evaluate data in a way that imitates the functions of human brain, and this structure is referred to as an ANN (Artificial Neural Network). When compared to ML, the structure of DL is more complicated, and it requires less human interaction because of its sophisticated design, which allows it to learn characteristics autonomously. Compared to traditional ML, DL takes more data to develop a model, and its sophisticated multi-layered structure learns diverse aspects from data while eliminating data fluctuations. Although DL systems take longer to train, they may produce results almost instantly. There are many different neural network topologies, and we'll go through Convolution Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).

2.6.1 Convolution Neural Network

Convolution neural network (CNN) also known as ConvNet designed to process grid-like topological data such image. The first CNN was developed by Yann LeCun, where he proposed LeNet-5 which was able to recognize handwritten characters like postal code [28]. CNN consists of multiple layers where earlier layers are responsible to extract features and later layers combines the features and make predictions. Figure 2.9 shows that a CNN model consists of various types of layers where convolution and pooling are used for feature extraction and fully connected layers are responsible for classifying the results.

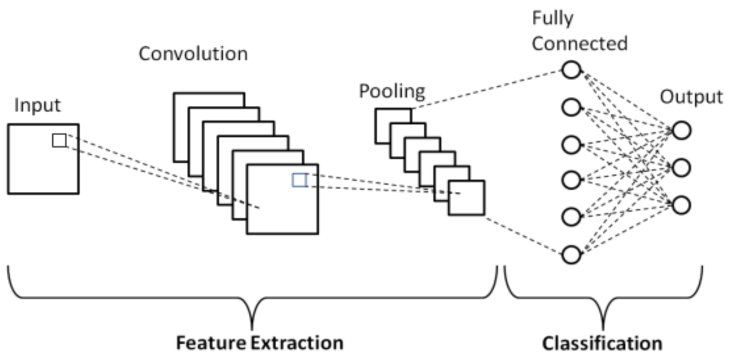


Figure 2.9: Convolution Neural Network [31]

1. **Convolution Layer:** It is the computational load-bearing building component of the CNN model. A filter or kernel slides across the input picture using stride and combines the filter values with original pixel values using dot product multiplication to form a new dimensional activation map, and the input of a convolution layer is reshaped to an ideal size. We add padding to the image's outer frame to provide additional space for the filter to cover the image, and it enables a more accurate image analysis by CNN. If we have a CNN input shape $i \times i \times d$, where i is the input size with d dimension Convolution layers works as follows:

$$i_{out} = \frac{i-f+2p}{s} + 1 \tag{2.12}$$

Where, f is the kernel size, s is stride, p is padding, i is the input, d is the dimension and i_{out} is the output of convolution layer.

2. **Pooling Layer:** Several pooling layers can be piled with convolution layers one after the other in a CNN architecture. Max pooling is the most used pooling layer. By sweeping a 2-D filter across the feature map, down sampling it, and aggregating the maximum output from the feature map, max-pooling minimizes the spatial size of the image. If we have a feature map of $i \times i \times d$, a kernel size of f, and stride s the output of max pooling can be:

$$i_{out} = \frac{i-f}{s} + 1 \tag{2.13}$$

Where, f is the kernel size, s is stride, i is the input.

3. **Fully Connected Layer:** Fully connected layers, also known as Dense layers, are made up of neurons, biases, and weights, and they interconnect the neurons between two layers. Before forwarding the input images from the previous layers to the fully connected layer, they must be flattened. Dropout layers are provided in between the fully connected layers to protect the model from overfitting. Finally, several activation functions such as the sigmoid or SoftMax activation function are used to derive class label predictions from fully linked layers.

2.6.2 Recurrent Neural Network

A recurrent neural network (RNN) is an artificial neural network (ANN) that works well with a sequence of data or time-series data. Furthermore, ordinal or temporal issues are frequently addressed using it. The RNN has at least one feedback link to allow activations to circulate in a loop, allowing the RNN to learn data sequences. When making future predictions, RNN has the capacity to recall the connection between the current input and the inputs that have already been used. RNN, however, is only capable of taking a little stride backward [36]. Figure 2.10 shows the basic architecture of RNN.

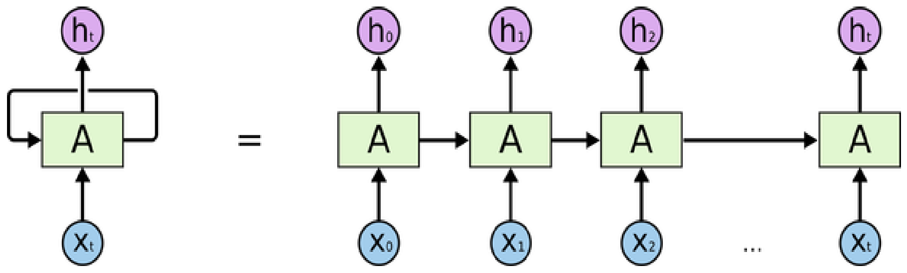


Figure 2.10: Basic Architecture of RNN [36]

It simply implies that the network's layer structure is repeated by the architecture throughout the entire period. Here, X_t is the vector of size N at timestamp t. A is the hidden state at time step t, and it is considered as the memory of the network. The calculation of the memory state is based on previous hidden state and the input at the current time step. The calculation is shown in Eq. 2.14.

$$A_t = f\ (WX_t\ +\ UA_{t-1}\ ) \tag{2.14}$$

Where, W is the input weights and U are is a weight of previous state. f is a non-linearity used to generate final cell state.

Theoretically, RNN can manage long-term dependencies and is effective for time series forecasting. It can handle input of any length, and even if the input size grows, the model size stays the same. The weights are constant across time and give historical data weight. RNN computations move more slowly, and training might be challenging. It struggles with exploding and gradient disappearing.

2.6.3 Long Short-Term Memory

Hochreiter and Schmidhuber initially proposed the Long Short-Term Memory (LSTM) as an advanced variant of RNN in 1997, to address the problem of exploding and disappearing gradients [29]. The only objective of LSTM is to avoid long-term reliance issues, and it is capable of automatically remembering information for lengthy periods of time. Figure 2.11 shows that the LSTM is made up of three cells, also known as gates:

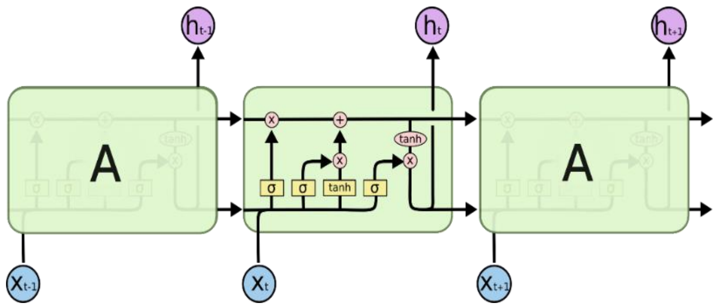


Figure 2.11: Long-Short Term Memory [31]

1. **Forget Gate:** This LSTM cell decides whether the cells need to keep information gained from the previous cell or forget it. A sigmoid function is utilized and it looks at the hidden state of the previous timestamp h_{t-1} and current timestamp and give a number between 0 and 1 as output, where 1 represents storing the state and 0 represents removing the state. Eq. 2.15 represents the calculation of forget gate.

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \tag{2.15}$$

2. **Input Gate:** The input gate, also known as the store gate, is in control of storing and quantifying new information. The first layer incorporates a sigmoid function to

determine which values in the cell need to be updated, while the second layer obtains the new data and applies the tanh activation function, which translates the new data between -1 and 1. After that both two parts are combined and updated the state.

$$i_t = \sigma\,(W_i.[h_{t-1}, x_t] + b_i) \tag{2.16}$$

$$\hat{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.17}$$

The formulation input gate is presented in Eq. 2.18.

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.18}$$

3. **Output Gate:** A sigmoid function is utilized in the output layer to determine which elements of the cell state will be the output. The cell state is sent through the activation function tanh, which outputs values ranging between -1 to 1.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.19}$$

$$h_t = o_t * tanh(C_t) \tag{2.20}$$

2.6.4 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is another short-term memory solution, and its essential principle is virtually identical to that of the LSTM. GRU used a hidden state to transport information instead of a cell state, and its design consists of two hidden states: the reset gate and the update gate. GRU's architecture is shown in Figure 2.12.

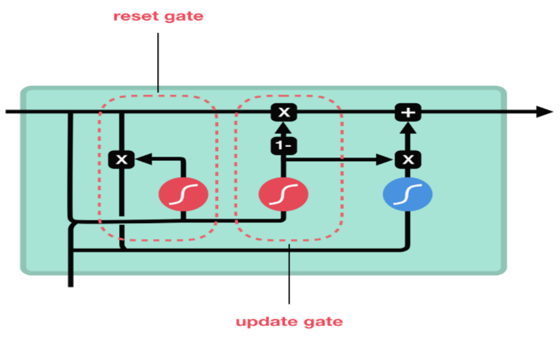


Figure 2.12: Gated Recurrent Unit [32]

2.7 Stacking Ensemble Architecture

Ensemble methods are strategies that are made up of numerous models that are then combined to improve classification and regression accuracy. When compared to a single model, a collection of models provides higher accuracy.

In a classification problem, we could use the hypothesis value H to categorize the new samples residing in the training data. The principal purpose of the hypothesis is to translate the features of the training data into relevant labels or classes that need to be predicted. Furthermore, it should reduce the generalization error by closely approaching the genuine unknown functions. To construct hypothesis H in the supervised learning technique, known data labels must be given in the training data instances. In the DL model, the purpose of producing hypothesis H is to predict unknown labels from test data. Additionally, incorporating predictions from several models can increase overall performance, and this combination of different models is referred to as ensemble learning or ensemble model. Several types of ensemble techniques developed such as voting and averaging based and stacking based ensemble methods. In our thesis, we are using stacking-based ensemble methods because it uses a second label learner also known as meta learner through which it can define which classifiers are appropriate and which are not. While bagging and boosting employ homogeneous weak learners, ensemble uses heterogeneous weak learners to train them in parallel and aggregate them. One drawback of this method is that each model's contribution to ensemble forecasts can occasionally be the same. Figure 2.13 shows the basic architecture of stacking ensemble technique.

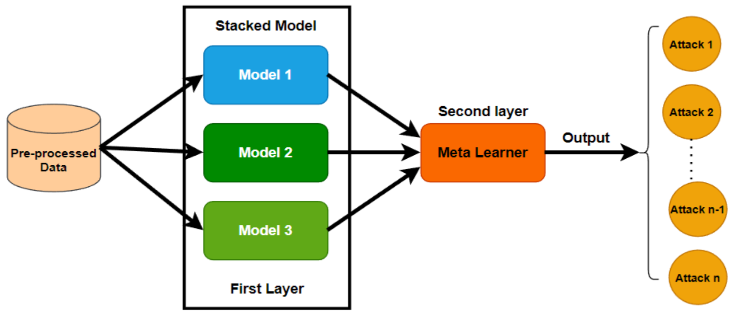


Figure 2.13: Stacking Ensemble Model Architecture

The idea of stacking was initially presented by David H Wolpert in 1992 when he divided the dataset into J equal pieces and utilized Jth fold cross-validation during training while the remaining samples were used for the testing purpose [33]. He later trained numerous models using the training test pairs as input for the meta-model using the training test pairs. Deep Convex Net (DCN) was proposed in 2011, and it was made up of many approaches layered together, including a linear input layer, non-linear hidden units, and a second layer containing target classes [34]. The output from lower modules is passed as the input of adjacent higher modules, and all the modules are connected to each other. Combining model-based approach along with hill climbing approaches, a data-driven framework for selecting parameters and models in a deep-stacked neural network was suggested in 2016 [35]. Parameter estimates, model selection, and hyperparameter tuning are all part of the stacking framework. An ensemble model uses the predictions of the previous layers models and combine them together and takes them as input to build a new model, and the calculation is presented in Eq. 2.21.

$$f(y|x) = \sum_{m \varepsilon M} w_m f_m(y|x) \tag{2.21}$$

Where, f_m is the m’th base model.

2.8 Time Series Forecasting

A time series is a group of observations or a set of data points that monitors samples across time. The time series' data points are used to depict changes over time. They have often repeated observations of various data measures from comparable sources. These observations are generally done on a minute-by-minute, hour-by-hour, daily, weekly, bi-weekly, monthly, and annual basis. Because so many enterprises across a wide range strongly rely on time series to disclose the patterns changes over time, it is becoming more and more important in our day and age. Additionally, we must carry out time series analysis in order to estimate the future behavior and patterns of the time dependent data. Time series are generally two types a) additive time series, and b) multiplicative time series [37].

Time series forecasting is currently widely utilized and has a variety of uses in different sorts of organizations. There are several uses for it, some of which are listed below:

1. Weather forecasting

2. Stock Price predictions.
3. Climate forecasting
4. Economic forecasting
5. ISP traffic predictions.
6. Network QoS Predictions
7. Healthcare predictions

2.8.1 Types of Time Series Data

There are mainly three types of time series data, and they are trend, seasonality, and cyclic. An overview of that data is given below:

a) **Trend:** A time series' trend shows how patterns change over a long period of time, either upward or downward. When there is an increase or decrease in behaviour over time, it is overserved rather than repeated over time. A general trend may be upward or downward trending. Some techniques for eradicating trends from time-series data include log transformation, smoothing, and linear regression.

b) **Seasonality:** When some observations in a time series are impacted by seasonal elements that recur throughout time, a seasonal pattern is formed. Seasonality can occur at a certain time of the week, month, or year. Additionally, seasonality can be seen at various points throughout the day. For instance, power usage peaks during the day and troughs early in the day, or online sales rise during Christmas before declining once more. We need to eliminate seasonality, a process known as deseasonalizing, to construct a better forecasting model. Some techniques for eliminating seasonality include average de-trended values and temporal differencing.

c) **Cyclic:** When the frequency of data is not fixed and rises and falls in the data are very frequent, then a cycle occurs. If the fluctuations are not fixed period, then the time series is cyclic. Generally, the average length of a seasonal pattern is shorter than the cyclic pattern.

d) **Random:** Time series data occasionally exhibit a random pattern, which is characterized by irregularity and has no discernible pattern. It is also known as white noise and white noise is not predictable.

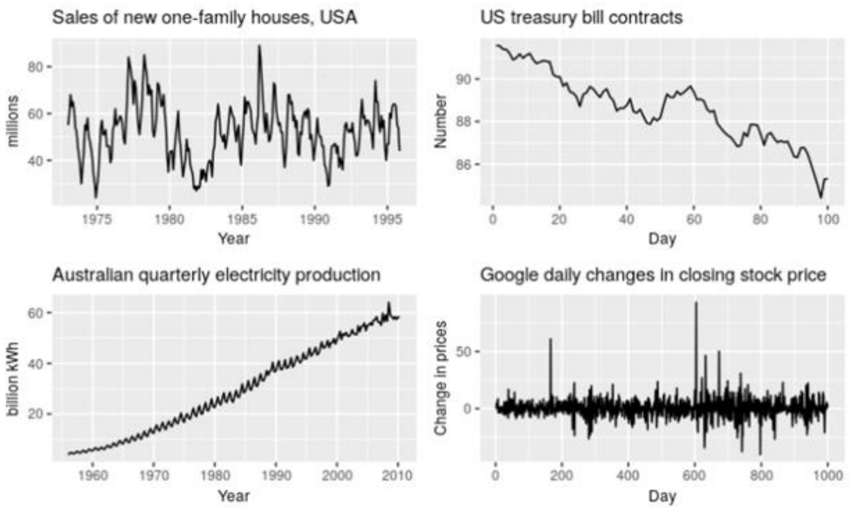


Figure 2.14: Different patterns of Time Series [38]

Figure 2.14 depicts the different patterns of time series data, in which the top left plot shows the monthly house sales where it shows strong seasonality every year with strong cyclic behaviour. The top right and bottom left plots do not show strong seasonality but downward and upward trend. Finally, the bottom right plot does not have trend, seasonality, or cyclic patterns rather it has random fluctuations over time.

2.8.2 Basics Steps in Time Series Forecasting

At the very first step of time series forecasting it is necessary to convert the date column into datetime datatype. Plotting the data is thus crucial to examining the time series data. The data's behaviour, including patterns, unexpected observations, missing values, trends, seasonality, changes in patterns through time, and correlations between variables, may well be understood using the time plot. We go through some of the fundamental procedures for forecasting from any time series data in the sections below.

a) **Stationarity:** A time series is considered stationary when its observations are independent of time. Additionally, seasonality and trends are time-dependent. Time series with trends and seasonality are not stationary because of this. A cyclic time series will always be stationary. Moreover, a time series that contains random data or white noise is likewise stationary.

The fundamental prerequisite for defining time series data as stationary is that it will not vary over time and that its general pattern should not change. Furthermore, the predictions perform poorly when fitted with non-stationary data. The non-stationary data must thus be converted into a stationary format. The two most often employed transformation methods are the Difference transform and the Logarithmic transform.

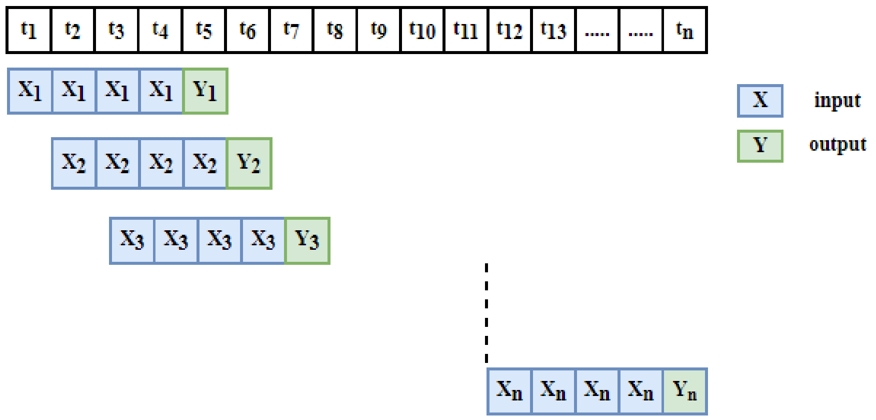
The Augmented Dicky-Fuller test, or ADF test, is the approach that is most frequently used to determine whether a time series is stationary or not. An ADF is a unit root test that verifies the existence of a unit root under the null hypothesis.

If it is present, then $p > 0$, the process is not stationary, otherwise
If $p \leq 0$, the null hypothesis is not accepted, and the process is stationary.

b) **Autocorrelation:** A time series' degree of resemblance to a lag version of itself over a subsequent period is defined by autocorrelation. The lagged versions of two variables, such as x_t and x_{t-1}, have a linear connection. It is employed to find trends and patterns in time series data. Serial correlation, which mimics the entire correlation of the time series, is another name for autocorrelation.

Partial correlation is a conditional correlation where the linear dependence of one observation is measured after removing the effect of other observations. Partial autocorrelation often declines rapidly. Due to its effectiveness in identifying and elucidating the internal connection between various variables in time series data, autocorrelation, and partial autocorrelation are both crucial components of time series. Additionally, it aids in determining the internal connection at a specific period.

c) **Data Windowing:** The supervised learning model requires that time series data be prepared in a certain way. The time series data reformatting will easily fit into the deep learning and machine learning algorithms, both linear and non-linear. We can reconstruct a sequence of time-series data so that it resembles a supervised learning problem if we already know the data. To achieve that, we may use data from earlier timestamps as input and the result of the subsequent phase as output. Data windowing is the process of using the prior timestamp as input and the subsequent timestamp as output [39]. The terms "window width" or "the number of lags" both refer to the quantity of preceding timestamps. The sliding window can be used in both univariate and multivariate time series analysis.



t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} .... t_n
X_1 X_1 X_1 X_1 Y_1
X_2 X_2 X_2 X_2 Y_2
X_3 X_3 X_3 X_3 Y_3
X_n X_n X_n X_n Y_n
X input
Y output

Figure 2.15: Data Windowing Technique

When a sliding window is used to forecast the next step, it is known as one-step forecasting, and when multiple timestamps are forecasted, it is known as multi-step forecasting. Figure 2.15 shows the data windowing technique in time series data.

d) **Outlier Detection:** An observation must meet requirements before being labelled an outlier. Time-series observations are considered outliers when there is a sudden peak and a quick decline in the data. Moreover, when any time-series observations differ significantly from the pattern of other data, the observations are considered as outliers. Researchers may establish certain standards or metrics for how they will treat outliers. The Rolling Statistical Bound-based method, Isolation Forest, and K-means There are three ways to identify outliers in time series data [40].

References

[1] Liao, H., Richard Lin, C., Lin, Y., & Tung, K. (2013). Intrusion detection system: A comprehensive review. *Journal Of Network And Computer Applications*, *36*(1), 16-24. https://doi.org/10.1016/j.jnca.2012.09.004

[2] Kushner, D. (2013). The real story of stuxnet. *IEEE Spectrum*, *50*(3), 48-53. https://doi.org/10.1109/mspec.2013.6471059

[3] Gervais, M. (2011). Cyber Attacks and the Laws of War. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.1939615

[4] *Digital Identity and Security*. Thales Group. (2022). Retrieved 29 April 2022, from https://www.thalesgroup.com/en/markets/digital-identity-and-security.

[5] (2022). Retrieved 29 April 2022, from https://www.senki.org/ddos-attackpreparation- workbook/history-of-denial-of-services-dos-attacks/.

[6] (2022). Retrieved 28 April 2022, from https://www.washingtonpost.com/archive/business/2002/10/23/large-scale-attack-cripples-internet-backbone/dc172cf4-a83e-4137-a74d-acc402d5b10c/.

[7] Trautman, L. (2016). Corporate Directorss and Officerss Cybersecurity Standard of Care: The Yahoo Data Breach. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2883607

[8] *Amazon 'thwarts largest ever DDoS cyber-attack'*. BBC News. (2022). Retrieved 28 April 2022, from https://www.bbc.com/news/technology-53093611.

[9] Kemmerer, R., & Vigna, G. (2002). Intrusion detection: a brief history and overview. *Computer*, *35*(4), supl27-supl30. https://doi.org/10.1109/mc.2002.1012428

[10] Yost, J. (2016). The March of IDES: Early History of Intrusion-Detection Expert Systems. *IEEE Annals Of The History Of Computing*, *38*(4), 42-54. https://doi.org/10.1353/ahc.2016.0043

[11] Barber, R. (2001). The Evolution of Intrusion Detection Systems — The Next Step. *Computers &Amp; Security*, *20*(2), 132-145. https://doi.org/10.1016/s0167-4048(01)00204-8

[12] McHugh, J. (2001). Intrusion and intrusion detection. *International Journal Of Information Security*, *1*(1), 14-35. https://doi.org/10.1007/s102070100001

[13] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1). https://doi.org/10.1186/s42400-019-0038-7

[14] Samrin, R., & Vasumathi, D. (2018). Hybrid Weighted K-Means Clustering and Artificial Neural Network for an Anomaly-Based Network Intrusion Detection System. *Journal Of Intelligent Systems*, *27*(2), 135-147. https://doi.org/10.1515/jisys-2016-0105

[15] Significant Cyber Incidents | Center for Strategic and International Studies. Csis.org. (2022). Retrieved 28 April 2022, from https://www.csis.org/programs/strategic-technologies-program/significant-cyber-incidents.

[16] Gupta, B., & Badve, O. (2016). Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a Cloud computing environment. *Neural Computing And Applications*, *28*(12), 3655-3682. https://doi.org/10.1007/s00521-016-2317-5

[17] Bosman, E., & Bos, H. (2014, May). Framing signals-a return to portable shellcode. In *2014 IEEE Symposium on Security and Privacy* (pp. 243-258). IEEE.

[18] Jafarian, J. H., Al-Shaer, E., & Duan, Q. (2015). An effective address mutation approach for disrupting reconnaissance attacks. *IEEE Transactions on Information Forensics and Security*, *10*(12), 2562-2577.

[19] Ochieng, N., Mwangi, W., & Ateya, I. (2014). A Tour of the Computer Worm Detection Space. *International Journal Of Computer Applications*, *104*(1), 29-33. https://doi.org/10.5120/18169-9045

[20] Krenker, A., Volk, M., Sedlar, U., Bešter, J., & Kos, A. (2009). Bidirectional Artificial Neural Networks for Mobile-Phone Fraud Detection. *ETRI Journal*, *31*(1), 92-94. https://doi.org/10.4218/etrij.09.0208.0245

[21] *5G*. Future Networks. (2022). Retrieved 28 April 2022, from https://www.gsma.com/futurenetworks/ip_services/understanding-5g/.

[22] *What is 5G? Benefits of 5G Network Technology Explained*. (2022). Retrieved 28 April 2022, from https://www.verizon.com/about/our-company/5g/what-5g.

[23] *Kevin Ashton Invents the Term "The Internet of Things" : History of Information*. Historyofinformation.com. (2022). Retrieved 28 April 2022, from https://www.historyofinformation.com/detail.php?id=3411.

[24] Gai, R., Du, X., Ma, S., Chen, N., & Gao, S. (2021, March). A Summary of 5G applications and prosprcts of 5G in the Internet of Things. In *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)* (pp. 858-863). IEEE.

[25] Qualcomm, W. (2022). *What is 5G | Everything You Need to Know About 5G | 5G FAQ | Qualcomm*. Qualcomm. Retrieved 29 April 2022, from https://www.qualcomm.com/5g/what-is-5g.

[26] Osseiran, A., Boccardi, F., Braun, V., Kusume, K., Marsch, P., & Maternia, M. et al. (2014). Scenarios for 5G mobile and wireless communications: the vision of the METIS project. *IEEE Communications Magazine*, *52*(5), 26-35. https://doi.org/10.1109/mcom.2014.6815890

[27] Chen, D., & Varshney, P. K. (2004, June). QoS Support in Wireless Sensor Networks: A Survey. In *International conference on wireless networks* (Vol. 233, pp. 1-7).

[28] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings Of The IEEE*, *86*(11), 2278-2324. https://doi.org/10.1109/5.726791

[29] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[30] *Support Vector Machine (SVM) Algorithm - Javatpoint*. www.javatpoint.com. (2022). Retrieved 29 April 2022, from https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.

[31] *Binary Image classifier CNN using TensorFlow*. Medium. (2022). Retrieved 29 April 2022, from https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697.

[32] *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. Medium. (2022). Retrieved 29 April 2022, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[33] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5*(2), 241-259.

[34] Deng, L., & Yu, D. (2011). Deep convex net: A scalable architecture for speech pattern classification. In *Twelfth annual conference of the international speech communication association*.

[35] Welchowski, T., & Schmid, M. (2016). A framework for parameter estimation and model selection in kernel deep stacking networks. *Artificial intelligence in medicine*, *70*, 31-40.

[36] Basic architecture of RNN and LSTM. (2022). Retrieved 19 July 2022, from https://pydeeplearning.weebly.com/blog/basic-architecture-of-rnn-and-lstm

[37] How to Remove Trend & Seasonality from Time-Series Data in Python? by Sunny Solanki. (2022). Retrieved 19 July 2022, from https://coderzcolumn.com/tutorials/data-science/how-to-remove-trend-and-seasonality-from-time-series-data-using-python-pandas

[38] Time Series Analysis with Theory, Plots, and Code Part 1. (2022). Retrieved 19 July 2022, from https://towardsdatascience.com/time-series-analysis-with-theory-plots-and-code-part-1-dd3ea417d8c4

[39] Brownlee, J. (2022). Time Series Forecasting as Supervised Learning. Retrieved 19 July 2022, from https://machinelearningmastery.com/time-series-forecasting-supervised-learning/

[40] Pre-processing of Time Series Data. (2022). Retrieved 19 July 2022, from https://medium.com/enjoy-algorithm/pre-processing-of-time-series-data-c50f8a3e7a98

[41] Sadasivam, G. K., Hota, C., & Anand, B. (2016, September). Classification of SSH attacks using machine learning algorithms. In *2016 6th International Conference on IT Convergence and Security (ICITCS)* (pp. 1-6). IEEE.

[42] Li, J., Zhao, B., & Zhang, C. (2018). Fuzzing: a survey. *Cybersecurity*, *1*(1), 1-13.

[43] Linear Regression — Detailed View. (2022). Retrieved 28 July 2022, from https://towardsdatascience.com/linear-regression-detailed-view-ea73175f6e86

Chapter 3

AI-Based Network Intrusion Detection System

3.1 Introduction

Since its inception, the internet has evolved into a very effective platform for almost all human activities. Moreover, when the Covid-19 outbreak began two years ago, the internet became a lifeline for people in practically every industry. According to the Pew Research Center, 90% of Americans regarded the internet as a vital part of their daily life during the coronavirus pandemic, with the majority of them using it for business, video meetings, academic, and communication purposes [1]. As per Datareportal, a total of 4.94 billion individuals, or 62.4 percent of the world's people, use the Internet [2]. Besides that, the deployment of 5G and its advanced features is expected to boost the number of 5G users to 1.02 billion in the coming year [3]. As the number of internet users grows and the network infrastructure becomes more complicated, safeguarding the network from various internet threats is becoming more complex. Cyberattacks on both network and host levels have increased in recent years, disrupting a broad range of business and government domains. According to Check Point Research, the weekly incidence of cyberattacks soared to 50% during the Covid-19 pandemic, with education and research being the most afflicted sectors [4]. Meanwhile, attackers are adopting novel strategies to bolster 5G networks. It is critical to develop a system like an Intrusion Detection System (IDS) that can detect the vulnerabilities of a network effectively and efficiently to protect against cyberattacks. Because of the dynamic nature of diverse cyber-attack patterns, AI-based technologies like machine learning (ML) and deep learning (DL) can become viable alternatives for constructing NIDS.

A cyberattack is an offensive activity directed toward different systems including, targeted government information systems, private network infrastructures, or personal computer devices, to access or modify valuable data, install malicious software, or disrupt regular services without the permission of the administrator, to gain financial benefits or as a form of cyber-warfare. A cyberattack can be targeted or untargeted, with targeted attacks attempting to infiltrate specific businesses to disrupt their services or get access to sensitive data, and untargeted attacks trying to break as many systems or devices as possible to disrupt the victim's services. Targeted cyberattacks include phishing, ransomware, and scanning, while untargeted assaults include building a botnet and spear-phishing. According to Cyber Kill Chain, most cyber-attacks begin with four basic steps: survey, delivery, breach, and affect [5].

Nowadays, a cyberattack is widespread, and attackers take many different approaches and methods to initiate an attack. Malware, Phishing, DoS, DDoS, Reconnaissance, Backdoors etc. are some common techniques of cyberattacks. In 2021 a large-scale cyber-attack hampered around nine US government agencies, where attackers deployed untraceable backdoors in Microsoft Exchange and compromised the access of the server to gain all information residing inside the server [6]. According to CSIS (Center for Strategic and International Studies), a Russian hackers group launched an "information attack" in February 2022 to obtain access to two of Ukraine's financial institutions, causing service outages [7]. Throughout the first half of 2021, 5,591 network layer DDoS attacks were reported, according to the "2021 DDoS Threat Landscape Report," and the attack technique is changing in dimension, volume, regularity, and complexity [8].

In order to detect and prevent cyberattacks on the networks, firewalls are used as the primary defence mechanism, and a network intrusion detection system, or NIDS, is employed as an advanced security mechanism. NIDS is mainly a security application that can analyze network traffic, detect security threats, and take measures when an anomaly is detected. Based on Peng et al. [9] IDSs should consist of three main steps: first, IDS needs to collect and track the network flow data. After that, IDS needs to reformat and reshape the network data by cleaning the raw data, processing them, and making them feasible in the next step. Finally, using the processed data as input, a model will determine if network traffic is normal or possesses any abnormalities. Developing an effective and efficient model is the most important part of building a NIDS. Models developed by integrating ML and DL can learn the attack signatures from which it can create the attack pattern. However, DL techniques outperform traditional machine learning-based approaches when the dataset is too large and complex. DL has the capability to handle high-dimensional imbalanced data [9, 10]. Based on the above context, to build an effective IDS model, in this thesis, we employed DL approach and used Ensemble techniques.

Many previous studies have used UNSW-15 and CICIDS-2017 to detect network abnormalities, however, those works used either binary classification or shallow ML and DL approaches. To address the previous challenges, this thesis proposes a DL-based stacking ensemble technique to build an efficient and dynamic IDS which can detect a wide range of cyberattacks. In this context, the main contributions of this thesis are as follows:

- Our suggested model uses two publicly available large-scale datasets containing eleven different types of cyberattacks, ensuring that it is robust enough to be used in real-world circumstances.
- We performed a sampling technique using SMOTEENN to under-sample the majority class and oversample the minority attack types.
- Our research proposes a multi-classifier for network intrusion detection that uses a hybrid deep learning model to accurately classify various cyberattack types.
- Our ensemble stacking deep learning model outperforms any single deep learning model or a shallow machine learning model in terms of accuracy and error rate.

The rest of the thesis is organized as follows: An overview of the current literature is provided in section 3.2. The suggested system model and technique are then discussed in section 3.3. Following that, the system design and evaluation metrics are presented in section 3.4 and section 3.5, respectively. Section 3.6 discusses the performance comparison of different deep learning models, followed by concluding remarks in section 3.7.

3.2 Relevant Work

This chapter will review the related literature of previous works in the network intrusion detection system, or NIDS, field as well as pinpoint the shortcomings of existing studies, which is the motivation for this research. This section discusses three distinctive phases of developing a NIDS: statistical based approach, machine learning based approach and deep learning based approach.

3.2.1 Statistical Based Approach

In 1988, Stephen E. Smaha introduced "Haystack," an intrusion detection strategy that could notify any unusual events in a multiuser computer system [15]. Haystack was designed to detect any inappropriate user behaviour as an infiltration that could compromise an organization's security and administrative policies. Haystack was inspired by a previous study proposed by Jim Anderson in 1980, in which the system scanned a computer's daily operations data to uncover usage trends [16]. Both Haystack and Jim's work used the statistical approach, relying on massive trail files compiled from all of the business’s computers. Some big IT-based companies began using IDSs to safeguard their networks around the beginning of 1990. IDES was used as a basis model by AT&T Bell Labs while developing its IDS. Nong Ye proposed a multivariate statistical analysis to detect host-based intrusion in 2002, where he recorded the normal activities for a long time and utilized it to detect abnormal activities [17]. To analyze the normal activities and determine the anomalies, this work used T^2 test proposed by Hotelling’s and compared the performance with the Chi-squared distance test. In 2007, Sathish Kumar and other researchers provided a methodology for analyzing raw network data in which thresholds are set using the six-sigma technique and these thresholds are used to identify the network's normal, uncertain, and anomalous states. [18]. The lack of real-time network traffic resources made developing an IDS difficult in the early stage. However, developing an efficient IDS system requires a well-labeled dataset, and the University of California's publication of the KDD-99 intrusion detection dataset paved the way for researchers to detect network intrusion in a more sophisticated manner using data mining and machine learning techniques.

3.2.2 Machine Learning Based Approach

In 1999, Chris Sinclair used machine learning algorithm based expert system to automatically classify a network connection by utilizing network patterns [19]. They employed genetic algorithm and decision tress algorithm to develop the IDS to detect “low and slow” attacks which may contain intrusion behaviour. Zheng et al. presented a Hierarchy-based Network Intrusion Detection System (HIDE) using a hybrid model comprised of perceptrons and backpropagation to distinguish normal, as well as abnormal traffic, flows [21]. The HIDE model is composed of three tiers and each tier has multiple agents to detect network intrusion. Tier-1 preprocesses the network traffic collected by probe and sends periodic report about the traffic into Tier 2. Tier 2 observes the LAN to check network status and finally Tier 3 receives data from both Tier 1 and Tier 2. After preprocessing, all data are sent to the statistical processor to convert them into stimulus vectors and feed them to the perceptron and backpropagation processor to classify traffic. Maheshkumar et al. employed an ML-based approach in the KDD-99 datasets to detect cyber-attacks launched from user-to-root as well as remote-to-local [20]. Their research focused on comparing the effectiveness of different machine learning methods in detecting various forms of cyber-threats, and they concluded that their four distinct classification algorithms perform better in detecting four different attack types.

A network intrusion detection technique using unsupervised machine learning (ML) was first presented by Eskin et al. in 2002. It includes clustering algorithms, a support vector machine (SVM), and the K-Nearest Neighbor algorithm [22]. This research introduces a geometrical paradigm for unsupervised anomaly detection that maps typical metadata into a feature space. Chih-Fong Tsai and his research team reviewed 55 machine learning-related intrusion detection systems developed between 2000 to 2007 [23]. They divided the system into the single, hybrid, and multi-classifier and compared them based on datasets used, classification method, and experimental setup. Their findings conclude that K-NN and SVM are more popular single classifiers, integrated-hybrid classifiers are the most used hybrid classifiers and ensemble techniques did not receive much attention. Phurivit et al. [24] developed a real-time IDS named RT-IDS, which could distinguish between normal traffic and anomalies. Authors identify 12 essential features from network traffic that are important for detecting network anomalies, and several ML algorithms such as DT (Decision Tree), Neural Network with back-propagation are used, where DT with rippler rule outperforms all other algorithms. A. Haque et al. [25] proposed a hybrid NIDS by incorporating random forest algorithm-based machine learning methodologies to detect misuse and anomaly in the network. For misuse detection, they used random forests to identify patterns from intrusions and compare network activities with patterns to identify intrusions. The authors used an unsupervised learning technique to train the anomaly detection components to detect anomalies and outliers in network traffic flow. Finally, this work combines misuse detection with anomaly detection, allowing anomaly detection to detect novel threats while misuse detection filters out known intrusions. P. Mishra et al. compared and analyzed ML techniques to detect different types of cyberattacks [26]. Authors mentioned ML techniques have drawbacks during multi-class attack detection such as low attack detection rates, algorithmic failure during learning patterns, and high computational cost.

3.2.3 Deep Learning Based Approach

A research team from the University of Toronto led by Geoffrey Hinton proposed ImageNet and addressed that deep learning can outperform any machine learning algorithms in image classification tasks [27]. In the LSVRC-2010 contest, they trained 1.2 million images using Convolutions neural networks (CNN) with thousand features and achieved low error rate than any other advanced machine learning model and in ILSVRC -2012 contest, they achieved only 15.3% error rate in the test data which outperforms every other model. In 2015 Yann et al. addressed traditional machine learning cannot process raw forms of natural data like pixel values of images, matching news items, user interests in a particular product and many more [28]. However, deep learning makes up multiple levels of representation layers, which takes raw input in one representation layer and transforms it into a higher layer that helps them learn complex functions more abstractly.

In 2016 Niyaz et al. proposed a DL-based strategy named STL (self-taught learning) to detect network anomalies using NSL-KDD dataset [29]. Their self-taught learning approach can learn about features from different network sources. These obtained preprocessed features are passed through the auto-encoder and regression with the SoftMax function to classify normal and attack traffics. Their experiments outperform some other research works conducted using deep learning techniques. Ana et al. [30] made a comparison between DL along with ML-based methods in order to identify cyberattacks on the networks using CICIDS-2018 and CICIDS-2017 dataset. Authors used KDD Cup 1999 dataset for their IDS by using DL based method named deep neural network (DNN) and ML based method named support vector machine (SVM), where the accuracy of DNN is 15% higher than SVM. Before 2016 most of the researchers used KDD98, KDDCUP99, and NSLKDD dataset for their IDS which are decade-old and don’t match the current network scenario. This dataset crisis was solved when in 2015 Moustafa et al. published the UNSW-15 Network flow datasets, which have both normal as well as anomalies network traffic [31]. The UNSW-15 network dataset is more useful than previous datasets to evaluate NIDS perfection because it represents contemporary network traffic contexts.

Hanif et al. utilized Artificial Neural Network (ANN) on detecting network intrusion in Internet of Things (IoT) devices, and in their experiment, they used UNSW-15 dataset [32]. In their experiment, instead of predicting different types of network attacks, they only detect the normal attribute of the traffic. In order to build an IDS for Internet of Things network traffic characteristics in 2018, Moustafa et al. utilized the UNSW-15 dataset where the authors used AdaBoost ensemble algorithms to discern between normal and abnormal traffic [33]. To construct the NIDS dedicated to identifying attacks in IoT networks, this framework focuses on DNS, HTTP protocols together with MQTT and associated flows. They merged three techniques in their framework: ANN, Naive Bayes, and Decision Trees, and then passed them through AdaBoost ensemble methods. The deep learning-based NIDS got more attention to the research community when the Canadian Institute of Cybersecurity (CIC) published two large network traffic datasets, namely CICIDS-2017 [34] and CICIDS-2018 [35].

In 2021 Aleesa et al. [36] UNSW-15 datasets and proposed a deep learning technique for both two levels for network intrusion detection [36]. The authors' suggested framework had two levels: level 1 could determine if traffic was normal or abnormal, and level 2 could classify the many forms of assaults when abnormal traffic was discovered. In this work, instead of incorporating any deep neural networks, shallow machine learning models are used. In order to evaluate deep learning and machine learning-based methods for detecting network anomalies, Liu et al. presented a taxonomy for NIDS [37]. The proposed taxonomy in this work can answer several questions about feature selection, data type selection to predict certain types of attacks as well as ML and DL model selection based on the type of network data available. Injadat et al. [38] introduced a multi-level optimum ML approach for network anomaly detection. Their strategy showed a 99 percent detection accuracy and reduced the number of false positives by 1–2 percent using the CICIDS-2017 and UNSW-15 datasets. Faker et al. employed Big Data as well as DL methods and used

UNSW-15 and CICIDS-2017 datasets to evaluate the model performance [39]. They incorporated Deep Feed-Forward Neural Network (DNN), Random Forest and Gradient Boosting Tree (GBT) where DNN shows better performance in both datasets for multiclass classification and GBT shows better performance in CICIDS-2017 during binary classification compared to DNN.

Unlike the above-mentioned research efforts, we used two different recent network traffic datasets, UNSW-15 released by the University New South Wales [11] and CICIDS-2017 published by the Canadian Institute of Cybersecurity (CIC) [12], to train our DL-based stacking ensemble model and analyze how it performs in different attack scenarios. Many researchers proposed ML and DL-based techniques to detect and predict network anomalies. Moreover, some researchers proposed DL-based methods, but most of them are focused on binary classification and used shallow ML and DL methods. Moreover, some researchers used an out-of-date dataset where traffic patterns no longer resemble today's diverse network traffic [13 ,14]. In our proposed approach, we used data resampling technique to balance the imbalanced data samples and employed stacking ensemble technique composed of different DL models to detect the different kinds of cyberattacks.

3.3 Proposed Model and Methodology

In this section, we will explore our proposed models along with approaches directed toward developing a unique multi-classifier-based NIDS system. The various deep learning architectures demonstrated in this thesis will be described in Section 3.3.1. The stacking ensemble methodologies will be discussed in Section 3.3.2 to ensure that our IDS system achieves optimal accuracy in the real-world context.

3.3.1 Deep Learning Architecture

In this thesis, we employed the stacking ensemble technique, which is comprised of different deep learning models. Our overall stacking ensemble architecture is comprised of two layers, in the first layer we employed three different DL based architectures for our IDS system, which consists of Convolution Neural Nets (CNN), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM). In the second layer, we used Deep Neural Network (DNN) which takes prediction results from the previous three DL models. Each of the DL architectures has its input layers, hidden layers, fully connected layers, and output layers for multi-classification.

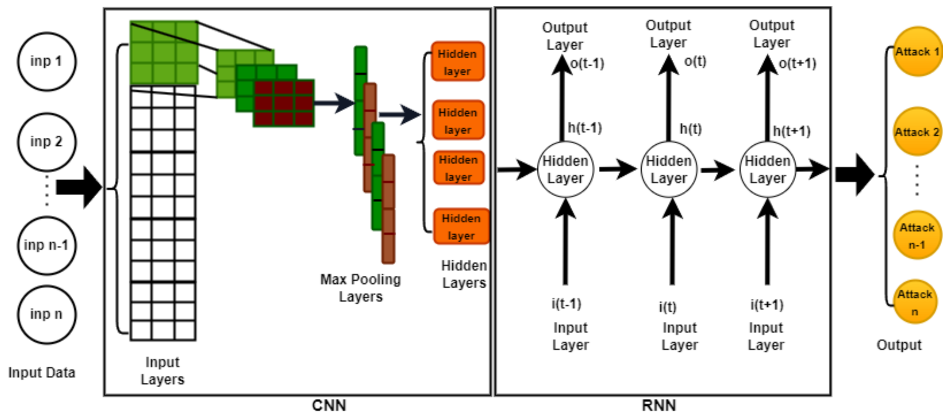


Figure 3.1: IDS Deep Learning Model

Figure 3.1 represents the comprehensive architecture of DL models utilized in our NIDS system, proposed DL models first take the preprocessed input data and feed them into the deep learning models. We employed both CNN and RNN in our IDS Model. The CNN model combines the extracted features with different size of kernels and pass it to the pooling layers. Finally, from the pooling layers all the information fit to the hidden layer which can apply weights to input and utilize activation function as the output. The hidden layers section comprises densely linked layers where every neuron receives the output of the previous layer as their input. Various terms and building blocks are explained below for a better understanding of the models.

RNN takes the output of a particular layer and feed them back to the input to predict the output of a particular layer. The input layer i takes input, process it and pass it to the hidden layer h and output layer o generates the results. In any time period t, the input combination is current timesteps i(t) and and previous timesteps i(t-1), and to improve the performance, the output is again fetched back to the network.

Activation Function: By using the activation function, the output of a neural network can be either yes or no. The output values are ranged from 0 to 1 or from -1 to 1. For neural networks, an activation function is necessary so that the model can understand relevant features and noise**.** Every layer of LSTM as well as CNN is composed of an activation function named Rectified Linear Unit or ReLU, which is by default incorporated in CNN. It also aids the CNN model in learning faster and performing better, as well as overcoming the problem of disappearing gradients. The key rationale for using ReLU for CNN and LSTM is that it has a simple computation complexity because it merely compares input data to the value 0. It also has a result of 0 or 1, depending on whether the input is positive or negative. This feature aids the model during backpropagation and prevents it from growing exponentially.

A differentiable hyperbolic tangent activation function (Tanh) is dealt with by each layer of GRU. Tanh reduces the range of real numbers between -1 to +1. Despite being a system with non-linear functionalities, it provides zero centric output, different from the sigmoid function. Because of their symmetry around the origin, this is the case. As a result, they produce outputs that are close to zero. The Outputs that are closed to zero are considered the best output because, during model optimization, the rate of weight swing is fewer, which provides faster model convergence.

The Dropout Layer: Any DL model has a dropout layer, which is a frequent feature. The Dropout layer is a mask that removes some neurons' contributions to the following layer while leaving all others unchanged. It can be used to negate some neurons from hidden layers in either the input layers or the hidden layer. Another key function of the dropout layer is to prevent the training dataset from becoming overfit. For hidden layers, the ideal dropout value is between 0.5 and 0.8, whereas for the input layer, it is 0.8. In our DL models, we used different settings of the dropout layer in both the dense and input layer.

The SoftMax Activation Function: A model's output layer is created using a SoftMax activation function, which combines multiple sigmoids. The model's fundamental purpose, which is classed in a multiclass environment, is related to the function's selection. The SoftMax function returns an array containing the highest probability values, which may be considered the sample's most precise probabilistic label.

In our experiments, we will construct a unique stacking ensemble-based DL model for our Network Intrusion Detection System using a structured method. We applied various DL strategies to adopt the best prototype for our NIDS. We then evaluated every model’s performance and combined them in the first layer of our stacking ensemble model. The first layer of our proposed model is comprised of CNN, LSTM, and GRU. CNN model generally takes an image as input. We reshaped the train and test dataset to (5, 5, 1) dimensions for the UNSW-15 dataset where the model is taking 5×5 matrix and ImageDataGenerator from Keras library generating an image type data. For CICIDS-2017 train and test reshape is (7, 7, 1) dimensions. In our CNN model of the first layer, we used six two-dimensional filters with the kernel size of (2 × 2) for the UNSW-15 dataset and kernel size of (3 × 3) for the CICIDS-2017 dataset. The input samples have the shape of (5,5,1) UNSW-15 dataset and (7,7,1) for the CICIDS-2017 dataset. Every layer of CNN deals with a ReLU activation function. After every two convolution layers, we set one two-dimensional max-pooling layer with different pool sizes.

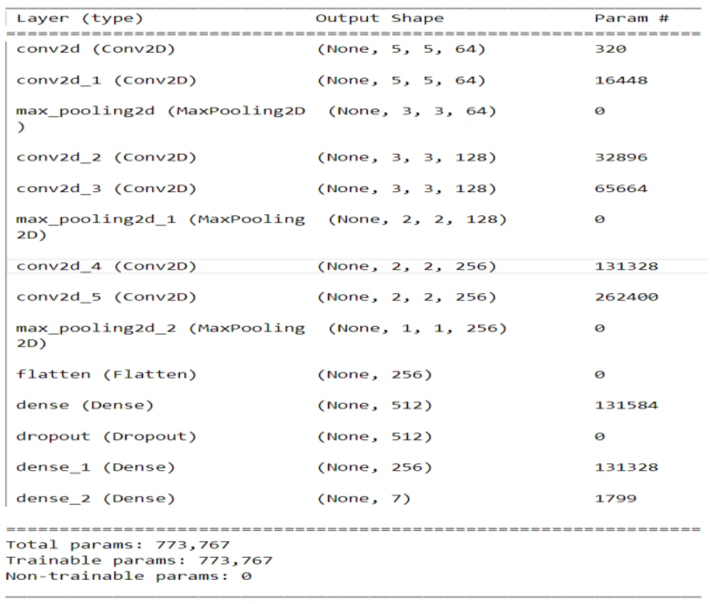
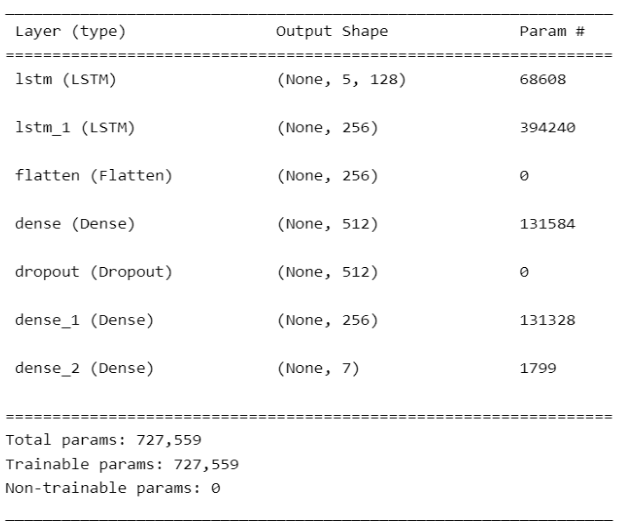

```
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 5, 5, 64)          320

conv2d_1 (Conv2D)           (None, 5, 5, 64)          16448

max_pooling2d (MaxPooling2D  (None, 3, 3, 64)         0
)

conv2d_2 (Conv2D)           (None, 3, 3, 128)         32896

conv2d_3 (Conv2D)           (None, 3, 3, 128)         65664

max_pooling2d_1 (MaxPooling  (None, 2, 2, 128)        0
2D)

conv2d_4 (Conv2D)           (None, 2, 2, 256)         131328

conv2d_5 (Conv2D)           (None, 2, 2, 256)         262400

max_pooling2d_2 (MaxPooling  (None, 1, 1, 256)        0
2D)

flatten (Flatten)           (None, 256)               0

dense (Dense)               (None, 512)               131584

dropout (Dropout)           (None, 512)               0

dense_1 (Dense)             (None, 256)               131328

dense_2 (Dense)             (None, 7)                 1799

=================================================================
Total params: 773,767
Trainable params: 773,767
Non-trainable params: 0
```

Figure 3.2: CNN Model Architecture

The output is then delivered to fully connected layers, where it is used to train representations of higher-order features that may be used to classify the output into distinct class labels. Figure 3.2 shows the configuration of the CNN model used in the first layer of the proposed hybrid model.

The LSTM model of the first layer of our proposed stacking ensemble model consists of two LSTM layers, one dense layer, followed by one dropout layer. Like the CNN model, every layer in the LSTM model interacts with a corrected ReLU activation function. The LSTM model's loss function is categorical cross-entropy, and an Adam optimizer with a learning rate of 0.001 was employed for optimization. The LSTM model's configuration, which is employed in the first layer of the proposed hybrid model, is shown in Figure 3.3.



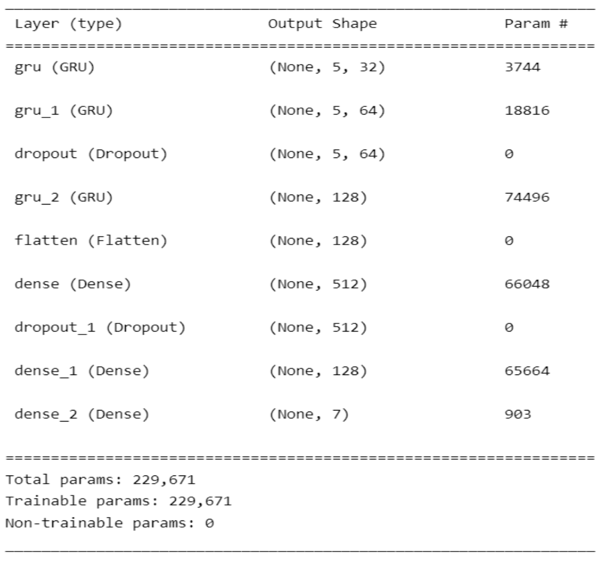
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 5, 128)            68608

 lstm_1 (LSTM)               (None, 256)               394240

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 512)               131584

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 7)                 1799

=================================================================
Total params: 727,559
Trainable params: 727,559
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.3: LSTM Model Architecture

The input data for an RNN (LSTM/GRU) should have three dimensions: (batch size, sequence length or timesteps, and input dimensions or features). Instead of combining the input and prior hidden state, the LSTM cell's four internal neural networks receive both before applying distinct weight matrices to them. The four matrices are referred to as the kernel when they are multiplied by the input and as the recurrent kernel when they are multiplied by the prior hidden state. When an LSTM cell is defined it has hidden units size, activation, and input shape. For LSTM, the input shape is (batch, timestamps, features), and here batch size means the number of samples we send to the model at a time. For UNSW-15, the timesteps are 5, features are 25 and the input shape is (5, 25); for CICIDS-

2017, it is (5, 49). In figure 3.3, we have the output shape of (None, 5, 128), where none means the batch size, 5 represents the number of timesteps and 128 represents the output features.

GRU is used as the last model in the first layer of the proposed stacking ensemble model. The GRU model of the first layer of the proposed hybrid model consisted of three GRU layers with 32, 64, and 128 filters, one dropout layer, and two dense layers. Tangen's hyperbolic activation function (Tanh), as opposed to CNN and LSTM models, is employed with the GRU model. With a learning rate of 0.001, Adam was utilized as the optimizer in this model, while categorical cross-entropy was used as the loss function. Figure 3.4 depicts the GRU model's configuration.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 5, 32)             3744

 gru_1 (GRU)                 (None, 5, 64)             18816

 dropout (Dropout)           (None, 5, 64)             0

 gru_2 (GRU)                 (None, 128)               74496

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 512)               66048

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 7)                 903

=================================================================
Total params: 229,671
Trainable params: 229,671
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.4: GRU Model Architecture

GRU also takes an input of a 3D tensor, with shape (batch size, timesteps, features), here batch size means the number of samples we send to the model at a time. In USNW-15, we take 5 timesteps, and 25 features and the input shape become (5, 25); for CICIDS-2017 we take 5 timesteps and 49 features and the input shape becomes (5, 49). In the model architecture of GRU (figure 3.4), we have output shape of (None, 5, 32), where none means the batch size, 5 reprsents the number of timesteps and 32 represents the output features.

3.3.2 Proposed Stacking Ensemble Architecture

In our proposed hybrid deep learning model called the stacking ensemble model, the meta learner layer is composed of a Deep Neural Network (DNN), and all the results obtained

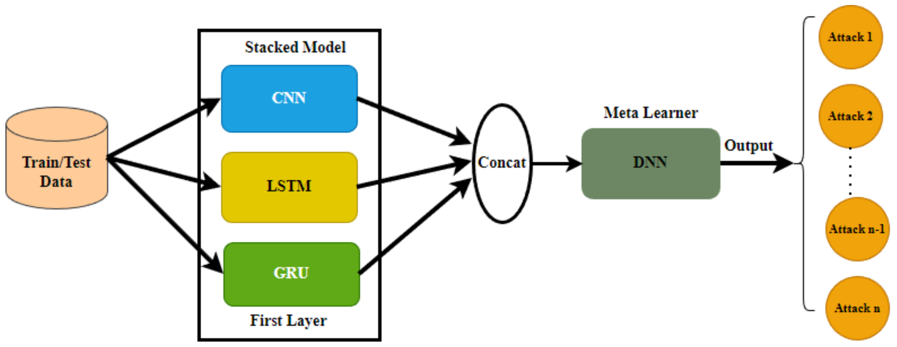


Figure 3.5: Stacked Ensemble Model

from the first three DL models in the first layer are concatenated and delivered to the DNN model. Figure 3.5 is our proposed stacking ensemble technique.

The weights of the prior models used in the first layer will not be changed while the new model is being trained since all the models from the previous layer are loaded as a list and are tagged as being untrainable. This new model uses the output from the prior three models as a separate input head. Using a single concatenation merge, which creates a single vector of 21 components from seven class probabilities, the output of each model is combined. Three distinct DL models, each with seven class labels, were used to build this vector of 21 elements. The meta learner will then use these two hidden layers to analyze the input, and an output layer will use this information to forecast the probability distributions. The training dataset can be directly fitted to the model after it has been defined. The new stacking model is used to predict the new data once the model has been fitted.

In order to create predictions based on the new data, we may finally employ the layered model. Adam worked as the DNN model's optimizer with a learning rate of 0.001, while categorical cross-entropy functioned as the loss function. The suggested stacking ensemble model's architecture is depicted in Figure 3.6.

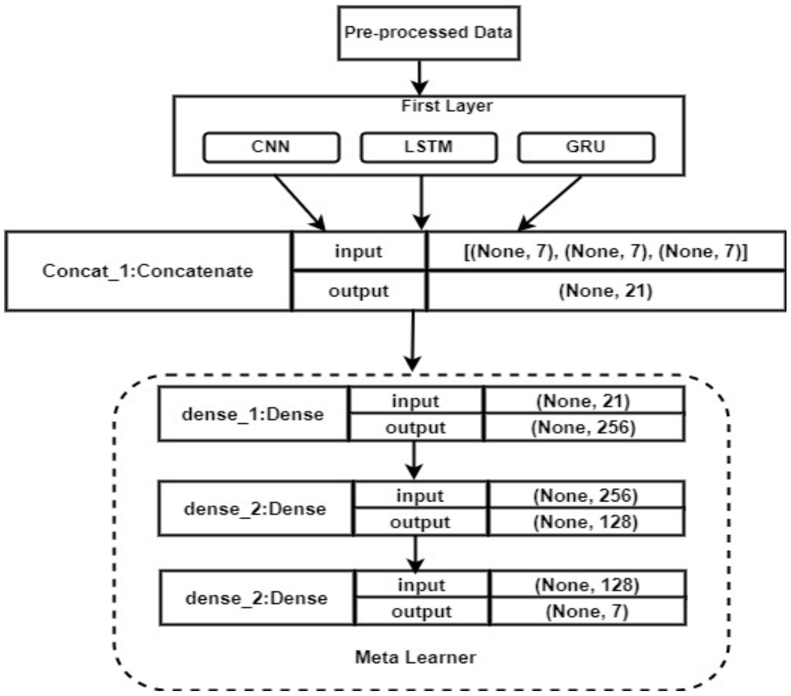


Figure 3.6: Stacking Ensemble System Architecture

3.4 System Architecture

This section will present the proposed Network Intrusion Detection System's (NIDS) architecture. The design of this NIDS system is shown in Figure 3.7, which incorporates stacking ensemble methods made up of different deep learning models to find network anomalies.

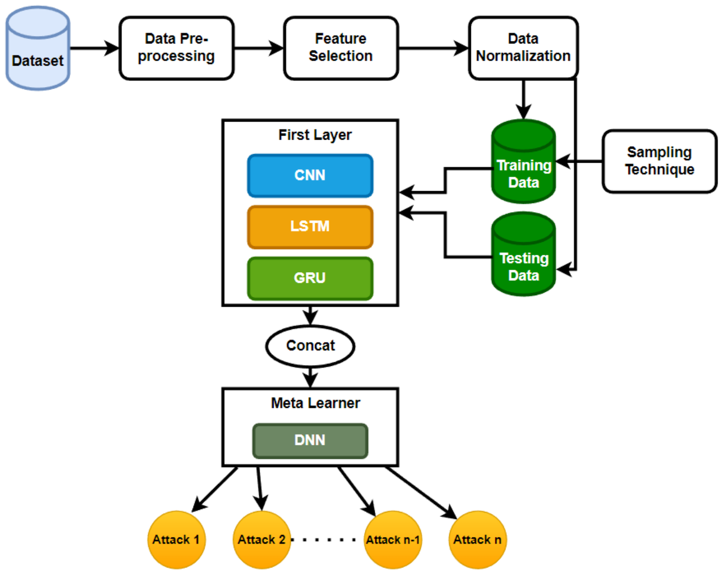


Figure 3.7: IDS System Architecture

3.4.1 Dataset Description

The stacking ensemble-based IDS system was built using two up-to-date datasets, UNSW-15 and CICIDS-2017. The UNSW-15 dataset captures raw packets with the IXIA PerfectStorm program. The Australian Centre for Cyber Security (ACCS) made this network flow data available for public use, which includes both regular and aberrant traffic. Backdoor, Analysis, Worms, Generic, Exploits, Fuzzers, Reconnaissance, DoS, and Shellcode are among the nine categories of network attacks classified as anomalous traffic. In our experiments, we worked with Worms, Generic, Fuzzers, Reconnaissance, DoS and Shellcode. ACCS used three networks with 45 different IP addresses to create the dataset, which took 31 hours to collect. The UNSW-15 dataset is divided into four CSV files and comprises 2.5 million records with 49 features. They also supply a 257,673 normal and abnormal records training and testing subset of the dataset. In our experiment, we used 70% data for training and 30% data for validating and testing the model. The attack distributions in the UNSW-15 dataset are similar to Figure 3.8.

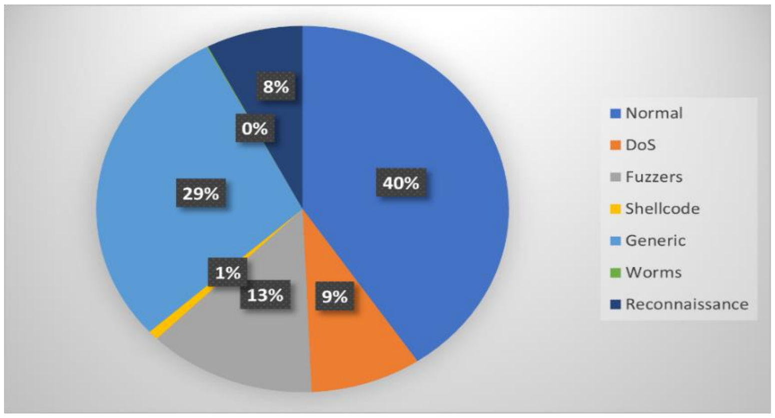


Figure 3.8: UNSW-15 Attack Distribution

The Canadian Institute of Cybersecurity (CIC) published the CICIDS-2017 network traffic data, which contains eight separate CSV files containing five days of regular and aberrant activity. They build an attack-network with a router and a switch for the testbed, and a victim-network with a firewall, router, and switches. They produced normal and anomalous traffic using CICFlowMeter software, and 79 features were collected from traffic-generated pcap files. The Heartbleed attack, Infiltration attack, Brute Force assault, DDoS attack, DoS attack, Web attack, and Botnet are among the cyber-attacks covered in

CICIDS-2017. There are various forms of DoS attacks, including DoS Slowhttp, DoS Hulk, DoS Slow loris, and DoS GoldenEye, and we merged all of them into DoS in our implementation. CICIDS-2017 comprises 209417 records, of which we used 70% in training and 30% is used to validate and test the model. The attack distributions in the CICIDS-2017 dataset are like Figure 3.9.

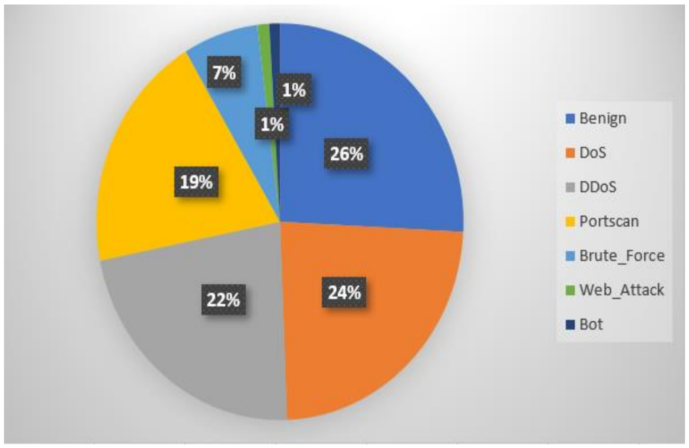


Figure 3.9: CICIDS-2017 Attack Distribution

3.4.2 Data Pre-Processing

We will exclude various features from both datasets that have little impact on normal or abnormal traffic in the initial round of our data pre-processing phase. Time-based features ('stime' and 'ltime') are removed from the UNSW-15 dataset since they are redundant with switch-related information like 'sport', 'srcip', 'dstip', and 'dsport'. Additionally, categorical features such as 'proto' and 'service' have a wide range of values, thus we employ the label-encoder approach, which generates new, unique numerical values for each category. We also eradicate features like 'ct ftp cmd,''is ftp login,' and 'ct flw http method,' which have a large number of null values.

Less relevant features such as 'timestamps' and 'IP addresses' were eliminated from the CICIDS-2017 dataset. As the network flow of this dataset is created using CICFlowMeter, it collects some network related redundant features such as ‘Bwd PSH Flags’, ‘Bwd Avg Bytes/Bulk’, ‘Bwd Avg Packets/Bulk’, ‘Bwd Avg Packets/Bulk’, ‘Bwd URG Flags’, ‘Fwd Avg Bytes/Bulk’, ‘Fwd Avg Packets/Bulk’, ‘Fwd Avg Bulk Rate’. These features are removed from the dataset because they only contain null values. The 'Fwd Header Length' feature is also excluded as it is existing twice in the dataset.

3.4.3 Feature Selection

Selecting important features is critical for developing a better IDS, as it allows us to exclude less significant aspects from the dataset. We used two very large datasets with many features. We used Random Forest (RF), a supervised machine learning method which employs both the bagging method and decision trees, to identify the optimal features for our IDS model. To calculate the soft voting for classification, RF takes the original columns, fits them into decision trees, and then mixes them. We may also calculate the feature importance using the random forest by increasing the purity of the child nodes. When the purity of its related child nodes is improved, one feature gains relevance. Each tree estimates the importance of each feature, which is then averaged to obtain the total feature importance. We perform feature selection using Random Feature Elimination (RFE) after determining the relevance of all features. RFE performs k-fold cross-validation, removing characteristics that are less significant for the model with each cross-validation. The process continues until RFE has read all of the features from the dataset and only keeps those that improve cross-validation performance overall. Random Feature Elimination with Cross-Validation (RFECV) is a time-consuming method for selecting features, yet it yields the best results. The feature importance of 15 best features of the UNSW-15 and CICIDS-2017 datasets is shown in Figures 3.10 and 3.11.

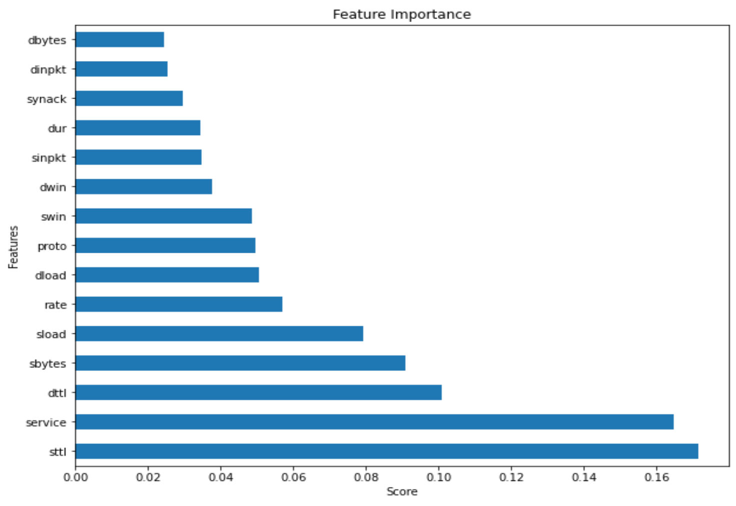


Figure 3.10: Feature Importance of UNSW-15

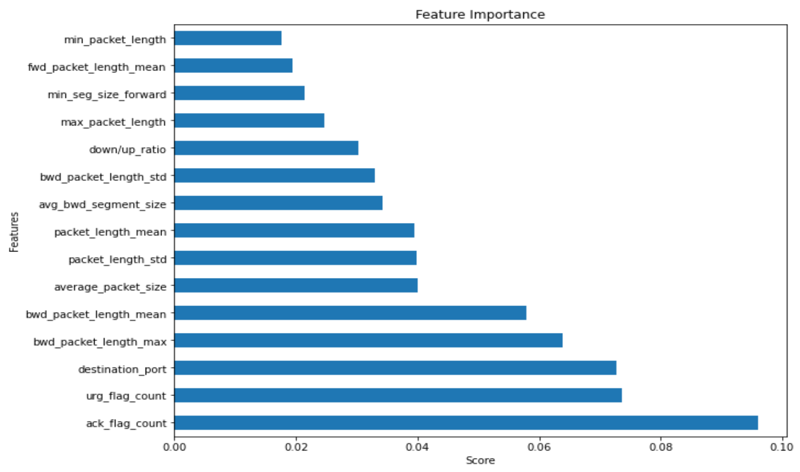


Figure 3.11: Feature Importance of CICIDS-2017

We selected 25 features from UNSW-15 and 49 features from CICIDS-2017 employing the feature selection technique using RFECV with Random Forest.

3.4.4 Data Normalization

Data normalization, also known as scaling the features, is a data pre-processing strategy where all the data transforms into the same scale. In deep learning normalization is a process to transform data within a range between 0 and 1. Moreover, normalization is a good approach when we do not know about the data distribution. Moreover, when we do not know about the data distribution then normalization is a good approach. If we do not normalize our data that are measured in different scales, they will not contribute during model training. Scaling the data equalizes all features, which also assists the algorithm to converge faster, along with optimizing with the gradient descent algorithm. Data distribution is different in both UNSW-15 and CICIDS-2017 dataset, and to fit the data appropriately into the model to get a better accuracy we need to normalize our data. In our model, we employed a data normalization technique named min-max scaler where, it scales the data from minimum range 0 to maximum range 1. Eq. 3.1 is the mathematical formulation for min-max scaler.

$$min - max = \frac{x - min(x)}{max(x) - min(x)} \tag{3.1}$$

3.4.5 Sampling

In a machine learning-based approach, imbalanced data is a classification problem. All the classes are not distributed equally in imbalanced data, meaning that the dataset is biased towards one or more classes, with only a few samples for others. As a result, while training a model using imbalanced data, the model was biased towards one or two classes. The majority class is balanced using the under-sampling technique, while the minority class is balanced using the over-sampling technique. To overcome the imbalanced class problems Synthetic Minority Oversampling Technique (SMOTE) is used [42] [43]. The less frequent samples were oversampled using SMOTE, whereas the more frequent samples were under sampled using Edited Nearest Neighbors (ENN). Imbalanced learn, imported as imblearn relying on scikit-learn library, is a widely known machine learning library that deals with imbalanced classes [44]. Both of our datasets are highly skewed. Worms and shellcode traffic have only 130 and 1133 samples in the UNSW-15 dataset, which is relatively low when compared to normal and generic traffic, which includes 56,000 and 40,000 samples, respectively. On the other hand, only 1% of samples in the CICIDS-2017 dataset belong to Web-attack and Bot attack traffic, while Benign, DoS, and DDoS attack traffic each include 26%, 24% and 22% samples, respectively.

SMOTE takes many steps to oversample the minority class:

I. It calculates the distance between each sample using Euclidean distance and then modifies these samples using k-nearest neighbor.

II. Then they take n samples and calculate the imbalance ratio from them, as well as the number of samples that need to be made from the samples.

$$n = round(imbalancedratio) - 1 \tag{3.2}$$

$$imbalancedratio = \frac{S_{max}}{S_{min}} \tag{3.3}$$

III. Finally, the set of produced samples y is taken from the k-nearest neighbor, and new synthetic samples are constructed from those neighbors.

In order to under-sample, the majority class was chosen. The majority of their k-nearest neighbor's sample is removed by ENN. If one sample is owned by a most frequent class also if the classification of the original class is disputed by its three nearest neighbors, it is

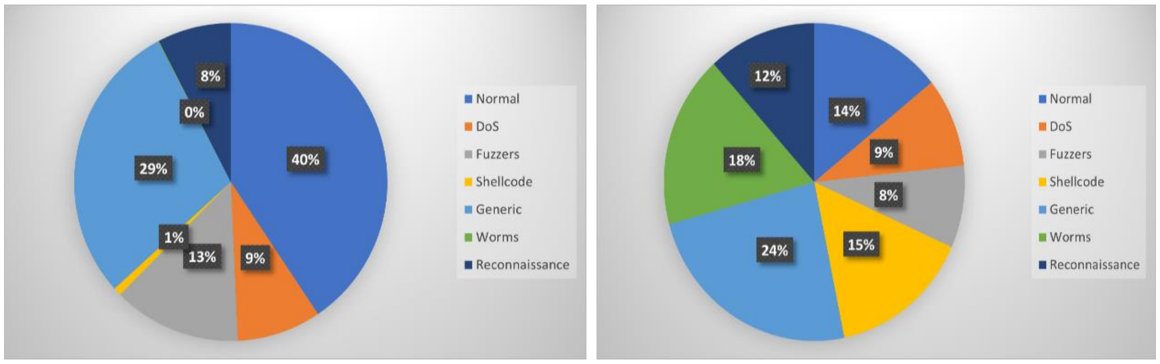


Figure 3.12: a) Before SMOTEENN, b) After SMOTEENN in UNSW-15

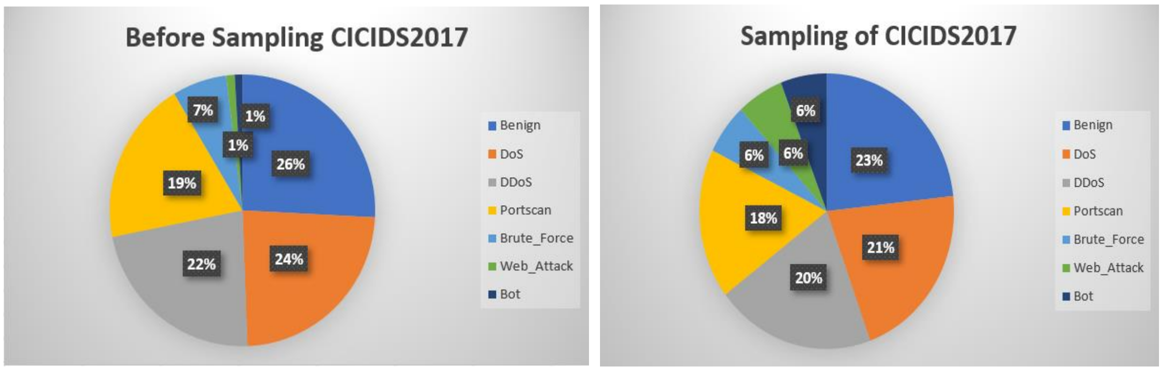


Figure 3.13: a) Before SMOTEENN, b) After SMOTEENN in CICIDS-2017

deleted from the samples; otherwise, it belongs to the minority class. Figure 3.12 depicts the attack distribution of the UNSW-15 dataset before and after data resampling using SMOTEENN and figure 3.13 depicts the attack distribution of the CICIDS-2017 dataset before and after data resampling using SMOTEENN.

3.4.6 Development Environment

We used a Windows 10 PC with an AMD Ryzen 9 5900HX with built-in AMD Radeon graphics to create the stacking ensemble-based IDS model. Our central processing unit has 16 logical processors, 16 GB of RAM, and 512 GB of a solid-state drive (SSD) and runs at 3.30 GHz (SSD). Our IDS model creation is easier and faster now that we have 4GB of NVIDIA GeForce RTX 3050 graphics processing unit (GPU). Throughout the construction of our model, we used a Jupyter notebook and the Visual Studio Code IDE. Python 3.7 is the primary programming language for IDS development, with deep learning framework TensorFlow 2.8.0 [40] and Keras [41] in the backend. We utilized the Pandas package for data analysis, NumPy for numerical analysis, and Matplotlib and seaborn to construct graphs for the experiment results. Furthermore, we employ the Sci-Kit learn machine learning library.

3.5 Evaluation Measures

The evaluation metrics for our suggested IDS stacking ensemble deep learning models will be covered in this section.

3.5.1 Classification Accuracy

In order to evaluate the performance of classification models the term Classification accuracy is interchangeably used with term accuracy. It is the ratio of all correctly predicted values to all input samples that the model has drawn.

$$Accuracy = \frac{Total\, number\, of\, Correct\, predictions}{Total\, number\, of\, predictions} \times 100 \tag{3.4}$$

The formula converts the classification accuracy of the model into a percentile number that may be used to evaluate how well the model was applied. However, classification accuracy does not always correspond to the model's real performance, especially when the misclassification rate for minor classes is high. It also ignores the problem of class imbalance in a dataset, which occurs when the number of positive and negative levels is vastly different. As a result, various performance evaluation indicators must be considered to achieve actual model performance.

3.5.2 Confusion Matrix

A confusion matrix is a graphical depiction of a classification problem's performance that gives the output a matrix format. A confusion matrix is made up of four separate actual and expected values arranged in a matrix. A classification model's outcomes can be characterized as follows using a confusion matrix:

1. **True Positive:** These values are predicted and labelled as positive. For example, if traffic is forecasted as a DoS assault in our IDS system and it is indeed a DoS attack, we can conclude that the IDS made an accurate prediction. A higher true positive number denotes better model performance.

2. **True Negative:** These are the values labeled as negative and also predicted as negative and correct. For instance, in IDS model, if traffic is predicted not as a DoS attack and it was not a DoS attack then the model correctly predicted the traffic. Again, higher true negative rates are the indicator of a good model.

3. **False Positive:** This occurs when a model predicts a positive value for a class, but the actual value is negative. For example, if our IDS model predicted a DoS assault but it turned out to be typical traffic, we'd have a false-positive result. False alarms are generated by a high false-negative rate, which lowers model performance. For a better-performing model, a lower false-positive rate is preferable.

4. **False Negative:** This is used to describe results that were anticipated to be negative but ended up being positive. The model projected the packet to be regular traffic in the NIDS. However, the traffic was actually a DoS attack. Higher false-negative numbers indicate a defective model, and for NIDS false positive is the most important metrics to evaluate system's performance.

Figure 3.14 shows a perceptible representation of a binary classification confusion matrix.

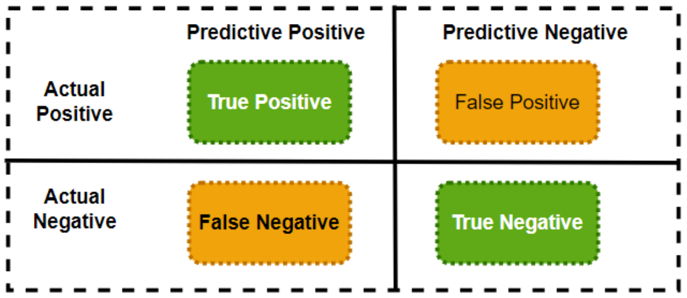


Figure 3.14: Confusion Matrix

The confusion matrix for multi-class classification will be the same size as the number of classes the model must predict.

From the confusion matrix, we can calculate three other important performance metrics namely Recall, Precision and F1-score.

Precision: It's determined by dividing true positives predicted to belong to a specific class by the total number of positive outcomes predicted by the classifier.

$$Precsion = \frac{true\ positives}{true\ positives + false\ positives} \tag{3.5}$$

Recall: It's a metric that measures how well our model detects true positives. It is calculated by dividing the total number of positive values by the number of real positive findings.

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{3.6}$$

F1-Score: It's a metric for determining how well our model detects genuine positives. It's calculated by dividing the number of real positive results by the total number of positive values.

$$f1\ score = \frac{2 * precession * recall}{precison + recall} \tag{3.7}$$

3.6 Experimental Results

In addition to evaluating the efficacy of the suggested strategy utilized in NIDS, findings from individual models and stacking ensemble models will also be examined in this section. The performance of the stacking ensemble model on the UNSW-15 dataset is covered in Section 3.6.1. The effectiveness of the suggested model in assessing the CICIDS-2017 dataset is discussed in Section 3.6.2. In section 3.6.3, we will finally talk about the general findings and outcomes.

As mentioned, deep learning models have the ability to do feature engineering on their own and extract crucial features. Additionally, deep learning algorithms' intricate network designs enable them to construct and learn deeper representations. DL algorithms are selfadjusting, in contrast to ML algorithms, which may need human involvement to be programmed toward an appropriate result. They can be hardcoded the traits for better performance without explicit human interaction. We mainly used three DL algorithms in the first layer of stacking ensemble model and used Deep Neural Network (DNN) in the second layer of the stacking ensemble model as a meta learner.

We will use the two preprocessed partitions of the training and validation dataset as mentioned in section 3.4.1 for training and validating our model. In our experiments, during training the DL models, we used 30 epochs for UNSW-15 dataset and 50 epochs for CICIDS-2017 dataset.

3.6.1 Experimental Results of UNSW-15

In the first layer of the proposed method, the first model we deployed is CNN. During training, we found training accuracy of 91.7% and validation accuracy of 91.0%. As the

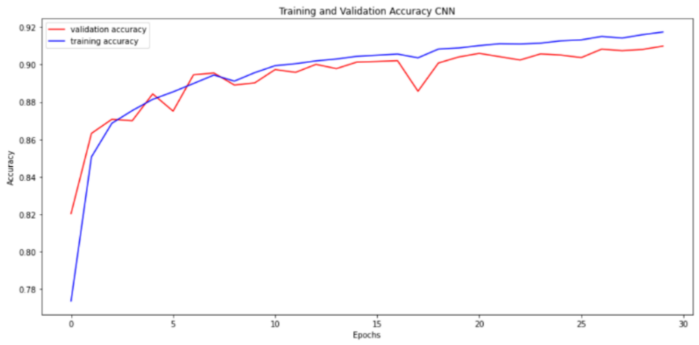


Figure 3.15: Training and validation accuracy of CNN

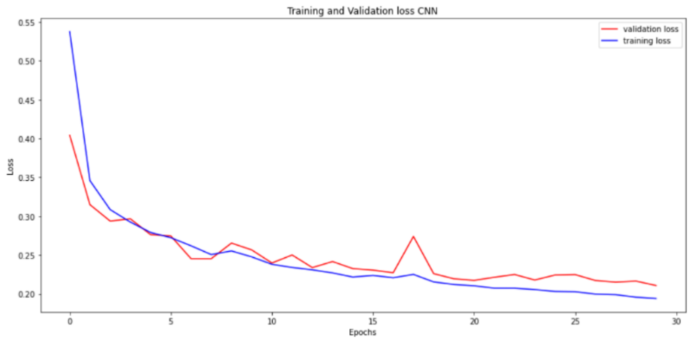


Figure 3.16: Training and validation loss of CNN

model training proceeds, both training and validation loss decrease significantly. The loss and accuracy curves of the CNN model during training are shown in Figures 3.15 and 3.16.

After a predetermined number of epochs, the model training gives the same result for validation accuracy and loss, we stop training CNN in order to avoid our model from overfitting. As the number of epochs rises, the CNN model loses its ability to accurately generalize new data, and we stop training our CNN model at that point.

Our experiments showed that CNN demonstrated 89.1% classification accuracy in test data. The model exhibits the best performance for detecting Generic and Normal traffic, only 2% and 6% detected wrongly. More than 85% of DoS, Shellcode, and Worms assaults can be identified using this methodology. However, this model confounded reconnaissance attack with dos attack and fuzzers attack with normal traffic. The confusion matrix for the CNN model is displayed in Figure 3.17, with the true label acting as the x-axis and the predicted label acting as the y-axis.

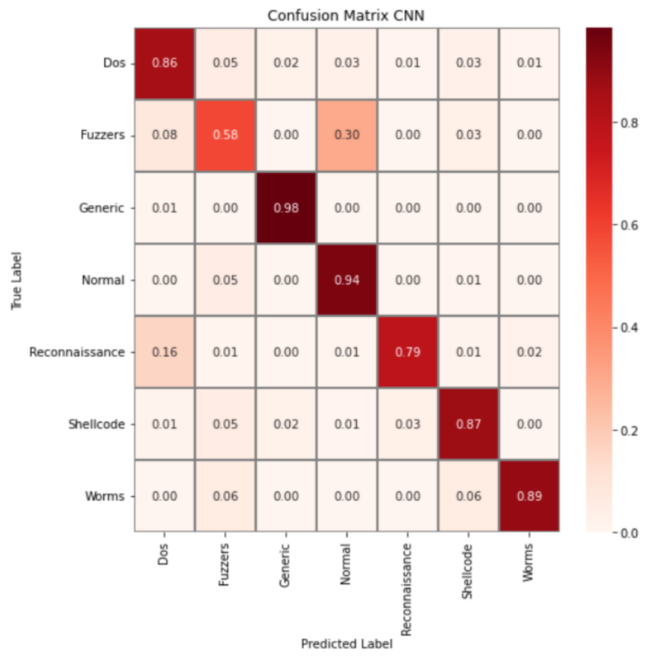


Figure 3.17: Confusion Matrix of CNN

The LSTM model of the first layer exhibits slightly lower performance than the CNN model, where the training and validation accuracy is almost same, and the training and validation loss decreases as the number of epochs increases. Figure 3.18 and 3.19 shows the accuracy and loss curves of training and validation of our LSTM model.

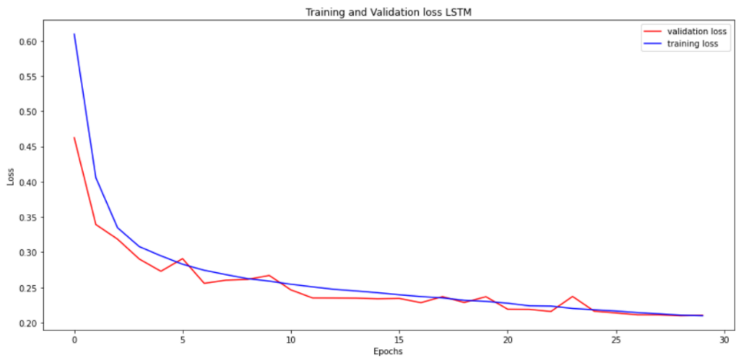


Figure 3.18: Training and validation accuracy of LSTM

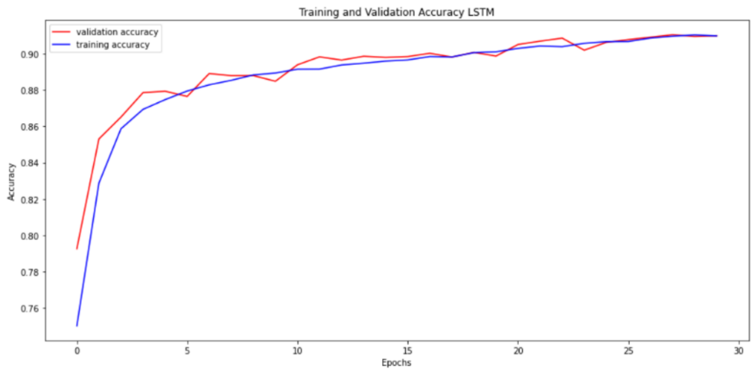
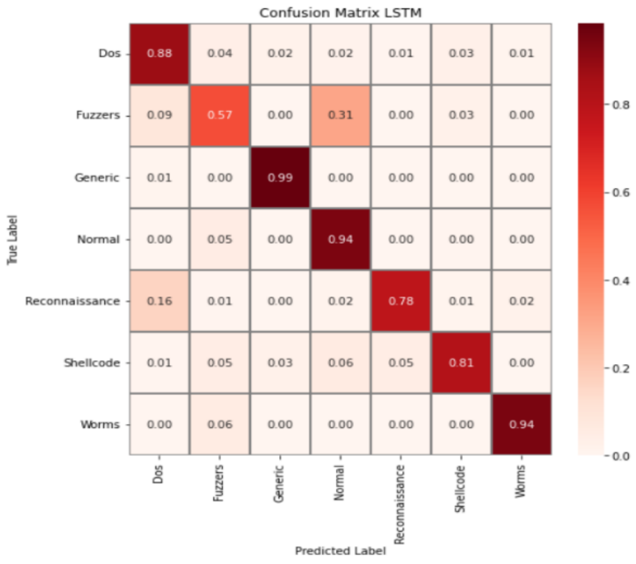


Figure 3.19: Training and validation loss of LSTM

In test data LSTM depicts 89.0% classification accuracy. LSTM performs better in detecting DoS attacks and Worms compared to CNN, while the detection rate of Shellcode attacks is lower than CNN.



	Dos	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms
Dos	0.88	0.04	0.02	0.02	0.01	0.03	0.01
Fuzzers	0.09	0.57	0.00	0.31	0.00	0.03	0.00
Generic	0.01	0.00	0.99	0.00	0.00	0.00	0.00
Normal	0.00	0.05	0.00	0.94	0.00	0.00	0.00
Reconnaissance	0.16	0.01	0.00	0.02	0.78	0.01	0.02
Shellcode	0.01	0.05	0.03	0.06	0.05	0.81	0.00
Worms	0.00	0.06	0.00	0.00	0.00	0.00	0.94

Figure 3.20: Confusion Matrix of LSTM

Apart from that, the detection rate of CNN and LSTM is almost similar for other types of attacks. The confusion matrix of the LSTM in Figure 3.20 displays the distribution of different attacks.

The last model of the first layer is GRU, and its training and validation accuracy is lower than CNN and LSTM. There is no significant difference between the training accuracy and validating accuracy of GRU. Also in both cases, losses decrease significantly. Figure 3.21 and 3.22 depicts the loss and accuracy curve of the GRU model during the training process.

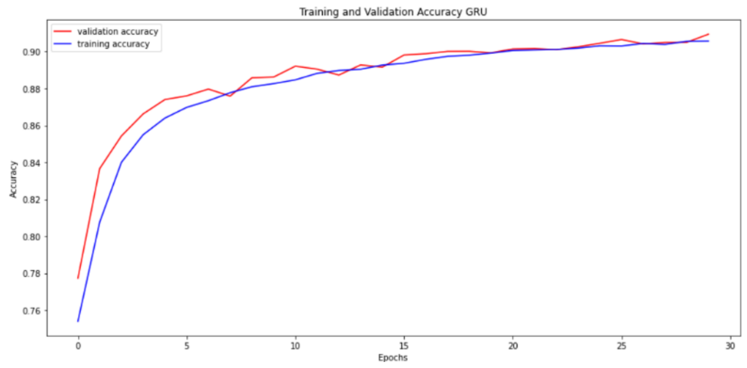


Figure 3.21: Training and validation accuracy of GRU

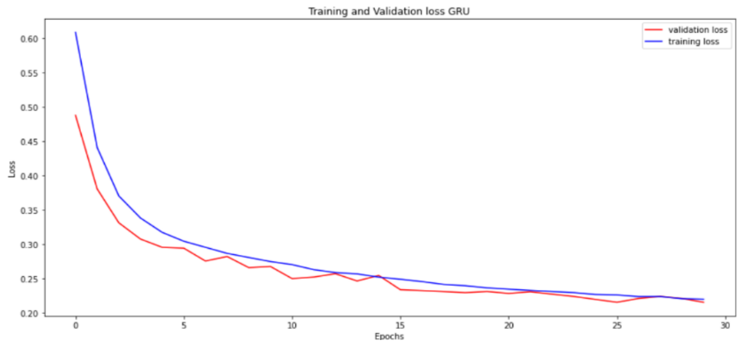


Figure 3.22: Training and validation Loss of GRU

The overall classification accuracy of GRU is 88.4%, which is lower than LSTM and CNN. The GRU model showed better performance than CNN and LSTM in detecting Worms, where it can detect all worms' attacks and it performs better than LSTM in detecting Shellcode. But it is unable to improve the detection rate of Fuzzers attacks and Reconnaissance attacks. The confusion matrix of the GRU is shown in Figure 3.23 where it represents the distribution of different attacks in UNSW-15 dataset.

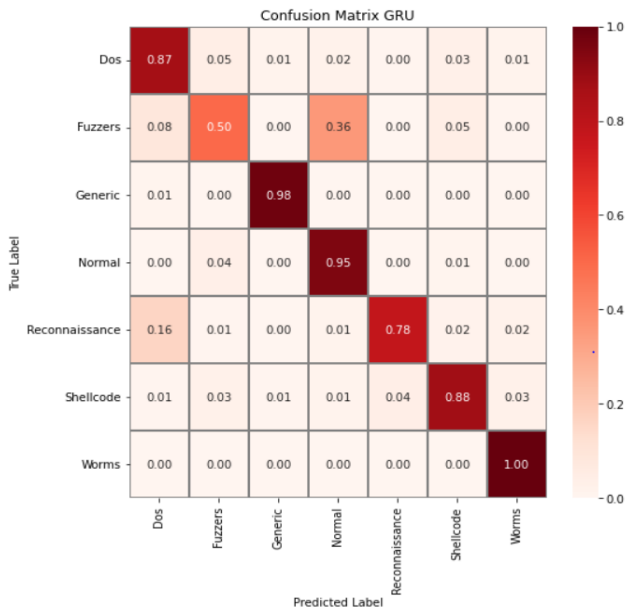


Figure 3.23: Confusion Matrix for GRU

Finally, we deployed our proposed stacking ensemble model where we combined the output of all the models of the first layer and feed them as input into a DNN model of the second layer where DNN is working as a meta learner. The overall accuracy of the stacking ensemble model is 90.4%, which is 1.3, 1.4, and 2.0 percent higher than CNN, LSTM, and GRU models respectively. The detection rate of Reconnaissance and Shellcode increases significantly. Figure 3.24 shows the confusion matrix of the stacking ensemble model.

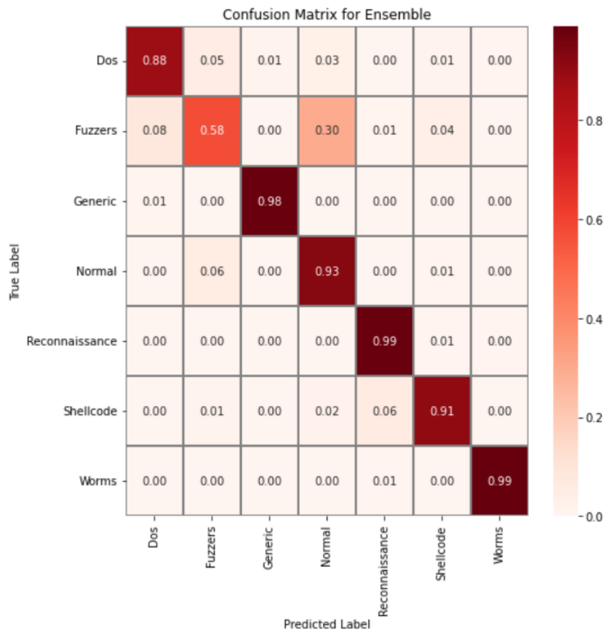


Figure 3.24: Confusion matrix of Ensemble Model

Table 3.1 depicts the accuracy, recall, F1-score, and precision of the three models along with the ensemble method, where the ensemble model outperforms other models with higher accuracy.

Table 3.1: Performance Comparison Models for UNSW-15 dataset

Model	Accuracy	Recall	Precision	F1-score	Training Time (S)	Testing Time (S)
CNN	89.1	89.1	89.7	89.1	322	7.3
LSTM	89.0	89.0	89.5	89.0	381	7.8
GRU	88.4	88.4	89.5	88.2	364	7.8
ANN [32]	84	-	-	-	-	-
RNN-LSTM [36]	85.38	-	-	-	-	-
Proposed Ensemble	**90.4**	**90.5**	**90.4**	**90.0**	**460**	**6.9**

3.6.2 Experiments Results of CICIDS-2017

In evaluating the CICIDS-2017 dataset the first model we used is CNN. The training and validation accuracy are the same, and over the whole training procedure, both the training and validation losses progressively reduce. The accuracy and loss curves for the CNN model are shown in Figures 3.25 and 3.26.

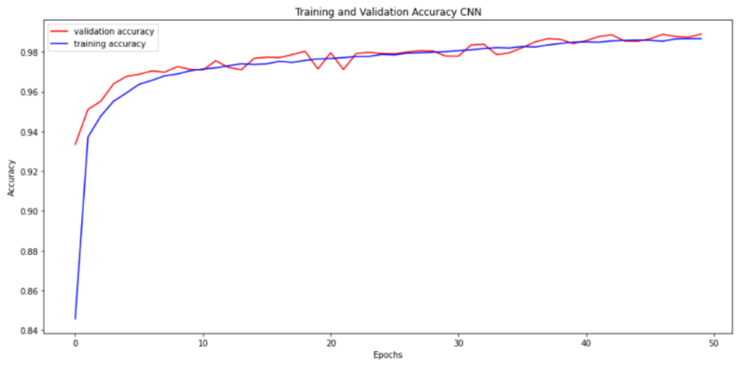


Figure 3.25: Training and validation accuracy of CNN

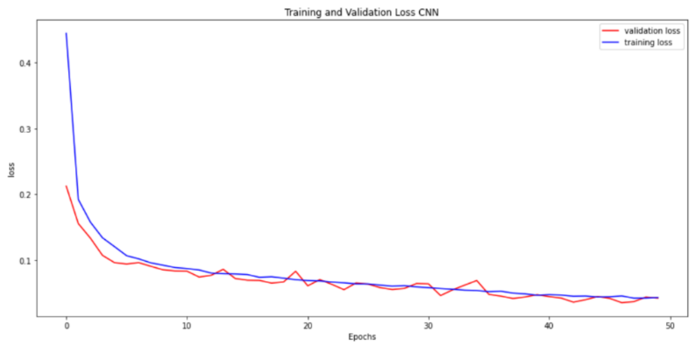


Figure 3.26: Training and validation loss of CNN

The overall classification accuracy for CNN is 97.5%, where it can detect all the DoS and Port-Scan attacks. Moreover, the detection rate of CNN for all other attacks is more than 95%, except for Webattacks. Figure 3.27 shows the confusion matrix of CNN.

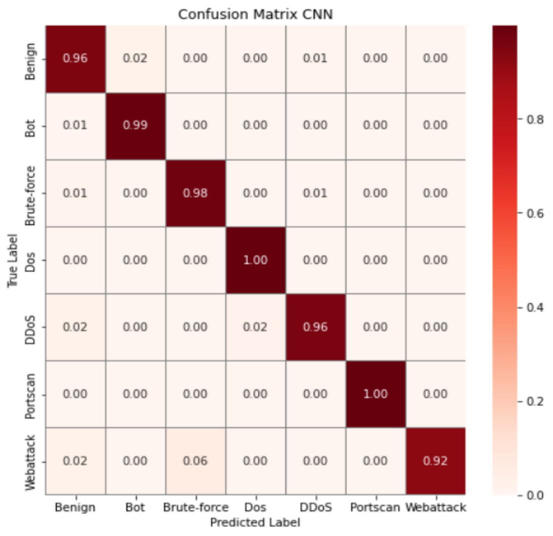


Figure 3.27: Confusion matrix of CNN

There is no significant gap between accuracy in training and validation of the LSTM model. Moreover, the loss in training and validation decreases as the number of epochs increases.

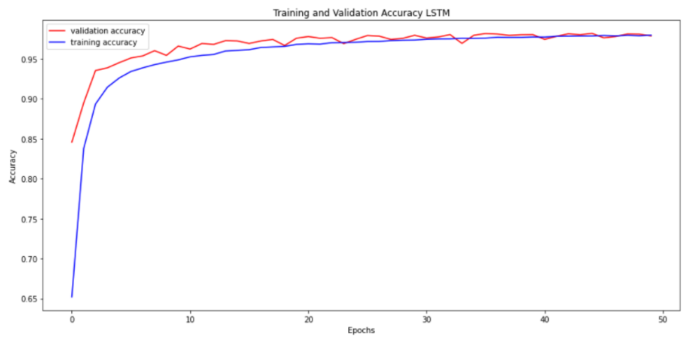


Figure 3.28: Training and validation accuracy of LSTM

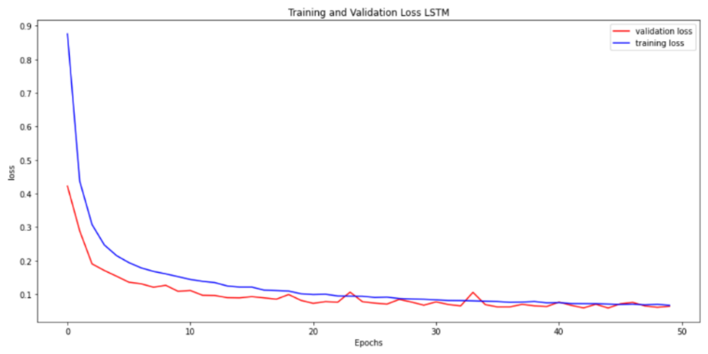


Figure 3.29: Training and validation loss of LSTM

Figure 3.28 and 3.29 shows the accuracy and loss curve of the LSTM model.

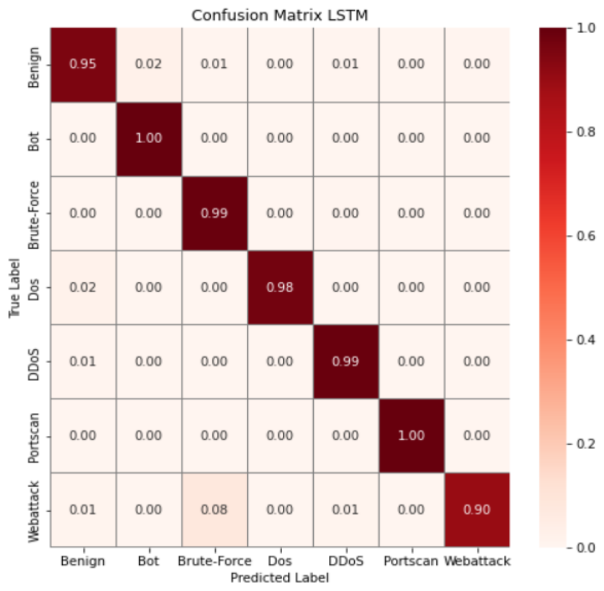


Figure 3.30: Confusion matrix of LSTM

The LSTM model of the first layer gives 97.2% classification accuracy which is slightly lower than CNN. Figure 3.30 shows the confusion matrix of LSTM model, where it shows that compared to CNN it misclassified 8% of the Web attacks and classified them as brute force attacks. However, LSTM able to detect almost all other attacks perfectly.

The GRU model of the first layer has almost the same training and validation accuracy as LSTM; however, compared to CNN the training accuracy and validation accuracy are lower. The training loss and validation loss decrease significantly as the number of epochs increases. Figure 3.31 and 3.32 depicts the accuracy and loss curves of GRU model.

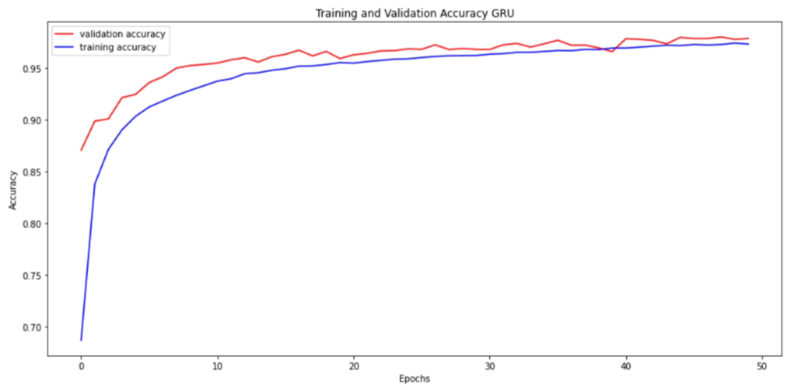


Figure 3.31: Training and validation accuracy of GRU

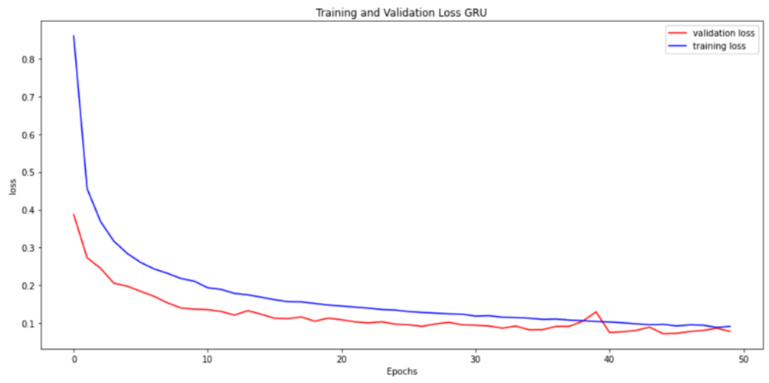
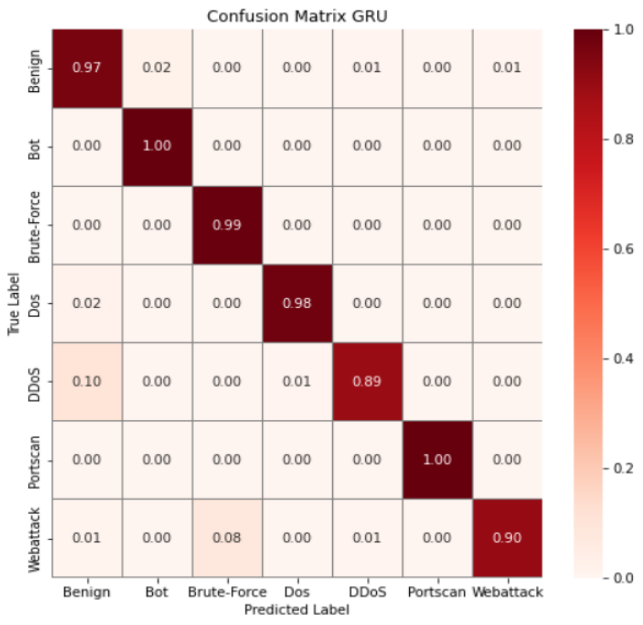


Figure 3.32: Training and validation loss of GRU



True \ Predicted	Benign	Bot	Brute-Force	Dos	DDoS	Portscan	Webattack
Benign	0.97	0.02	0.00	0.00	0.01	0.00	0.01
Bot	0.00	1.00	0.00	0.00	0.00	0.00	0.00
Brute-Force	0.00	0.00	0.99	0.00	0.00	0.00	0.00
Dos	0.02	0.00	0.00	0.98	0.00	0.00	0.00
DDoS	0.10	0.00	0.00	0.01	0.89	0.00	0.00
Portscan	0.00	0.00	0.00	0.00	0.00	1.00	0.00
Webattack	0.01	0.00	0.08	0.00	0.01	0.00	0.90

Figure 3.33: Confusion matrix of GRU

The overall classification accuracy of GRU is 96% which is lower compared to CNN and LSTM. Like LSTM, it can detect all the Bot attacks and Port scan attacks. However, overall, 10% of the DDoS attacks are misclassified as Benign traffic and 8% of Web attacks are misclassified as Brute-force attacks. Figure 3.33 shows the confusion matrix of the GRU model.

Finally, we again deployed our proposed stacking ensemble model where predictions from the first layer models were passed to the DNN model of the second layer, where DNN performed as meta learner. Cyber-attack detection with the ensemble is highly accurate, where it can detect all DDoS and Port scan attacks. Moreover, the ensemble model can detect more than 95% of Benign, Bot, Brute-force, and DoS attacks. Though all other models in the first layer cannot significantly detect webattack, ensemble outperforms other models in detecting webattacks. Figure 3.34 shows the confusion matrix of the ensemble

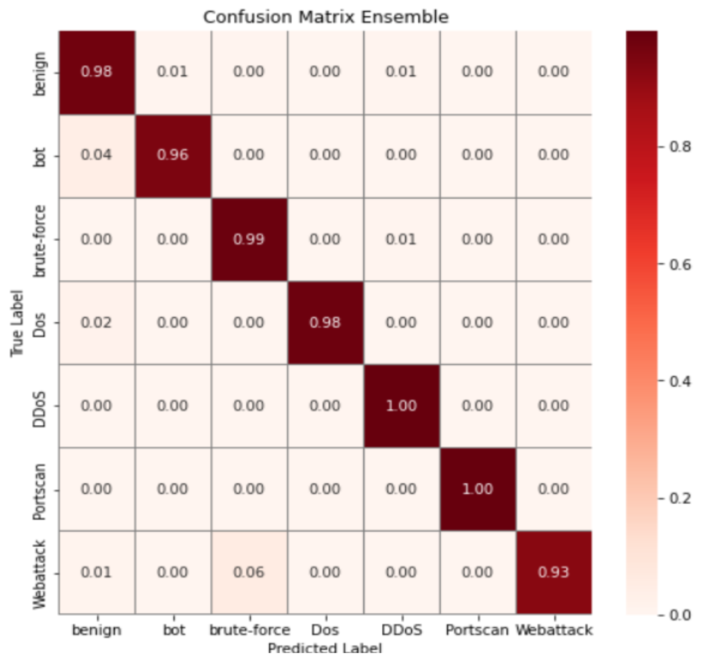


Figure 3.34: Confusion matrix of Ensemble model

model. Table 3.2 shows the accuracy, recall, precision, and F1-score of all the three models along with the ensemble method, where the ensemble model outperforms other models with higher accuracy.

Table 3.2: Performance Comparison models for CICIDS-2017 dataset

Model	Accuracy	Recall	Precision	F1-score	Training Time (S)	Testing Time (S)
CNN	97.5	97.5	97.7	97.6	499	8.6
LSTM	97.2	97.3	97.4	97.2	677	12.7
GRU	96.0	96.0	96.4	96.0	637	12.3
RBF-BLS [30]	96.63	-	-	96.87	-	-
DNN [39]	97.04	92.7	-	-	-	-
Proposed Ensemble	**98.7**	**98.7**	**98.7**	**98.5**	**1063**	**8.6**

In our experiments, LSTM shows better performance than GRU because LSTM performs better with large datasets and both datasets have a large number of samples. LSTM consists of three gates and is more complex than GRU, which consists of two gates. The GRU exposes the entire hidden contents without any control since it lacks internal memory and output gates, unlike the LSTM. LSTM performs better than GRUs because it can remember longer sequences. However, GRU has a faster training time and is computationally more efficient than LSTM.

3.6.3 Discussion

This section discusses the experiments and methodologies we used to develop a real-time, reasonably intelligent intrusion detection system that could identify a wide range of cyberattack types. A data preprocessing method and a multi-classifier model for identifying cyber threats using a stacking ensemble DNN are the key contributions of this thesis. In contrast to earlier suggestions based on literature studies, the current model not only classifies traffic flow as dangerous or harmless but also identifies the type of attack that may be present in the traffic flow. This categorization is crucial when creating security policies for deployment in a network that has experienced or is at risk from cyberattacks. The final layer obtains a collection of predictions from several other deep learning models that improve the final classification, using an ensemble stacking DNN technique is proposed. In this instance, additional indicators like the improvements in accuracy are also discernible. Figures 3.35 and 3.36 display the metrics that were used to compare each model's performance across the UNSW-15 and CICIDS-2017 datasets. In this instance, additional indicators like the improvements in accuracy are also discernible. Figures 3.35 and 3.36 display the metrics that were used to compare each model's performance across the UNSW-15 and CICIDS-2017 datasets.

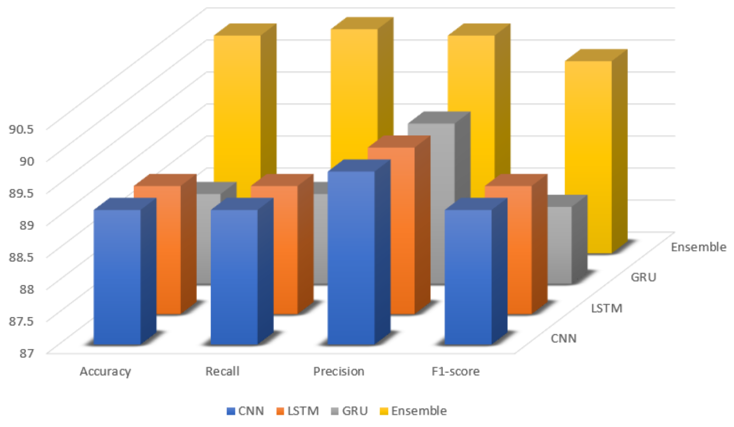


Figure 3.35: Comparative performance metrics for UNSW-15

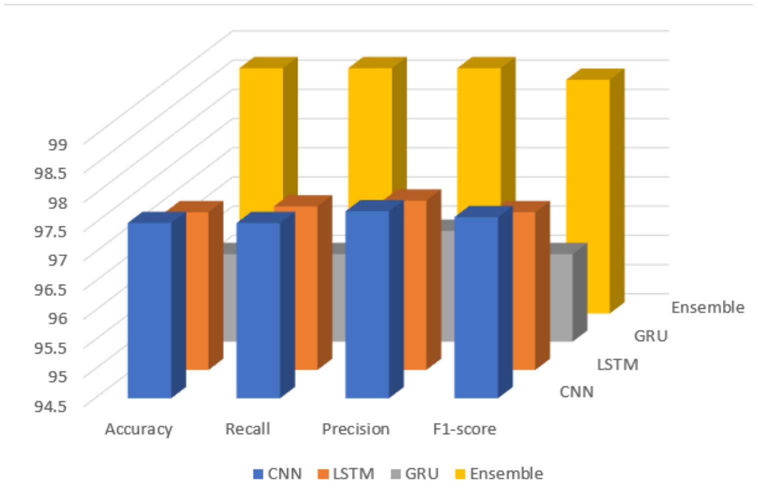


Figure 3.36: Comparative performance metrics for CICIDS-2017

3.7 Conclusion

Driven by the advancement in ultra-high speed network technologies such as 5G, the number of connected devices to the Internet is growing rapidly. Hackers/intruders continue to use new techniques to launch large-scale cyberattacks, making networks more vulnerable. This research suggested a multi-classifier approach for identifying various types of cyberattacks. In contrast to earlier research, our proposed novel network intrusion detection techniques not only determine if the network traffic is benign or normal, but also the type of assault in the flow. Compared to the existing models, our suggested stacking ensemble model provides a more accurate forecast. Overall, Stacking Ensemble DNN, a hybrid deep learning approach, and a well-defined data preprocessing technique are presented in this study. Two well-known datasets, UNSW-15 and CICIDS-2017, were selected for this study because they closely mimic real network traffic flow, which improves the efficiency, robustness, and practicality of our approach.

We select the best suitable features from the network traffic flow and preprocess them using different data pre-processing techniques like data imputation, data normalization, encoding, and resampling techniques. With the help of the suggested technique, we can train various robust and large deep learning models, concatenate their outputs, and then pass them to the second layer of the stacking ensemble technique, which consists of a DNN model. By retraining the model, we can increase the classification accuracy for identifying various cyberattacks. The proposed model aims to simulate the real-world environment where we use two well-known datasets, resamples the data, and train them to detect cyberattacks.

References

[1] *The Internet and the Pandemic*. Pew Research Center: Internet, Science & Tech. (2022). Retrieved 29 April 2022, from https://www.pewresearch.org/internet/2021/09/01/the-internet-and-the-pandemic.

[2] *Digital Around the World — DataReportal – Global Digital Insights*. DataReportal – Global Digital Insights. (2022). Retrieved 29 April 2022, from https://datareportal.com/global-digital-overview.

[3] *5G Security Solutions: Protect devices, identity and Data*. Thales Group. (n.d.). Retrieved April 13, 2022, from https://www.thalesgroup.com/en/markets/digital-identity-and-security/mobile/5g-security

[4] *Check Point Research: Cyber Attacks Increased 50% Year over Year - Check Point Software*. Check Point Software. (2022). Retrieved 29 April 2022, from https://blog.checkpoint.com/2022/01/10/check-point-research-cyber-attacks-increased-50-year-over-year/.

[5] *Cyber Kill Chain®*. Lockheed Martin. (2022). Retrieved 29 April 2022, from https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html.

[6] *What is a Cyber Attack | Types, Examples & Prevention | Imperva*. Learning Center. (2022). Retrieved 29 April 2022, from https://www.imperva.com/learn/application-security/cyber-attack/.

[7] *Strategic Technologies Program | Center for Strategic and International Studies*. Csis.org. (2022). Retrieved 29 April 2022, from https://www.csis.org/programs/strategic-technologies-program.

[8] *2021 DDoS Threat Landscape Report | Resource Library*. Resource Library. (2022). Retrieved 29 April 2022, from https://www.imperva.com/resources/resource-library/reports/ddos-threat-landscape-report/.

[9] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. *Cluster Computing*, *22*(1), 949-961.

[10] Ahmad, J., Farman, H., & Jan, Z. (2019). Deep learning methods and applications. In *Deep learning: convergence to big data analytics* (pp. 31-42). Springer, Singapore.

[11] Moustafa, N., & Slay, J. (2015, November). 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)* (pp. 1-6). IEEE.

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

[13] Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection system. *Eai Endorsed Transactions on Security and Safety*, *3*(9), e2.

[14] Dong, B., & Wang, X. (2016, June). Comparison deep learning method to traditional methods using for network intrusion detection. In *2016 8th IEEE international conference on communication software and networks (ICCSN)* (pp. 581-585). IEEE.

[15] Smaha, S. E. (1988, December). Haystack: An intrusion detection system. In *Fourth Aerospace Computer Security Applications Conference* (Vol. 44).

[16] Anderson, J. P. (1980). Computer security threat monitoring and surveillance, James P. *Anderson Co., Fort Washington, PA*.

[17] Ye, N., Emran, S. M., Chen, Q., & Vilbert, S. (2002). Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on computers*, *51*(7), 810-820.

[18] Kumar, S. A. P., Kumar, A., & Srinivasan, S. (2007). Statistical based intrusion detection framework using six sigma technique. *IJCSNS*, *7*(10), 333.

[19] Sinclair, C., Pierce, L., & Matzner, S. (1999, December). An application of machine learning to network intrusion detection. In *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)* (pp. 371-377). IEEE.

[20] Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context

[21] Zhang, Z., Li, J., Manikopoulos, C. N., Jorgenson, J., & Ucles, J. (2001, June). HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *Proc. IEEE Workshop on Information Assurance and Security* (Vol. 85, p. 90).

[22] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security* (pp. 77-101). Springer, Boston, MA.

[23] Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *expert systems with applications*, *36*(10), 11994-12000.

[24] Sangkatsanee, P., Wattanapongsakorn, N., & Charnsripinyo, C. (2011). Practical real-time intrusion detection using machine learning approaches. *Computer Communications*, *34*(18), 2227-2235.

[25] *Random-Forests-Based Network Intrusion Detection Systems*

[26] Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Communications Surveys & Tutorials*, *21*(1), 686-728.

[27] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

[28] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

[29] Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection system. *Eai Endorsed Transactions on Security and Safety*, *3*(9), e2.

[30] Rios, A. L. G., Li, Z., Bekshentayeva, K., & Trajković, L. (2020, October). Detection of denial of service attacks in communication networks. In *2020 IEEE international symposium on circuits and systems (ISCAS)* (pp. 1-5). IEEE.

[31] Moustafa, N., & Slay, J. (2015, November). 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)* (pp. 1-6). IEEE.

[32] Hanif, S., Ilyas, T., & Zeeshan, M. (2019, October). Intrusion detection in IoT using artificial neural networks on UNSW-15 dataset. In *2019 IEEE 16th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT and AI (HONET-ICT)* (pp. 152-156). IEEE.

[33] Moustafa, N., Turnbull, B., & Choo, K. K. R. (2018). An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, *6*(3), 4815-4830.

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

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

[36] Aleesa, A., Younis, M. O. H. A. M. M. E. D., Mohammed, A. A., & Sahar, N. (2021). Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques. *Journal of Engineering Science and Technology*, *16*(1), 711-727.

[37] Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, *9*(20), 4396.

[38] Injadat, M., Moubayed, A., Nassif, A. B., & Shami, A. (2020). Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Transactions on Network and Service Management*, *18*(2), 1803-1816.

[39] Faker, O., & Dogdu, E. (2019, April). Intrusion detection using big data and deep learning techniques. In *Proceedings of the 2019 ACM Southeast conference* (pp. 86-93).

[40] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

[41] Gulli, A., & Pal, S. (2017). *Deep learning with Keras*. Packt Publishing Ltd.

[42] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

[43] Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009, April). Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 475-482). Springer, Berlin, Heidelberg.

[44] SMOTEENN — Version 0.9.1. (2022). Retrieved 26 August 2022, from https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTEENN.html

Chapter 4

Exploring Artificial Intelligence (AI) Techniques for Forecasting Network QoS in 5G Network

4.1 Introduction

In this era of digital services and applications, one of the challenges for wireless networks is to provide even increasing connectivity bandwidth demand. According to global-newswire, global data traffic would reach 220.8 million terabytes per month by 2026, with a single user's data usage expected to exceed 41GB [15]. According to a recent analysis from Ericsson Mobility, by 2026, six billion people will be on the Internet, with smartphones accounting for 54% of all mobile data, expressing a pressing problem that must be addressed [16]. There is significant evidence that there is a link between global data traffic increases and the proliferation of new services like Voice over IP (VoIP), high-definition video on demand, IoT, Machine-to-Machine communication (M2M), and several other programs that have elevated the pressure on the spectrum, affecting network signal strength. The development of LTE to the most recent 5G technology has resulted in a diverse set of commercial entities with widely disparate features and capabilities. In order to satisfy the aforementioned requirement, the 5th generation (5G) of cellular networks is being developed. It will be able to handle the large increase in capacity demand, the number of connections, and the developing use cases of a connected society for 2020 and beyond [17]. When compared to 4G LTE, 5G is predicted to give a tenfold increase in bandwidth and ultra-low latency of less than one millisecond, which might support a wide range of the previously mentioned applications [18].

As the next-generation wireless technology, 5G is the solution, which overcomes current 4G/Long-Term Evolution (LTE)'s constraints since the current LTE network cannot keep up with consumers' rising demand for ultra-low latency, expanded capacity, and high data speeds. As the 5G network becomes available, it is expected that the demand for big bandwidth and ultra-low latency along with device connectivity will continue to grow. These requirements are expected to be met by the 5G for supporting varieties of next-gen digital services and applications. 5G introduced millimeter wave (mm-wave) communications and network densification to increase the capability [19]. Throughput and latency, in particular, are sensitive metrics to use when evaluating a network's overall performance with QoS [20]. 5G is expected to offer the followings: a) very high throughput, more than 1 Gbps to support Virtual reality and HD video streaming, b) ultra-low latency, possibly lower than 1 millisecond, c) high availability and reliability, and d) low energy consumption.

Throughput, latency, jitter, and packet loss are common QoS measures [22]. According to Chen et al. [23], for assessing the end users experience, throughput and latency is the most important metric and throughput has the ability to directly influence productivity of any system connected to the network. Moreover, the overall QoS of the network solely depends on the throughput as well as latency. In many digital manufacturing applications, for example, insufficient throughput may result in the production being totally suspended because lack of bandwidth to support video streaming [13]. The next-generation digital services and applications that to be supported by 5G networks must guarantee stringent QoS to its services, including many mission-critical applications such as autonomous and connected vehicles, autonomous drones, remote telehealth etc.

Recently, Machine Learning (ML)-based approaches have been used to assist the autonomous operations of cellular networks in a variety of ways, including resource allocation [23], video streaming [24], and energy-efficient networks [25]. Moreover, Both ML and AI strategies can predict network QoS in the cellular network. ML models assist network planners in predicting network QoS status in both the short and long term. Authors in [28] and [14], for example, used historical network QoS metrics to anticipate network latency and throughput using User Equipment (UE). Authors in [1] and [13] collected data from simulated NSA 5G networks and predicted throughput, and compared it with different ML models and Deep Neural Networks (DNN). However, the lack of real-time network traffic data, 5G in particular, has always been a challenge to develop an efficient ML model to predict 5G cellular network latency and throughput. A key shortcoming of current research is the lack of large-scale field tests to measure the QoS considering the seasonality behaviour of the traffic.

It remains a challenge for the network service providers to guarantee better service as the number of network users grows continuously. Additionally, the arrival of 5G networks has increased the bandwidth demand, and traffic circumstances are constantly changing. To overcome these issues and ensure desired network performance, the ability to forecast the network QoS metrics is expected to assist the network service providers in their service planning and capacity provisioning. Nowadays, digital services utilizing 5G networks, including IoT devices, autonomous vehicles, drone technology, real-time video streaming services, e-health, etc., are highly dependent on the QoS of the network, and to ensure better services, forecasting the QoS like throughput is a challenge for digital service providers [21]. However, limited research is being done in this area due to the lack of a live 5G platform available in the public domain and lack of 5G QoS data. In this thesis, we addressed the above research gap by conducting the first-ever large-scale field test in a 5G network environment. We developed a novel framework to collect 5G network QoS data and a QoS forecasting framework to predict the major QoS metrics such as throughput, latency, packet loss, and jitter. In this regard, the main contribution of our research is as follows:

- We developed a tool to collect 5G Quality of Service (QoS) metrics in real-time where we set up the experimental test bed using a high-performance server, a 5G router, and Wi-Fi 6 integrated PCs/laptops.
- We imitated the real-world 5G use scenarios by uploading and downloading five various types of files to and from our server and collecting 5G QoS data simultaneously.
- To avoid biases in the data, we collected data from multiple campus locations and our collected QoS metrics are: throughput, latency, jitter, and packet loss.
- We collected data for almost two months from 8:30 AM to 6:30 PM on the weekdays and 10:30 AM to 5:00 PM on the weekends, with three-minute intervals.
- Finally, we employed a time-series forecasting approach by using various Deep Learning (DL) models to forecast throughput during various periods of the weekdays or weekend, including morning, evening, and afternoon.

The rest of this chapter is organized as follows: Section 4.2 begins with a complete overview of related work. In Section 4.3, the experimental setup of our proposed data collection strategy is discussed. Next, an AI-based QoS predictions mechanism is presented in section 4.4. Section 4.5 depicts the experimental results along with a discussion. Finally, concluding remarks are presented in section 4.6.

4.2 Related Work

5G is a new technology and it has just begun its rollout that will gradually replace the 4G/LTE network. The 5G claims to provide high throughput, low latency, high availability, and reliability according to the 3GPP standard. However, from a research standpoint, its large-scale field test for measuring the QoS in a live public setting is not reported in any literature yet. Some limited studies have been conducted to predict the throughput of non-standalone (NSA) 5G networks using statistical and machine learning (ML) models to predict the QoS of 5G. Some prior similar studies are based on 4G and LTE networks using popular test applications like Speedtest, OpenSignal and nPerf.

Verma et al. [1] proposed a ML based methodology to predict the user throughput in LTE networks by utilizing different network parameters. In their experiments, they used a dataset provided by Nokia Network Pvt. Ltd. which contains information of 50,000 records collected from Base Transceiver Sites (BTS) with 135 features. They employed 14,000 records of Time Division Duplexing (TDD) in their experiments. They categorize their dataset into four different categories based on the throughput value. Finally, they predict average upload and downlink throughput values from the dataset by implementing three ML model namely Support Vector Machine (SVM), Naïve Bayes and K-Nearest Neighbors (KNN) where the accuracy of SVM was 96.17% and 96.10% for download and upload which is better compared to other two models. Mohammed et al. [2] analyzed 4G/LTE networks and proposed a ML-based and Network Functions Virtualizations (NFV) framework to predict the behaviour of the network. The cells of their model were categorized based on criteria such as download throughput and elevation hours. Alho et al. [3] proposed ML and DL based model to capture the cells with anomalies. Their model analyzes the problematic cells to classify the throughput by monitoring data with low number of labels. In their experiments, they create their own dataset which contains information of serving cell of Mobile Networks and User Equipment (UE).

It is difficult for programmers to track the network resources of a complex network like 5G because there is a gap between network events and QoS abnormalities. To addresses this gap Zhu et al. [4] proposed a supervised ML model which is capable to track the network environment and give decisions in dynamic situations. Their model has been trained with past QoS-related information and anomalies and developed a relationship between current QoS data and abnormalities, and as a result, their model can reliably and effectively forecast potential QoS and anomalies in 5G networks. In [5], researchers utilized queuing models to evaluate the data transmission speed of 5G networks to clarify that the QoS of service of the network is depending on the overall data speed of the networks. For their experiments, they used mobile systems and explained how the data speed of 5G has an impact on the overall QoS of the 5G networks.

Mollel et al. [6] focused on Handover (HO) because it is one of the major factors of mobile communications which has a greater impact on QoS of the networks, especially on throughput and service availability. Because the network is becoming more densified with the growing number of base stations (BS) per unit area, as well as the number of connections, HO management is very difficult in 5G. To address this challenge, researchers proposed intelligent HO management by utilizing ML techniques. In-network data-based HO management concentrates on beam selection and BS station selection, whereas ML-based HO management is divided into two major groups, including visual data-based HO management approaches and network-data-based HO management. In [7] published a comprehensive and exhaustive assessment in which HO management was discussed for both LTE and 5G networks, with contrasting comments. Moreover, Step-by-step HO methods in both LTE and 5G were described, and HO types were addressed in depth. Although some ML techniques were highlighted while discussing state-of-the-art approaches, the spectrum of the article was purely HO management, not ML-based solutions to HO management. Mardian et al. [8] suggested some approaches to measure the QoS and QoE in the 5G cellular networks where they focused on the importance of location, devices, and network scenarios. They also proposed Self Organizing Networks

(SON) which consists of three features: reading network status, predicting user behaviour and dynamic adjustments of the networks has an impact on the QoS/QoE of the network. Additionally, objective network parameters such as packet loss and delay and subjective parameters such as Mean Opinion Score (MOS) are merged to harmonize the QoS/QoE.

Ye et al. [9] proposed in how the network slicing methodology affects End-to-End (E2E) QoS provisioning in both wireless and wired NSA 5G networks. In the wireless network environment, dynamic radio resource slicing is used to split the network's bandwidth and distribute it among multiple BS in order to maximize network utility. When traffic flows in numerous network function virtualization (NFV) nodes, a bottleneck-resource generalized processor sharing (Br-GPS) is employed to minimize the latency in the wired network. Bui et al. [10] reviewed the most contemporary network performance prediction algorithms, emphasizing throughput as critical contextual network information. There are two types of throughput prediction methods: active and passive. The previous work requires connected user equipment (UEs) to stream packets, whereas the latter generates predictions with low or no network intrusion [11]. Furthermore, in-vehicle scenarios, the dynamic wireless environment would increase the sampling of active tests, overloading the network.

Pan et al. [12] proposed the very first large-scale comparative study between 5G (NSA) and LTE on a high-speed railway route. They used cellular data to cover three major carriers in China and a dataset containing throughput, RTT, packet loss rate, signal quality, and physical resource utilization. Their findings conclude that 5G gains better throughput with a low loss rate and can also tolerate weak channel conditions. They also found that 5G has a more significant impact on handover management than LTE networks and that 5G reduces average radio access network (RAN) latency from 40.0 milliseconds to 15.8 milliseconds when compared to LTE. Minovski et al. [13] proposed an approach to predict the throughput of end-users using NSA 5G and compared it with LTE by incorporating network slicing strategy. They did their study in four scenarios: urban, sub-urban, rural areas as well as in densely crowded areas. Finally, they employed several ML methods to train a model combining various lower layer radio environment parameters to predict the throughput in both uplink and downlink. Their model was validated in the LTE networks and then applied to NSA 5G networks to predict throughput in different slice. Daengsi et al. [14] used three popular network speed test applications such as Speedtest by Ookla, OpenSignal and nperf to collect different QoS parameters data such as speed, latency from a 5G network. They collect 180 data points per application and use ANOVA and a t-test to determine the difference between average upload and download speeds as well as latency, with the results revealing that the speeds and latency varied in three applications for the same 5G networks.

Unlike the above-mentioned research efforts, our study is the first approach to conduct the large-scale field test of a non-stand-alone (NSA) 5G network to measure the different QoS of the networks. Most of the above-mentioned works focused on 4G and LTE networks and most of them used different simulation tools or queuing models to predict the throughput. A few previous research utilized ML techniques and used real-time datasets to predict the QoS of 4G networks, and there is no dataset available for 5G networks. Some of the previous studies employed machine learning approaches to predict network QoS, but the datasets were generated using simulation tools rather than a large-scale field test. Moreover, their proposed strategy did not cover the impact of QoS of the 5G network while handling multiple files. Other researchers used the LTE network to anticipate the throughput of the NSA 5G network, but these trials are not representative of real-world 5G scenarios. In our study, we develop a system through which we continuously upload and download different files to and from a server using NSA 5G networks, and from this, we measure throughput, latency, packet loss, and jitter by sending and receiving ping. We did our test in a controlled environment where the number of people varies from 50 to 1200 in different time frames of a day. We gathered all the QoS parameters to generate a dataset in which the time frame is divided into weekends and weekdays because network behaviour may change on weekdays and weekends. Finally, to validate our dataset, we employed time series forecasting strategy using deep sequence models to predict the throughput of the 5G networks in the morning, afternoon, and evening of a day.

4.3 Experimental Setup

In this section, we will provide a high-level description of the conducted experiments. We will briefly describe the tools, development library, and platform that were used to conduct the experiment and collect different QoS metrics of 5G network. All our experiments were carried out in the Western University 5G network environment. Figure 4.1 depicts the high-level experimental setup to conduct the experiment.

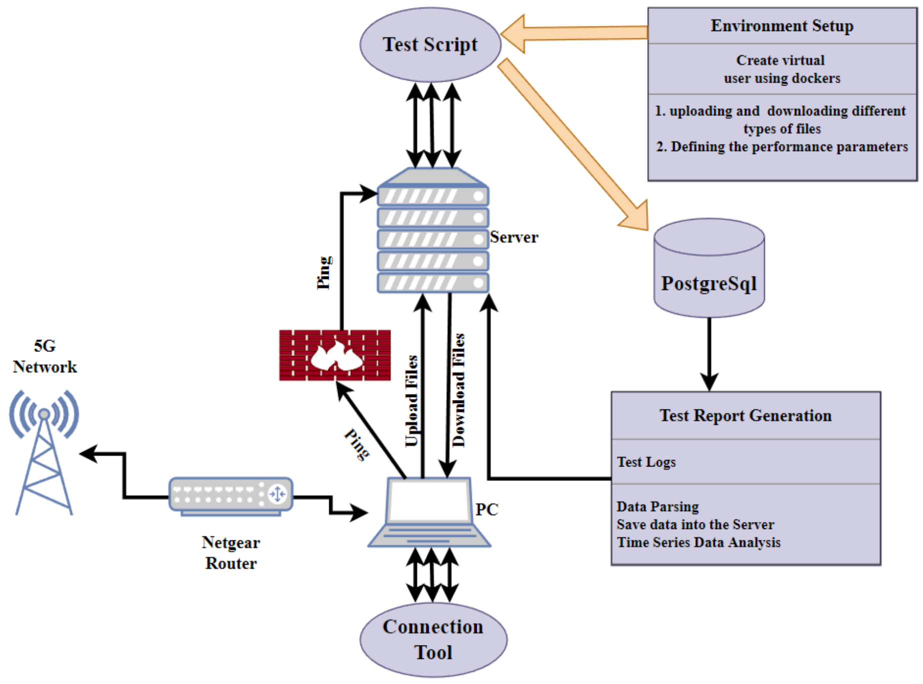

Test Script
Environment Setup
Create virtual user using dockers
1. uploading and downloading different types of files
2. Defining the performance parameters
Server
PostgreSql
Ping
5G Network
Upload Files
Download Files
Ping
PC
Test Report Generation
Test Logs
Data Parsing
Save data into the Server
Time Series Data Analysis
Netgear Router
Connection Tool

Figure 4.1: Experimental Setup

4.3.1 Hardware Requirements

In this section, we will briefly discuss the hardware tools we used to conduct the experiments.

Netgear Router: We connected our devices to a 5G network using a router called "NETGEAR Nighthawk M5" in our experiment. The router has Wi-Fi 6 built-in, as well as the ability to connect to a 5G network, and it can connect 32 devices at once. This router's Wi-Fi 6 capability boosts its speed to 7 Gbps, and it's powered by Qualcomm's Snapdragon X55 Mobile platform. The router can also connect to several types of devices, such as smartphones, tablets, and laptops, also send out signals with a long-range with a consistent coverage. It also includes a long-lasting battery and a touch screen for controlling the gadget. We utilized this router to connect our PC to 5G networks throughout our tests through Wi-Fi and Ethernet.

Server: We employed a high-speed dedicated server in our test to continually upload and download various sorts of files to and from the server. The server is running Ubuntu server 20.04 LTS and linked to a high-speed Internet connection with a maximum throughput of 1.0 Gbps in our lab [34]. The server's metal frame, often known as the chassis, includes an HPE Edgeline EL8000t 2U platform and two C15 - NEMA 6-20P 250V 15Amp Black 2.5m US Power Cord. One HPE Edgeline EL8000t chassis controller manages the server's chassis. The server we're utilizing is a Blade Server, which helps reduce physical space and energy use. Furthermore, the Blade server offers improved management features such as cooling, minimum wiring, lower power consumption, and a small footprint. An Intel Xeon-Silver 4210R CPU, 96 GB of RAM, two HPE 120GB SATA M.2 2242 Solid State Drives, and an NVIDIA T4 16GB GPU make up our Blade server.

Personal Computer (PC): In our tests, we used a PC with a Ryzen 9 5000 series processor, 16 GB of RAM, 4GB of NVIDIA GeForce GPU, and a Wi-Fi 6 module incorporated. Wi-Fi 6 provides faster speeds even in congested areas, with longer battery life. The PC comes with Windows 10 operating system. Our PC serves as a connecting tool as well as a controller for the entire experiment. Using the Netgear Router, we were able to connect our PC to a 5G network during the tests. Moreover, we're also connected to the server and uploading files to it with our custom-built connection tool.

4.3.2 Development Platform

We used a variety of software tools to calculate the QoS metrics of the 5G networks in our study. We developed our exploratory tools, including a connection tool and a server-side scripting tool, to conduct the experiments. Moreover, all the tools were developed with Python 3.7 [29] along with different Python libraries.

Our connection tool is primarily responsible for conducting two operations: file sharing and metric calculations. In metrics calculation, we use various functions with different computations for each QoS measure to calculate the QoS metrics. We measured 5G internet connection performance like throughput for both upload and download using a common application called Speedtest CLI [30] API (Application program interface). Furthermore, Speedtest CLI is used all over the world to assess network performance such as download and upload speeds, latency, and packet loss. The Python version of the Speedtest CLI is used in our script. Moreover, after integrating Speedtest CLI into our system, we tried to compare the throughput we are getting in our system with the throughput we are getting at "speedtest.net", which is an official website of Speedtest by Ookla to test the current QoS metrics of the network. Our system's download and upload throughput are virtually identical to "speedtest.net's" performance. Our connection tool also uses PostgreSQL [31], commonly known as Postgres, a sophisticated open-source object-relational database created by the University of California, Berkeley, to construct a database table where we will store the computed QoS metrics of the 5G networks. With automatic updates of views, triggers, and stored procedures, PostgreSQL offers all of the ACID (Atomicity, Consistency, Integrity, and Durability) features of a relational database. We utilized ping, a well-known network software program, to evaluate the network reachability of our host server. Ping sends an ICMP echo request to the target host and waits for a response using ICMP (Internet Control Message Protocol) packets. It also gives a statistical breakdown of packet loss and latency.

In our research, we tried to present a real-life scenario of 5G use-cases. We employed five distinct types of files in our experiments: image, video, audio, text file, and zip file. Our connection tools regularly upload and downloads these files to and from the server to monitor the performance of QoS metrics.

During uploading and downloading of the files to and from the server, the server-side scripting tool receives ping requests from the connection tool, and as a result, our connection tool collects various network QoS parameters. We used docker [32] on the server, which is a toolkit for developing, running, and managing containers. Docker is also well-known for a variety of advantages, including portability, performance, agility, isolation, and scalability. We used Docker to establish numerous virtual users in our research.

4.3.3 Experimental Description

The high-level infrastructure utilized to perform the research is depicted in Figure 3.1. A personal computer (PC) serves as a connecting tool, interacting natively across a non-standalone 5G network via the NetGear 5G router. All of our tests were carried out in a controlled setup on the Western University campus, which was covered by non-standalone 5G networks. Furthermore, we conducted our experiments in a variety of places to ensure that our data was not skewed by a single site. The primary spot to collect 5G QoS data is the D.B. Weldon Library, and we also took the number of users using the network to check how the performance of the 5G network differs when the number of users increases. The following are the methodologies we used to calculate 5G QoS data:

a) The connection tool downloads and uploads distinct types of application files to and from the server linked to a high-speed network continuously. The server has built-in firewalls that prevent any other incoming requests; nonetheless, a special port is available to allow file sharing from outside the campus network.
b) The connection tool sends multiple ICMP ping requests to the server while downloading and uploading the files, and the test scripts on the server receive those requests and respond to the connection tool.
c) Docker generates a virtual environment with multiple containers that have been used here as multiple users to create a real time environment. While uploading and downloading the files, the server's test script, defines various 5G network performance factors which have been declared inside all of the containers in Docker. Our tool collects all the network QoS metrics from all the containers simultaneously.
d) When a download and upload cycle is completed, the metric calculations functions of the test script calculate the QoS metrics value of the 5G network for the present period continuously.
e) We use the Speedtest API to calculate upload and download throughput and convert it to megabits. Ping results are used to calculate latency, jitter, and packet loss. We received the packets transmitted by ping after an upload and download cycle was completed, and computed the latency, packet loss, and jitter based on that.
f) Finally, the computed QoS metrics and time records are saved in a PostgreSQL database table, which creates a test report and saves the data in CSV (comma separated value) file format on the server.

g) Apart from the four QoS metrics of the 5G network, we have three other attributes in the dataset; they are the number of users, and part of the day. We collect data on both weekdays and weekends from D.B. Weldon Library along with the real-time user number of the library. The data collection process continues from 8:30 AM to 6:30 PM every day for almost two months. Moreover, we divided a day into three periods. Table 4.1 shows the different parts of the day with their respective time ranges.

Table 4.1: Different day parts with time range

Day Part	Time Range
Morning	8:30 AM to 11:59 AM
Afternoon	12:00 PM to 3:59 PM
Evening	4:00 PM to 6:30 PM

4.3.4 Collected QoS Metrics

In our studies, we used four network performance indicators as our core QoS metrics for the 5G network: upload throughput and download throughput, jitter, latency, and packet loss. The above-mentioned four network QoS measures determine the overall performance of the network in our daily operations. In this section, we'll go through the four different QoS metrics utilized in our studies:

Throughput: To calculate throughput we utilized the Speedtest API, and it is calculated as the proportion of total amount of data transferred to and from connection tool to server in a unit of time. Figure 4.2 shows throughput values from our experiments during different periods of a day. Eq. 4.1 shows the mathematical formula to calculate the throughput.

$$Throughput = \frac{Total\ data\ sent}{Delivery\ time} \tag{4.1}$$

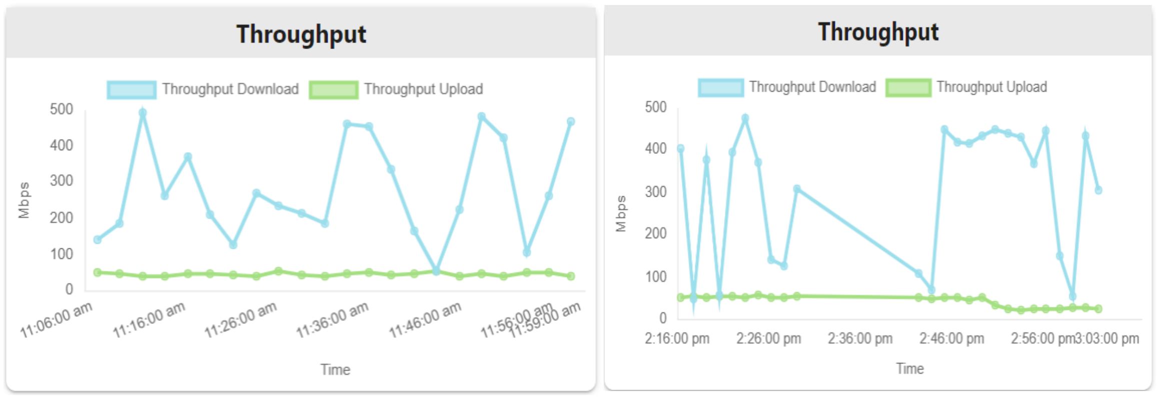


Figure 4.2: Upload and Download Throughput

Latency: The term latency is used to define the delays in network communication. Latency is the total proportion of transmission time taken by a packet initiated by ping to reach from connection tools to server. Figure 4.3 shows the outcome of latency from our experiments in different periods of day. The formula for calculating latency is depicted in Eq. 4.2.

$$Latency = \frac{packets\ size}{Link\ Bandwidth} \tag{4.2}$$

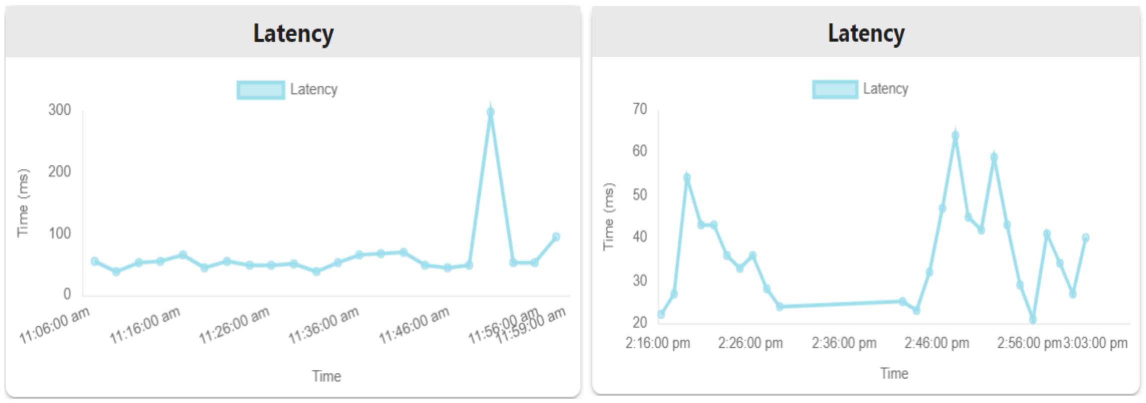


Figure 4.3: Latency in different periods of time

Packet Loss: A packet loss happens when a delivered packet initiated by a ping fails to reach its intended destination. It is measured as a proportion of lost packets compared to total messages delivered. The packet loss that happened during various periods of our experiment is depicted in Figure 4.4. The packet loss is formulated in Eq. 4.3.

$$packet\ loss = \frac{packets\ transmitted - packets\ received}{packets\ transmitted} \times 100 \tag{4.3}$$

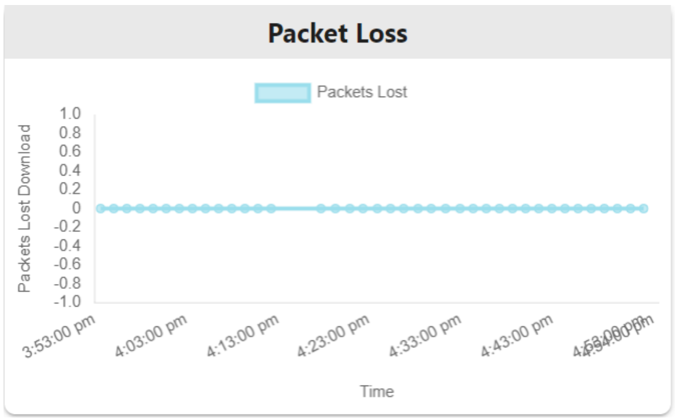


Figure 4.4: Packet loss in different periods of time

Jitter: When a packet is sent from one source to another, it is sent in regular intervals over a set period. When there is a variation in the time delay in sending packets, it is called jitter. Figure 4.5 demonstrates the jitter that occurs throughout various phases of our studies.

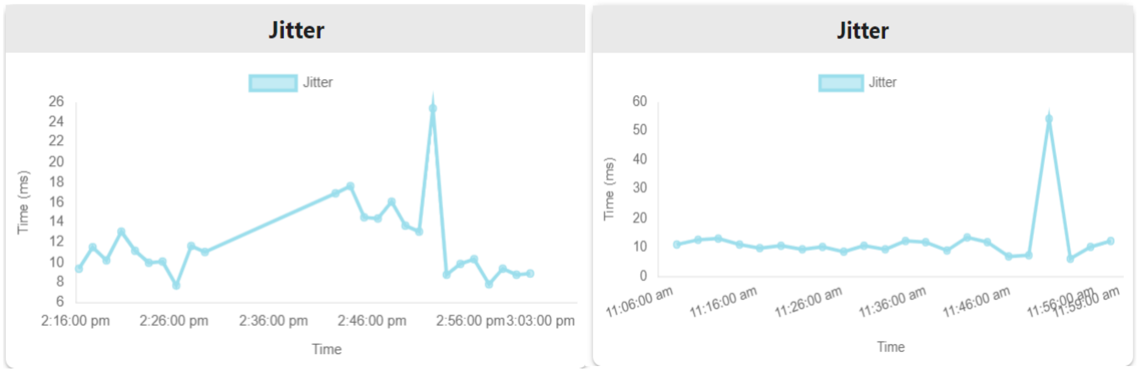


Figure 4.5: Jitter in different periods of time

Our proposed 5G QoS data collection tool differs from the previously developed data collections mentioned in section 4.2. Most previous works developed a tool to collect QoS metrics data from a simulated environment. However, our experimental testbed is based on an actual operational NSA 5G network. The network behaviour generated in the simulated environment differs from that generated in the real-life setting. Moreover, some previous studies collected data using different network speed test applications, but their data collection methodology is unknown. To address the shortcomings of the previous strategy, we utilized the NSA 5G network to handle different types of files. Additionally, none of the previous research did consider changing network behaviours at various times when the number of users varies, but our study showed that the bahaviour of the network is not the same at different times. For the reasons mentioned above, our developed tool is novel where we focus on the real-life use cases of the 5G network; thus, our proposed and developed data collection tool differs from the previous data collection strategies.

4.4 QoS Prediction Methodology

This section will discuss our proposed methodology to predict the QoS service of the 5G network. Our goal is to develop an Artificial Intelligence (AI) based model to predict the throughput of the 5G network for different periods of the day, including weekends and weekdays. Figure 4.6 resembles the architecture of QoS prediction of the 5G network, where we did time series forecasting by employing various Deep Learning (DL) techniques.

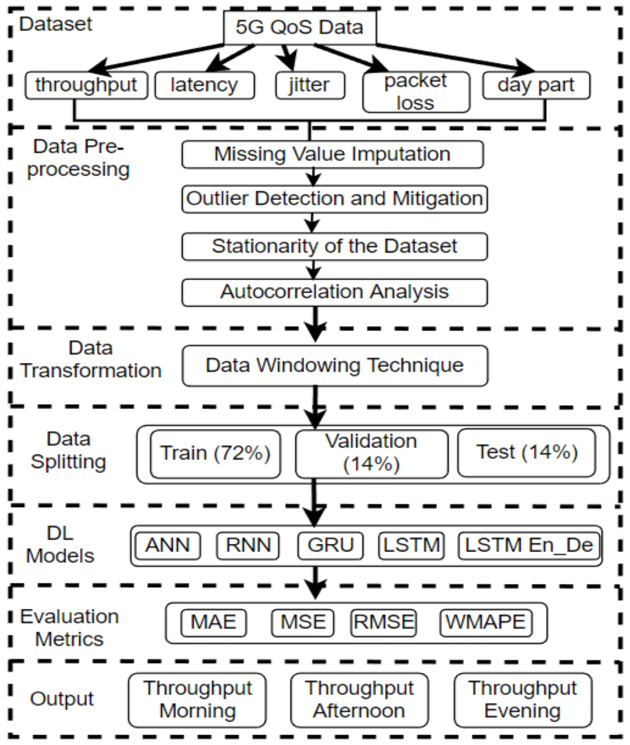


Figure 4.6: Our Proposed 5G QoS Metrics Model

In section 4.4.1, we'll briefly discuss the dataset and pre-processing steps, in section 4.4.2, we'll go over the outlier detections of the dataset, and in sections 4.4.3 data normalization and windowing techniques quickly to fit data into the DL models. The discussion of different DL models is presented in section 4.4.4. Finally, the model evaluations and predictions criteria will be discussed in section 4.4.5.

4.4.1 Data Pre-processing

As mentioned in section 3, we developed a QoS data collection tool through which we collected different QoS service metrics of 5G networks, including throughput, latency, packet loss, and jitter. All of our data was collected in a controlled setting and saved in a CSV file for subsequent analysis. The data are collected every three minutes at time intervals for 50-day time periods. We have a total of 8,386 data samples over 50 days, where weekdays have 200 data samples per day and weekends have 140 data samples. Among the 8,385 samples, morning, afternoon, and evening have 2,764, 1,728 and 3,893 data points, respectively. However, there is some missing data in the dataset. We considered four QoS metrics, including throughput, latency, jitter, and packet loss, and daypart in our experiments. To develop the prediction model, we performed a multi-variate time series analysis by utilizing all 50 days of data. The timestamps in the dataset are in GMT format. The unit of our throughput data is in Mbps (Megabits per second), and latency and jitter are in millisecond format. Figure 4.7 shows the plot of throughput download of our collected data from 03/05/2022 to 22/06/2022.

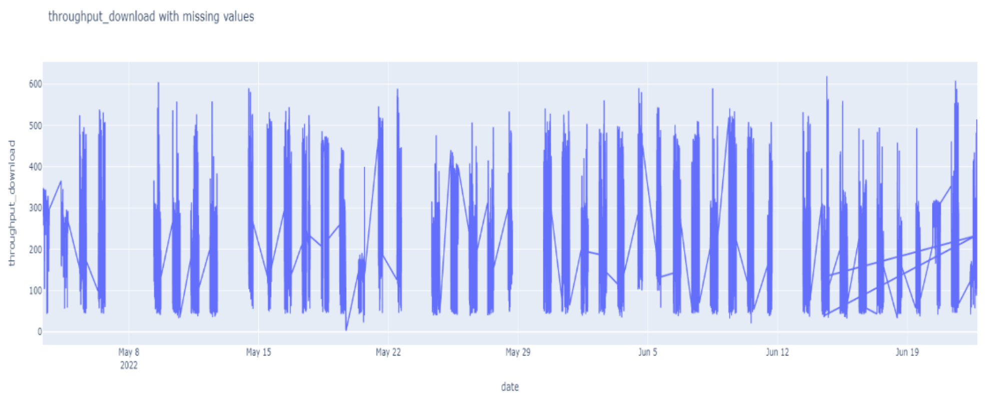


Figure 4.7: Throughput Download with missing values

In this section, we will discuss some of the data pre-processing techniques which are important for time series analysis and to fit our data into DL models.

1) **Imputing Missing Values:** Due to server-related issues and the library's closure, we cannot gather data for 6 days, but it is crucial to fill in the gaps in the next two days so that our time series analysis can make accurate forecasts for the future. Many methods can be used to close the gap, including backward and forward filling, linear and quadratic interpolation, and the mean of seasonal counterparts. However, the observations in our dataset depend on various times of the day, making typical interpolation algorithms impractical for our experiments. For instance, data from the afternoon and evening cannot be used in place of morning data. In our experiments, to fill in the data from a missing day, we considered the same days' values, calculated three separate means for three different portions of the days, and then filled in the missing day's data with those values. For instance, if data for one Monday is missing, we take the means of the morning, afternoon, and evening data obtained on Monday, considering all other Monday data, and then fill in the missing Monday's data with newly generated mean values. However, we also consider the traditional imputation techniques mentioned above, but those techniques do not perform well in our dataset because we have long sequences of

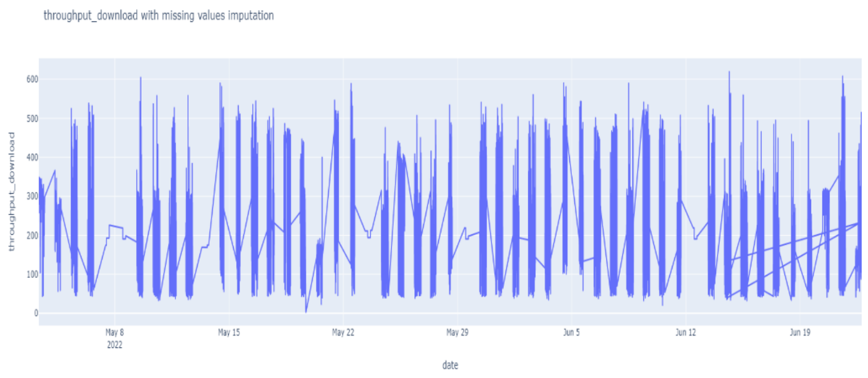


Figure 4.8: Throughput Download after missing value imputation

missing values. Figure 4.8 shows the plot of throughput download after imputing the missing values.

2) **Stationarity of the Dataset:** The predictions model is designed to work with stationary data and to become stationary, a time series data cannot be dependent on time. When a time series data has a trend or seasonality that cannot be stationary, it affects the overall time series predictions. The first step of time series forecasting is to check whether the data is stationary or non-stationary. In our experiments, we used Augmented Dicky Fuller Test (ADF test) to verify whether the dataset is stationary or not. In the ADF test, all our attributes in the dataset rejected the null hypothesis. The p value is less than 0.05 which is required to be stationary data; thus, our process is considered stationary. The p value for attribute "throughput_download" is $5.064114362981264e-13$ and the p value for attribute "throughput_upload" is $1.2180665142462095e-12$ and in the ADF test the number of lags we used is 26.

3) **Autocorrelation Analysis:** We employ autocorrelation to reveal the underlying patterns in the data. To fit the data into the supervised learning model, we must transform our time series features into an array format. We can determine the correlation between previous and future data that our future forecast depends on using the autocorrelation function (ACF) and partial autocorrelation function (PACF). The prior data values, usually referred to as lags, are utilized to identify the correlation. Future projections use various characteristics, including the number

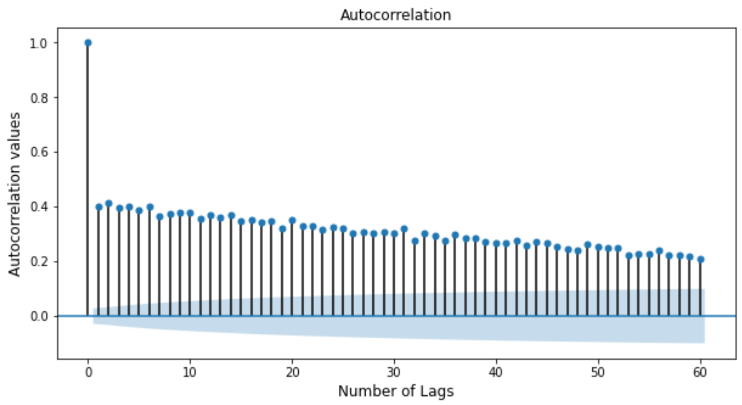


Figure 4.9: Autocorrelation Analysis for throughput

of lags that we will utilize in the ACF and PACF. In our tests, we used 60 lags. The ACF and PACF plots for the throughput of our time series data are shown in Figures 4.9 and 4.10, respectively.

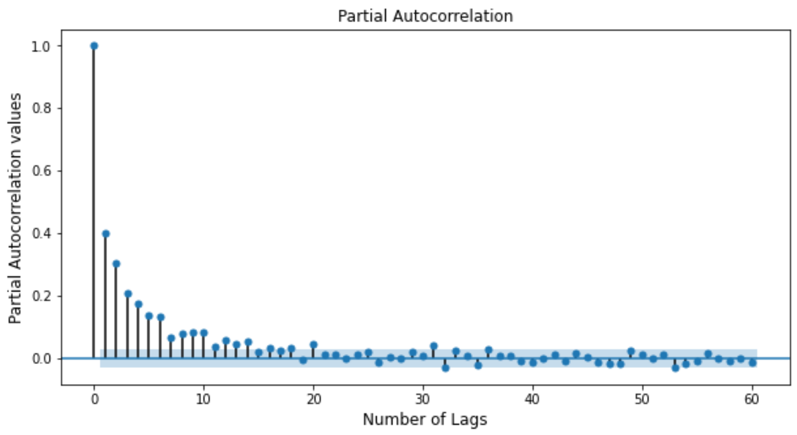


Figure 4.10: Partial Autocorrelation Analysis for throughput

We can calculate the number of previous periods necessary to predict future observations by examining the ACF and PACF plots. For instance, our PACF graphs demonstrate that to foresee the following data points, we require the previous 15 data points from lag 0 to lag 15.

4) **Outlier Detection and Mitigation:** Data may contain several outliers in 5G network traffic, such as sharp peaks or falls in throughput or latency. In time-series analysis, outliers are any data points with unusual values that do not follow a pattern resembling other data points. There are several standard techniques for finding outliers, including the Three Sigma rule and Isolation Forest. However, because of server problems, our data occasionally experience dramatic increases or decreases in throughput and latency, which are challenging to identify using conventional outlier identification techniques. We considered the top and bottom 6% of the data points outliers in our experiments and removed them. For instance, the throughput can fluctuate between 500 Mbps and 50 Mbps, which is completely unrealistic when compared to other data points.

4.4.2 Data Windowing Technique

To incorporate our time series data into the supervised learning models, we must format it correctly. We need to consider eight variables every time in the current formatting of our multi-variate time-series data. The time-series data's structure consists of a number of tuples (date, throughput, latency, jitter, packet loss, and day part), which the deep learning models are unable to accommodate. Therefore, we use a sliding window technique to reformat the time-series data, which forecasts the following observation based on the data from the past 15 observations.

4.4.3 Data Normalization

Data normalization, also known as feature scaling, is a pre-processing data strategy in which all data is transformed to the same scale. If our data has variable scales and the algorithm is unable to make assumptions about the data, normalization is necessary. Furthermore, normalization is a suitable strategy when we don't know the data distribution. Furthermore, data variables measured on disparate scales will not contribute to model training if they are not normalized. Scaling the data equalizes all features, allowing the algorithm to converge faster and optimize with the gradient descent approach, which is far more pleasant. Process data is changed from 0 to 1 in the normalization process. We used a min-max scaler to normalize our 5G networks QoS metrics dataset, which comprises throughput, latency, jitter, packet loss, and user number numbers from various ranges. Eq. 4.4 shows the formula to calculate min-max scaler.

$$min - max = \frac{x - min(x)}{max(x) - min(x)} \tag{4.4}$$

4.4.4 Deep Learning Models

We selected a variety of approaches for DL model types that are typically used for time-series analysis. We will employ five different deep learning algorithms, which are well known for time-series forecasting. In all of our DL models, we used an Adam optimizer with a learning rate of 0.0001. The overview of DL models is given below:

1) **Artificial Neural Network (ANN):** The input units, also known as receptors, and output units, also known as effectors, are interconnected by the layers that make up an artificial neural network. In our research, we employed a simple ANN with two hidden layers, each with 16 units. We employed the linear activation function in the output layer and the ReLU (rectified linear unit) activation function in the hidden layer.

2) **Recurrent Neural Network (RNN):** RNN only considers the current input and is designed to deal with sequential data, such as time-series analysis, language modeling, etc. In order to predict the output of that layer, it retains the output of that layer and sends it back to the input layer. In our experiments, we employed the ReLU activation function, two RNN layers, one dense layer, and one dropout layer and the output layer came up with a linear activation function.

3) **Long Short-Term Memory (LSTM):** LSTM is very well known for processing sequential patterns like time-series data. It overcomes the vanishing gradient problem of RNN and has memory cells that can store information for longer periods, which makes it superior to RNN. Our experiments used two LSTM layers with two 64 units, one hidden layer and one dropout layer. Both LSTM layers and dense layers come up with a ReLU activation function.

4) **Gated Recurrent Unit (GRU):** GRU is substantially the same as LSTM, although it is less complicated. The reset and update gates make up the two gates that make up GRU. However, LSTM contains three gates: an input gate, an output gate, and a forget gate. The gates are in charge of handling the information flow. In our experiments, we used two GRU layers with a tanh activation function and one hidden layer with a ReLU activation function. The output layer has a linear activation function.

5) **LSTM Encoder-Decoder (LSTM En_De):** It is also known as sequence-to-sequence modeling, designed to solve complex sequence-related problems. The most common architecture to build a sequence-to-sequence model is the encoder-decoder architecture, where the encoder is responsible for reading the input sequences, converting the information into internal state vectors, and passing it to the decoder. The decoder takes the internal state vector and output from the previous time step as input and returns a sequence of output. In our experiments, the encoder consists of one LSTM layer followed by one RepeatVector layer and the output of the encoder is passed to the decoder which is also an LSTM layer. We used the linear activation in the output layer.

4.4.5 Model Evaluation

Model evaluation is important to assess the performance of the model in both train data and test data. The popular model evaluation techniques for regression are discussed below:

1. **Mean Absolute Error (MAE):** It is used to calculate the difference between actual values and predicted values. It takes the sum of all actual value errors and divides them by the total number of observations. It is calculated as follows:

$$MAE = \frac{1}{N} \sum |y_{actual} - y_{predicted}| \tag{4.5}$$

Where, N is the total number of observations.

2. **Mean Squared Error (MSE):** It is almost the same as the MAE but instead of taking the absolute difference between actual and predicted values, it takes the squared difference between actual and predicted values. The formula to calculate MSE is shown in Eq. 4.6.

$$MSE = \frac{1}{N} \sum (y_{actual} - y_{predicted})^2 \tag{4.6}$$

3. **Root Mean Squared Error (RMSE):** RMSE is the simple square root of MSE. The formula of RMSE is given below:

$$RMSE = \sqrt{\frac{1}{N} \sum (y_{actual} - y_{predicted})^2} \tag{4.7}$$

MAE, MSE, RMSE are calculated using the scikit-learn library [35].

4. **Weighted Mean Absolute Percentage Error (WMAPE):** One of the most popular techniques to evaluate forecast accuracy is through WMAPE. Since time series include zero values, we do not employ Mean Absolute Percentage Error (MAPE), which yields infinite values. Additionally, MAPE does not take into account the amount of time or any variances in priority between the items.

However, WMAPE prioritizes quickly moving goods and provides a solution for dividing by zero problems [36]. Eq. 4.8 shows the formula of WMAPE.

$$WMAPE = \frac{\sum_{t=1}^{n}(w_t|A_t - F_t|)}{\sum_{t=1}^{n}(w_t|A_t|)} \times 100\ \% \tag{4.8}$$

Here, A_t is the real value and F_t is the predicted value.

To calculate the accuracy from WMAPE we can follow the below equations:

$$Accuracy = 1 - WMAPE$$

When compared to MSE or RMSE, MAE provides a direct depiction of the total of error components. By squaring the actual and forecasted values, MSE is differentiable and offers a bigger penalization error. MAE, on the other hand, treats all errors equally. WMAPE evaluates the error across all data points before averaging them. It is the weight of absolute error normalized over every data points. Additionally, WMAPE can detect more errors and outliers than MAE and RMSE. In our experiments, all the error results and accuracy of different deep sequence models are compared using WMAPE.

4.4.6 Development Platform

We used a Windows 10 PC with an AMD Ryzen 9 5900HX with built-in AMD Radeon graphics to create the stacking ensemble-based IDS model. Our central processing unit has 16 logical processors, 16 GB of RAM, and 512 GB of a solid-state drive (SSD) and runs at 3.30 GHz (SSD). Our IDS model creation is easier and faster now that we have 4GB of NVIDIA GeForce RTX 3050 graphics processing unit (GPU). Throughout the construction of our model, we used a Jupyter notebook and the Visual Studio Code IDE. Python 3.7 is the primary programming language for 5G networks QoS prediction. For data processing, we used the Pandas package, NumPy for numerical analysis, and Matplotlib and seaborn for graphing the experiment results. In addition, the Sci-Kit learn machine learning package is used.

4.5 Experiment Results and Discussion

We used the 5G QoS data to foresee throughput after gathering data from a controlled live 5G environment. 50 days of 5G QoS data were utilized in our studies; the first 43 days of data were used to train our deep learning models, and the final seven days of data were used to assess the effectiveness of our deep learning models. The four stages of our experiments are as follows: I. data preparation; II. outlier detection and mitigation; III. data normalization; and IV. model deployment and forecasting. We initially impute the missing data values during the preprocessing stage. The Augmented Dicky-Fuller test (ADF test) is then used to examine a number of traffic parameters, including trend, seasonality, and stationarity. The number of prior observations needed to forecast the future observation is calculated using the ACF and PACF plots in figures 4.11 and 4.12. The data windowing approach was then used, which is necessary to fit our data into the deep learning models. To improve our deep learning models' capacity for learning and prediction, we eliminated the outliers from the 5G QoS data in the second stage of our experiments. After that, we use a min-max scaler to normalize our data before fitting it into deep learning models.

Finally, in stage four, we deployed five different deep learning models, such as ANN, RNN and its three variants, including GRU, LSTM and LSTM En_De. The models were trained once, and three different periods of the day including morning, afternoon and evening were predicted. We predicted the throughput of the last seven days from 16-06-2022 to 22-02-2022. The model architecture is unique for each model and each model is trained with 100 epochs with a batch size of 16. The performance evaluation metrics for all models are presented in Table 4.2. The performance is measured in MSE, RMSE, MAE, and WMAPE, through which we calculated the error and from WMAPE we calculated the prediction accuracy of the models.

Table 4.2: Performance of all models

Model	MSE	RMSE	MAE	WMAPE	Accuracy
ANN	1.54	12.40	8.90	18.69	81.31
RNN	1.28	11.32	8.46	16.82	83.18
LSTM	0.99	9.96	7.34	14.57	85.43
GRU	1.01	10.05	7.49	15.89	84.11
LSTM En_De	0.89	9.46	6.92	13.75	86.24

Our findings demonstrate that the LSTM Encoder-Decoder outperforms all other models, with an average prediction error of approximately 13.75% between real and anticipated 5G QoS metrics. The RMSE and MAE error rate is also lower in the LSTM En_De compared to other models. In comparison to GRU models, which have an average error rate between actual and predicted 5G QoS of 15.89%, LSTM models had an average prediction error between actual and predicted 5G QoS of around 14.57%. There is no significant difference between LSTM and GRU in terms of RMSE and MAE error rate. Additionally, the error rate for the simple RNN model is 16.82%. But compared to other models, ANN's average error rate between real and forecasted traffic is greater at 18.69%. We observed significant RMSE and MAE error rate in ANN model. Overall, the LSTM Encoder Decoder achieved 86.24% accuracy in predicting the throughput. On the other hand, LSTM achieved 85.43% accuracy. Figure 4.11 illustrates the comparative performance of different deep learning models.

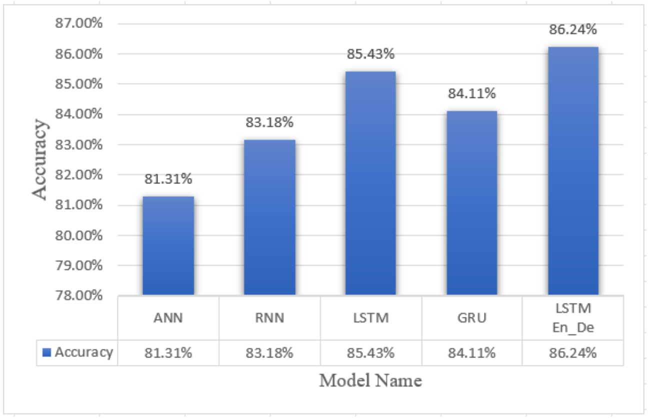


Figure 4.11: Comparative performance of sequence models

Table 4.3 shows the comparative graphical depiction of actual and predicted throughput in the morning using LSTM and LSTM Encoder-Decoder models.

Table 4.3: LSTM and LSTM En-De performance in the Morning

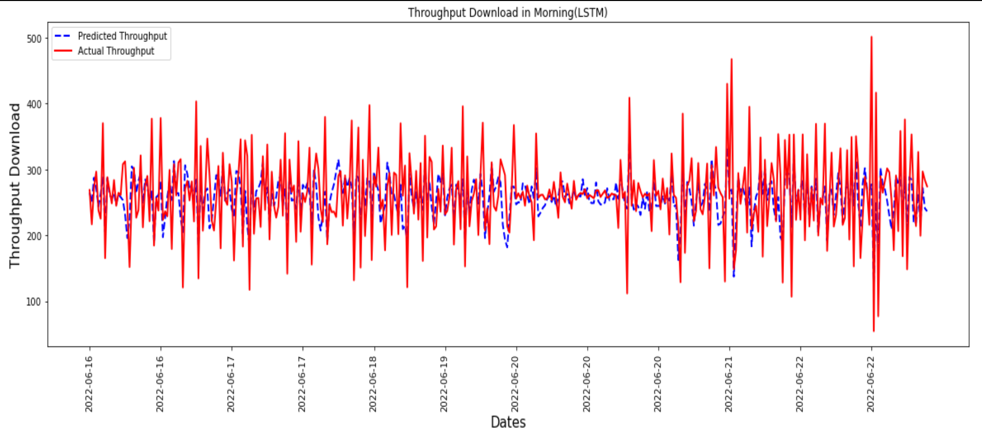
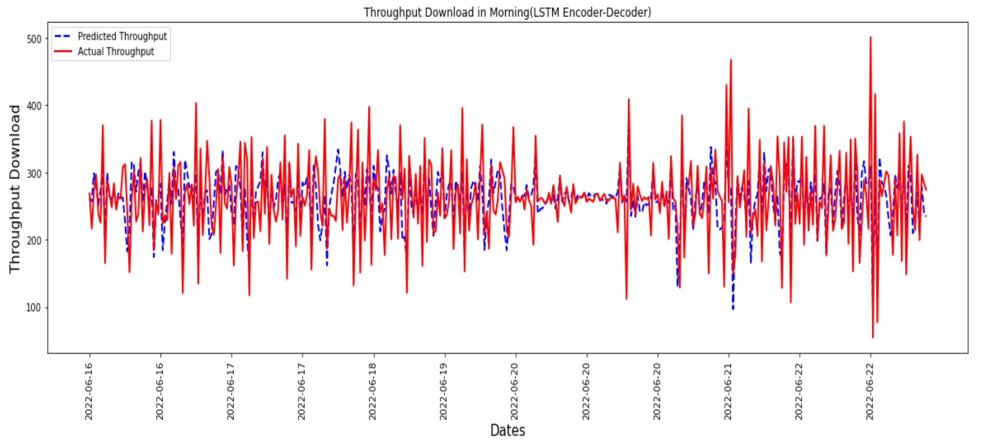
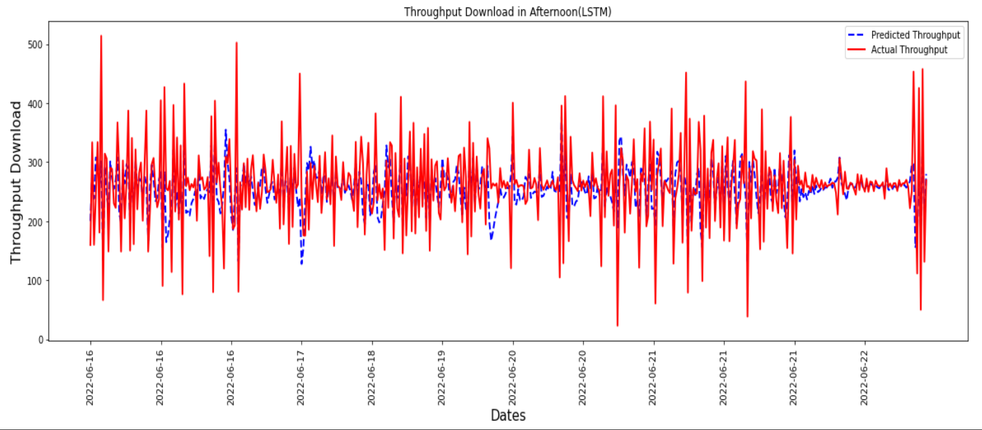
Models	Daypart: Morning
LSTM	
LSTM En_De	

Table 4.4 shows the comparative graphical depiction of actual and predicted throughput in the afternoon using LSTM and LSTM Encoder-Decoder models.

Table 4.4: LSTM and LSTM En-De Performance in the Afternoon

Models	Daypart: Afternoon
LSTM	

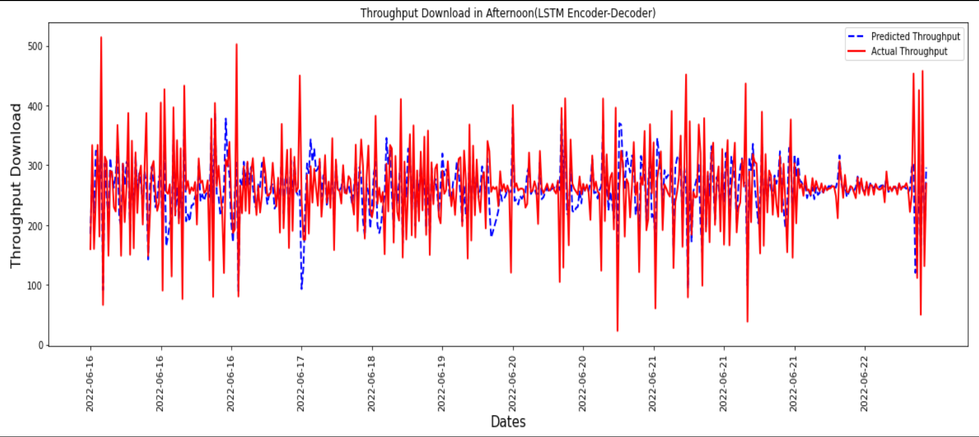
LSTM En_De	

Table 4.5: LSTM and LSTM En-De Performance in the evening

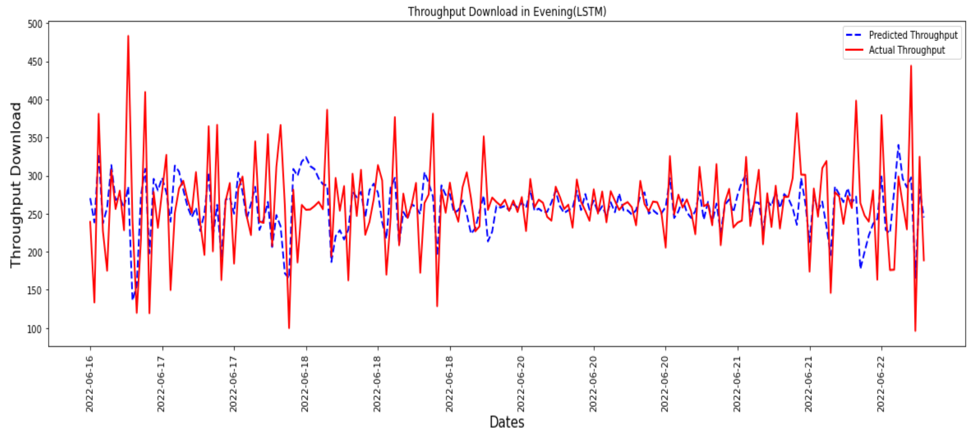
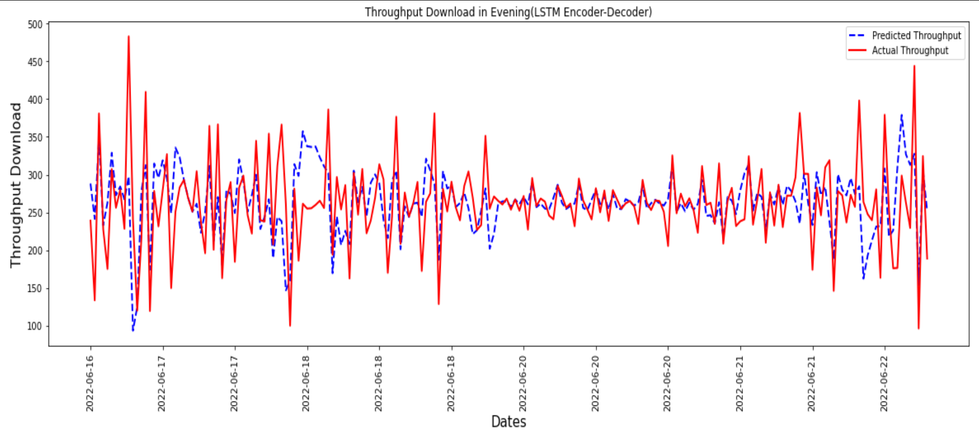
Models	Daypart: Evening
LSTM	
LSTM En_De	

Table 4.5 shows the comparative graphical depiction of actual and predicted throughput in the evening using LSTM and LSTM Encoder-Decoder models.

Our studies illustrate how well artificial intelligence methods perform when forecasting the throughput of 5G networks. However, a network's QoS is influenced by a variety of internal and external circumstances. Moreover, modifications in network behaviour, particularly in the 5G network, significantly influence the network's overall QoS. Training a deep learning model with abnormal QoS data of 5G will affect the performance of the model during prediction. In order for the model to learn the best possible features, it is crucial to analyze the data points in an appropriate way. Exploring the series' pattern and handling the missing values correctly is crucial for time series analysis. Moreover, detecting outliers and mitigating them before fitting the data into the model will increase the overall performance of the models; otherwise, there is a chance of learning from abnormal traffic, which will affect the performance of the model. Finally, the selection of models is essential because all deep learning models do not perform well with sequential data.

4.5.1 Applications of QoS Forecasting Tool

QoS forecasting based on time series data is a newer concept. Since the advent of 5G, it is becoming very popular as the number of users is increasing significantly, and internet service providers are facing challenges in supporting an increasing number of consumers. To overcome the challenges, our proposed QoS forecasting tool will assist both network service providers and end users in the following areas:

1) It will give predictions based on user mobility and service requirements.
2) By utilizing this tool, network service providers can decide the necessary deployment of network resources to support future traffic demand.
3) Service providers can automatically optimize and fine-tune the network service parameters based on the prediction results.
4) Digital service providers, such as autonomous cars, drone technologies, etc., rely heavily on the network's QoS. To ensure the uninterrupted operations of their services, digital service providers will need an effective and accurate QoS forecasting tool.

4.6 Conclusion

Predicting the QoS of 5G networks is critical for network planning and provisioning. 5G is reshaping next-generation digital services, infrastructure, and applications, including industrial-scale Internet of Things (IoT), autonomous transportation, mission-critical communications, and industrial automation. The stringent QoS requirement of a 5G network is critical for supporting the above-mentioned digital service sectors. This thesis contributes to developing a framework for collecting QoS metrics in a controlled 5G network environment and proposed AI techniques for predicting QoS patterns.

Our proposed 5G QoS metrics data collection tool is developed in a way that can mimic the real-life usage of a network, like uploading and downloading various types of files by a variety of users to and from a fixed server by utilizing a non-standalone (NSA) 5G network. In the process of collecting 5G QoS data, we considered four important QoS metrics of the network, including download throughput, upload throughput, latency, jitter, and packet loss. Additionally, factors like the number of network users, the days of the week, and different times of day like the morning, afternoon, and evening are taken into account. The data collection process accumulates 8385 data points over the course of 50 days during specific time periods, with a three-minute interval between each data point. However, several internal and external circumstances, such as server failure, the library's closing, etc., influence this procedure.

Time series analysis is a popular approach for analyzing sequential data collected over time. We preprocessed the collected 5G QoS metrics data, including missing value imputation and outlier mitigation, and fitted them into the deep learning models, including RNN, LSTM, GRU, and LSTM encoder-decoder, which are well known for analyzing time-series data. Our experiments demonstrate the effectiveness of predicting the QoS metrics for LSTM and LSTM Encoder-Decoder models.

References

[1] Verma, D., Saraf, H., & Gupta, S. (2022). Prediction of user throughput from Network Parameters of LTE Network using machine learning. *Mobile Networks And Applications*. https://doi.org/10.1007/s11036-022-01934-6

[2] Mohammed, N. H., Nashaat, H., Abdel-Mageid, S. M., & Rizk, R. Y. (2020, October). A framework for analyzing 4G/LTE-A real data using machine learning algorithms. In *International Conference on Advanced Intelligent Systems and Informatics* (pp. 826-838). Springer, Cham.

[3] Alho, L., Burian, A., Helenius, J., & Pajarinen, J. (2021, March). Machine learning based mobile network throughput classification. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1-6). IEEE.

[4] Zhu, G., Zan, J., Yang, Y., & Qi, X. (2019). A supervised learning based QoS assurance architecture for 5G networks. *IEEE Access*, *7*, 43598-43606.

[5] Baryun, A. N., & Hamed, K. B. (2021, September). Evaluating Data Speed for 5G Mobile Technologies using Queuing Models. In *2021 IEEE Asia-Pacific Conference on Geoscience, Electronics and Remote Sensing Technology (AGERS)* (pp. 155-159). IEEE.

[6] Mollel, M. S., Abubakar, A. I., Ozturk, M., Kaijage, S. F., Kisangiri, M., Hussain, S., ... & Abbasi, Q. H. (2021). A survey of machine learning applications to handover management in 5G and beyond. *IEEE Access*, *9*, 45770-45802.

[7] Tayyab, M., Gelabert, X., & Jäntti, R. (2019). A survey on handover management: From LTE to NR. *IEEE Access*, *7*, 118907-118930.

[8] Mardian, R. D., Suryanegara, M., & Ramli, K. (2019, June). Measuring quality of service (QoS) and quality of experience (QoE) on 5G technology: A review. In *2019 IEEE International Conference on Innovative Research and Development (ICIRD)* (pp. 1-6). IEEE.

[9] Ye, Q., Li, J., Qu, K., Zhuang, W., Shen, X. S., & Li, X. (2018). End-to-end quality of service in 5G networks: Examining the effectiveness of a network slicing framework. *IEEE Vehicular Technology Magazine*, *13*(2), 65-74.

[10] Bui, N., Cesana, M., Hosseini, S. A., Liao, Q., Malanchini, I., & Widmer, J. (2017). A survey of anticipatory mobile networking: Context-based classification,

prediction methodologies, and optimization techniques. *IEEE Communications Surveys & Tutorials*, *19*(3), 1790-1821.

[11] Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C., & Liu, Y. (2018). A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Communications Surveys & Tutorials*, *21*(1), 393-430.

[12] Pan, Y., Li, R., & Xu, C. (2022). The First 5G-LTE Comparative Study in Extreme Mobility. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, *6*(1), 1-22.

[13] Minovski, D., Ogren, N., Ahlund, C., & Mitra, K. (2021). Throughput prediction using machine learning in lte and 5g networks. *IEEE Transactions on Mobile Computing*.

[14] Daengsi, T., Ungkap, P., Pornpongtechavanich, P., & Wuttidittachotti, P. (2021, August). QoS Measurement: A Comparative Study of Speeds and Latency for 5G Network Using Different Speed Test Applications for Mobile Phones. In *2021 IEEE 7th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)* (pp. 206-210). IEEE.

[15] *Global Mobile Data Traffic Market to Reach 220.8 Million Terabytes per Month by the Year 2026*. GlobeNewswire News Room. (2022). Retrieved 29 April 2022, fromhttps://www.globenewswire.com/news

[16] (2022). Retrieved 29 April 2022, from https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast.

[17] Rodrigo Muñoz, L., Mora, D., & Barriga, R. (2017). Teachers Cisco certification and their impact in Cisco Networking Academy program. *Revista Científica De La UCSA*, *4*(1), 50-56. https://doi.org/10.18004/ucsa/2409-8752/2017.004(01)050-056

[18] *IMT-2020 - Wikipedia*. En.wikipedia.org. (2022). Retrieved 29 April 2022, from https://en.wikipedia.org/wiki/IMT-2020.

[19] Shafi, M., Molisch, A. F., Smith, P. J., Haustein, T., Zhu, P., De Silva, P., ... & Wunder, G. (2017). 5G: A tutorial overview of standards, trials, challenges,

deployment, and practice. *IEEE journal on selected areas in communications*, *35*(6), 1201-1221.

[20] Samba, A., Busnel, Y., Blanc, A., Dooze, P., & Simon, G. (2017, May). Instantaneous throughput prediction in cellular networks: Which information is needed?. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)* (pp. 624-627). IEEE.

[21] *LTE UE Category & Class Definitions - CableFree*. CableFree. (2022). Retrieved 29 April 2022, from https://www.cablefree.net/wirelesstechnology/4glte/lte-ue-category-class-definitions/.

[22] Mitra, K., Zaslavsky, A., & Åhlund, C. (2013). Context-aware QoE modelling, measurement, and prediction in mobile computing systems. *IEEE Transactions on Mobile Computing*, *14*(5), 920-936.

[23] Chinchali, S., Hu, P., Chu, T., Sharma, M., Bansal, M., Misra, R., ... & Katti, S. (2018, April). Cellular network traffic scheduling with deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*.

[24] Zorzi, M., Zanella, A., Testolin, A., De Grazia, M. D. F., & Zorzi, M. (2015). Cognition-based networks: A new perspective on network optimization using learning and distributed intelligence. *IEEE Access*, *3*, 1512-1530.

[25] Li, R., Zhao, Z., Zhou, X., Ding, G., Chen, Y., Wang, Z., & Zhang, H. (2017). Intelligent 5G: When cellular networks meet artificial intelligence. *IEEE Wireless communications*, *24*(5), 175-183.

[26] Chen, T., Matinmikko, M., Chen, X., Zhou, X., & Ahokangas, P. (2015). Software defined mobile networks: concept, survey, and research directions. *IEEE Communications Magazine*, *53*(11), 126-133.

[27] Chen, Y., Wu, K., & Zhang, Q. (2014). From QoS to QoE: A tutorial on video quality assessment. *IEEE Communications Surveys & Tutorials*, *17*(2), 1126-1165.

[28] Raca, D., Zahran, A. H., Sreenan, C. J., Sinha, R. K., Halepovic, E., Jana, R., & Gopalakrishnan, V. (2020). On leveraging machine and deep learning for throughput prediction in cellular networks: Design, performance, and challenges. *IEEE Communications Magazine*, *58*(3), 11-17.

[29] Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

[30] *Speedtest CLI: Internet speed test for the command line*. Speedtest.net. (2022). Retrieved 29 April 2022, from https://www.speedtest.net/apps/cli.

[31] Stonebraker, M., & Rowe, L. A. (1986). The design of Postgres. *ACM Sigmod Record*, *15*(2), 340-355.

[32] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, *2014*(239), 2.

[33] The Concise Encyclopedia of Statistics. (2009), *23*(2), 34-35. https://doi.org/10.1108/09504120910935282

[34] *Anwar Haque | Western University UWO*. Csd.uwo.ca. (2022). Retrieved 29 April 2022, from https://www.csd.uwo.ca/~ahaque32/projects.php.

[35] API Reference. (2022). Retrieved 19 July 2022, from https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

[36] Ekambaram, V., Manglik, K., Mukherjee, S., Sajja, S. S. K., Dwivedi, S., & Raykar, V. (2020, August). Attention based multi-modal new product sales time-series forecasting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 3110-3118).

Chapter 5

Conclusion and Future Directions

This thesis proposed two AI-based approaches to protect the network against cyber attacks and predict the network quality of service (QoS) for efficient network planning and provisioning. The two interrelated studies presented innovative approaches, methods, and findings in their respective domains. The first study proposed a novel network intrusion detection system that utilized two popular network traffic datasets, UNSW-15 and CICIDS-2017, and employed the stacking ensemble technique to detect cyberattacks. The best possible traffic flow features are selected using random feature elimination with cross-validation (RFECV) and the SMOTEENN technique is employed to resample the imbalance classes. Finally, three different deep learning models are employed in the first layer of the proposed stacking ensemble model, then the output of the first layer is concatenated and passed to the second layer of the stacking ensemble model, which is a deep neural network model. The network anomaly detection accuracy is higher for stacking ensemble techniques compared to the single deep learning models used in the first layer of the proposed model. The second study developed a 5G network QoS prediction framework based on our collected 5G network data. Various data preparation techniques were employed, such as missing value imputation and outlier detection, while autocorrelation and data windowing techniques were used to uncover the underlying data patterns. The preprocessed QoS data are utilized through five different deep sequence models to forecast the throughput of the 5G network at various times of the day, including the morning, afternoon, and evening. The five different deep sequence models were compared, and LSTM Encoder-Decoder and LSTM demonstrated their accuracy in predicting throughput.

Some of the limitations of our study and future directions of this research are discussed in the following sections.

5.1 AI-based Intrusion Detection system

Although deep learning and stacking ensemble approaches have the potential to be used to design data-driven intrusion detection systems, there may be certain restrictions and difficulties when implementing the systems in real-world production settings. One limitation of the proposed method is that it can't detect new attacks before they are detected since it is constrained by the labels used in the training phase. If an attack is a part of a traffic flow designated for evaluation by the model, it will detect it as malicious but report the wrong type of assault. Additionally, if an intrusion detection system acquires new traffic and cannot distinguish it from legitimate traffic or other types of intrusions, the performance of the stacking ensemble may deteriorate. However, this may be prevented by using the right feature engineering and data preprocessing techniques.

Future research in this field will focus on the following areas:

1. Deploy the hybrid model proposed in this thesis to live network traffic, so that the model can analyze the traffic flow of networks and distinguish between normal and malicious traffic.
2. The IDS system may be installed on any edge devices that are linked to networks. Additionally, by using various feature selection and dimensionality reduction approaches in the pre-processing phases, the IDS model will be able to identify low-frequency assaults as well as intrusions with regularly changing traffic patterns.
3. Employed the transfer learning technique for our models so that we can assess how well they function on unknown network traffic. The capacity to understand data is crucial for practical intrusion detection systems and future research on intrusion detection systems may focus significantly on the interpretability of models.

5.2 5G QoS Data Collection and Predictions

Network data collection can be impacted by some internal and external factors, including hardware and service availability-related issues. Our proposed data collection mechanism utilized a high performance server, a wi-fi 6-enabled router, and NSA 5G networks. Though we collected 5G QoS data continuously within a fixed period, our process sometimes terminated unexpectedly due to server unavailability and Docker container-related issues. The behavior of the network is unpredictable, such as sometimes we face a sharp drop in throughput or a sudden increase in throughput, which results in creating unrealistic data points for 5G networks. The sharp increase or decrease in throughput and other QoS metrics creates significant random fluctuations in data patterns, and it is difficult for deep learning models to learn those patterns during predictions.

This research introduced a method for collecting data on 5G QoS measures from a large-scale field test and a mechanism for predicting the throughput of 5G networks using deep sequence models. Our data collection tools gathered four major 5G QoS metrics, including throughput, latency, jitter, and packet loss, and deep learning models forecasted the network's throughput. Nevertheless, several internal and external network elements have an influence on the process as a whole. The following items are expected to be part of our upcoming research:

1. Further improvement in the automation of the data gathering process will make it easier to monitor server utilization, network availability, and network anomalies in real-time, and it will also resume the entire process when a process terminates.
2. In this research we used NSA 5G networks. However, in the future, we can collect 5G QoS metrics using stand-alone (SA) 5G networks and make a comparison of how the performance of the two networks fluctuates.
3. In predicting the throughput of the 5G network, the efficient data preprocessing mechanism, including feature selection, dimensionality reduction, and outliers' detection, will increase the prediction accuracy. The proposed deep learning models performed well, and in the future, we would like to use statistical analysis and attention mechanisms in single-step and multi-step throughput predictions.
4. The development of a hybrid model to gather data on 5G QoS metrics from diverse user groups, evaluate the data, and suggest end users' and service providers' real-time predictions of the network's QoS.

Curriculum Vitae

Name:	Ibrahim Mohammed Sayem
Post-secondary Education and Degrees:	University of Western Ontario London, Ontario, Canada 2021 – 2022 M.Sc. (Computer Science) University of Chittagong, Chittagong – 4331, Bangladesh 2013-2018 Bachelor of Computer Science and Engineering
Honors and Awards:	Western Graduate Research Scholarship (WGRS) 2021-2022
Related Work Experience	Graduate Research and Teaching Assistant University of Western Ontario 2021 – 2022

Publications: M. Sayed, I. Sayem, S. Saha, and A. Haque, "A Multi-Classifier for DDoS Attacks Using Stacking Ensemble Deep Neural Network", Accepted, *IEEE 18th International Wireless Communications and Mobile Computing (IWCMC 2022).*