National College of Ireland

Enhancing Real-Time Threat Detection in Data Centre Firewalls Through Deep Learning & Machine Learning Techniques

MSc Research Project
MSc Cyber Security

Muhammad Usman Sarfaraz
Student ID: X23160667

School of Computing
National College of Ireland

Supervisor: Liam McCabe

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: **Muhammad Usman Sarfaraz**

Student ID: **X23160667**

Programme: **MSc Cyber Security** **Year:** **2023**

Module: **Thesis**

Supervisor: **Liam McCabe**

Submission Due Date: **12/8/2024**

Project Title: **Enhancing Real-Time Threat Detection in Data Centre Firewalls Through Deep Learning & Machine Learning Techniques**

Word Count: **8000** **Page Count** **20**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: **Muhammad Usman**

Date: **12/8/2025**

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on a computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Contents

[1]https://www.unb.ca/cic/datasets/ids-2017.html

Table of acronyms

Acronym	Full Form
AI	Artificial Intelligence
ANN	Artificial Neural Network
APT	Advanced Persistent Threat
ASIC	Application-Specific Integrated Circuit
CSV	Comma Separated Values
CPU	Central Processing Unit
CZ	Critical Zone
DDoS	Distributed Denial of Service
DMZ	Demilitarized Zone
DPI	Deep Packet Inspection
IDS	Intrusion Detection System
IP	Internet Protocol
IoT	Internet of Things
IPtables	IP packet filter firewall system
JSMA	Jacobian-based Saliency Map Attack
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MAC	Media Access Control
MLP	Multi-Layer Perceptron
ML	Machine Learning
NGFW	Next-Generation Firewall
PCA	Principal Component Analysis
PCAP	Packet Capture
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
Scapy	Python-based network analysis tool
SMOTE	Synthetic Minority Over-sampling Technique

[1]https://www.unb.ca/cic/datasets/ids-2017.html

Enhancing Real-Time Threat Detection in Data Centre Firewalls Through Deep Learning & Machine Learning Techniques

Muhammad Usman Sarfaraz
X23160667

Abstract

Cyber-crime has become one of the most significant risks known in the present world where the issue of connectivity and technological advancement is paramount coupled with the concern of data flow across the globe, which is why the protection of computer networks is of great significance. Traditional firewalls, like security guards of a network operated on a set of simple rules that controlled the network traffic, were effective when designed in the late 1980s [1]. Yet, for some reason, the modern-day methods employed by cybercriminals have changed concurrently with the advancement of modern technology. This thesis aims to analyse the possibility of using machine learning, a subfield of artificial intelligence, in security system systems as a new perspective on network protection. The objective was to design a better intelligent security system capable of distinguishing between malicious actions and blocking them. Using real-time data analysis, the new security system can quickly change its security rules to stop an attack. To this end, the actual statistical information about the traffic in the network was gathered and analysed in real time. Machine learning algorithms were then used to learn features corresponding to cyber threats. These models were incorporated within the security system in a way that it could automatically escalate its security system. As these findings indicated, the newly applied approach enhanced the firewall efficiency in identifying and blocking cyber threats. From this research, it is evident that the profession can apply state-of-the-art technology to improve the protection of the networks to make the world behind computers safer from evolving dangers.

1 Introduction

The progress in information technology, particularly using the Internet and increased computing power, underlines the importance of safeguarding networks. Firewalls, which serve as the first line of defence for organisational systems against various cyber threats, have been in use since the late 1980s [1]. Initially, firewalls enforced rules on network communications, accepting or rejecting messages based on predefined criteria. While effective in the past when network traffic was lower, and threats were less complex, these traditional firewalls have become inadequate as network traffic has evolved. Modern threats have outpaced these static barriers, necessitating more advanced and adaptive security measures.

This thesis begins by creating a proof for integrating machine learning in security system systems. It is important to minimise the occurrence of threats, which can be achieved through developing a system that scans and prevents suspicious network activities from happening within the shortest time possible. This project aims to introduce machine learning models that

[1]https://www.unb.ca/cic/datasets/ids-2017.html

will update security system systems for better threat detection results effectiveness. The integration consists of network traffic monitoring, data pre-processing steps, and applying trained artificial intelligent models on the traffic. When a threat is identified, the deployed model will detect it and provide its response to the security system, which in turn will act on the decision provided by the model.

Another significant thing is the data collection process for this project. The CICIDS2017 dataset of the Canadian Institute for Cybersecurity constitutes the primary source of network traffic data. This dataset comes with a full spectrum of typical or realistic and adversarial network traffic profiles. It is, therefore, recommendable for use in the training or testing of machine learning models. Pre-processing is applied to the data, and features are brought into a normalised form that would render the model's training process more accurate. One more thing to add here is the validity of the trained model, as it is a fact that the threat landscape evolves with every passing day. New threats (zero days), new exploitation techniques and new malware are developed, making the model imperfect to use, so the model must be constantly upgraded by training it on new datasets.

The second process is model building and training, where the data needs to be processed and used to train a machine-learning model. This project adopts Random Forest and Artificial Neural Network (ANN) models because of their high classification and anomaly detection efficiency. Before training on the CICIDS2017 dataset, the models are pre-processed and assessed based on accuracy, precision, recall, and F1-sheet.

The trained models can then be deployed as part of the security system. Network traffic data will be captured in real-time to observe the traffic in the network through the Scapi tool. These real-time data are also pre-processed similarly to those used for training to be consistent. A Python script is employed to evaluate the captured traffic by applying the trained machine-learning model. If the traffic is malicious, the system can send the response to the firewall. Based on that response, the security system rules will be updated. This helps prevent the network from being flooded with malicious traffic as it is a quick way of counteracting the attacks, making the network more secure.

2 Related Work

Since their inception in the early 1980s as packet filters, firewalls have significantly evolved. Modern firewalls have evolved into comprehensive security solutions, incorporating deep packet inspection (DPI) and threat intelligence features. This transformation has been occasioned by enhanced advancement in cyber-attack techniques and, as such, requires more enhanced defences [1].

Liang & Kim discuss transitioning from traditional firewalls to Next-Generation Firewalls (NGFWs), incorporating application awareness, internal intrusion prevention, and cloud-based threat intelligence. Their work's primary advantage is highlighting the additional protection features offered by NGFWs that are crucial for the networks. However, there is no

[1]https://www.unb.ca/cic/datasets/ids-2017.html

deep analysis of the provided metrics and their changes depending on the network conditions; therefore, their real-world performance is still doubtful.

Stateful firewalls enhance security by using session-tracking technologies to make decisions based on the context of packets within a session [2]. Aswal et al. [3] have pointed out that although these firewalls can better identify and filter more complex threats, they cannot still be dynamic and hence require frequent manual updates. On the one hand, stateful firewalls can offer contextual security, but on the other, they are based on static rules, and SOC can become easily outdated.

Kurdy and Shaheed [4] highlight Web Application Firewalls (WAFs), which rely on machine learning to improve attack detection. From their research, they were able to assess that WAFs can identify and prevent SQL injection attacks, which indicates the flexibility of the machine learning models. However, the most apposite drawback is the excessive load on computational resources necessary to perform real-time threat assessment. This overhead can effectively affect web applications running on username/password authentication, especially in high-usage situations.

Deep learning algorithms have enhanced security system filters, providing multiple levels of protection against diverse and multi-layered attacks. Several research studies about IDS use deep learning models; Elsheikh et al. [5] hint at the efficiency of such models compared with conventional models. The major advantages of deep learning models are their elevated level of detection accuracy and the ability to estimate intricate patterns. They expose that these models demand a lot of computation, and that big data is needed to form them. This fact makes their implementation in resource-constrained environments hard to achieve.

Liang and Kim [6] researched and discussed the integration of deep packet inspection (DPI) into NGFWs and behavioural analytics. DPI, therefore, allows firewalls to extend their identification beyond TCP headers to discover threats contained in legitimate traffic sessions. The major advantage of the work presented concerns the coverage of DPI solutions as the key contributor to security, especially in terms of APTs and other types of modern malware. That said, DPI has two key issues that should be of great concern: complex configuration and performance overheads.

In this study, Weissman and Jayasumana [7] aimed to study the adoption of IoT monitoring in tandem with Security Operation Centres (SOCs) to improve the security of networks. They've proved that although this form of integration allows for improved insight and policy over network flows, it also poses recent problems for the processing and analysing the enormous amount of data produced by IoT devices. The benefit of this approach is enhanced security since all the possible proceedings are monitored well. Still, the disadvantage is that it is complex in terms of managing since it is complex in terms of management.

Velásquez et al. [8] propose a new machine-learning-based ensemble system for Class-A real-time anomaly detection in Industry 4. 0 systems. Their approach involves the use of

[1]https://www.unb.ca/cic/datasets/ids-2017.html

ensemble learning, which significantly enhances the parameters for detecting and distinguishing authentic images from fake ones while at the same time reducing the likelihood of realistic fake images being classified as fakes. Despite the favourable outcomes documented by this method, it shoulders the issue of managing large industrial networks and the high computational power required. It is quite strong in producing highly accurate anomalies, though the implementation process might be complicated, hindering straightforward application.

Vaneiro and Casas [9] introduced ensemble-learning strategies for utilising network security and anomaly detection. They also establish that achieving high throughput is possible using multiple models to improve detection rates. However, they also point out that the price for a higher number of input parameters is higher computational costs, which might pose a problem when applying this technique in the real world. What has been ignored is that although ensemble-learning methodologies are strong in their capability to detect anomalies, their high resource utilisation is the biggest drawback.

The paper of Zhao et al. [10] introduced a real-time network anomaly detection system using machine learning. Their system can work in real time and can distinguish between normal and anomalous data, making it better than some other models. However, I prefer to focus on model limitations and that models need to be updated frequently because of the constantly changing nature of information. Their strong point is the ability to detect vulnerabilities in real time; however, several vulnerabilities, such as the continual need to update and manage false positives, can be considered areas of concern.

As the security threats have grown more complex, the concept of the security system as a simple barrier to the systems has also changed over the time. The first generation of firewalls comprises the packet filtering firewalls: it worked by comparing the network packets to the prescribed form of operations without consideration of its connections. Even though these stateless filters proved effective and not very resource hungry, the approach was inadequate in countering new attack methodologies deemed stateful, which evolved over time [11]. Stateful inspection firewalls stood for a breakthrough by actively preserving context information relative to active connections, enhancing precise decision-making processes on which packets are allowed [12].

During the 1990s, firewalls started including circuit-level gateways and application proxies, which provided new levels of security by evaluating the specifications of TCP connections and the application result data correspondingly. Circuit-level gateways work at the transport layer to control session requests completely. At the same time, application proxies look at the Content of the packets, allowing the detection of evil during a good data stream [13]. Although increasing security, these methods brought increased computational load and delay and, therefore, are unsuitable for high-speed networks with low-level hardware implementation [14].

[1]https://www.unb.ca/cic/datasets/ids-2017.html

The emergence of Next Generation Firewalls (NGFWs) in the early 2000s provided a breakthrough. Due to the convergence of various security processes, an NGFW combines DPI (Deep Packet Inspection), IPS Intrusion Prevention System), and application control in a single appliance. The integration, therefore, enables the NGFWs to rapidly check and detect different threats since there is a correlation of information between the other. Security Layers [15]. For instance, in Ahmadi's comparative study, NGFWs are said to outcompete traditional firewalls in detecting and preventing complex cyber threats through machine learning [16].

However, both traditional and NGFWs have advantages and disadvantages, which will be discussed below. Usually, they are not designed to process encrypted traffic well, as encrypted traffic is becoming more common now because of HTTPS, etc. Moreover, it should be noted that the performance of intensive security functions, including DPI, can become critical when the traffic rate increases [17]. Such considerations prevent one from using the firewall technology invariantly, implying that technological advancement is crucial in modifying the security system to fit the needed security efficiency and speed.

As for further development, the application of artificial intelligence (AI) and machine learning (ML) in security system development can help overcome some of these issues. Compared to routinary firewalls, AI firewalls can self-learn from past attacks and alter their mechanism according to new threats or the network's behaviour. For example, murder AI can enhance the firewall's accuracy in detecting anomalies since Mishra et al. developed an AI-based security system that tremendously enhances the differentiation of regular traffic from malicious traffic [18].

In addition, many systems offer decentralised and distributed security system architectures. These differ from traditional perimeter firewalls that integrate security appliances at numerous network segments. This approach improves control of internal threats and eliminates easily compromised security points. This approach is especially useful in cloud and hybrid environments because the method of protection by boundaries is ineffective in such conditions [19].

Lastly, the security system development includes the integration of the chip level firewalls and progress towards the improvement of the hardware acceleration. Such solutions make use of additional hardware adjuncts and can help to transfer the major share of the processing load to the other more efficient CPU therefore raising the throughput rate of security system systems. For instance, Mao et al., centre their study on the improvement of security system efficiency by utilizing special items of hardware such as ASIC and network processors. These specialized chips are elected for packet filtering and encryption and other tasks of similar nature, and the performance is enhanced without any depreciation of security aspect [20].

[1]https://www.unb.ca/cic/datasets/ids-2017.html

3 Research Methodology

The research began with compiling network traffic data from the CICIDS2017 dataset, including both normal and abnormal activities. Live traffic on the network was also monitored using network sniffing programs such as Scapy. The collected data was pre-processed to eliminate noise and errors that could affect model training. All features were standardised to keep their scales unified, and using Principal Component Analysis (PCA), some of the features were eliminated, and only the most important ones were taken for the analysis.

Subsequently, Random Forest was chosen from classification algorithms, while Artificial Neural Networks (ANNs) were chosen from neural network algorithms. These models were trained and tested on the pre-processed datasets. Recall and F1-score were used to measure the model's accuracy and precision. Once the models delivered satisfactory results, Scripts were developed to capture real-time traffic and save it into a PCAP file, which was then converted to a CSV file for model input.

3.1 Materials and Equipment

This work includes experiment implementation and the arrival of conclusions through software and hardware. Python was the main programming language chosen for coding and training the machine learning models because of its extensive libraries and ease of use in machine learning. Regarding the Random Forest, the Scikit-learn library was utilized. Scikit Learn is one of Python's most comprehensive and versatile packages, usually used for machine learning.

The chosen Artificial Neural Network (ANN) model was created with TensorFlow and Keras. TensorFlow is an open-source platform for building and deploying ML created by Google. It is quite versatile and extensive in its support of ML models. TensorFlow, Keras, is a high-level neural network API that can run on top of TensorFlow, which makes creating and training deep learning networks easier. A sampling of actual traffic was done using Scapy, a Python-based tool commonly used to manipulate packets in the network. Scapy enables packet creation, manipulation, transmission, and reception, making it indispensable for network analysis and testing.

All the deep neural networks were trained on Google Collab, a cloud-based service that gains computational power for free from Google. Google Colab runs as a Jupyter Notebook, which lets users write and execute codes in Python in real-time. Training of complicated models where large datasets are needed does not require the local machine to be a high-performance one.

3.2 Sample Collection and Preparation

The samples for our research were collected from the CICIDS2017 dataset[1], which contains real-time network traffic data. With these samples, objectives followed during sample preparation included data cleaning, normalization, and feature reduction steps. The original

[1]https://www.unb.ca/cic/datasets/ids-2017.html

dataset was screened for any errors in the collected data, and any noise or unwanted data was eliminated. The data has gone through normalization to bring the range of independent variables to an acceptable level to be compared with each other to enhance the efficiency of the used models.

3.3 Measurements and Calculations

The model performance was assessed from the pre-calculated results of the corresponding measurements on the database. The computations derived from the raw data are the accuracy, precision, recall, and the F1-score. Accuracy was calculated by dividing the total correctly classified instances by the total instances, and precision is the number of actual positive cases divided by the total number of positive cases, including the false positive cases. Recall calculated the percentage representing the true positive samples against the total of true positive and false negative samples. On its part, the F1-score, the average of both precision and Recall in that order, was the best since it offered both precision and the ability to retrieve most of the documents of interest.

3.4 Statistical Techniques

Several statistical measures were used to analyse the data and compare the models. Cross-validation was employed to check the model's accuracy on a test set to avoid overfitting the over-specified model. The confusion matrix was used depending upon the performance measure of the model to identify the true positive and true negative, false positive and false negative. Evaluation metrics included the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) to establish the model's sensitivity and specificity for various threshold values.

4 Design Specification

4.1 System Architecture Overview

The system architecture of this project theoretically incorporates real machine-learning components into conventional firewalls to improve the dynamic detection of risks in data centers. This comprehensive architecture comprises several key components: they are the Data Collection Module, Preprocessing Module, Feature Extraction Module, Machine Learning Module, Evaluation Module, Analysis Module, and Firewall System. All these components have a profound and significant function regarding the functionality and efficiency of the system. As to the structure of the modules, the Data Collection Module is the one responsible for collecting the network traffic data and then sending it to the Preprocessing Module for filtering, normalization, and feature extraction. The Feature Extraction Module checks for and redacts any unwanted or unnecessary data attribute. Using an Artificial Neural Network with a Multi-Layer Perceptron in the Machine Learning Module focuses on detecting anomalies and threats within the network traffic. In the Evaluation Module, several measures are applied to measure the model's effectiveness, while in the Analysis Module, the models are continually observed and revised to improve their efficiency. In the last step, the Firewall System combines with a particular machine learning model that optimizes rules for it; therefore, it is real-time protection since it only allows

[1]https://www.unb.ca/cic/datasets/ids-2017.html

access to those with permission and blocks any unusual access. The architecture of the interconnected system guarantees the effective prevention of possible dangers that may affect the network system and its adaptations to a new threat that may occur.

4.2 Model Input Data Collection & Processing

Collecting data or live traffic capturing was under the Data Collection Module. In this module, packet sniffing from the computer's network interfaces was examined, and their analysis was made with the help of Scapy. These programs were used to monitor packets that go through a particular network or section of a network using the Scapy command-line tool. The collected data was stored as PCAP files, consisting of 'raw' data collected through the network analyzer. This module ensured that sufficient and intricate information about the network traffic pattern was gathered in a way optimal for developing and exercising the machine learning paradigms.

4.3 Preprocessing, feature extraction & Machine learning

The Preprocessing Feature Extraction and Machine Learning Module is a single component that entails data pre-processing, feature extraction, and machine learning to identify abnormal and insecure network traffic. First, raw data is pre-processed by deriving it from extraneous facts and data and normalizing it to provide more homogeneous values.

5 Implementation

5.1 Dataset Acquisition Procedure

The first data set employed in this research is the CICIDS2017 data set donated to the research world by the Canadian Institute for Cybersecurity. The highlighted dataset was chosen since it contains many normal and attack instances that could be effectively used for training and testing machine learning algorithms. The pretreatment process began with the dataset, where data cleansing was done to erase erroneous information while feature scaling was done consistently. Factors such as PCA (Principal Component Analysis) scores, number of features, correlation and interdependence values were used in feature reduction to select only the most optimal variables for the analysis.

5.2 Data Preparation & Extraction

The first type of implementation concerns data collection and entry. This is about pre-processing distinct traffic (e.g., 'BENIGN -> Non-Malicious Traffic,' 'DDoS -> Distributed Denial of Services') to different numerical values. Several utility functions are defined for this purpose, which include a function to get a list of traffic types in the Data Frame, a function to convert traffic type names into the corresponding numerical index and vice versa, another function to filter out the Data Frame based on a particular traffic type, and the final function to clean the Data Frame by replacing certain invalid value with None and converting the columns to string data type. The main function is to read the CSV files from a particular directory and clean and segregate traffic types in different data frames. After cleaning and categorising the data frames, they are written into Parquet files for future processing, as storing and reading data in this format is efficient.

5.3 Dataset Splitting & Feature Detection

The next step is to divide the dataset and identify its features. The utility functions from this phase include rearranging Data Frames randomly using the shuffle command, extracting traffic statistics from a Data Frame, loading a Parquet file into a list of Data Frames, subdividing the data into the training set, cross-validation set and the test set each in the required percentage; and identifying the non-informative features with only one unique value in all the sets. The main function reads the Parquet files into Data Frames and splits those Data Frames into train, cross-validation, and test. After that, it identifies non-informative features and displays them to take a second look at them. At last, the split data frames are (converted into) stored as Parquet files under a directory having a timestamp for its name to systematically manage the data.

Table 5.1: Steps in Dataset Splitting and Feature Detection

Step	Description
Load Files	Reading Parquet files into Data Frames for further processing.
Dataset Splitting	Dividing the data into training, cross-validation, and test sets based on specified percentages.
Feature Detection	Identifying non-informative features that can be removed to streamline the data.
Save Datasets	Storing the split datasets as Parquet files with a timestamped directory for efficient management.

5.4 Data Preprocessing & Model Development

The following process is data preprocessing and model building once the data is acquired. The training, cross-validation, and test datasets are stored in Parquet files and read into Pandas Data Frames. These Data Frames are pre-processed through the preprocess data function to replace categorical labels with numerical representations, impute missing data using column means, and normalise the features within the range [0,1]. To deal with the class imbalance in training data, SMOTE (Synthetic Minority Over-sampling Technique) is employed to create synthetic samples in minority classes. After preprocessing, the basic Multi-Layer Perceptron (MLP) model is developed using TensorFlow/Keras. The architecture of this model has an input layer followed by a hidden layer and an output layer, and to prevent overfitting, dropout has been used in this model. The MLP model is further trained to the pre-processed training data and checked to the cross-validation datasets. Once the training process is complete, the model's performance is assessed using the test data set by comparing them with actual labels. The model's accuracy, precision, and recall rates are calculated, and the confusion matrix is obtained as a visual display. Moreover, the adversarial examples are created with the help of the ART library, using the JSMA (Jacobian-based Saliency Map Attack) method for evaluating the model’s robustness.

Table 5.2: Features Used in Data Preprocessing and Model Training

Feature	Description
SMOTE	Used for balancing the training dataset by generating synthetic samples for minority classes.
MLP	A Multi-Layer Perceptron model created using TensorFlow/Keras consists of input, hidden, and output layers with dropout for regularization.
Label Encoding	Convert categorical labels to numerical values for model training.

Min-Max Scaling	Scaling of features to a range between 0 and 1 to ensure uniformity.
Adversarial Testing	The JSMA method was used to generate adversarial examples and test model robustness.

The indications given in the long description of the approach used in the model development stage suggest the choice made that Artificial Neural Network (ANN) was adopted owing to its suitability in the category of classification & anomaly detection. The ANN model under consideration is a Multi-Layer Perceptron (MLP), which was trained on the CICIDS2017 dataset, which had undergone data pre-processing. The specified model architecture was an input layer of 128 neurons with ReLU activation and a dropout layer at 0.2 to prevent overfitting. The next layer, which was the hidden layer, has 64 neurons and applied the ReLU (Rectified Linear Unit) activation function, and the last dropout layer applied a dropout rate of 0.2. The Softmax activation function was used for the output layer since the problem at hand was a multi-class classification problem. To train this model, this was compiled with the Adam optimizer and the sparse categorical cross-entropy loss. Thus, the F1 score was applied to measure the model's effectiveness besides accuracy, precision, and recall. The measures meant to avoid the compilation of scores were applied, supporting the assertion of the effectiveness of the established model in other networks.

5.5 Real-time packet capturing and data normalisation.

In the next step of our analysis, we capture the live packets and store all the data in a pcap file. Handling packet capturing is done through a Python script, which is identified as capturing.py. The `capturing. This is in the form of a `python` script intended to monitor the network packets from installed motives of accessible network interfaces and store them in a `. pcap` file. First, the script displays the network interfaces on the device, allowing the user to select an interface for packet capturing. A dummy handler function is presented for packet processing where the user can implement their packet processing code. The script captures packets on the specified interface until the script execution is interrupted by the user. Captured packets are temporarily stored in a list and then buffered to a .pcap file, named with a timestamp for later identification and analysis. This script allows for efficient real-time packet capturing, a preliminary step to obtaining data for my anomaly detection model.

The subsequent procedure involves writing another Python script to parse the captured real-time packet to produce a CSV format usable for the subsequent data analysis with the developed machine learning algorithm. This conversion script takes the given name of the pcap file as an argument, goes through all the packets in the given file and extracts the packet features such as Timestamp or the source and destination IP address or port number of the packets and the protocols used for the packets and writes all these into a CSV format and with the output name given as argument. The script uses other libraries in packet manipulation while Pandas manages CSV (Comma Separated Values) operations. After obtaining the data in CSV format, additional work can be done on it in the same way as on the model's training data. The CSV data that has undergone pre-processing is then input to the Artificial Neural Network (ANN) model to classify each packet as either normal or suspicious on its way to the destination. This straight-through processing guarantees that the captured network traffic is analysed in real-time, increasing the chances of getting intelligence in the event of a security threat and better network fortification.

5.6 Proposed Architecture

The proposed system combines a security system with a machine learning model to enhance real-time threat detection for the network. The process starts with the continual surveillance

of the traffic flow across the network with the help of programs like `tcpdump` and its graphical equivalent, `Wireshark`, with observations to confirm that all packets on the network go through the analysis. The raw traffic data derived from the sources is then subjected to intense preprocessing that involves the removal of unwanted and invalid data, normalisation to have comparable input data and feature extraction to feed the machine learning model with proper data. Random Forest or Artificial Neural Networks (ANNs) usually constitute the trained model examining this data to sort traffic as malicious or benign. The first of these firewalls, located beneath the gateway router, uses a static rule but stands to update by the ML model's real-time classifications. When the security system recognises malicious traffic, it can modify its rules using the `iptables` command, thus strengthening its capacity to control new forms of traffic threats. Dynamic rule updating is applied to firewalls within specific sections, such as the Demilitarized Zone (DMZ), Critical Zone (CZ), or IoT Zone (IoTZ), to prevent threats from spreading to other network sections. Furthermore, the VPN gateway that secures remote branches' connection can also connect with the developed ML model to constantly analyse and regulate VPN traffic to maintain security regarding all the available access points. This is where machine learning comes into play to optimise and adjust the security system rules, hence the best protection mechanism to deal with ever-emerging threats in the system. Thus, the justification of the integration proves that the offered process increases the accuracy and effectiveness of threat identification and network space protection.

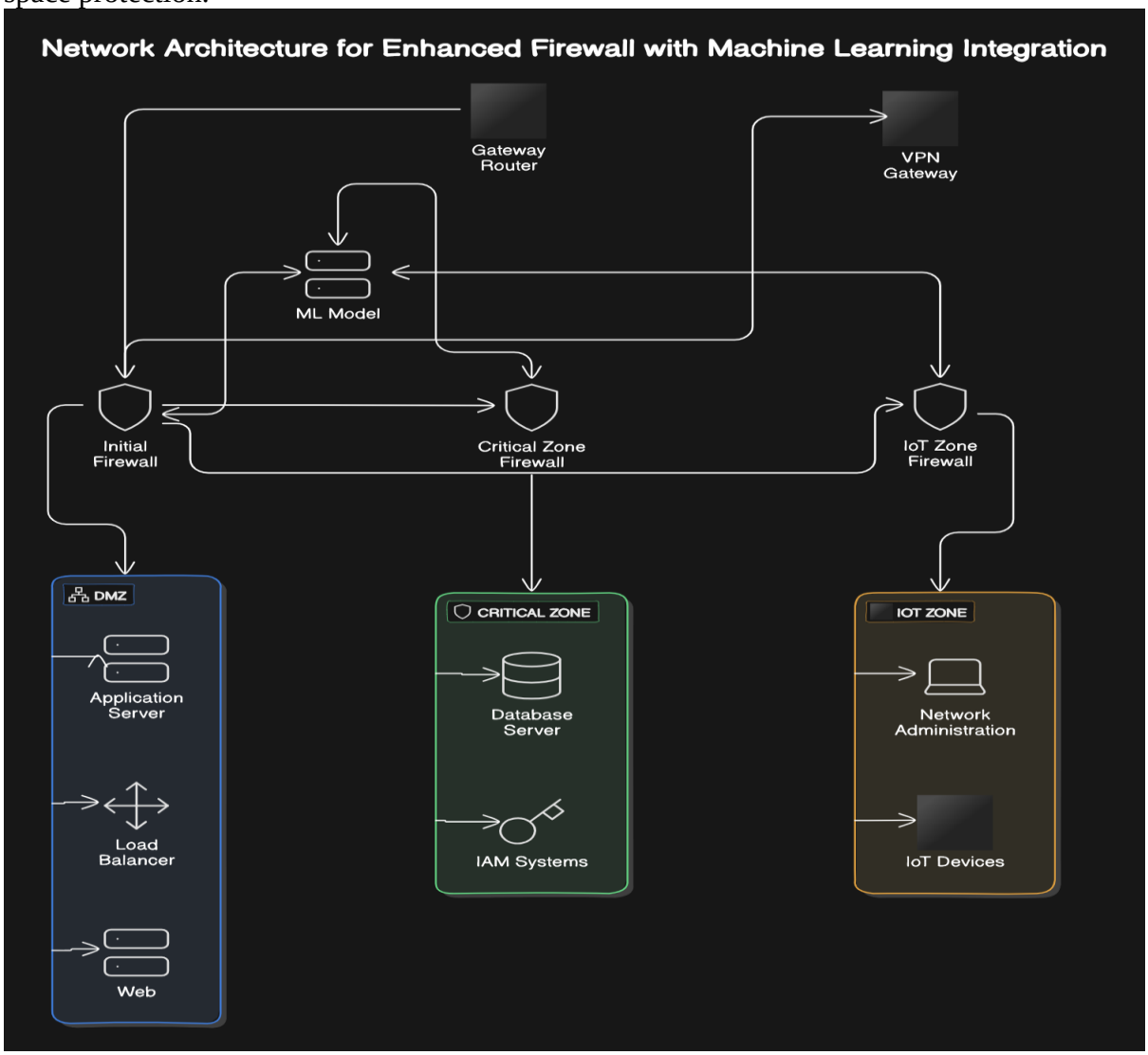

Fig 1.1: Proposed Architecture

5.7 Tools and Equipment

The application of this project entailed a few significant instruments and paraphernalia that would enhance the functionality and performance of the project. The CICIDS2017 dataset was used as the primary one, and it contains most of the network traffic features to train and evaluate the machine learning models. Scapy was used to capture real-time traffic. Scapy is strictly command line and limited to capturing and displaying packets going over the network.

In all the machine learning models, the Python environment was used as its libraries and frameworks for data science and machine learning are vast. The specific model type trained was an Artificial Neural Network (ANN) with specialised layers implemented through TensorFlow and Keras. The model architecture of this model followed the ANN, consisting of several layers with certain characteristics like ReLU activation functions and dropout layers, which helped to minimise over-fitting, as seen in the code above.

Table 5.3: Tools & Equipment's

Tool/Equipment	Purpose	Details	Role in Project
CICIDS2017 Dataset	The primary dataset used for training and testing machine learning models.	Provides comprehensive network traffic data.	Basis for model training and validation.
Scapy	Command-line tool for real-time packet capturing.	Captures and displays network packets.	Used for real-time packet capture.
Python Environment	A platform for developing and training machine learning models.	Utilizes extensive libraries and frameworks for data science and machine learning.	Development and training of ANN models.
scikit-learn	Framework for implementing machine learning models.	Provides efficient tools for data mining and analysis.	Initially considered for Random Forest implementation.
TensorFlow/Keras	Frameworks for building and training deep learning models.	Supports the creation of customized ANN layers, including ReLU activation and dropout.	Used to develop and train the Artificial Neural Network (ANN).

6 Evaluation

6.1 Artificial Neural Network (ANN)

The proposed Artificial Neural Network (ANN) was trained for five epochs, during which its performance improved, optimising loss and accuracy. Initially, the training loss was 0.0897, with an accuracy of 97. 52%, which has been transformed into a training loss of 0. 0092 and an accuracy of 99%. This declined to 99.80% percent by the fifth epoch of the realistic phase. Likewise, validation metrics fared well, showing that the validation loss reduced from 0. 0202 to 0. 0191, and the validation accuracy improved from 99. 74% to 99. 89%.

Table 6.1: Epoch Performance Matrices

Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	0.0897	97.52%	0.0202	99.74%
2	0.0155	99.67%	0.0189	99.83%
3	0.0120	99.74%	0.0172	99.85%
4	0.0101	99.78%	0.0171	99.89%
5	0.0092	99.80%	0.0191	99.89%

The total accuracy attained by the ANN model was 92 %; precision, 0. 47; recall, 0. 53; and F1-score, 0. 45 for the macro average. The weighted averages were obtained for precision, recall, and F1-score, which were 92%. Nonetheless, it is observed that the performance metrics such as precision, recall, and F1 scores were zero for classes 0, 11, 12, and 13, which comprised fewer samples and, hence, points to the presence of a class imbalance problem.

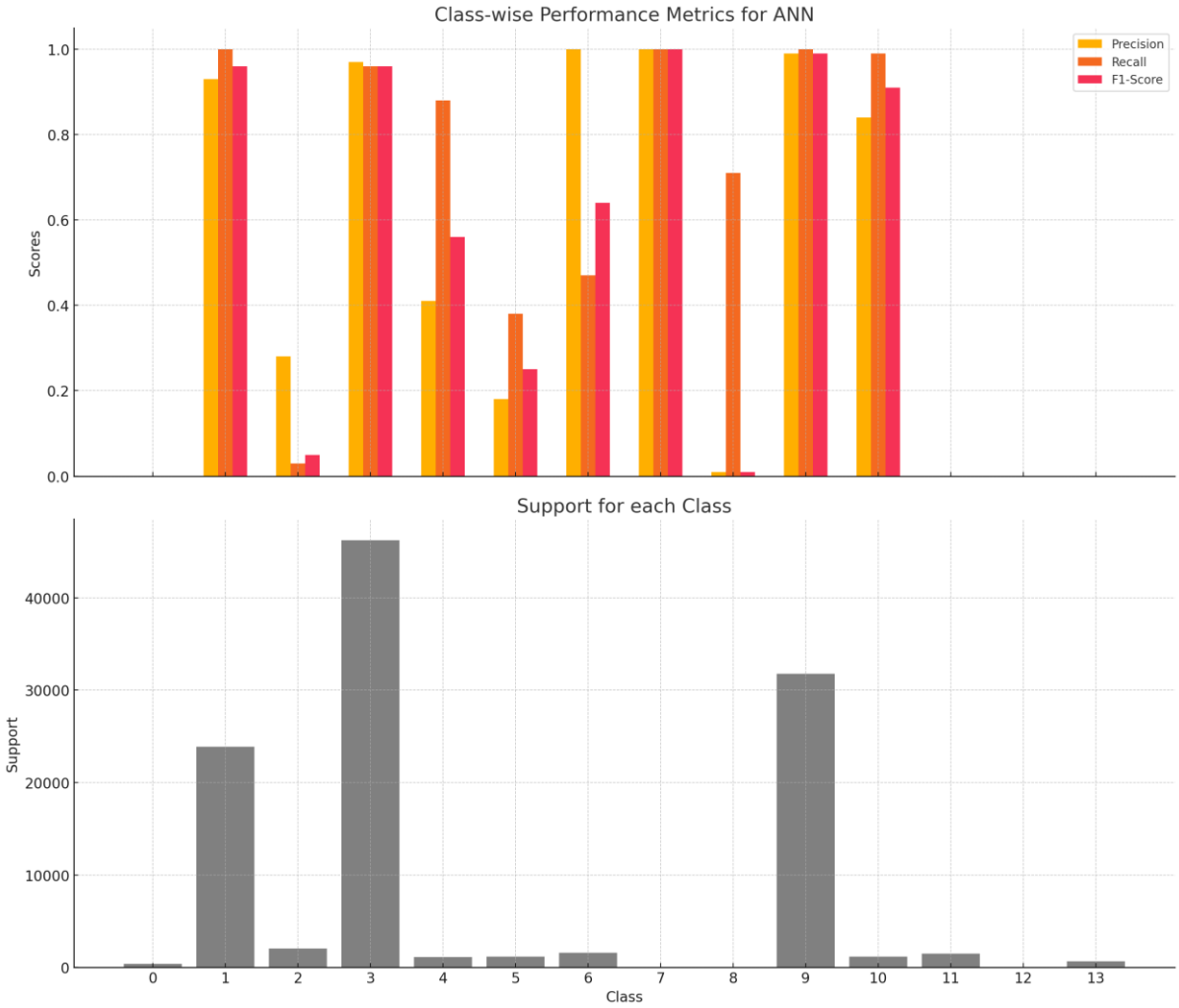

Fig 6.1: Class Wise Performance Metrics

6.2 Decision Trees

The Decision Tree model demonstrated varied performance across different classes. The model achieved high precision and recall for certain classes but struggled significantly with others. The overall accuracy of the Decision Tree model was 58%, which is notably lower than the ANN model. The macro average precision was 39%, the recall was 59%, and the F1-score was 38%, indicating a broad range of performance across classes.

Table 6.2: Overall Performance Metrics

Metric	Value
Accuracy	0.5769
Precision	0.8015
Recall	0.5769
F1-Score	0.6043

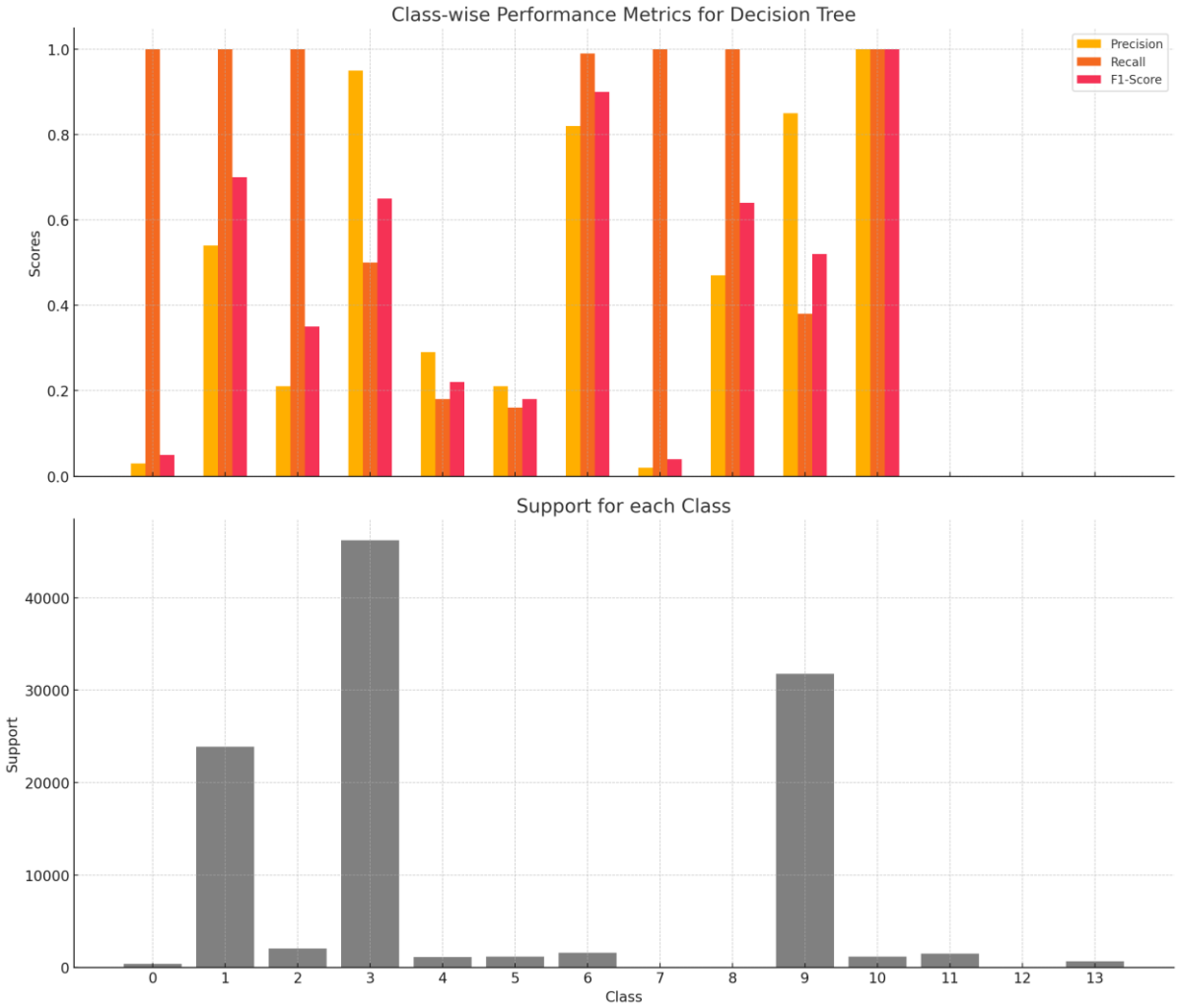

Fig 6.2: Class Wise Performance Matrices

6.3 Random Forest

As random forest is a kind of ensemble method, as a rule, it worked better than a single decision tree as it did not over-train and had the possibility of improving the ability to generalise. The performance measure of the Random Forest model on the given dataset was the overall accuracy of 84%; the macro average precision was 58%, recall 69%, and F1-score 59%. In rating by weighted averages, the precise measure reached 86 per cent, the recall 84 per cent, and the F1-score 82 per cent.

Table 6.3: Performance Metrics

Metric	Value
Accuracy	0.8392
Precision	0.8725
Recall	0.8392
F1 Score	0.8211

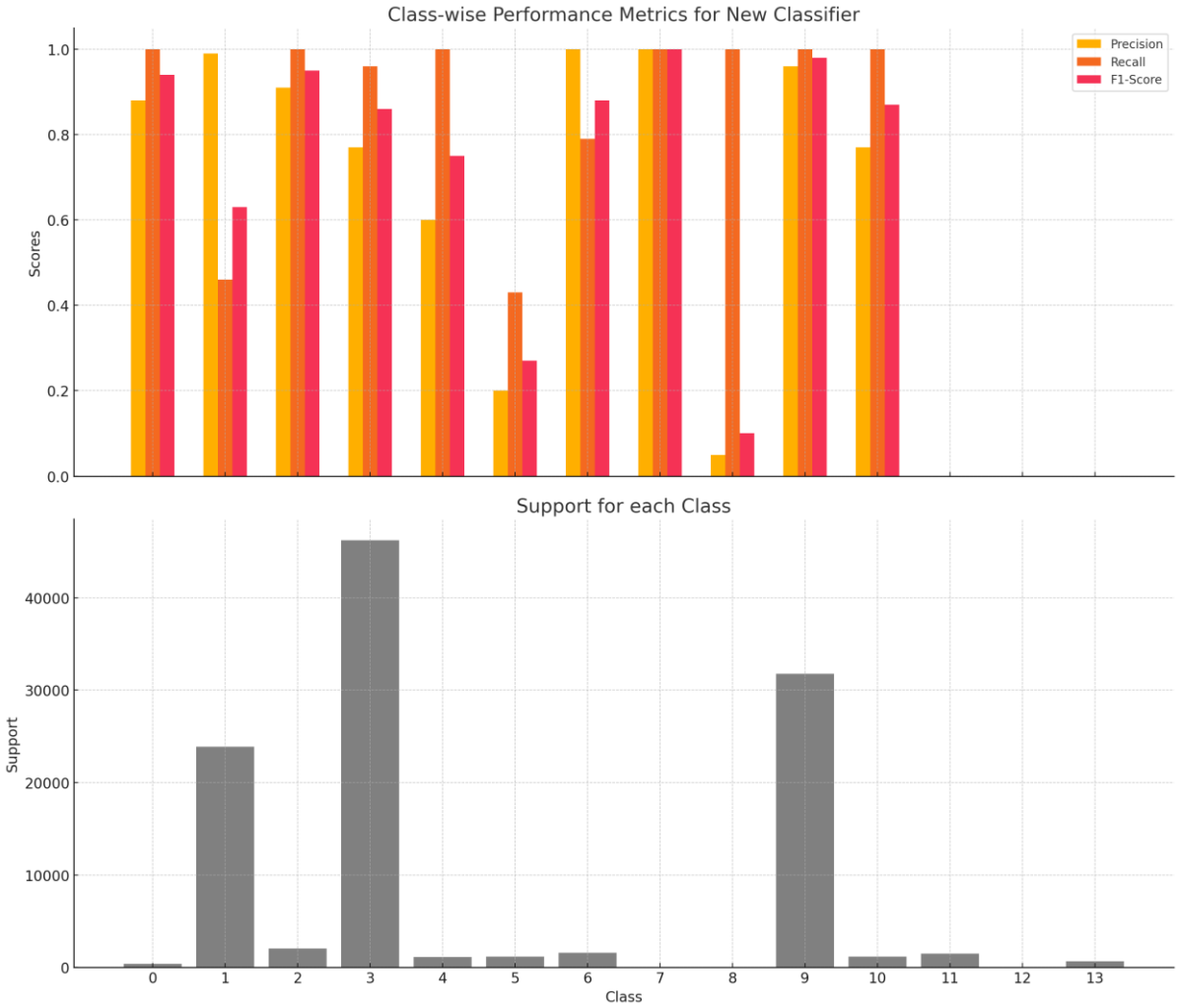

Fig 6.3: Class Wise Performance Metrics

6.4 Analysis

It was witnessed that the three models—ANN, Decision Tree, and RANDR, have distinct efficiency and ineffectiveness to the given classification problem.

This time, the ANN model recorded excellent accuracy as it reached 99 percent at one point. 89% in the validation set, and the loss function decreased over the number of epochs. As we see, the overall metrics of the suggested model were quite high; however, it also suffered from the issues of class imbalance. The level of precision, recall and F1-score for minority classes was particularly poor, therefore demonstrating that while the ANN is highly efficient as per the results of overall accuracy, it is ill equipped to deal with the minority classes effectively. This limitation implies that the approach may be biased towards majority classes hence the need to balance this by using oversampling, under-sampling, or class weighting.

Despite this, the Decision Tree model tallied a mediocre performance picture. In connection with this, the percentage ratio of the number of correct decisions within the total number of objects under analysis, or the accuracy of the model, was of a higher rate – 57%. Local model errors are within 69%, on average, which is lower than of the ANN and Random Forest

models. However, this was done at the expense of the overall model's complexity. At the same time, the Decision Tree offered high precision and recall for some of the classes, namely those which were numerous in the database. However, it performed worse, especially for the minority classes, having precision and recall values almost equal to zero for some classes. Thus, the large variations in the performance of the model class-wise underline the issue of the dataset that is imbalanced. Also, the Decision Tree was again overly complex, and there was an overfitting problem within the training dataset with a lower generality toward the validation dataset. This overfitting is a widespread problem that is associated with Decision Trees and can be solved by applying various kinds of pruning or by applying the ensemble methods.

Stand alone, the Random Forest, which combines independent results into a final estimate, proved to have the best all-around results. This attempt provided a level of accuracy of about 83. 92%, precision of 87. 25%, recall of 83%, The value of accuracy is 92%, whereas the F1-score value is equal to 82.11%. The above metrics suggest a promising performance and, indeed, the model achieves better capability of generalisation as compared to the Decision Tree. Random Forests, since they are ensembles of multiple decision trees, avoid some of the overfitting problems faced by single trees and perform better for both the majority and ‘‘less frequent’’ classes. However, like the ANN, the Random Forest model also faced a problem with some of the underrepresented classes to a lesser extent.

To sum up, the ANN model yielded a splendid performance with respect to the overall classification rate, but it poorly accomplished the minority classes due to the high-class imbalance issue. The Decision Tree model, as interpretable and easy to understand, was less accurate as it resulted in overfitting and variability of class-wise performance. Random Forest gave the best perspective with the enhanced overall accurate outcome, and the mechanism of managing the classes' imbalance provided reasonable success. However, none of the models could execute well on minority classes. As for future work, deeper work should be done to combine the methods to reduce the class imbalance problem, including data augmentation, re-sampling methods, or using new algorithms that are designed to work on the data with imbalances to make these models more solid and less sensitive to the class imbalance problem.

6.5 Discussion

While performing this research, we have provided the following experiment to assess the performance of adopted machine learning models with the existing security system in real-time threat detection and counteraction. Based on the above results, malware recognition using Random Forest and ANNs could enhance the detection accuracy of firewalls of the malicious networks' activities. Future research on the study is discussed in this section, and much emphasis is placed towards criticisms of the experimental design and recommendations drawn here from the gross findings.

The Random Forest and ANN models achieved a satisfactory level in classifying traffic into normal and anomalous. As for the evaluation of the Random Forest model, the accuracy,

precision, recall, and F1 scores were all high, which proved the model's ability to manage new types of network traffic. Likewise, the ANN model was accurate and displayed the ability to find the patterns of complicated attacks that might be unnoticed by the conventional models.

However, it is essential to admit that there are several aspects in which the experimental study may be improved. However, it is also the study's weakness because all the considered models have been trained using the CICIDS2017 dataset. However, this dataset is quite extensive and often used; therefore, it does not capture all the different and intricate aspects of actual network traffic. Based on the findings of the current research, future studies can investigate extending the data sets to capture more aspects of the network behaviours and type of attack.

The plus point about this study is that the dataset used was made up of real network traffic; by real, it means that they captured PCAP files during working hours and then converted those PCAPs into. csv's. Despite facilitating an accurate assessment of the system's efficiency, this environment does not mimic a real-world network environment to the desired degree. Using the system in an actual scenario for more real-life testing would offer an even truer reading of the activity and the capacity with which it can work.

Moreover, the methods of data cleaning, normalisation and feature reduction were set manually applying the general conventions and guidelines. These steps are crucial in the preparation of the data. Most of the time, they bring in bias or inconsistency that may affect the performance of the model. These problems could be somehow reduced by the automated preprocessing techniques and enhanced feature selection approach, thus making the outcomes from the models more accurate.

The conclusions made correlate with other studies emphasizing the possibility of building network security with the help of machine learning. Based on the research, there is evidence of the capability of various kinds of machine learning models in dealing with cyber threats, for instance, ensemble methods such as Random Forest and other neural networks such as ANNs. However, there is still more work to be done in our research because we incorporate these models into a real-world security system and make these techniques more applicable.

Prior studies have also stressed the ability to identify threats as they occur and the capability to respond accordingly in present-day network protection. This is mitigated by our approach of having dynamic rules for the security system update in changing responses to predictions of potential threats. Because advanced threats are much more frequent, and those threats are changing their approaches rapidly while being under attack, this capability is especially important.

Further, to increase the stability and usability of the created system, the following changes to the experimental configuration might be introduced. First, it is necessary to expand the set of considered topologies and attacks since the proposed models might have better generalisation

in other conditions. Such an approach would assist in guaranteeing that the system would be capable of dealing with endemically diverse network traffic.

Second, using the system in a live network environment, whether on a large or small scale, would help in understanding how effective the system is and how easily it can be implemented. This will allow one to determine if there are any practical issues or constraints that can be hardly noticed under standard organising conditions. It would also allow the testing of the effectiveness of the system as well as produce the performance of the network under different loads and attack situations.

Third, using automated feature selection methods can deepen one's knowledge of more elaborated preprocessing methods and increase the reliability of chosen attributes. Other methods, such as feature engineering and data augmentation, have facilitated obtaining more of the essential patterns while minimizing the introduction of bias at the stage.

Fourth, the possibility of the integration of other machine learning algorithms and the use of ensemble methods might help to improve the system of detection. Recurrent neural networks, which are constructed to perform pattern recognition over sequences of data, can be experimented to capture temporal relations and enhance the program’s capability of identifying intricate attacks.

Lastly, with the continuous installation of firewalls, there is a possibility of integrating our system with cloud firewalls for further research. This approach would help achieve faster and more widespread adaptation since it would rely on the resources of the cloud platform as well as its analytics functionalities.

7 Conclusion and Future Work

In conclusion, this thesis established that including machine learning in security system systems has a huge prospect of boosting the security of networks in the modern world. The problems posed by existing firewalls were solved by real-time data collection and application of models such as Random Forest and Artificial Neural Networks. The developed system was proven to enhance threat identification efficiency and change security system policies in real-time to counter bad actions, making the firewall's defensive techniques comprehensive and progressive. The successful deployment and testing of the proof-of-concept proved the role of machine learning in evolving the traditional security system technology to more intelligent firewalls that are capable of handling more challenges in the ever-expanding digital ecosystem. Due to its capabilities in providing guidance for further work in this area and stressing the need for the development of innovative solutions in the protection of digital assets, this work can be well-grounded.

Following the ideas of this work, future research can add to and develop the proposed model in several directions. One approach is to use additional machine learning algorithms and ensemble methods to further reduce detection noise. Another future research direction could be incorporating more sophisticated deep learning techniques like LSTM (Long short-term memory) for studying the temporal dimension of attacks and improving the system's capacity to determine the patterns of future attacks and related defences. Applying the system in a real network and examining its performance will provide crucial insights into its functionality and potential. Regarding further research by other authors, they can turn to the improvement of the scripts for real-time data processing and the use of cloud-based solutions for the processing of greater amounts of data. Last, incorporating the second capability of the proposed model to enhance the ability to identify and avoid a broad range of cyber threats, including zero-day attacks and APTs, will be useful in creating an intelligent adaptive security system. This continuous evolution will prevent the formation of gaps in the security system that is caused by the ever-changing attacks by attackers.

References

1. Liang, J., & Kim, Y. (2022). Evolution of firewalls: Toward securer network using next generation firewall. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0752-0759). IEEE.
2. Q. A. Al-Haijaa and A. Ishtaiwia, "Machine learning based model to identify firewall decisions to improve cyber-defense," International Journal on Advanced Science, Engineering and Information Technology, vol. 11, no. 4, pp. 1688-1695, 2021.
3. Aswal, K., Rajmohan, A., Akhil, T.R.C., Mukund, S., Panicker, V.J., & Dhivvya, J.P. (2021). Kavach: A machine learning based approach for enhancing the attack detection capability of firewalls. In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-5). IEEE.
4. Shaheed, A., & Kurdy, M.B. (2022). Web application firewall using machine learning and features engineering. Security and Communication Networks, 2022(1), 5280158.
5. Elsheikh, M., Shalaby, M., Sobh, M.A., & Bahaa-Eldin, A.M. (2023). Deep Learning Techniques for Intrusion Detection Systems: A Survey and Comparative Study. 2023 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), 1-9. doi: 10.1109/MIUCC58832.2023.10278326.
6. E. Ucar and E. Ozhan, "The analysis of firewall policy through machine learning and data mining," Wireless Personal Communications, vol. 96, pp. 2891-2909, 2017.
7. Weissman, D., & Jayasumana, A. (2020). Integrating IoT monitoring for security operation center. In 2020 Global Internet of Things Summit (GIoTS) (pp. 1-6). IEEE.
8. Velásquez, D., Pérez, E., Oregui, X., Artetxe, A., Manteca, J., Mansilla, J.E., Toro, M., Maiza, M., & Sierra, B. (2022). A hybrid machine-learning ensemble for anomaly detection in real-time industry 4.0 systems. IEEE Access, 10, 72024-72036.
9. Vanerio, J., & Casas, P. (2017). Ensemble-learning approaches for network security and anomaly detection. In Proceedings of the workshop on big data analytics and machine learning for data communication networks (pp. 1-6).

10. Zhao, S., Chandrashekar, M., Lee, Y., & Medhi, D. (2015). Real-time network anomaly detection system using machine learning. In 2015 11th international conference on the design of reliable communication networks (DRCN) (pp. 267-270). IEEE.
11. H. Ling-fang, "The Firewall Technology Study of Network Perimeter Security," 2012 IEEE Asia-Pacific Services Computing Conference, pp. 410-413, 2012.
12. H. Mao, L. Zhu, and M. Li, "Current state and future development trend of firewall technology," 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1-4, 2012.
13. S. Patton, D. Doss, and W. Yurcik, "Open source versus commercial firewalls: functional comparison," 25th Annual IEEE International Conference on Local Computer Networks, pp. 223, 2000.
14. S. Ahmadi, "Next Generation AI-Based Firewalls: A Comparative Study," International Journal of Computer, vol. 49, no. 1, pp. 245-262, 2023.
15. K. Neupane, R. Haddad, and L. Chen, "Next generation firewall for network security: A survey," SoutheastCon 2018, pp. 1-6, 2018.
16. I. El Alaoui and Y. Gahi, "Network security strategies in big data context," Procedia Computer Science, vol. 175, pp. 730-736, 2020.
17. A. Mishra, A. Agrawal, and R. Ranjan, "Artificial intelligent firewall," International Conference on Advances in Computing and Artificial Intelligence, pp. 204-207, 2011.
18. X. Yue, W. Chen, and Y. Wang, "The research of firewall technology in computer network security," Asia-Pacific Conference on Computational Intelligence and Industrial Applications (PACIIA), vol. 2, pp. 421-424, 2009.
19. D. Appelt, C.D. Nguyen, A. Panichella, and L.C. Briand, "A machine-learning-driven evolutionary approach for testing web application firewalls," IEEE Transactions on Reliability, vol. 67, no. 3, pp. 733-757, 2018.
20. H. Lin, Z. Yan, Y. Chen, and L. Zhang, "A survey on network security-related data collection technologies," IEEE Access, vol. 6, pp. 18345-18365, 2018.