



Ministry of Higher Education
Kabul Polytechnic University
Faculty of Computer Science
Department of Network Engineering

Title:

Network Traffic Classification using Machine learning

Supervisor:

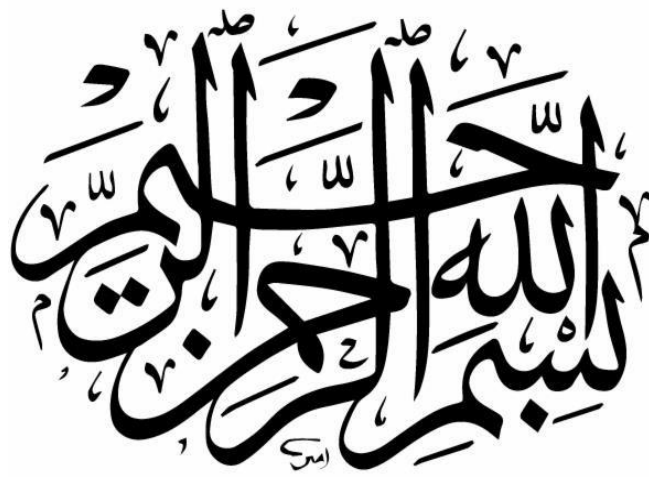
Asst. Prof. Sayed Shafiullah "Sadat"

By:

Mohammad Wasil "Jalali"

Years:

August 2024



تقریظ استاد رهنما

اینجانب سید شفیع الله "سادات" استاد رهنمای پروژه دیپلوم محترم محمد وصیل فرزند غلام عباس محصل صنف چهارم دیپارتمنت انجنیری شبکه میباشم، رساله موصوف را که تحت عنوان تقسیم بندی ترافیک شبکه توسط روش ماشین لیرنینگ در 44 صفحه تحریر گردیده بود مطالعه نمودم. محترم محمد وصیل "جلالی" فرزند غلام عباس مشوره های لازم اینجانب و سایر استادان را در جریان کار پروژه خویش بطور متداوم در نظر گرفته و از اخلاق نیکو و روابط خوب اجتماعی برخوردار میباشد. رساله مذکور مطابق معیارهای علمی و اکادمیک آماده گردیده و این پروژه به نسبت تحقیقی بودن آن از اهمیت ویژه برخوردار است، اینجانب پروژه موصوف را برای پایان نامه تحصیلی محترم هوشام کافی دانسته و پروژه ایشان را ارزیابی و به دفاع پروژه دیپلوم سفارش نموده و برای ایشان لقب انجنیر لسانس را در رشته دیپارتمنت انجنیری شبکه به کمسیون دفاع دیپلوم پیشنهاد مینمایم.

تاریخ : 1403

با احترام

Acknowledgement

Before all, I would like to be grateful from merciful Allah who has given me the power to finish my final year project. And also, I want to show massive appreciation to my family members especially my mother, father and old brother who have never stopped encouraging me in harsh circumstances next special thanks to my professors who are stayed beside me and helped me, I was not able to gain knowledge without them.

Additionally, the success of every project needs commitment and continual struggle as well the right and appropriate guidance of teachers, I would like to show my pure thanks and appreciation to most respected professor dear “Sayed Shafiullah Sadat” who has been declared the idea of project for me and guided me a lot in every step of project and answered my question related to the project on time and dedicated his time to give me interesting suggestions and pointed out tiny details for me, also special thanks to every other professors who learned us more and more and stayed beside students despite lots of problems. At the end, I would like to thank from KPU family especially Network Engineering department of Computer Science faculty for providing facilities and educational environment for students.

Abstract

The "Network Traffic Classification using Machine Learning" project aims to enhance network security and management by classifying network packets into normal or abnormal categories. If a packet is identified as normal, the project further classifies it by application type; if abnormal, it identifies the type of network attack. The project encompasses three distinct classification models: binary classification, attack type classification, and application type classification. Utilizing Convolutional Neural Networks (CNN) for binary and attack type classification, and Long Short-Term Memory (LSTM) networks for application type classification, the models leverage four datasets: CICIDS 2017, CICIDS 2018, CIC-ToN-IoT, and the IP Network Traffic Flows dataset labeled with 75 applications. The binary classification model was trained on the CICIDS 2017 dataset augmented with a subset of CIC-ToN-IoT and further refined using transfer learning with CICIDS 2018. The application type model was developed using the IP Network Traffic Flows dataset, and the attack type model was trained on the CICIDS 2017 dataset. The models achieved high accuracy rates, with the binary model reaching 99.71%, the attack type model 96.83%, and the application type model 99.02%. This project provides a comprehensive solution for real-time network traffic analysis and classification, aiding network administrators in efficient network management and robust security measures.

Table of Contents

title	page
Chapter 1	1
1.1 Introduction.....	2
1.2 Motivation.....	2
1.3 Aim	3
Chapter 2	4
2.1 Introduction to project technologies	5
2.1.1 Network.....	5
2.1.1.1 What is Network	5
2.1.1.2 What is Packet.....	5
2.1.1.3 What is Payload	5
2.1.2 Machine Learning	5
2.1.2.1 What is ML	5
2.1.2.2 Types of ML.....	5
2.1.2.3 Supervised Learning	6
2.1.2.4 Deep Learning.....	6
2.1.2.5 Convolutional Neural Network.....	6
2.2 Project Scope	6
2.2.1 Objective.....	6
2.1.2 Dataset	7
2.1.3 Methodology.....	7
2.1.4 Expected Outcomes	8
2.3 Technologies Used in Project	8
2.4 Algorithms Formula.....	8
2.4.1 CNN	8
2.4.2 LSTM	10
2.4.3 Precision	11
2.4.4 F1 score.....	12
2.4.5 Recall.....	12
Chapter 3	13
3.1 Related work	14
3.2 Traditional Methods.....	16
3.3 Proposed Method	17
Chapter 4	18

4.1 Proposed Method Architecture	20
4.1.1 Dataset.....	20
4.1.1.1 CICIDS2017.....	20
4.1.1.2 CICIDS2018.....	21
4.1.1.3 IP Network Traffic Flows 75 Apps	21
4.1.1.4 CIC-ToN-IoT	21
4.1.2 Preprocessing	21
4.1.3 Feature Engineering	23
4.1.3.1 Attack type Feature Engineering.....	24
4.1.3.2 Application Type Feature Engineering	24
4.1.4 Data Split	25
4.1.5 Training.....	26
4.1.5.1 Binary Classification.....	26
4.1.5.1.1 Transfer Learning.....	27
4.1.5.2 Attack Types Classification	32
4.1.5.3 Application Type.....	33
4.1.6 Evaluation	36
4.1.6.1 Binary.....	36
4.1.6.2 Attack Types	38
4.1.6.3 Application Type.....	39
4.1.7 Proposed Interface.....	42
Chapter 5.....	44
5.1 Conclusion	45
5.2 Future Work.....	45
5.3 References.....	46

Table of Figures

title	page
Figure 1: Convolutional Layer Formula	9
Figure 2: Pooling Layer Formula.....	9
Figure 3: Fully Connected Layer Formula.....	10
Figure 4: Forget, Input and Cell state	10
Figure 5: Output and Hidden	11
Figure 6: Formula Info.....	11
Figure 7: Proposed Method Architecture.....	20
Figure 8: CICIDS 2017 Types Data.....	23
Figure 9: Label Attack Type after Oversampling	24
Figure 10: Label Values After Dropping Values less than 1000 records	25
Figure 11: Binary Model Architecture.....	27
Figure 12: Binary Model Accuracy and Epochs	27
Figure 13: First Transfer Architecture Model.....	29
Figure 14: Second Transfer Architecture Model	29
Figure 15: Transfer-learning Accuracy	30
Figure 16: Classification Report of Transfer	30
Figure 17: Confusion Matrix of Transfer.....	30
Figure 18: Attack Classification Architecture	32
Figure 19: Accuracy and Evaluation Metrics	32
Figure 20: Application Classification LSTM Architecture	35
Figure 21: Confusion Matrix of Binary	36
Figure 22: Classification Report of Binary	36
Figure 23: ROC Curve of Binary	37
Figure 24: Confusion Matrix of Attack	37
Figure 25: Classification Report of Attack	38
Figure 26: Application Classification Evaluation Metrics.....	39
Figure 27: Classification Report.....	40
Figure 28: Proposed Method Interface	41

Table of Tables

title	page
Table 1: Related Work Details	16
Table 2: Label Value Types in CICIDS 2017, 2018 Before and After Augmentation.....	23
Table 3: Proposed Method Result.....	40

Chapter 1

(Introduction)

1.1 Introduction

The number of connections and devices worldwide is growing faster than the population and the Internet users, increasing network traffic exponentially[1]. Furthermore, the adoption of new devices with significant capabilities and intelligence (e.g., smartphones, smart televisions, video game consoles), combined with the proliferation of Machine-to-Machine (M2M) communications and the consequent development of new services and applications, have significantly changed traffic flow patterns and network performance[2].

Intrusion detection plays a vital role in the network defense process by aiming security administrators in forewarning them about malicious behaviors such as intrusions, attacks, and malware. Having IDS is a mandatory line of defense for protecting critical networks against these ever-increasing issues of intrusive activities[3].

Managing different network infrastructures to satisfy the requirements of new devices, applications, and services has become a complex task. Therefore, there is great interest in building autonomous networks, characterized by their self-configuring, self-repairing, self-optimizing, and self-protecting abilities, using cognitive techniques and Machine Learning (ML)[4].

To address these challenges by developing a comprehensive network traffic classification system. The primary objectives are to classify network packets as either normal or abnormal, further classify normal packets into layer seven applications, and identify the specific types of network attacks for abnormal packets. Achieving these objectives will provide network administrators with a powerful tool to enhance their decision-making process and effectively manage network security.

1.2 Motivation

The motivation behind the "Network Traffic Classification using Machine Learning" project stems from the exponential growth in the number of network-connected devices and the consequent surge in network traffic. As modern networks become increasingly complex due to the proliferation of intelligent devices and Machine-to-Machine (M2M) communications, managing network performance and ensuring security has become a formidable challenge. Traditional methods of traffic classification and intrusion detection are no longer sufficient to handle the dynamic and sophisticated nature of contemporary network traffic.

Network administrators are faced with the task of maintaining optimal network performance while protecting against an ever-evolving landscape of cyber threats. Accurate and efficient network traffic classification is essential for enabling Quality of Service (QoS), content filtering, lawful interception,

and malicious behavior identification. The ability to distinguish between normal and abnormal traffic, and further classify the types of applications and attacks, is crucial for proactive network management and defense.

This project leverages advanced machine learning techniques, specifically Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, to provide a robust solution for real-time network traffic analysis. By utilizing multiple datasets that encompass a wide range of normal and malicious traffic patterns, the project aims to deliver high-accuracy models capable of enhancing network security and performance. The ultimate goal is to equip network administrators with the tools needed to make informed decisions, optimize resource allocation, and protect their networks from cyber threats.

1.3 Aim

The aim of the "Network Traffic Classification using Machine Learning" project is to develop an advanced, machine learning-based system capable of accurately classifying network traffic into normal and abnormal categories. For normal traffic, the system further identifies the specific types of applications in use, while for abnormal traffic, it classifies the types of network attacks. This system is designed to enhance network security and performance by providing network administrators with precise, real-time insights into network activity, enabling proactive management and defense against cyber threats.

Chapter 2

(Introduction To Project Technologies)

2.1 Introduction To Project Technologies

This section declares technologies used in the project in addition, there is an introduction to network and some components related. The section is divided in two parts. First part, about Network. Second part, about Machine learning.

2.1.1 Network

2.1.1.1 What is Network:

A network is a set of devices (often referred to as *nodes*) connected by communication links. A node can be a computer, printer, or any other device capable of sending and/or receiving data generated by other nodes on the network[5].

2.1.1.2 What is Packet:

A packet is defined as a formatted unit of data carried by a packet-switched network. In general, a packet consists of control information and user data, also known as the payload. Control information provides data for delivering the payload, such as source and destination addresses, error detection codes, and sequencing information[5].

2.1.1.3 What is Payload

Payload refers to the actual data that is being transmitted in a network packet. This excludes any headers or metadata added for routing and managing the packet. Essentially, the payload is the core content that the sender wants to deliver to the receiver, which could be anything from a simple text message to a complex multimedia file [5].

2.1.2 Machine Learning

2.1.2.1 What is Machine learning (ML):

Machine learning is the science (and art) of programming computers so they can learn from data[6].

2.1.2.2 Types of ML:

There are so many different types of machine learning systems that it is useful to classify them in broad categories, based on the following criteria:

- How they are supervised during training (supervised, unsupervised, semi-supervised, self-supervised, and others)
- Whether or not they can learn incrementally on the fly (online versus batch learning)

- Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do (instance-based versus model based learning)[6].

2.1.2.3 Supervised Learning:

In supervised learning, the training set you feed to the algorithm includes the desired solutions, called labels[6].

2.1.2.4 Deep Learning:

Deep learning refers to training neural networks with more than two non-output layers[7].

2.1.2.5 Convolutional Neural Network (CNN):

CNN is a special kind of FFNN that significantly reduces the number of parameters in a deep neural network with many units without losing too much in the quality of the model[7].

2.2 Project Scope

This project focuses on the development and implementation of a network traffic classification system using deep learning techniques. The system aims to enhance network security by accurately classifying network traffic into normal and abnormal categories, identifying specific application types for normal traffic, and detecting various attack types for abnormal traffic.

2.2.1 Objectives:

1. **Binary Classification:** Develop a model to classify network traffic as either normal or abnormal.
2. **Application Classification:** For normal traffic, further classify the data into specific application types.
3. **Attack Classification:** For abnormal traffic, identify and classify the specific types of network attacks.

2.2.2 Datasets

- **CICIDS 2017:** Used for initial binary classification and attack classification. This dataset includes a comprehensive set of benign and malicious traffic, simulating real-world network scenarios.
- **CICIDS 2018:** Employed for transfer learning to improve model accuracy on unseen data.
- **CIC-ToN-IoT:** Used to augment the dataset and evaluate model performance on different network environments.
- **IP Network Traffic Flows Labeled with 75 Apps:** Used for application type classification.

2.2.3 Methodology

1. Data Preprocessing:

- **Combining and Cleaning Data:** Integrate multiple datasets, handle missing and infinite values, and ensure data consistency.
- **Data Augmentation and Oversampling:** Address class imbalance by augmenting underrepresented classes using techniques such as oversampling.
- **Feature Engineering:** Transform features into appropriate formats, normalize data, and handle outliers to improve model performance.

2. Model Development:

- **Binary Classification Model:** Utilize a Convolutional Neural Network (CNN) to distinguish between normal and abnormal traffic.
- **Application Classification Model:** Implement an LSTM (Long Short-Term Memory) model to classify normal traffic into various application types.
- **Attack Classification Model:** Develop a CNN model to identify specific attack types in abnormal traffic.

3. Transfer Learning:

- Enhance model performance by applying transfer learning techniques using CICIDS 2018 datasets to adapt to new data patterns.

4. Evaluation and Validation:

- **Performance Metrics:** Assess model accuracy, precision, recall, and F1-score.
- **Cross-Validation:** Use train-validation-test splits to ensure robust model evaluation.
- **Real-world Testing:** Evaluate models on CIC-ToN-IoT and other unseen datasets to verify generalization capability.

2.2.4 Expected Outcomes

- Achieve high accuracy in classifying network traffic, with specific benchmarks:
 - Binary Classification: > 99% accuracy
 - Application Classification: 99% accuracy
 - Attack Classification: > 96% accuracy
- Develop a reliable, scalable network traffic classification system that can be integrated into network security infrastructures to detect and mitigate threats in real-time.

2.3 Technologies Used in Project

- TensorFlow

- Scikit-learn
- Python
- NumPy
- Pandas
- Matplotlib
- Seaborn
- Imblearn
- Elevenlabs
- HTML
- CSS
- Bootstrap
- JavaScript
- Google Colab

2.4 Algorithms Formula

2.4.1 CNN

A Convolutional Neural Network (CNN) primarily consists of convolutional layers, pooling layers, and fully connected layers (Figure 1,2,3).

1. Convolutional Layer

In a convolutional layer, the input image is convolved with a set of filters (kernels) to produce feature maps.

Formula:

$$\text{output}_{ij}^k = \sigma \left(\sum_{m=1}^M \sum_{n=1}^N x_{(i+m-1)(j+n-1)} \cdot w_{mn}^k + b^k \right)$$

Where:

- output_{ij}^k is the output at position (i, j) in the k -th feature map.
- $x_{(i+m-1)(j+n-1)}$ is the input at position $(i + m - 1, j + n - 1)$.
- w_{mn}^k is the (m, n) element of the k -th filter.
- b^k is the bias term for the k -th filter.
- σ is the activation function (e.g., ReLU).

Figure 1: Convolutional layer formula

2. Pooling Layer

A pooling layer performs down-sampling by applying a pooling operation (e.g., max pooling) over each feature map.

Max Pooling Formula:

$$\text{output}_{ij} = \max \left(\{x_{(i+p)(j+q)} \mid 0 \leq p < P, 0 \leq q < Q\} \right)$$

Where:

- output_{ij} is the output at position (i, j) .
- $x_{(i+p)(j+q)}$ are the input values within the pooling window.
- P and Q are the height and width of the pooling window.

Figure 2: Pooling layer Formula

3. Fully Connected Layer

A fully connected layer computes the output as a weighted sum of the inputs, similar to a traditional neural network.

Formula:

$$\text{output}_j = \sigma \left(\sum_i x_i \cdot w_{ij} + b_j \right)$$

Where:

- output_j is the output at the j -th node.
- x_i is the input from the i -th node of the previous layer.
- w_{ij} is the weight between the i -th input node and the j -th output node.
- b_j is the bias term for the j -th output node.
- σ is the activation function (e.g., ReLU, sigmoid).

Figure 3: Fully Connected Layer Formula

2.4.2 LSTM

LSTM (Long Short-Term Memory) algorithm is made of different gates and their description are in figures bellow (Figure 4,5,6).

1. **Forget Gate:**

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

The forget gate determines which information from the cell state should be discarded. Here, σ is the sigmoid activation function, W_f is the weight matrix, b_f is the bias, h_{t-1} is the previous hidden state, and x_t is the current input.

2. **Input Gate:**

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

The input gate determines which values will be updated. W_i and b_i are the weight matrix and bias for the input gate, respectively.

3. **Cell State Update:**

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Figure 4: Forget, Input and Cell state

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

Here, \tilde{C}_t is the candidate cell state, \tanh is the hyperbolic tangent function, W_C and b_C are the weight matrix and bias for the cell state update, and C_{t-1} is the previous cell state.

4. **Output Gate:**

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

The output gate determines which values will be output. W_o and b_o are the weight matrix and bias for the output gate, respectively.

5. **Hidden State:**

$$h_t = o_t \cdot \tanh(C_t)$$

The hidden state h_t is the final output of the LSTM cell, combining the output gate and the updated cell state.



Figure 5: Output and Hidden

The hidden state h_t is the final output of the LSTM cell, combining the output gate and the updated cell state.

In these formulas:

- σ is the sigmoid activation function, which outputs values between 0 and 1.
- \tanh is the hyperbolic tangent function, which outputs values between -1 and 1.
- \cdot denotes the dot product of the weight matrices with the concatenated vectors of the hidden state and input.

These operations allow LSTMs to maintain and update a cell state over time, effectively capturing long-term dependencies in sequential data.

Figure 6: Formulas info

2.4.3 Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is calculated as:

$$\text{Precision} = \frac{TP}{TP+FP} \dots\dots\dots 1$$

where TP is the number of true positives and FP is the number of false positives.

2.4.4 Recall

Recall, also known as Sensitivity or True Positive Rate, is the ratio of correctly predicted positive observations to all the observations in the actual class. It is calculated as:

$$\text{Recall} = \frac{TP}{TP+FN} \dots\dots\dots 2$$

TP is the number of True positive and FN is the number of false negative.

2.4.5 F1-Score

The F1 Score is the harmonic mean of Precision and Recall. It provides a single metric that balances both concerns and is especially useful for imbalanced datasets. It is calculated as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \dots\dots\dots 3$$

Chapter 3

(Related works)

3.1 Related Work

This section briefly overviews previous works related to the research area of traffic classification with ML techniques, (summarized in table 1).

The training and testing times needed for classification are significantly decreased using the transfer learning approach. The proposed model successfully identifies 20 different attacks with an accuracy of 93.75% on the NF-UQ NIDS dataset. Additionally, we have verified our proposed model in real-time on an edge device with limited resources[8].

Three ML models are built using the Spark ML library; decision tree, random forest, and logistic regression. These models are evaluated in four terms of evaluation metrics; accuracy, precision, recall, and f1-score. Results show that the decision tree model has the best accuracy with 0.98. Spark streaming is incorporated to stream the data to the elected ML model to replicate the online network traffic flow of data [9].

We consider two different scenarios, a regular data delivery over the network (Scenario A) and a malicious network that sporadically attacks the receiver node by flooding too many requests (Scenario B).

Network traffic data collected from the system that we implemented. We collected 53,171 traffic data from Scenario A and 2,036 traffic data from Scenario B in ONOS controller. In order to train the machine learning model, 70% of data samples are used for the train set and the rest of the data for the test set. In scenario A, the classification accuracies of RF and LDA are 95% and 98%, respectively, while DNN shows 69% accuracy. Although the DNN performance is lower than other algorithms, it is improved from 47% accuracy where the DNN structure was not optimized for network traffic data. In Scenario B, on the other hand, it is observed that the performance of abnormal behavior detection with the same algorithms is significantly degraded, where the accuracies of RF, LDA, and DNN and RF are 42%, 76%, and 74%, respectively. Specifically, RF shows lower than 50% accuracy since the traffic data collected from an abnormal scenario incurs a large variation in the structure of the optimal decision tree in RF. In case of DNN, we also compare the DNN performance with blindly used DNN and found that the accuracy of the DNN in this paper is 74% which is improved compared to the performance in [20]. The reason why the performance degradation of the DNN algorithm is because the network data for analysis includes the unintended data generated by various network protocols such as ARQ messages. Hence, these experiments show that deploying machine learning algorithms in traffic data classification, without understanding the detail of network protocols and target scenarios, does not guarantee good performance in real physical and virtual networks[10].

As K-means clustering requires the number of clusters, the Davies-Bouldin Index (DBI), which is an evaluation metric for clustering algorithms, is used to determine the optimum number of clusters by measuring inter and intra-cluster distances. Three clusters, which also represent three network slices, are used as K=3 gets the lowest Davies-Bouldin score about 0.28 which will achieve the highest clustering accuracy. To avoid this problem and overfitting too, only 1200 flows for each cluster were chosen to solve class imbalance problem and were enough to train the model.

This is due to the usage of the suitable algorithm for scaling the data and solve the problem of imbalance class distribution. ANN achieved the highest accuracy 98.2%. In the proposed model, tree-based algorithms such as decision tree and random forest weren't used because they aren't affected by scaling. After the 8th epoch, the ANN model converged, indicating that the model was fitted well by the dataset and the fine-tuned parameters. The precision, recall, f1_score and kappa score for each model. ANN and SVM with linear function gave outstanding results and gave better results than other models in all metrics. The confusion matrices for all the models are represented in figure 10. From this figure we can deduce that even though the ANN model can classify the second and the third cluster (slice) with high accuracy of 99.1% and 99.7% respectively, it has some confusion to classify the first cluster only 95.9%[11].

Three different supervised machine learning models for data traffic classification in the SDN platform. The accuracy obtained for SVM is 92.3%, Naïve Bayes is 96.79% and the nearest centroid is 91.02%. The challenges faced are in the live network data traffic capture and classification of the applications in the SDN platform. The training error in SVM is least, 2% error in Naïve Bayes and 7% error in the nearest centroid learning model.

The data traffic classification using supervised learning model is proposed in the SDN environment. The three different models used are SVM, nearest centroid and Naïve Bayes. The traditional approaches of traffic classification have a limitation due to 1000-fold increase in the network data traffic. The accuracy of traffic classification is greater than 90% in all the three supervised learning models[12].

Table 1: Related work details.

Ref	Dataset	Area of focus	Models	Accuracy	Year
[9]	IP network traffic flows labelled with 75 Apps	Architecting a machine learning pipeline for online traffic classification in software defined networking using spark	DT, RF, LR	98.71%, 84.9%, 82.7%	2023
[8]	NF-ToNIoT, NF-BoTIoT, NF-CSECICIDS2018, NF-UNSW-NB15, NF-UQ-NIDS, CIC flowmeter-based IoT DS2, IoT Network Intrusion, MQTTIoTIDS2020, CIC-ToNIoT	A Transfer Learning based Intrusion detection system for Internet of Things	CNN	99.66%, 93.82%, 95%33%, 98.62%, 98.34%, 97.17%, 96.20%, 95.33%, 86.96%	2023
[10]	Captured with ONOS SDN controller	Traffic Data Classification using Machine Learning Algorithms in SDN Networks	RF, LDA, DNN	Normal 95%, 98%, 69% Ab-normal 42%, 76%, 74%	2020
[11]	IP Network Traffic Flows, Labeled with 75 Apps	Network Slicing Based on Real-Time Traffic Classification in Software Defined Network (SDN) using Machine Learning	LR, SVM (linear), SVM (RBF), KNN, ANN	85.5%, 96.7%, 90.7%, 92.5%, 98.2%	2022
[12]	Captured	Data Traffic Classification in Software Defined Networks (SDN) using supervised-learning	SVM, NB, NC	92.3%, 96.79%, 91.02%	2020

3.2 Traditional Methods:

Traditional network traffic classification approaches, such as the port-based approach, identify an application by examining the packet header. However, this approach is unreliable since the current applications flow with unusual port numbers or select dynamic ports, leading to a rise in the false-negative rate of the classifier. In some situations, illegitimate applications hide using standard ports to avoid being filtered, increasing the false-positive results of classifiers due to undetectable applications. In addition, it is unfeasible to recognize the actual port numbers when handling encrypted data [9].

Deep packet inspection (DPI) is developed to overcome the insufficiency of the port-based approach. It is based on inspecting the contents of the packet rather than its header. Although this approach is considered reliable; it has some weaknesses. First, it is computationally expensive since it needs several accesses to the packet content. Second, it is impossible to examine an encrypted packet using this method. Finally, privacy challenges are encountered when inspecting packet contents [9].

3.3 Proposed Method:

Deep learning algorithms, such as Convolution Neural Networks (CNN), have proven their efficiency through the unnecessary of extracting any statistical feature and through their reliance on the employment of the raw network traffic as their input[13]. Therefore, the proposed method for network traffic classification involves a three-step process using deep learning models. Firstly, a Convolutional Neural Network (CNN) is employed for binary classification to distinguish between normal and abnormal traffic. Secondly, the abnormal traffic is further analyzed using another CNN model to classify the specific type of network attack. Lastly, for normal traffic, an LSTM (Long Short-Term Memory) model is utilized to identify the specific application type. This multi-tiered approach leverages transfer learning and various datasets, including CICIDS 2017, CICIDS 2018, CIC-ToN-IoT, and IP Network Traffic Flows, to achieve high accuracy in both binary and multi-class classification tasks.

Chapter 4

(Project Implementation Stages)

The exponential growth in the number of connections and devices worldwide, combined with the rise of sophisticated, intelligent devices like smartphones and smart televisions, has significantly increased network traffic and altered traffic flow patterns and network performance. This surge in network traffic, coupled with the proliferation of Machine-to-Machine (M2M) communications, presents considerable challenges for network management and security. In addition, managing diverse network infrastructures to meet the demands of new devices and applications has become increasingly complex, driving interest in autonomous networks capable of self-configuration, self-repair, self-optimization, and self-protection through cognitive techniques and Machine Learning (ML).

In related work, various ML approaches have demonstrated potential in addressing these challenges. For instance, the transfer learning approach has significantly reduced training and testing times, achieving 93.75% accuracy in identifying 20 different attacks on the NF-UQ NIDS dataset and showing effective real-time performance on resource-limited edge devices. Other studies have employed decision trees, random forests, and logistic regression using the Spark ML library, achieving accuracies up to 98%. Additionally, clustering algorithms and neural networks have been used to optimize traffic classification and handle class imbalance, achieving high accuracy rates of up to 99.7%.

Building on these foundations, my project leverages Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to classify network traffic, detect anomalies, and identify specific types of network attacks. Using datasets such as CICIDS 2017, CICIDS 2018, and CIC-ToN-IoT, my models achieve high accuracy rates—99.7% for binary classification, 99.02% for multi-class application type classification, and 96.83% for multi-class attack type classification. This comprehensive approach not only enhances network security but also provides a robust framework for real-time network traffic monitoring and classification, addressing the complexities of modern network environments.

4.1 Proposed Method Architecture:

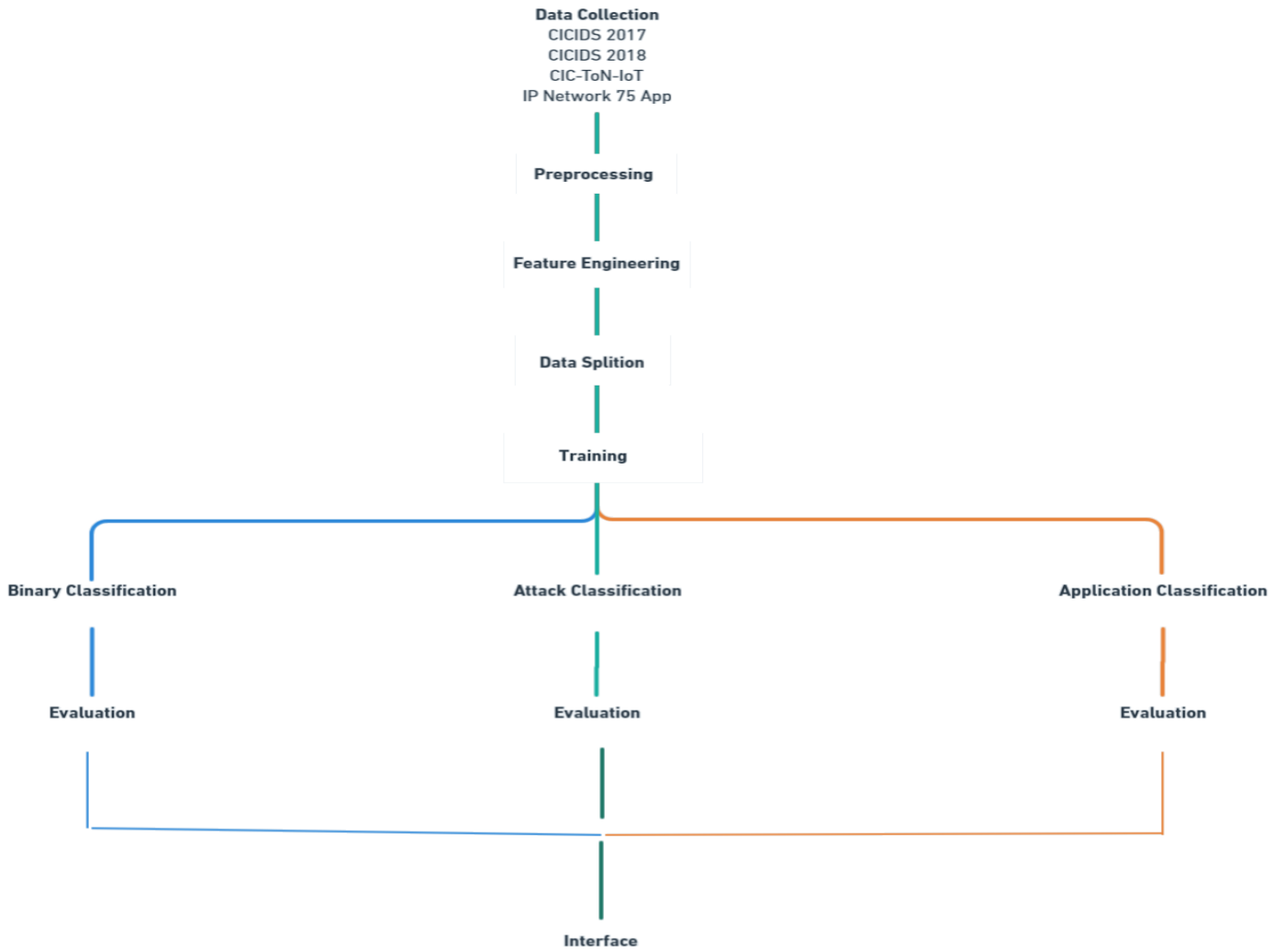


Figure 7: Proposed Method Architecture

4.1.1 Datasets

4.1.1.1 CICIDS2017:

The CICIDS2017 dataset contains benign (Normal) and the most up-to-date common attacks, closely resembling real-world data. Data capturing started at 9 a.m. on Monday, July 3, 2017, and ended at 5 p.m. on Friday, July 7, 2017, spanning a total of 5 days. Monday exclusively contains benign traffic, while the remaining days include a mix of benign and attack traffic. The implemented attacks are Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. These attacks were executed both in the morning and afternoon on Tuesday, Wednesday, Thursday, and

Friday. The dataset comprises 2,830,743 records and 80 features [3]. The CICIDS2017 dataset is available for download from the University of New Brunswick.

4.1.1.2 CICIDS2018:

The CICIDS2018 dataset, also available from the University of New Brunswick, contains 80 features and consists of different datasets captured on various days. It is a comprehensive dataset designed to include a wide range of attack scenarios and normal traffic, providing a diverse set of data for network traffic analysis and model training[14].

4.1.1.3 IP Network Traffic Flows, Labeled with 75 Apps:

The 75 Apps dataset was collected in a network section at Universidad Del Cauca, Popayán, Colombia. Data capturing occurred at different hours, both morning and afternoon, over six days: April 26, 27, 28, and May 9, 11, and 15 of 2017. The dataset comprises 3,577,296 instances stored in a CSV (Comma Separated Values) file, containing 87 features. Each instance includes information about an IP flow generated by a network device, such as source and destination IP addresses, ports, interarrival times, and the layer 7 protocol (application) used on that flow as the class. Most attributes are numeric, with some nominal types and a date type due to the Timestamp[15].

4.1.1.4 CIC-ToN-IoT:

The CIC-ToN-IoT dataset was generated using the CICFlowMeter-v4 tool, which extracted 83 features from the pcap files of the ToN-IoT dataset. The dataset contains 5,351,760 data samples, with 2,836,524 (53.00%) labeled as attacks and 2,515,236 (47.00%) as benign. This dataset provides a significant amount of data for training and evaluating intrusion detection models. The CIC-ToN-IoT dataset is available for download from the University of Queensland website[8].

4.1.2 Preprocessing:

The CICIDS 2017 dataset consists of various subsets captured on different days, requiring their combination into a single dataset for analysis. The dataset includes 15 types of data: one normal and 14 types of attacks, (as detailed in Figure 8). However, certain attack types, such as XSS, Brute Force, Injection, and Port-scan, have fewer than 2000 instances (except for Port-scan) out of a total of 2,830,743 rows, potentially leading to bias[16].

To address this issue, data augmentation was chosen over oversampling[17]. Data augmentation is considered superior in maintaining the integrity of the dataset while increasing the representation of underrepresented attack types. Specifically, the CICIDS 2017 dataset was augmented with a subset of the CIC-ToN-IoT dataset, adding records for "XSS," "Brute Force,"

"SQL Injection," and "PortScan." A total of 660,711 records were randomly selected from the CIC-ToN-IoT dataset and integrated into the CICIDS 2017 dataset with corresponding labels, (as detailed in Table 2).

Next, the combined dataset underwent a series of checks. The total number of records and columns was verified, and each label's value count and percentage were calculated. The dataset was examined for null values, revealing 1358 null values in the "Flow Bytes" feature. Given the low number and their distribution across different rows, these null values were addressed appropriately.

The dataset was also checked for infinity values, identifying 1509 infinity values in the "Flow Bytes" and "Packets" features. These values appeared in various rows across different data types, including Normal, Portscan, Bot, FTP-Patator, and DDoS data.

Statistical analysis played a crucial role in understanding the dataset's characteristics. A function was created to compute the minimum, maximum, mean, median, standard deviation, and mode for each feature. This analysis informed the decision-making process regarding normalization.

Features were categorized into two groups: flag features and those whose maximum values are 0 or 1 and minimum values are 0. These features did not require normalization, as determined by the `max_two_feature` function. The remaining features required normalization, but first, they needed to be checked for outliers. The `outlier_feature` function was employed to identify outliers in each feature, ensuring the dataset was properly prepared for model training.

```

Label
BENIGN                2273097
DoS Hulk              231073
PortScan              158930
DDoS                  128027
DoS GoldenEye         10293
FTP-Patator           7938
SSH-Patator           5897
DoS slowloris         5796
DoS Slowhttptest      5499
Bot                   1966
Web Attack 💎 Brute Force  1507
Web Attack 💎 XSS         652
Infiltration           36
Web Attack 💎 Sql Injection  21
Heartbleed             11
Name: count, dtype: int64

```

Figure 8: CICIDS 2017 types data

Table 2: label value types in CICIDS 2017, 2018 before and after augmentation

Dataset	Attack Types	Before values	After Values
CICIDS 2017	Brute Force, SQL Injection, XSS, Port Scan	1507, 21, 652, 158930	125052, 106999, 415862, 173908
CIC-ToN-IoT	Brute Force, SQL Injection, XSS, Port Scan	340208, 277696, 2149308, 36205	216663, 170718, 1734098, 21227

4.1.3 Feature Engineering:

After identifying null values and infinity values in the dataset, the null values were dropped[18], while the infinity values were replaced with the mean of their respective columns and corresponding label types. Two features, "Protocol" and "Forward Header Length 1," were found to be unrelated or duplicate features and were thus removed from the dataset.

Statistical analysis informed the decision-making process regarding the scaling method. Outlier features were identified and prepared for scaling using Robust Scaler normalization[19, 20], ensuring that the dataset's features were appropriately normalized despite the presence of outliers.

Next, the dataset was prepared for further analysis by converting the "Label" feature from an object type to an integer type. In this process, all non-normal values were converted to 1, representing attacks, while normal values were converted to 0, representing benign traffic. This binary classification setup was essential for subsequent model training and evaluation.

4.1.3.1 Attack type Feature Engineering:

Model's dataset is CICIDS 2017 so its machine learning pipeline is the same as previous but dataset is preparing for attack multi class classification such that Normal data are dropped, Oversampling method[21] is used for balancing types of attacks which has values less than 11000 and oversample them between 30,000 and 50,000 randomly. and One Hot Encoder

method is used for converting label feature from object type to numerical type. Its total data was 2829385 (Figure 8) and after dropping Normal data it became to 556697 and after Oversampling become to 951120 (Figure 9) and 91 features after One-Hot-Encoder.




Label	
DoS Hulk	230124
PortScan	158930
DDoS	128027
DoS GoldenEye	49556
Heartbleed	47760
FTP-Patator	47378
SSH-Patator	42145
Infiltration	41424
Web Attack  XSS	40349
DoS slowloris	36466
Bot	33813
Web Attack  Brute Force	32864
Web Attack  Sql Injection	32177
DoS Slowhttptest	30107
Name: count, dtype: int64	

Figure 9: Label attack types after oversample.

4.1.3.2 Application type Feature Engineering:

IP Network Traffic Flows Labeled with 75 Apps dataset Label values with less than 1,000 instances are removed from the dataset, except for Google-Maps, which has 807 records. This step is taken to prevent model bias [15]. Initially, the dataset contained 78 applications, but after filtering, it was reduced to 32 applications (Figure 10). Subsequently, label values with fewer than 60,000 instances are oversampled randomly to achieve a range between 60,000 and 80,000 instances for dataset balancing. Additionally, irrelevant features (Flow ID, Source IP, Source Port, Destination IP, Protocol, Timestamp, Fwd. Header Length 1, and Label) are dropped from the dataset.

GOOGLE	959110
HTTP	683734
HTTP_PROXY	623210
SSL	404883
HTTP_CONNECT	317526
YOUTUBE	170781
AMAZON	86875
MICROSOFT	54710
GMAIL	40260
WINDOWS_UPDATE	34471
SKYPE	30657
FACEBOOK	29033
DROPBOX	25102
YAHOO	21268
TWITTER	18259
CLOUDFLARE	14737
MSN	14478
CONTENT_FLASH	8589
APPLE	7615
OFFICE_365	5941
WHATSAPP	4593
INSTAGRAM	2415
WIKIPEDIA	2025
MS_ONE_DRIVE	1748
DNS	1695
IP_ICMP	1631
NETFLIX	1560
APPLE_ITUNES	1287
SPOTIFY	1269
APPLE_ICLOUD	1200
EBAY	1192

Figure 10: Label values after dropping values less than 1000 records.

4.1.4 Splitting dataset:

The dataset is split into training, validation, and test sets with the following proportions: 70% for training, 15% for validation, and 15% for testing. This ensures that the model has a substantial amount of data for learning while also having enough data for validation and testing to evaluate its performance.

Next, the training, validation, and test datasets are converted into tensors and reshaped from 2D to 3D to fit the requirements of the model. The `TensorFlow` dataset method is used to combine `x_train` and `y_train` into a single training dataset, `x_val` and `y_val` into a validation dataset, and `x_test` and `y_test` into a test dataset.

Finally, these datasets are shuffled and a batch size of 32 is selected, which is a common choice for training deep learning models. This setup ensures efficient training and evaluation of the model.

4.1.5 Training:

4.1.5.1 Binary Classification:

The binary classification model is trained using a Convolutional Neural Network (CNN) with 7 layers, implemented in Keras. The architecture is as follows:

1. **First Layer:** A 1D convolutional layer with 8 filters, a kernel size of 3, and ReLU activation function. This layer processes the input data with the specified input shape.
2. **Second Layer:** A 1D max pooling layer with a pool size of 2, which reduces the dimensionality of the data and helps in down-sampling.
3. **Third Layer:** Another 1D convolutional layer with 16 filters, a kernel size of 3, and ReLU activation function, further extracting features from the data.
4. **Fourth Layer:** Another 1D max pooling layer with a pool size of 2, reducing the data size further.
5. **Fifth Layer:** A flattening layer, which converts the 2D data into a 1D vector, preparing it for the dense layers.
6. **Sixth Layer:** A dense (fully connected) layer with 32 neurons and ReLU activation function, learning complex patterns in the data.
7. **Final Layer:** A dense layer with a single neuron and sigmoid activation function, used for binary classification (normal vs. abnormal).

The model training took approximately one and a half hours, achieving an accuracy of 99.74% on the training dataset and 99.71% on the validation and test datasets (see architecture in Figure 11 and accuracy and epochs in Figure 12).

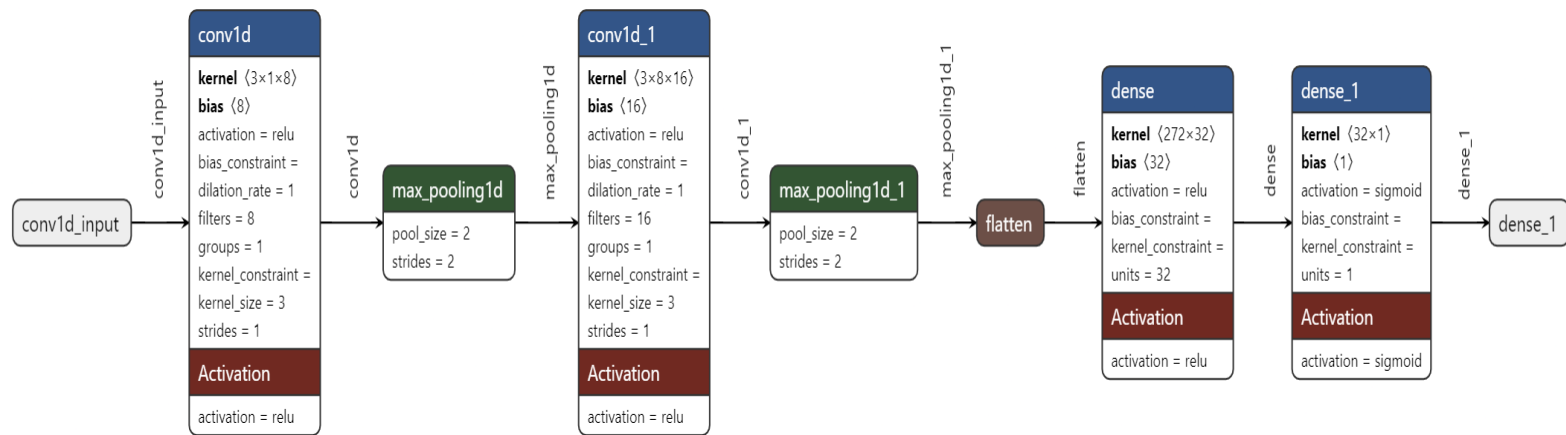


Figure 11: Binary model Architecture.

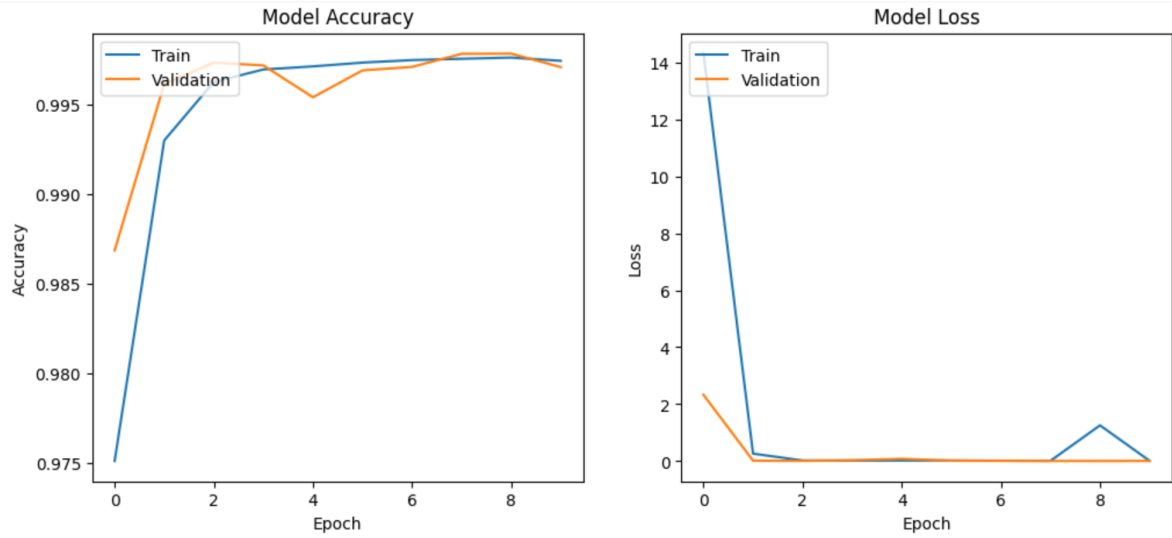


Figure 12: Binary model Accuracy and epochs.

4.1.5.1.1 Transfer Learning

The initial model was tested on an unseen dataset (CIC-ToN-IoT) and achieved an accuracy of 80%. This indicated that the model did not generalize well on patterns from the CICIDS 2017 dataset alone. To address this issue, transfer learning was applied.

First Transfer Learning Implementation

Datasets: CICIDS 2018 datasets from 02-16-2018 and 02-21-2018 were selected.

16-2018 Dataset:

- Features: 80 features
- Records: 1,048,575 values
- Distribution: 461,912 DoS Hulk, 446,772 Normal, and 139,890 DoS SlowHTTPTest values.
- Preprocessing: Similar to previous stages, but object features were converted to float32, keeping the Label feature for further processing. Two irrelevant features were removed.
- Model Architecture: The new model architecture mirrored the binary model. During transfer learning, all layers from the initial model were frozen except for the last two dense layers. The new model was trained with 10 epochs and a learning rate of 0.0001 (see Figure 13 for architecture).

- Results: After retraining, the model was tested again on the CIC-ToN-IoT dataset and achieved an improved accuracy of 83%.

Second Transfer Learning Implementation

Dataset: CICIDS 2018 dataset from 02-21-2018.

- Features: Same as the 02-16-2018 dataset.
- Records: 1,048,575 values
- Distribution: 686,012 DDoS Hoic, 360,833 Normal, and 1,730 DDoS LOIC-UDP records.
- Preprocessing: Similar preprocessing steps were followed, with the removal of two unrelated features.
- Model Architecture: The new model was trained based on the first transfer learning model's structure (see Figure 14 for architecture).
- Results: The model achieved an accuracy of 99.99% and evaluation metrics (see Figures 15, 16, and 17). When tested on the CIC-ToN-IoT dataset, it achieved an accuracy of 90%, which is a significant improvement.

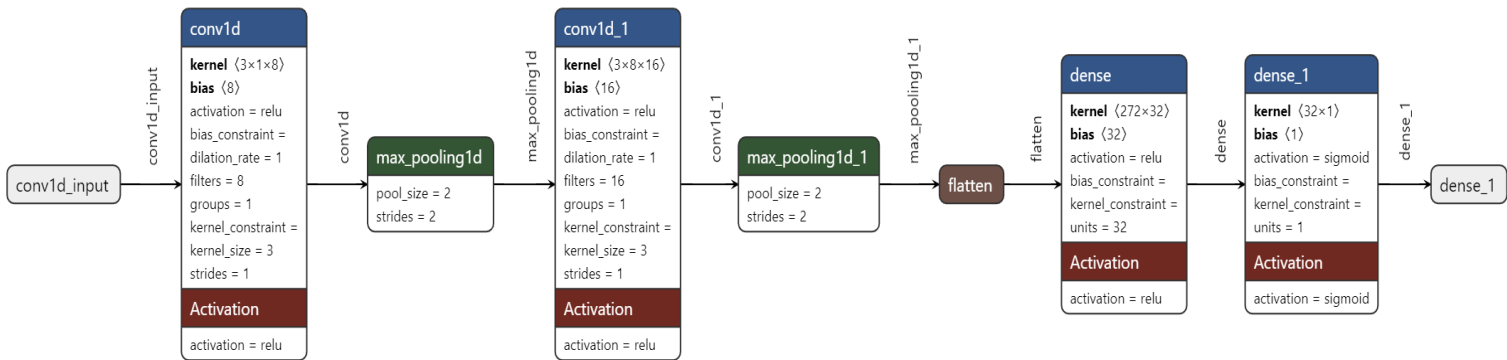


Figure 13: First Transfer Architecture Model.

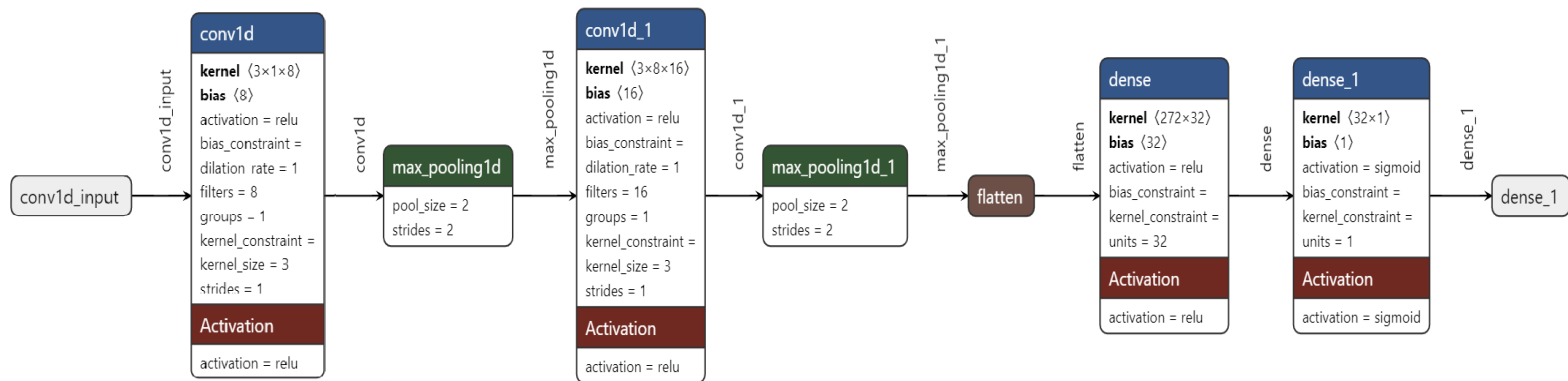


Figure 14: Second Transfer Architecture Model.

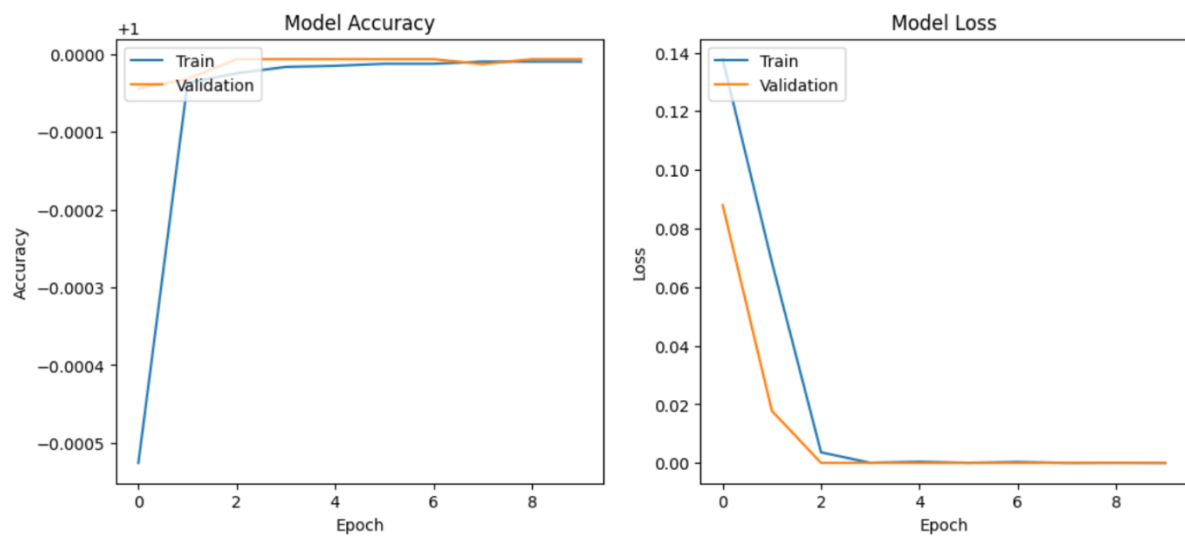


Figure 15: Transfer-learning Accuracy.

Classification Report:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	66890
1	1.00	1.00	1.00	90397
accuracy			1.00	157287
macro avg	1.00	1.00	1.00	157287
weighted avg	1.00	1.00	1.00	157287

Figure 16: Classification report.

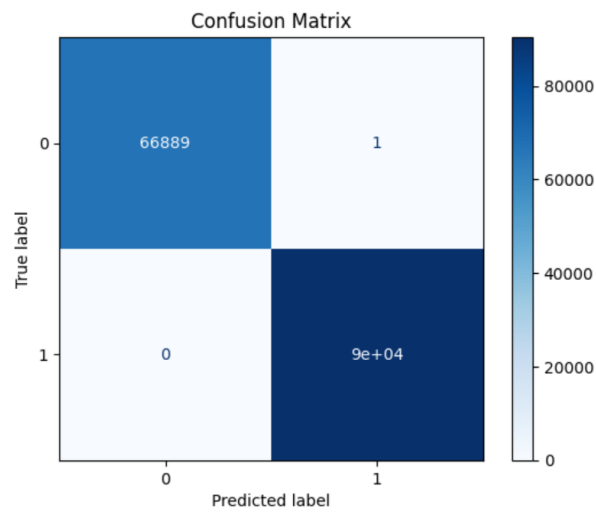


Figure 17: Confusion matrix.

4.1.5.2 Attack types Classification:

The second model in this project is designed to classify network attack types, capable of identifying 14 common attacks: DoS Hulk, PortScan, DDoS, DoS GoldenEye, HeartBleed, FTP-Patator, SSH-Patator, Infiltration, Web XSS, DoS Slowloris, Bot, Brute Force, SQL Injection, and DoS Slowhttptest. This model uses a Convolutional Neural Network (CNN) algorithm, but with a different architecture compared to the binary classification model.

The model architecture consists of 10 layers:

- **First Layer:** A 1D CNN layer with 32 filters, a kernel size of 3, ReLU activation function, and padding.
- **Second Layer:** A Batch Normalization layer to stabilize and accelerate training.
- **Third Layer:** A 1D Max Pooling layer with a pool size of 2 to reduce dimensionality.
- **Fourth Layer:** Another 1D CNN layer with 64 filters, a kernel size of 3, padding, and ReLU activation function.
- **Fifth Layer:** A Batch Normalization layer.
- **Sixth Layer:** A 1D Max Pooling layer with a pool size of 2.
- **Seventh Layer:** A flattening layer to convert the 2D data into a 1D vector.
- **Eighth Layer:** A dense layer with 64 neurons and ReLU activation function.
- **Ninth Layer:** Another Batch Normalization layer.
- **Tenth Layer:** A dense layer with 14 neurons and a softmax activation function for multi-class classification (architecture in Figure. 18).

The model achieved an accuracy of 96.7% on the training dataset, 96.77% on the validation dataset (accuracy in Figure. 19), and 96.83% on the test dataset.

This architecture and performance indicate that the model is robust in identifying various types of network attacks, making it a valuable tool for enhancing network security through detailed attack classification.

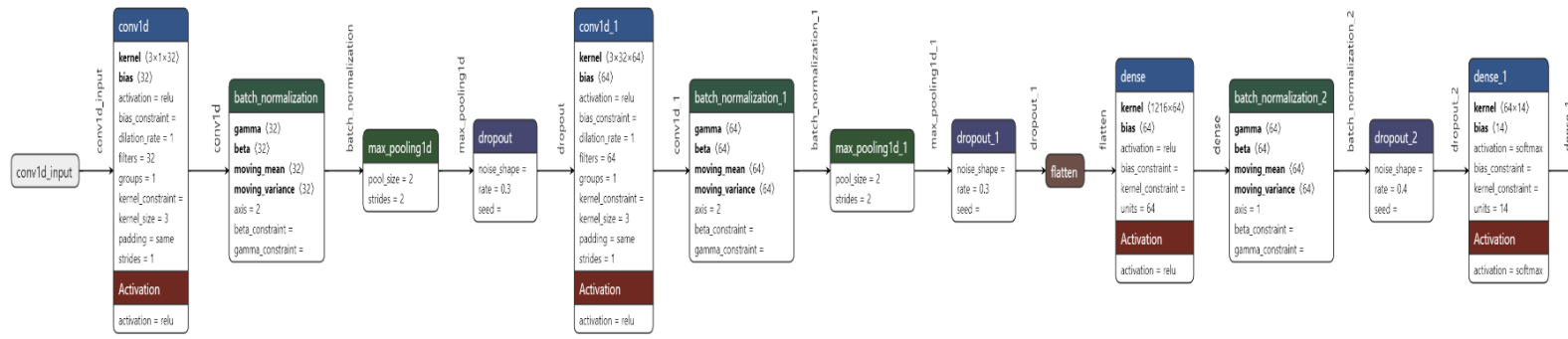


Figure 18: Attack Classification Architecture.

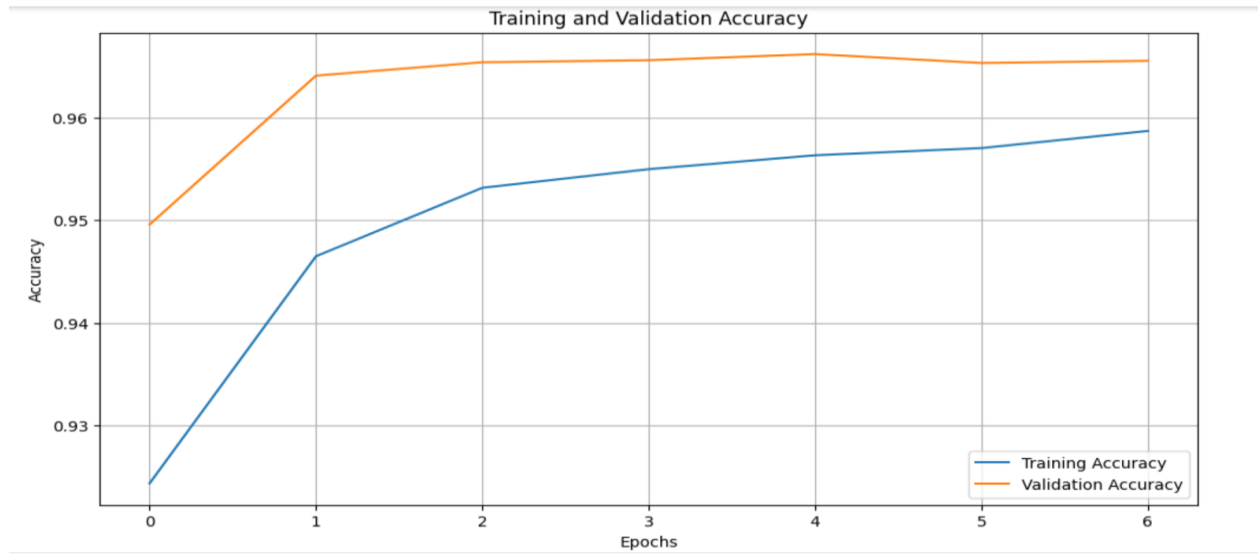


Figure 19: Accuracy and Evaluation Metrics.

4.1.5.3 Application Type:

Five models were tested on this dataset: four CNN models and one LSTM (Long Short-Term Memory) model, which is a type of Recurrent Neural Network (RNN).

First Model:

The first model consists of 7 layers:

Two 1D CNN layers: With 8 and 16 filters, kernel size of 3, and ReLU activation function.

Two 1D Max-Pooling layers: Each with a pool size of 2.

Flatten layer: Converts the 2D data to 1D.

Two Dense layers: The first dense layer has 32 neurons and ReLU activation, and the final dense layer has 32 neurons with a softmax activation function.

The model uses Adam optimizer with a learning rate of 0.0001, sparse categorical cross-entropy as the loss function, and accuracy as the metric. It was trained for 10 epochs using a TPU processor, with a total training time of 1 hour. The model achieved an accuracy of 76.54% on the training dataset, 75.53% on the validation dataset, and 76.21% on the test dataset.

Second Model:

The second model has 13 layers:

Two 1D CNN layers: With 64 and 128 filters, kernel size of 3, padding, and ReLU activation function.

Three Batch-Normalization layers.

Two 1D Max-Pooling layers.

Three Dropout layers.

Flatten layer.

Two Dense layers.

The compilation and fitting parameters were the same as the first model. It was trained using a CPU processor, with a training time of 2.5 hours. The model achieved an accuracy of 71% on the training dataset, 73.91% on the validation dataset, and 73.27% on the test dataset.

Third Model:

After adjusting the dataset using under sampling to reduce higher label values to 80,000 and oversampling lower label values between 60,000 to 80,000, a new CNN model was trained. The third model has 10 layers:

Two 1D CNN layers: With 32 and 64 filters, kernel size of 3.

Three Batch-Normalization layers.

Two 1D Max-Pooling layers.

Two Dense layers.

Flatten layer.

The compilation parameters remained the same, but the loss function was changed to categorical cross-entropy. The model was trained for 20 epochs, taking 1.5 hours. It achieved an accuracy of 78.8% on the training dataset, 71.84% on the validation dataset, and 71.83% on the test dataset. The low accuracy led to exploring another neural network algorithm, LSTM.

LSTM Model:

The LSTM model has:

Two LSTM layers: Each with 64 neurons.

A Dense layer: With softmax activation function.

This model was trained for 7 epochs, with the same compilation and fitting parameters as the previous models. The training time was 1 hour and 40 minutes (Figure. 20). It achieved an accuracy of 98.39% on the training dataset, 99.02% on the validation dataset, and 99.02% on the test dataset.

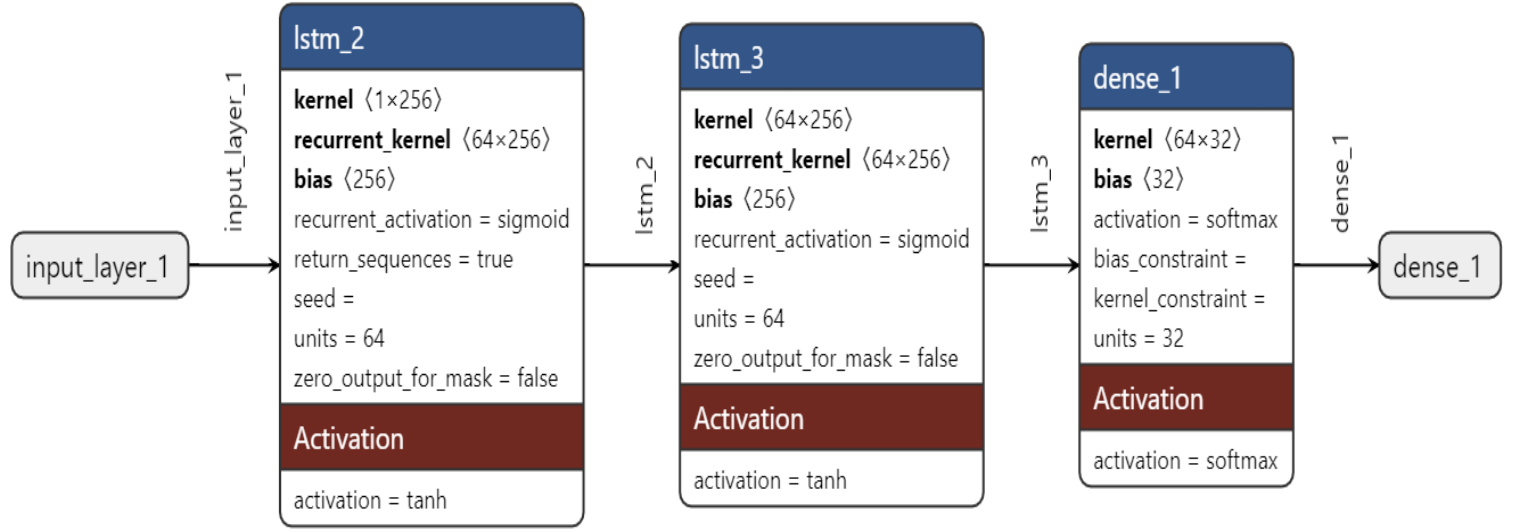


Figure 20: Application classification LSTM architecture.

4.1.6 Evaluation:

4.1.6.1 Binary

The binary classification model was evaluated using several metrics: confusion matrix, ROC curve, AUC, precision, recall, and F1 score.

Confusion Matrix: Out of 341,036 negative samples (label 0) in the test dataset, the model correctly predicted 340,705 and incorrectly predicted 331. For the 182,479 positive samples (label 1), it correctly predicted 181,341 and incorrectly predicted 1,138 (details in Figure. 21).

Precision, Recall, and F1 Score: The precision, recall, and F1 score were all 100% (details in Figure. 22).

ROC Curve and AUC: Both the ROC curve and AUC were also 100% (details in Figure. 23).

In addition, k-fold cross-validation was used to check for overfitting, underfitting, and bias. The training dataset was split into 5 parts and trained for 10 epochs, taking 6.8 hours. The results were consistent with the initial model, with minimum and maximum evaluation scores of 99.21% and 99.85%, respectively.

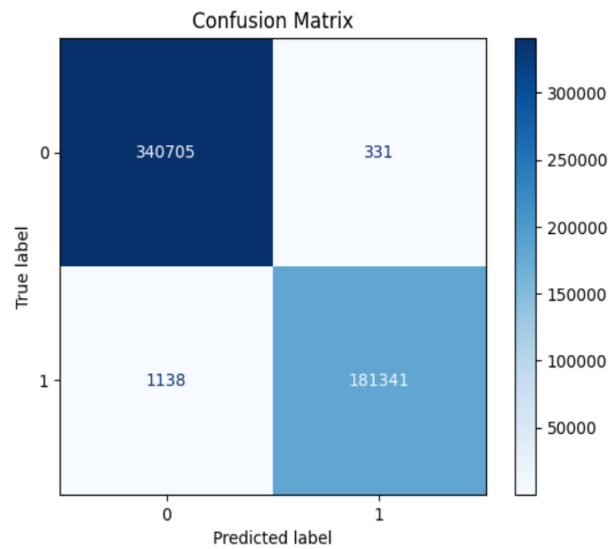


Figure 21: Confusion matrix of Binary.

Classification Report:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	341036
1	1.00	0.99	1.00	182479
accuracy			1.00	523515
macro avg	1.00	1.00	1.00	523515
weighted avg	1.00	1.00	1.00	523515

Figure 22: Classification Report of Binary.

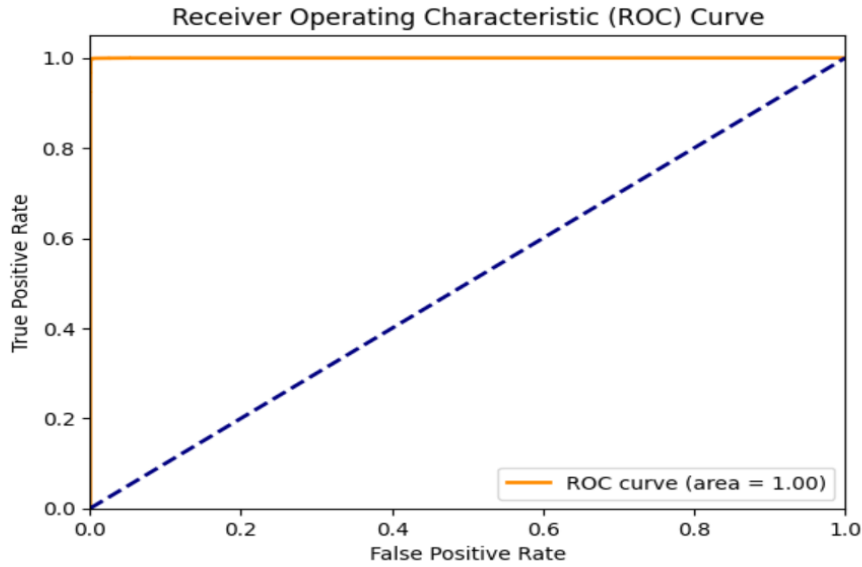


Figure 23: ROC Curve of Binary.

4.1.6.2 Attack types:

Evaluation Matrixes are confusion-matrix, ROC curve, AUC, Precision, Recall and F1 score (Figures 24, 25).

Confusion Matrix as List:

```
{'Class': 0, 'True Positive': 7135, 'False Positive': 9, 'False Negative': 7, 'True Negative': 192585}
{'Class': 1, 'True Positive': 26608, 'False Positive': 13, 'False Negative': 63, 'True Negative': 173052}
{'Class': 2, 'True Positive': 10162, 'False Positive': 151, 'False Negative': 111, 'True Negative': 189312}
{'Class': 3, 'True Positive': 48448, 'False Positive': 120, 'False Negative': 67, 'True Negative': 151101}
{'Class': 4, 'True Positive': 6349, 'False Positive': 174, 'False Negative': 39, 'True Negative': 193174}
{'Class': 5, 'True Positive': 7344, 'False Positive': 29, 'False Negative': 198, 'True Negative': 192165}
{'Class': 6, 'True Positive': 10054, 'False Positive': 8, 'False Negative': 31, 'True Negative': 189643}
{'Class': 7, 'True Positive': 10030, 'False Positive': 0, 'False Negative': 0, 'True Negative': 189706}
{'Class': 8, 'True Positive': 8760, 'False Positive': 6, 'False Negative': 11, 'True Negative': 190959}
{'Class': 9, 'True Positive': 33461, 'False Positive': 3, 'False Negative': 12, 'True Negative': 166260}
{'Class': 10, 'True Positive': 8713, 'False Positive': 77, 'False Negative': 54, 'True Negative': 190892}
{'Class': 11, 'True Positive': 944, 'False Positive': 21, 'False Negative': 5980, 'True Negative': 192791}
{'Class': 12, 'True Positive': 6697, 'False Positive': 480, 'False Negative': 2, 'True Negative': 192557}
{'Class': 13, 'True Positive': 8293, 'False Positive': 5647, 'False Negative': 163, 'True Negative': 185633}
```

Figure 24: Confusion matrix of Attack.

Classification Report:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	7142
1	1.00	1.00	1.00	26671
2	0.99	0.99	0.99	10273
3	1.00	1.00	1.00	48515
4	0.97	0.99	0.98	6388
5	1.00	0.97	0.98	7542
6	1.00	1.00	1.00	10085
7	1.00	1.00	1.00	10030
8	1.00	1.00	1.00	8771
9	1.00	1.00	1.00	33473
10	0.99	0.99	0.99	8767
11	0.98	0.14	0.24	6924
12	0.93	1.00	0.97	6699
13	0.59	0.98	0.74	8456
accuracy			0.97	199736
macro avg	0.96	0.93	0.92	199736
weighted avg	0.98	0.97	0.96	199736

Figure 25: Classification Report of Attack.

4.1.6.3 Application type:

Confusion matrix, f1 score, precision, recall are used to determine the accuracy Figure (26, 27).

Confusion Matrix as List:

```
{'Class': 0, 'True Positive': 18282, 'False Positive': 156, 'False Negative': 53, 'True Negative': 1030217}
{'Class': 1, 'True Positive': 15112, 'False Positive': 0, 'False Negative': 1, 'True Negative': 1033595}
{'Class': 2, 'True Positive': 14731, 'False Positive': 1, 'False Negative': 22, 'True Negative': 1033954}
{'Class': 3, 'True Positive': 14013, 'False Positive': 1, 'False Negative': 4, 'True Negative': 1034690}
{'Class': 4, 'True Positive': 12245, 'False Positive': 3983, 'False Negative': 1223, 'True Negative': 1031257}
{'Class': 5, 'True Positive': 15900, 'False Positive': 0, 'False Negative': 0, 'True Negative': 1032808}
{'Class': 6, 'True Positive': 13702, 'False Positive': 0, 'False Negative': 0, 'True Negative': 1035006}
{'Class': 7, 'True Positive': 14389, 'False Positive': 9, 'False Negative': 0, 'True Negative': 1034310}
{'Class': 8, 'True Positive': 14154, 'False Positive': 49, 'False Negative': 142, 'True Negative': 1034363}
{'Class': 9, 'True Positive': 13944, 'False Positive': 3, 'False Negative': 3, 'True Negative': 1034758}
{'Class': 10, 'True Positive': 16581, 'False Positive': 2, 'False Negative': 13, 'True Negative': 1032112}
{'Class': 11, 'True Positive': 201628, 'False Positive': 12, 'False Negative': 12, 'True Negative': 847056}
{'Class': 12, 'True Positive': 13611, 'False Positive': 9, 'False Negative': 5, 'True Negative': 1035083}
{'Class': 13, 'True Positive': 143920, 'False Positive': 0, 'False Negative': 0, 'True Negative': 904788}
{'Class': 14, 'True Positive': 66940, 'False Positive': 17, 'False Negative': 11, 'True Negative': 981740}
{'Class': 15, 'True Positive': 131211, 'False Positive': 10, 'False Negative': 22, 'True Negative': 917465}
{'Class': 16, 'True Positive': 11656, 'False Positive': 1099, 'False Negative': 1608, 'True Negative': 1034345}
{'Class': 17, 'True Positive': 12848, 'False Positive': 0, 'False Negative': 0, 'True Negative': 1035860}
{'Class': 18, 'True Positive': 11959, 'False Positive': 1619, 'False Negative': 1099, 'True Negative': 1034031}
{'Class': 19, 'True Positive': 15975, 'False Positive': 0, 'False Negative': 0, 'True Negative': 1032733}
{'Class': 20, 'True Positive': 9138, 'False Positive': 171, 'False Negative': 5475, 'True Negative': 1033924}
{'Class': 21, 'True Positive': 15954, 'False Positive': 6, 'False Negative': 0, 'True Negative': 1032748}
{'Class': 22, 'True Positive': 14155, 'False Positive': 3055, 'False Negative': 522, 'True Negative': 1030976}
{'Class': 23, 'True Positive': 12936, 'False Positive': 16, 'False Negative': 11, 'True Negative': 1035745}
{'Class': 24, 'True Positive': 15392, 'False Positive': 1, 'False Negative': 4, 'True Negative': 1033311}
{'Class': 25, 'True Positive': 84585, 'False Positive': 0, 'False Negative': 0, 'True Negative': 964123}
{'Class': 26, 'True Positive': 14130, 'False Positive': 3, 'False Negative': 8, 'True Negative': 1034567}
{'Class': 27, 'True Positive': 16132, 'False Positive': 22, 'False Negative': 2, 'True Negative': 1032552}
{'Class': 28, 'True Positive': 16564, 'False Positive': 4, 'False Negative': 14, 'True Negative': 1032126}
{'Class': 29, 'True Positive': 14769, 'False Positive': 9, 'False Negative': 1, 'True Negative': 1033929}
{'Class': 30, 'True Positive': 16159, 'False Positive': 0, 'False Negative': 0, 'True Negative': 1032549}
{'Class': 31, 'True Positive': 35732, 'False Positive': 4, 'False Negative': 6, 'True Negative': 1012966}
```

Figure 26: Application Classification Evaluation Metrics.

Classification Report:				
	precision	recall	f1-score	support
0	0.99	1.00	0.99	18335
1	1.00	1.00	1.00	15113
2	1.00	1.00	1.00	14753
3	1.00	1.00	1.00	14017
4	0.75	0.91	0.82	13468
5	1.00	1.00	1.00	15900
6	1.00	1.00	1.00	13702
7	1.00	1.00	1.00	14389
8	1.00	0.99	0.99	14296
9	1.00	1.00	1.00	13947
10	1.00	1.00	1.00	16594
11	1.00	1.00	1.00	201640
12	1.00	1.00	1.00	13616
13	1.00	1.00	1.00	143920
14	1.00	1.00	1.00	66951
15	1.00	1.00	1.00	131233
16	0.91	0.88	0.90	13264
17	1.00	1.00	1.00	12848
18	0.88	0.92	0.90	13058
19	1.00	1.00	1.00	15975
20	0.98	0.63	0.76	14613
21	1.00	1.00	1.00	15954
22	0.82	0.96	0.89	14677
23	1.00	1.00	1.00	12947
24	1.00	1.00	1.00	15396
25	1.00	1.00	1.00	84585
26	1.00	1.00	1.00	14138
27	1.00	1.00	1.00	16134
28	1.00	1.00	1.00	16578
29	1.00	1.00	1.00	14770
30	1.00	1.00	1.00	16159
31	1.00	1.00	1.00	35738
accuracy			0.99	1048708
macro avg	0.98	0.98	0.98	1048708
weighted avg	0.99	0.99	0.99	1048708

Figure 27: Classification Report of Application.

Proposed method result.

Table 3: Proposed Method Result

Proposed method	Dataset	Model	Accuracy
Binary Classification	CICIDS 2017, 2018, CIC-ToN-IoT	CNN	99.71%
Attack Classification	CICIDS 2017	CNN	96.83%
Application Classification	IP Network Traffic Flows Labeled with 75 Apps	LSTM	99.02%
Total average:			98.52%

4.1.7 Proposed Interface

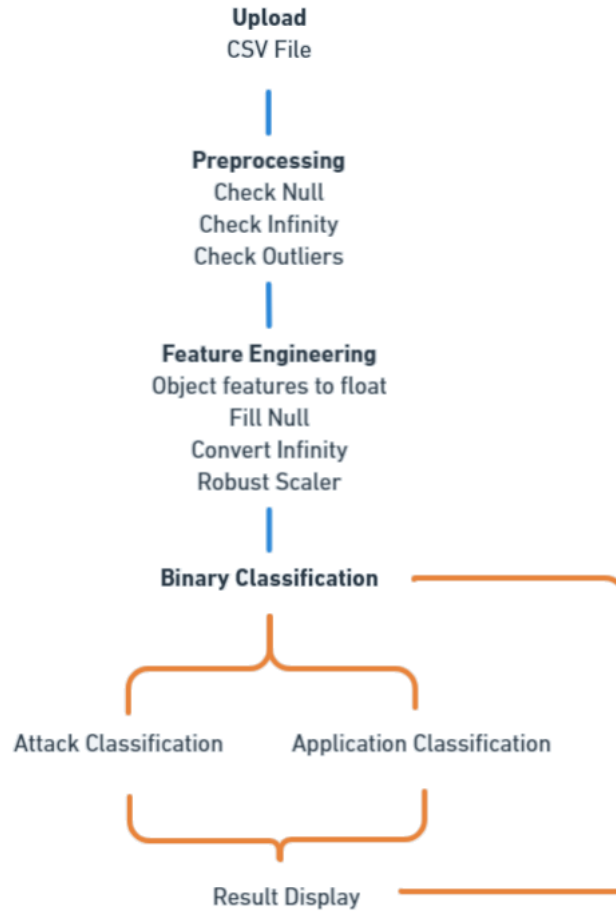


Figure 28: Proposed Method Interface

Interface:

The user interface is divided into six sections, utilizing HTML, CSS, Bootstrap, and JavaScript for the frontend, and Flask for the backend. The workflow begins when a network administrator uploads a CSV file. This file is processed by the initial classification function, which reads it using the pandas `read_csv` function. The preprocessing function is then called to prepare the dataset. If the dataset contains a 'Label' feature, it is dropped before being sent to the null values check function in a separate Python file.

The null values function examines the dataset for any missing values. If found, these are filled with the mean of their respective features, and the dataset is passed to the infinity values function. In this function, several steps are executed:

1. Convert any object (string) feature types to `float32`.
2. Fill any infinity values with the mean of their respective features.

Once these checks are completed, the dataset returns to the main file and is sent to the binary classification function for further preprocessing. Here, the feature names are checked and renamed to match the trained model's feature names if necessary. Outliers are identified and processed, with necessary features categorized to avoid normalization, as done during the model training. The remaining features are normalized using the RobustScaler method before returning to the main file. In the main file, the dataset first undergoes binary classification to determine if the data is normal or abnormal. If classified as normal, the Multi-Application-Types model is invoked for further classification; if abnormal, the Multi-Network-Attack-Types model is used to identify the type of network attack. The results of both the binary and multi-class classifications are then displayed on the screen for the administrator.

Chapter 5

(Conclusion)

5.1 Conclusion

This project successfully demonstrates the effectiveness of deep learning techniques in the realm of network traffic classification. By employing Convolutional Neural Networks (CNN) for binary classification and attack type detection, and Long Short-Term Memory (LSTM) networks for application type classification, we achieved high accuracy across multiple datasets, including CICIDS 2017, CICIDS 2018, CIC-ToN-IoT, and IP Network Traffic Flows Labeled with 75 Apps. The binary classification model attained an accuracy of 99.71%, the attack type classification model achieved 96.83%, and the application type classification model reached 99.02%. Moreover, the project highlighted the importance of handling imbalanced datasets through techniques such as data augmentation and oversampling, as well as the benefits of transfer learning in enhancing model performance on diverse datasets. The incorporation of statistical analysis and robust preprocessing steps ensured the reliability and robustness of the models. Overall, this project provides a comprehensive approach to leveraging machine learning for enhancing network security through accurate and efficient traffic classification.

5.2 Future Work

In future work, the developed models can be implemented in real-time network environments to enhance security measures. This would involve integrating the models into a network monitoring system capable of analyzing live traffic and providing immediate classification and alerts. Furthermore, combining these models with reinforcement learning techniques could significantly improve their adaptability and accuracy. Reinforcement learning can enable the system to dynamically learn and adapt to new types of traffic patterns and attacks, thus enhancing its ability to detect and mitigate evolving threats in real-time. By continually updating the models based on feedback from the network environment, the system could achieve higher levels of robustness and effectiveness in maintaining network security.

5.3 References

- [1] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, vol. 10, no. 1, pp. 1-35, 2020.
- [2] D. Nunez-Agurto, W. Fuertes, L. Marrone, and M. Macas, "Machine Learning-Based Traffic Classification in Software-Defined Networking: A Systematic Literature Review, Challenges, and Future Research Directions," *IAENG International Journal of Computer Science*, vol. 49, no. 4, 2022.
- [3] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, pp. 108-116, 2018.
- [4] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 3-10.
- [5] B. A. Forouzan, *Data Communications and Networking*, 4th ed. Alan R. Apt, 2007, p. 1171.
- [6] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, Third ed. United States of America.: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2023, p. 1351.
- [7] A. Burkov, *The Hundred-Page Machine Learning Book*. p. 152.
- [8] M. Vishwakarma and N. Kesswani, "A Transfer Learning based Intrusion detection system for Internet of Things," 2023.
- [9] S. S. Samaan and H. A. Jeiad, "Architecting a machine learning pipeline for online traffic classification in software defined networking using spark," *IAES Int. J. Artif. Intell. IJ-AI*, vol. 12, no. 2, p. 861, 2023.
- [10] J. Kwon, D. Jung, and H. Park, "Traffic data classification using machine learning algorithms in SDN networks," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020: IEEE, pp. 1031-1033.
- [11] A. A El-serwy, E. AbdElhalim, and M. A Mohamed, "Network Slicing Based on Real-Time Traffic Classification in Software Defined Network (SDN) using Machine Learning," *MEJ-Mansoura Engineering Journal*, vol. 47, no. 3, pp. 1-10, 2022.
- [12] M. M. Raikar, S. Meena, M. M. Mulla, N. S. Shetti, and M. Karanandi, "Data traffic classification in software defined networks (SDN) using supervised-learning," *Procedia Computer Science*, vol. 171, pp. 2750-2759, 2020.
- [13] A. Azab, M. Khasawneh, S. Alrabaee, K.-K. R. Choo, and M. Sarsour, "Network traffic classification: Techniques, datasets, and challenges," *Digital Communications and Networks*, 2022.
- [14] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, vol. 167, pp. 636-645, 2020.

- [15] J. a. R. G. Rojas Meléndez, Alvaro and Corrales, Juan, "Personalized Service Degradation Policies on OTT Applications Based on the Consumption Behavior of Users," *Lecture Notes in Computer Science*, pp. 543-557, 2018, doi: 10.1007/978-3-319-95168-3_37.
- [16] J. Chakraborty, S. Majumder, and T. Menzies, "Bias in machine learning software: Why? how? what to do?," in *Proceedings of the 29th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2021, pp. 429-440.
- [17] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?," in *2016 international conference on digital image computing: techniques and applications (DICTA)*, 2016: IEEE, pp. 1-6.
- [18] L. O. Joel, W. Doorsamy, and B. S. Paul, "A review of missing data handling techniques for machine learning," *International Journal of Innovative Technology and Interdisciplinary Sciences*, vol. 5, no. 3, pp. 971-1005, 2022.
- [19] L. T. Quang, B. H. Baek, W. Yoon, S. K. Kim, and I. Park, "Comparison of Normalization Techniques for Radiomics Features From Magnetic Resonance Imaging in Predicting Histologic Grade of Meningiomas," *Investigative Magnetic Resonance Imaging*, vol. 28, no. 2, pp. 61-67, 2024.
- [20] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, 2020.
- [21] Z. Zheng, Y. Cai, and Y. Li, "Oversampling method for imbalanced classification," *Computing and Informatics*, vol. 34, no. 5, pp. 1017-1037, 2015.