



**WALLAGA UNIVERSITY  
SCHOOL OF GRADUATE STUDIES  
INSTITUTE OF TECHNOLOGY  
DEPARTMENT OF COMPUTER SCIENCE  
PROGRAM MASTERS WEEKEND**

**Developing Deep learning based Predictive Model for Network Intrusion  
Detection System.**

**A THESIS IS SUBMITTED IN PARTIAL FULFILLMENT OF THE  
REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN  
COMPUTERSCIENCE**

**BY**

**Habtamu Tadese**

**ADVISORS : KEMAL.M(ASSISTANT PROFESSOR)**

**May, 2025  
NEKEMTE, ETHIOPIA**

|               |               |      |
|---------------|---------------|------|
| Name of Dean, | SGS Signature | Date |
|---------------|---------------|------|

## DECLARATION

I declare that this research titled “**Developing Deep Learning Based Predictive Model for Network Intrusion Detection system**” is my original work, and has not been submitted as a requirement for the award of any degree in the university or elsewhere.

As researcher Advisor, I hereby certify that I have read and evaluated this thesis prepared under my guidance, by Habtamu Tadesse entitled “**Developing Deep Learning Based Predictive Model for Network Intrusion Detection system**”, and recommended would be accepted as a fulfilling requirement for the Degree of Master of Science in computer science.

Adivisor(\_\_Prof \_\_\_\_\_)

---

Name Advisor

Signature

Date

## **ACKNOWLEDGEMENT**

I want to thank the Almighty God first and foremost for blessing me with the ability to complete all of my work in his perfect method throughout every minute of my life. Next, I would like to sincerely thank my advisor, Ass. Pro. Kama Mohamed , for his dedication, passionate support, insightful feedback, and helpful suggestions beginning with the early phases of conceptual inception and continuing guidance that helped me acknowledge and overcome the challenges I encountered to complete my proposal work. I couldn't finished my research work without his sincere and competent advice. Additionally, I have a special appreciation to all IT department and staff.

## Table of Contents

|                                                                |      |
|----------------------------------------------------------------|------|
| DECLARATION.....                                               | i    |
| ACKNOWLEDGEMENT.....                                           | ii   |
| Abstract.....                                                  | viii |
| LIST OF TBLE.....                                              | ix   |
| LIST OF FIGURE.....                                            | x    |
| LIST OF ACRONYM.....                                           | xi   |
| CHAPTER ONE: INTRODUCTION.....                                 | 1    |
| 1.1. Background of the study.....                              | 1    |
| 1.2. Motivation of the study.....                              | 2    |
| 1.3. Statement of the problem.....                             | 3    |
| 1.4. Research question.....                                    | 4    |
| 1.5. Objectives.....                                           | 4    |
| 1.5.1. General objective.....                                  | 4    |
| 1.5.2. Specific objective.....                                 | 4    |
| 1.6. Scope of the study.....                                   | 4    |
| 1.7. Limitation of the research study.....                     | 5    |
| 1.8. Contribution of the research.....                         | 5    |
| 1.9. Significance of the study.....                            | 6    |
| CHAPTER TWO.....                                               | 7    |
| REVIEW LITRETURE AND RELATED WORK.....                         | 7    |
| 2.1. Overview of intrusion detection system.....               | 7    |
| 2.2. History of Cyber Attacks.....                             | 8    |
| 2.4. Types of intrusion detection.....                         | 8    |
| 2.4.1. Signature-based intrusion detection systems (SIDS)..... | 9    |
| 2.4.2. Anomaly based intrusion detection systems.....          | 9    |
| 2.4.3. Hybrid based detection.....                             | 10   |
| 2.4.3. Network based IDS (NIDS).....                           | 10   |
| 2.4.5. Application based IDS.....                              | 11   |
| 2.4.6. Misuse detection (MD).....                              | 11   |
| 2.5. Challenges of intrusion detection System.....             | 12   |

|                                                             |    |
|-------------------------------------------------------------|----|
| 2.5.1. Impact of Network Intrusion .....                    | 14 |
| 2.6. Traditional Machine Learning Approaches.....           | 14 |
| 2.6.1. Types of machine learning.....                       | 15 |
| 2.6.2. Supervised machine learning algorithm.....           | 16 |
| 2.6.3. Unsupervised machine learning algorithms.....        | 16 |
| 2.6.4. Semi supervised learning ML algorithm.....           | 16 |
| 2.6.5 Artificial neural network (ANN).....                  | 17 |
| 2.8. Deep learning techniques for NIDS.....                 | 17 |
| 2.8.1. Deep learning architecture.....                      | 18 |
| 2.9. Types of deep learning algorithm.....                  | 21 |
| 2.9.1. Convolutional neural networks (CNN).....             | 21 |
| 2.9.2. Use of artificial neural network (ANN) in NIDS ..... | 23 |
| 2.9.3. ANN architectural layers.....                        | 23 |
| 2.9.4. Feed forward neural network architecture.....        | 24 |
| 2.9.6. Feed-back neural network neural architecture.....    | 24 |
| 2.9.7. Input layer.....                                     | 24 |
| 2.9.8. Hidden layer.....                                    | 25 |
| 2.9.9. Output layer.....                                    | 25 |
| 2.9.10. Convolutional layer.....                            | 25 |
| 2.9.11. Padding.....                                        | 25 |
| 2.9.12. Filter.....                                         | 25 |
| 2.9.13. Dimension.....                                      | 25 |
| 2.9.14. Pooling layer.....                                  | 26 |
| 2.9.15. Activation Layer.....                               | 26 |
| 2.9.16. Dropout Layer.....                                  | 26 |
| 2.9.17. Fully Connected Layer.....                          | 26 |
| 2.4. Deep learning based IDS.....                           | 27 |
| 2.4. LSTM.....                                              | 27 |
| 2.5. Combined Approach.....                                 | 28 |
| 2.10. Review of related work.....                           | 28 |
| CHAPTER THREE.....                                          | 33 |

|                                                  |    |
|--------------------------------------------------|----|
| RESEARCH METHOD .....                            | 33 |
| 3.1. Introduction.....                           | 33 |
| 3.2. Research Design .....                       | 34 |
| 3.3. Approaches of the research.....             | 37 |
| 3.4. Network Intrusion Data set.....             | 38 |
| 3.5. Preprocessing.....                          | 39 |
| 3.5.1. Dataset Preparation.....                  | 40 |
| 3.5.2. Data Cleaning .....                       | 42 |
| 3.5.3. Label Encoding.....                       | 42 |
| 3.5.4. Data Standardization.....                 | 42 |
| 3.5.5. Fix Imbalance Class.....                  | 42 |
| 3.5.6. Dataset Splitting.....                    | 43 |
| 3.5.7. Feature Selection .....                   | 43 |
| 3.6. Types of attack in CIC-IDS2018 Dataset..... | 46 |
| 3.6.1. Benign or normal .....                    | 46 |
| 3.6.2. Brute Force Attack.....                   | 47 |
| 3.6.3. DoS based attacks.....                    | 47 |
| 3.6.4. DoS attacks golden eye type.....          | 48 |
| 3.6.5. DoS attacks hulk type .....               | 48 |
| 3.6.6. DoS Attacks lowering HTTP Test.....       | 48 |
| 3.6.7. DoS attacks slowloris .....               | 49 |
| 3.6.8. Brute force web.....                      | 49 |
| 3.6.9. Brute force XSS type.....                 | 49 |
| 3.6.10. SQL Injection types.....                 | 49 |
| 3.6.11. Infiltration.....                        | 50 |
| 3.6.12. DDOS attack types .....                  | 50 |
| 3.6.13. DDOS-HOIC types.....                     | 50 |
| 3.7. Botnet attack types.....                    | 51 |
| 3.8. Preprocessing phase.....                    | 51 |
| 3.9. Preparation of the data set.....            | 54 |
| 3.10. Data set.....                              | 55 |

|                                                              |    |
|--------------------------------------------------------------|----|
| 3.11. Convolution layer .....                                | 57 |
| 3.12. Convolutional Neural Network(CNN) .....                | 57 |
| 3.12.1. Network Architecture .....                           | 58 |
| 3.12.2. Input layer .....                                    | 58 |
| 3.12.3. Max pooling.....                                     | 59 |
| 3.12.4. Reshaping Layer .....                                | 60 |
| 3.12.5. Dense Layer .....                                    | 60 |
| 3.12.6. Drop out layer .....                                 | 60 |
| 3.12.7. Fully connected (FC) layer .....                     | 61 |
| 3.12.8. Output layer .....                                   | 61 |
| 3.12.9. Activation function .....                            | 61 |
| 3.12. 10. Optimization Algorithm.....                        | 62 |
| 3.13. Learning Rate Scheduler.....                           | 63 |
| 3.13.1. Dropout Regularization .....                         | 63 |
| 3.13.2. Model quantization .....                             | 64 |
| 3.14. Classification of network attacks.....                 | 65 |
| 3.14.1. Binary VS Multiclass classification model.....       | 65 |
| 3.15. Model Training .....                                   | 66 |
| 3.16. Model Performance evaluation.....                      | 67 |
| 3.17 .Development tools.....                                 | 69 |
| CHAPTER FOUR.....                                            | 70 |
| RESULT AND DISCUSION.....                                    | 70 |
| 4.1. Introduction.....                                       | 70 |
| 4.2. Datasets.....                                           | 71 |
| 4.3. Pre-processing steps.....                               | 73 |
| 4. 4. Performance measure of the model .....                 | 73 |
| 4.4.1. Hyper parameters of the model.....                    | 75 |
| 4.5. Binary Classification .....                             | 77 |
| 4.6. Multi class Classification.....                         | 80 |
| 4.7. Model Training using Custom Class Weights.....          | 85 |
| 4.7.1. The sampling method of imbalanced Classification..... | 85 |

|                                                              |     |
|--------------------------------------------------------------|-----|
| 4.8. Experiment I : Majority Class Referenced .....          | 86  |
| 4.9.Experiment II using SMOTE with Class Weights method..... | 91  |
| CHAPTER FIVE .....                                           | 95  |
| CONCLUSION AND FUTURE WORK.....                              | 95  |
| 5.1. Conclusion.....                                         | 95  |
| 5.2. Contribution of the research .....                      | 96  |
| 5.3. Future work.....                                        | 97  |
| References.....                                              | 98  |
| Appendix.....                                                | 108 |

## **Abstract**

Now a days, Information and network security become a vital consideration throughout the world. Network intrusion detection (NIDS) is techniques that was used for preventing network attack. Intrusion prevention Systems plays a vital role for detecting connected networks by preventing malicious actors and activities. Various researcher have implemented a research using data mining techniques, statistical analysis and time series, rule based and traditional Machine learning method to detect and classify intrusion detection system. However, the trust made on information asset was growing the exposure of various institution for cyber-attacks that are extremely difficult active and critical. For the sake of protecting organizations information from the risk of various malicious attacks and minimizing its effect is important to confirm the safety of data and also network information. Through implementation of information security technologies, network attack detection become an important requirement. Therefore, one of the most important techniques for information security prediction is the utilization of deep learning approach . Deep learning method improve the limitation of machine learning model. The aim of this thesis is designing and developing a network intrusion detection model using deep learning techniques with high accuracy result. In this research study, we have detected one normally network traffic and 13 network attack by using convolution neural network algorithm and RNN. Our proposed model was implemented model that detects a network attacks using two classification method which is called binary classification and multiclass classification. This research study develops model that detect and predict network intrusion using deep learning method with eight layer architectures. For the aim of network intrusion detection techniques, we have utilized current standard dataset of the Canadian Institute of Cyber security intrusion detection system (CIC-IDS2017-2018). (Hagar & Gawali, 2022)

The model performance of this research study was evaluated using a predictive model outcomes. For implementation of this research study, we have utilized a compressive deep learning model including , CNN,RNN and LSTM . The performance evaluation result of the model were tested with the accuracy=0.985, precision=0.983,recall=0.985, F1 score=0.983, balanced accuracy=.0.816 respectively.

**Keywords:** Network intrusion Detection system, Machine learning, deep learning, cyber security.

## LIST OF TBLE

|                                                                                        |    |
|----------------------------------------------------------------------------------------|----|
| Table 3.2. Attacks Environment of the NID .....                                        | 41 |
| Table3.3. Description of Network intrusion detection Features .....                    | 44 |
| Table3.4. attack and Normal class distribution of data set .....                       | 53 |
| Table 3.5. the portion of data set for network attack class .....                      | 56 |
| Table 3.6 evaluation matrix of the model .....                                         | 68 |
| Table 4.1. network attacks classes with data set distribution .....                    | 72 |
| Table 4.2. hyper tune of the training model .....                                      | 75 |
| Table 4.3. the whole Performance Results of the model with Binary Classification ..... | 78 |
| Table 4.5. Performance Results of the model with Multi class Classification .....      | 81 |
| Table 4.6. network traffic dataset distribution After SMOTE #1 analysis .....          | 86 |
| Table 4.7. Overall Performance Results after SMOTE 1analysis .....                     | 88 |
| Table 4.8 The whole Performance Results of the model with SMOTE 2 1 .....              | 92 |

## LIST OF FIGURE

|                                                                                                        |    |
|--------------------------------------------------------------------------------------------------------|----|
| Figure 2.1. types of IDS system 1 (Bayar B. and Stamm M., 2017).....                                   | 12 |
| Figure 2.2 Intrusion Detection System procedure 1 (Bayar B. and Stamm M., 2017).....                   | 13 |
| Figure 2.6. CNN layer architecture 1 (Qazi, Faheem, & Zia, 2023).....                                  | 22 |
| Figure 2.7 Artificial neural network 1 (Devaraj, 2012).....                                            | 24 |
| Figure 3.1. Proposed architectural model of the this research study 1 (Qazi, Faheem, & Zia, 2023)..... | 36 |
| Figure 3.2.CNN Network architecture 1 (Gamage, 2020).....                                              | 58 |
| Figure 4.1Class Distribution Percentage for attack and benign traffic 1.....                           | 74 |
| Figure 4.3. Training Loss with Binary Classification 1.....                                            | 78 |
| Figure 4.4.confusion matrix of the model using binary classification 1.....                            | 79 |
| Figure 4.5.classification report using binary classification method 1.....                             | 79 |
| Figure 4.6 Training Accuracy with Multi-class Classification 1.....                                    | 80 |
| Figure 4.7. Training Loss Binary Classification 1.....                                                 | 80 |
| Figure 4.8. Confusion Matrix of proposed model with Multi-class Classification 1.....                  | 82 |
| Figure 4.8. Confusion Matrix of proposed model with Multi-class Classification 1.....                  | 82 |
| Figure 4.9 Classification Report with Multi-class Classification 1.....                                | 83 |
| Figure 4.11 with Multi-class Classification 1.....                                                     | 84 |
| Figure 4.12. Training Accuracy for SMOTE analysis 1.....                                               | 87 |
| Figure 4.13. Training Loss for SMOTE analysis 1.....                                                   | 87 |
| Figure 4.14 Confusion Matrix of the model after SMOTE 1 analysis 1.....                                | 88 |
| Figure 4.15. Classification Report for SMOTE 1analysis 1.....                                          | 89 |
| Figure 4.16. ROC AUC Curve after SMOTE #1 analysis 1.....                                              | 90 |
| Figure 4.17. Precision Recall Curve after SMOTE 1 analysis 1.....                                      | 90 |
| Figure 4.19. Training Loss using SMOTE #2 analysis 1.....                                              | 91 |
| Figure4.20. classification report of the model 1.....                                                  | 92 |
| Figure 4.21 Classification Matrix of the model with SMOTE 2 1.....                                     | 93 |
| Figure 4.22. ROC AUC Curve with SMOTE 2 1.....                                                         | 93 |
| Figure 4.23 Precision Recall Curve with SMOTE 2 1.....                                                 | 94 |

## LIST OF ACRONYM

|             |                                                                 |
|-------------|-----------------------------------------------------------------|
| IDS         | Intrusion detection system                                      |
| NIDS        | Network intrusion detection system                              |
| CNN         | convolution neural Network                                      |
| DL          | deep learning                                                   |
| ML          | Machine learning                                                |
| CIC-IDS2018 | Canadian cyber security institute of intrusion detection system |
| RNN         | Recurrent neural network                                        |
| LSTM        | long short term memory                                          |
| Epoch       | training round of the model                                     |
| Batch       | Division of data set into identical group                       |
| ANN         | artificial neural network                                       |
| DOS         | denial of service                                               |
| DDOS        | distributed denial of service                                   |
| Benign      | normal traffic                                                  |
| Attack      | malwares or attacked traffic                                    |
| F1score     | evaluation matrix                                               |
| SIDS        | signature based intrusion detection system                      |
| AIDS        | Anomaly based intrusion detection system                        |
| Accuracy    | measure of training model                                       |

## **CHAPTER ONE: INTRODUCTION**

### **1.1. Background of the study**

A software program or hardware device that sees an eye out of delicious action or policy breaches on a network is called an intrusion detection system (IDS). Usually, a security information and event management system is used to report or gather centrally any malicious activities or violations (J P. Vimalachandran, 2020). Some intrusion detection systems have the ability to react as soon as they find an intrusion (S. Eltanbouly and M. Bashendy, 2020). Monitoring network traffic for its disrupted behavior, is an intrusion detection system (IDS) that generates an alerts when it detects intrusion.

In today's modern world, networks are essential parts of data transferring tool. However, there are several security risks associated with data transfer, therefore in order to protect ourselves from vulnerabilities; we utilize cyber security (Moustafa N & 25:18-31, 2016). Cyber security measures such as firewalls and anti-virus software help to protect an attacks on networks, but they are not powerful enough to identify every new kind of un known attack (Selvakumar K & 35(1):87-107, 2019) The cyber security issue become a recent headache in various sector as the rapid growth and technological advancement increase.

The first intrusion detection system was committed in 1980 (Anderson, Report, & James P. Anderson Company: Philadelphia, 1980) Since then, many research have been done on IDS It has been standard procedure to use traditional machine learning techniques, like random forests (RF), self-organized maps, support vector machines (SVM), and artificial neural networks (ANN), to create NIS. Yet, conventional machine learning methods grow more and more unsuited to handle real-world network application contexts as data sets change in size and kind.

As a result, a lot of academics worked to create IDSs that had lower false alarm rates and higher detection rates. The inability of current IDSs to identify unknown attacks is another issue. (Alok Sharma, 2019)s. Novel attacks and attack variants are always emerging due to the rapid changes in network settings. Various scholars have done using conventional machine learning techniques to detect network issues. One form of artificial intelligence called machine learning is capable of automatically extracting valuable information from enormous datasets. (Bagui, Kalaimannan, Bagui, Nandi, & Pinto, 2019). When there is enough training data available and machine learning models are sufficiently generalizable to recognize attack variants and unique attacks,

machine learning-based intrusion detection systems can reach satisfactory detection levels. (Bhuyan, Bhattacharyya, Kalita, & Springer: Berlin/Heidelberg, 2017). Besides, machine learning algorithm, an intrusion detection systems do not rely heavily on domain specialists' knowledge. Consequently, it is crucial to develop NIDSs that detect unknown attacks using a deep learning-based prediction model.

Among machine learning techniques, deep learning is one that can yield remarkable outcomes. Large data sets are easier for deep learning algorithms to handle than for standard machine learning methods. Additionally, deep learning methods are helpful and end-to-end since they can automatically extract characteristics from raw data and then produce results. One characteristic that sets deep learning apart is its deep structure, which consists of multiple hidden layers. This research study employs a deep learning technique to identify NAIDS.

In this research study, we use the latest publically available data source which are called CSE-CIC-IDS2018 and CSE-CIC-IDS2019 to detect and classify network intrusion . We also focused on data processing method because a lot of data can have repetitive values. Furthermore, we employed various CNN models for network attack detection and classification. Finally, activities involving binary and multi-class classification was performed to determine whether traffic is indicative of a hostile attack.

## **1.2. Motivation of the study**

The networks of today are not completely safe and new security issues arise due to the constant growth of new technologies and modifications to the network infrastructure (H. Altwaijry and S. Algarny, 2020) . The cyber security issue become a recent headache in various sector as the rapid growth and technological advancement increase. So, it is necessary to create various layers of security level safely to solve these problem.

The reason for intrusion detection systems is to remain current and capable of identifying novel threats, they must be regularly improved. However, despite the fact that many efforts done to enhance intrusion detection in creating such systems with a high degree of effectiveness still presents difficulties. Because of this, the primary goal of this thesis will be multi-class detection, which involves several layer of deep learning models to identify type of network attack in computer network. in addition to this, it also simply identify faulty connections (H. Lui and B. Lang, 2020).

The identification and mitigation of security risks as well as anomaly intrusion detection systems are vital to the identification of computer networks. In traditional machine learning models, we can handle massive amounts of data and find trends instantly. But they low in performance to detect intrusion detection systems. NIDS can learn from prior data and identify new trends that can point to possible intrusions by utilizing deep learning algorithms like RNN CNN, LSTM and DNN (Mezina, Burget, & Travieso-González, 2022). This improves the accuracy of NIIDS and lowers false positives. It is very crucial to apply deep learning models in IDS when working with varied datasets (Zha, Liao, Liu, Xie, Tian, & Cao, 2021).

### **1.3. Statement of the problem**

Various Machine learning method offers a capable solution for improving intrusion detection techniques (IDS) (Cuelogic Technologies, 2022) . But, still there are some false positive alarm rate occurred in some networks. A number of publications pertaining specifically to machine learning-based intrusion detection that have been taken from various literature review. However, there are still issues with the current IDSs including, Low processing speed, long training times, low detection rates, and comparatively high false alarm rates (FAR) (S. Cooper, 2022).

Effective IDS development has been the details on multiple attempts in recent era to overcome these problems. But, these methods still have not potential for improvement (Kasongo & Sun, 2020). The two primary shortcomings of the intrusion detection systems in use today are their high false-positive alert rates and low zero-day attack detection rates (Elhanashi, et al., 2023). The complexity, assessment, false alarm rate, and detection rate of both new and known

Earlier attempts at intrusion detection systems tended to overlook threats (Garcia-Teodoro P & 28:18–28, 2019). The fact that, about 8% of the worldwide businesses today are unaware of their current cyber threats or alerts is also quite fascinating (Gamage, 2020). This indicates that either they have not been subjected to any cyber-attacks that make sense in light of the current cyber-attack scenario, or they lack the ability to detect cyber-attacks (Khraisat, 2020). Considering the aforementioned conclusions, it is critical to use appropriate technology and methodologies in order to improve cyber threats visibility and detection.

organization networks' safety systems is developed in tandem with technical progress in order to fortify network security defenses. The purpose of this study is to develop a predictive model for network intrusion detection using deep learning techniques using the CIC-IDS2017 and CIC-

IDS2018 sample datasets from the Canadian Institute for Cyber Security. A new dataset has to be used in order to address the recent significant advances in internet traffic variety and developing attack types. We require intrusion detection methods that are capable of learning and efficiently identifying intrusions to solve these risk. Many scholars have offered a traditional machine learning methods. These techniques benefit from the single detection.

All previous researchers uses Canadian Institute for Cyber security intrusion detection system (CIC-IDS2017) datasets which was only restricted to USA IDS. In fact, this distinction can be decreased the whole detection amounts as the inadequacies of one strategy might be remedied by another. Therefore, the goal of this research study is to develop a model that uses deep learning techniques to detect NIDS in order to improve the prediction model's detection accuracy.

#### **1.4. Research question**

1. What are the common cyber-attacks affecting the normal functioning of network?
2. Which deep learning model we utilized for network intrusion detection?
3. To what extent deep learning model perform network intrusion detection ?

#### **1.5. Objectives**

##### **1.5.1. General objective**

The general objectives of this study is to develop deep learning model for network intrusion detection using CNN model.

##### **1.5.2. Specific objective**

To identify network attacks that affect the normal functioning of a network.

To develop a predictive model using deep learning techniques for NIDS.

To compare different deep learning model for detecting NIDS.

To measure the performance of deep learning model for network intrusion detection using CNN model.

#### **1.6. Scope of the study**

The proposed study is focused on designing and implementing model for detecting network intrusion using deep learning method. This research study aims for classifying and detecting a potential network attacks. It also evaluate the model efficiency to detects an attacked and normal network traffic, but not to avoid and eliminate the attacker from being getting the network. This

research study was mainly concerned on possibly classifying and detecting timely cyber-attack, and inform the risk of that attack to the security department or Administrator. In this work we have increased the rate of NIDS detection through decreasing the false positive ratio. Network intrusion data set was collected and used from on line publically available dataset. However, most of the dataset that we have used in this research work were obtained from various local organization, enterprise, bank, telecom as well as real log files. Most of the dataset that we have used for network intrusion detection was used from international web attack standard called CSE-IDS2018. This study focused on attacker type classification and cyber-attack risk factor. however, since the confidentiality of local network data are based on local and current IDS log files, we prefer the standard and most current network traffic data from Canadian institutes for cyber security ( CSE-IDS2018).

### **1.7. Limitation of the research study**

The fact that the dataset for this research was obtained from the free open publically available website of network traffic dataset called Canadian Institute for Cyber security means that the trained model cannot be employed directly to a particular organization's network, which is one of the limitations of this work. Because each organization has a unique network configuration and equipment. Additionally, a lengthy computation time is required for the grid search with cross validation process.

The other limitation of the research work was the implementation was done only by using deep learning model than ensemble learning approach to detect network intrusion including the designs of cyber security attack. The model was developed only to detect and classify the types of Network-attack and predict future behavior of the cyber security for an organization. The model not build a prototype system that uses GUI in supporting cyber security measure.

### **1.8. Contribution of the research**

This research employs the colab and jupyter one note book GPU to speed up the training model. With an emphasis on data preprocessing, we employed the entire CSE-CIC-IDS2018 dataset to represent the actual network traffic situation in our tests and produce thorough test results. To address binary and multi-class classification job, we implemented the CNN, RNN and LSTM models. By using the suggested suitable techniques for preprocessing data and methodically

adjusting the hyper parameters of all three models, as it was discovered that the accuracy of each CNN,RNN and LSTM model exceeded 97%. The suggested model effectively enhances the detection performance when compared to the IDS of other works.

On other hand, to the empirical implementation of the research , the CNN models combinations have longer inference times than the individual CNN model. Taking into account how the algorithm is implemented using NID data set, we determine that each individual CNN algorithm.

### **1.9. Significance of the study**

The implication of this research is detecting an intrusion using deep learning method . Different entities like, data administrator, network Administrator and system administrator could use the application of the model as a bench mark to detect the possible intrusion detection system. It also assists the identification techniques to be active rather than reactive. The major significance of the research study includes the following areas. The bank enterprise, telecom enterprise, governmental sectors like, financial office, revenues, investment bureau. These entity can use the model to detect and predict the network security issues.

- ✓ Decreases cyber security risk, in providing knowledge about the behaviors of users to protect organizational resources from damage as early as possible by applying intrusion detection system based on specific information.
- ✓ Decreasing the staff fee and adjustment of the system using SIEM method.
- ✓ Minimize cyber security attack and its cause including, financial, political, and social problem.

## **CHAPTER TWO**

### **REVIEW LITRETURE AND RELATED WORK**

#### **2.1. Overview of intrusion detection system.**

The primary focus of this chapter's literature analysis and description of the basic intrusion detection systems (IDS) employing deep learning techniques with convolution neural networks, recurrent neural networks (CNN, RNN).

Network assaults are defined as a collection of malevolent actions intended to interfere with, prevent, deteriorate, or eliminate data and services that are present in computer networks. Network attacks are carried out via network data streams with the intention of jeopardizing the availability, confidentiality, or integrity of computer network systems.(V. Kanade, 2022) Computer attacks can take many forms, such as viruses that are attached to emails, internet worms, probing systems to gather information, unauthorized use of systems, denial of service attacks that abuse system features, or exploiting software bugs to change system data. (Anderson, Report, & James P. Anderson Company: Philadelphia, 1980). Intrusion detection is the technique of keeping an eye on computer events and network traffic to advert unlawful or malicious activity. An intrusion detection system (IDS) is any hardware or software program which aims to perform an intrusion detection (A. Yadav, 2020).

There are numerous attack detection systems that have been developed and are widely used that examine network data for any deviation from a system's or user's typical behavior.(C. Team ". 1., 2022). Hackers have established many mechanisms ranging from simple to sophisticated techniques to continue their criminal acts (A. Khraisat, 2019). Sometimes, intrusion prevention systems (IPSs) and firewalls tools are confused with intrusion detection systems (IDSs). Although they employ distinct techniques, these three security procedures are intended to safeguard a systems inside a network. To prevent an attacks before they reach the secured network, firewalls scan the outside world for them. They use packet header analysis to filter incoming and outgoing traffic according to preset criteria (IP address, protocol and port number (S. Cooper, 2022). However, intrusion detection systems (IDSs) can keep an eye on activity both within and outside of the secured network. IDSs are also limited to monitoring, unlike firewalls, and are consequently dependent on an administrator or intrusion prevention system (IPS) to

handle their notifications (W. Brown and . L. Stallings, 2018). IDSs are unable to prohibit suspicious activity.

In order to assess and protect computers, networks, programs, and data against attacks, unauthorized access, illegal read/write, and deletion/corruption intrusion detection systems techniques and applications. are used (Kasongo SM & 7(1):1–20., 2020). They may be categorized as host or network, depending on where and how it is implemented. It is important to distinguish between intrusion detection systems and other security features of a network or system, including firewalls and antivirus programs (A. Mouhammd, 2016). A kind of computer security management system is an intrusion detection system, or IDS. It is an attack prevention techniques. A detection system looks for possible security vulnerabilities treats by gathering and analyzing data from specific areas of a network or computer. (Springer, 2017).

## **2.2. History of Cyber Attacks**

The fact that, the first cyber-attacks occurred in the 1970s, when the majority of the people do not use and access the computers resource (Ferrag, 2019). The person named as Bob Thomas developed the computer program called Creeper in 1972 as an experimental tool for the ARPANET (The Advanced Research Projects Agency Network) research project (Singla, 2018). Then after, the person Ray Tommy, invented an email, and created the program Reaper to find and remove Creeper. Because of its ability to replicate itself, this program which is referred as the first computer worm and antivirus software (Morris, 2018).The amount of information stored online increased between 1990s and 2000s as the public's access to and use of the internet grow. Due to this situation, numerous attackers were drawn to steal data from institutions, governments, and individual people by making use of the internet's recent influence (Khraisat, 2019). However, as people began to place greater value on cyber security, a NASA researcher have developed the first firewall program. In addition, antivirus software was being produced in large quantities to address these novel security threats. But , these new security measures were only able to provide protection up to a certain point because attackers were using new strategies and tactics (Leevy, 2020).

## **2.4. Types of intrusion detection**

It makes sense that actions taken against the information system's security policy are considered intrusions, and the process of detecting intrusions is known as intrusion detection. The field of

intrusion detection has been explored for around 20 years. It is predicated on the ideas that an intruder's behavior will differ significantly from a legitimate user's and that many acts taken without authorization will be visible (Garcia-Teodoro P & 28:18–28, 2019). Intrusion detection systems fall into three different categories. Some of these system types are host-based intrusion detection, network intrusion detection, misuse intrusion detection, specification-based intrusion detection, anomaly-based intrusion detection, and signature-based intrusion detection. (] A. A. Aburomman and M. B. I. Reaz, 2018).

#### **2.4.1. Signature-based intrusion detection systems (SIDS)**

Traditional signature-based intrusion detection systems (S-IDS) use pattern matching techniques, also referred to as knowledge-based detection or misuse detection, to identify known attacks. These systems learn attack or misuse patterns from past activities and use the knowledge they have gained to identify similar patterns. Signature refers to the acquired attack and misuse patterns. Attack patterns are defined using both user-defined and pre-defined signatures. S-IDS generates an alert when malicious network traffic matches the corresponding signatures (Murphy, 2020).

The purpose of a signature database is to maintain detection efficiency by continually updating it with the latest attack types and known signatures of known attack types. Existing activities are compared using the signature database and an alert is sent upon finding a match. This approach offers exceptional detection accuracy and highly suitable for known attacks.

#### **2.4.2. Anomaly based intrusion detection systems**

A normal model of the behavior of the associated system is constructed using statistical, knowledge-based, or machine learning techniques. Any discrepancies between the model and the observations are considered anomalies that point to an incursion. The fundamental idea behind this strategy is that malicious user traffic should be separated from genuine user traffic, which can be generated by mimicking legitimate user behavior. Attacks like insider and zero-day vulnerabilities can be found more readily this way. This method's primary problem is raising false positive alerts. Both HIDS and NIDS can benefit from the application of A-IDS techniques to improve their detection capabilities.

It is capable of identifying undiscovered dangers. But, it also produces a high volume of difficult to manage false positives. Additionally, because the breach lacks a distinctive signature,

information about it is more difficult to obtain the anticipated system and network behavior is simulated via behavior-based detection. The administrator is notified when the behavior departs from the predetermined threshold. Since there are no rules need to be defined these methods in contrast to signature based detection, there may be able to advert zero-day attacks. It is more challenging for attackers to comprehend the capabilities of IDSs when this detection method is used.

We chose and used the most widely used deep learning model, the convolution neural network (CNN), which has multiple architectural layers, including input, hidden, and output layers. This research study focused on deep learning-based intrusion detection systems to enhance the conventional machine learning and data mining techniques in detecting network cyber-attacks using multiple deep learning models, including CNN, DNN, RNN, and DBN.

### **2.4.3. Hybrid based detection**

In order to reduce the vulnerabilities of each category, hybrid detection combines these two methods as abuse detection related to anomaly based detection, anomaly detection followed by misuse detection, or both at the same time. Identifying unexpected intrusions by anomaly detection and detecting known attacks using their signatures are the objectives.

Hybrid detection techniques sometimes referred to as specification based detection. It is a novel kind of detection made possible by deep learning techniques that integrate behavior and signature based detection. This combination reduces false negative and false positive alarm rate which in turn reduces the previously mentioned detection gaps. Research indicates that utilizing deep learning algorithms can enhance the identification of cyber-attacks and is regarded as a feasible approach to improve cyber security.

### **2.4.3. Network based IDS (NIDS)**

Using sensors spread throughout the network is a standard practice for network-based intrusion detection systems (NIDS). The beam analyses the traffic locally, or a central controller analyses it remotely. NIDSs are more frequently employed to protect an organization's IT assets than HIDSs since they are more scalable and cross-platform. Nevertheless, combining these technologies could offer a higher level of security (A. M. Z. a. M. J. A. A. Qais Saif Qassim, 2016).

One of the roles of a network-based intrusion detection system (NIDS) is to monitor and manage network traffic. By detecting harmful or unauthorized access to a local area network and keeping an eye on traffic between different hosts on the wire, it is primarily responsible for protecting the system against network-based threats. An NIDS alert notifies the administrator by using detection algorithms that look for any abnormal patterns in both incoming and outgoing packets. (M. S. a. Anuradha., 2012).

#### **2.4.5. Application based IDS**

Applications are monitored by a form of intrusion detection system (HIDS) known as an application-based intrusion detection system (IDS). IDS involves looking at logs, authorizations, file executions or modifications, and any other possibly suspicious activity. Their ability to generate user profiles enables them to identify unusual behavior. As a go-between for the application and the encrypted data for storage, application-based intrusion detection systems can also access data prior to encryption.

#### **2.4.6. Misuse detection (MD)**

Some of the existing MD techniques rely on the use of expert systems (ES), state-transition analysis (STA), signature analysis (SA), and data mining (DM). Regarding expert systems, they rely on a set of rules for the description of intrusions. The intrusion detection system (IDS) in this category looks for known attack patterns to detect intrusion as applied in the current commercial needs. However, the problem with MD is the inability to detect new attacks (S. M. H. Bamakan, 2016). An expert systems analyses audit events using a set of transitions and goals using a state transition diagram. An events are deemed intrusive if they cause an intrusion stat. An expert systems then translate audit events into facts that convey their semantic relevance based on these rules and facts and then conclusions are drawn using an inference engine (Y. Zhang, 2019).

Recent developments in intrusion detection systems (IDSs) have focused on the extraction of previously undiscovered and valuable patterns from massive data sources through data mining processes. These patterns can be displayed as rules, decision trees, neural nets, or instance-based examples.

## 2.5. Challenges of intrusion detection System

New trend for automatic classification of network intrusion detection are the topic of ongoing thesis. IDS also revealed the implementation of an intrusion-detection model that was proposed as a foundation for all-purpose of IDS. Numerous network ID automation algorithms have been created and put into use by experts since then. Furthermore, they've For this purpose has been a constant search for more accurate, quicker, and scalable methods. With the emergence of big data and the Internet of Things, it is projected that by 2020, there will be more than 26 billion linked devices (Al-Fawa'Reh, Al-Fayoumi, Nashwan, & Fraihat, 2022). It is projected that as a result of this trend, there will be a rise in the variety and volume of cyber security problems. An summary of the challenges in IDS is given in Figure 2.

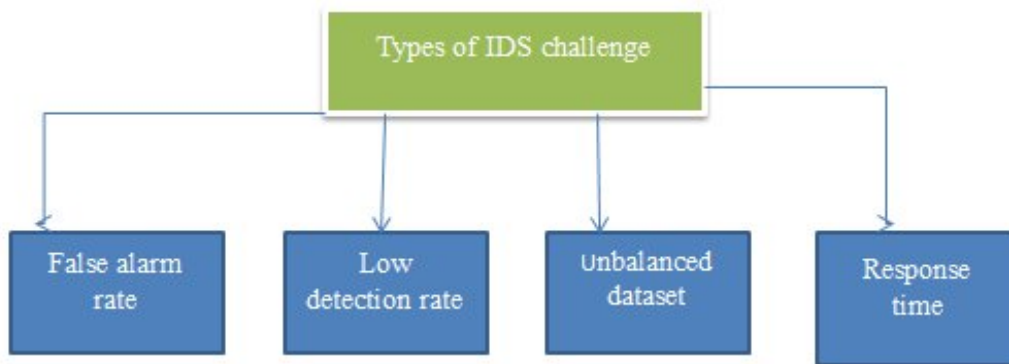


Figure 2.1. types of IDS system 1 (Bayar B. and Stamm M., 2017)

More IDS system has been proposed recently by various research. For example, (M. K. Khaleel, 2018) proposed that IDS should be further classified into five subcategories that might be included in any of the previously described classifications. The various subclasses of intrusion detection systems that are recommended are rule-based, pattern-based, state-based, statistics-based, and heuristic-based. Such a classification could cause mistake because there are many similarities between the strengths of the numerous procedures and there are no distinct criteria that set one technique apart from another. The FPR of a signature- or rule-based intrusion detection system (IDS) is usually low, but it cannot identify novel forms of intrusions. A low FPR detection rate is necessary for a system intended to identify anomalies. Depending on how well a profile is defined, ID systems using state full protocol analysis perform differently in terms of detection. Updated attack profiles are a major difficulty for this strategy since new protocols appear over time.

Establishing network based intrusion detection system with low false positive detection rates and high accuracy is the primary goal of this research study.

## **NIDS architectural part**

The generic part of network intrusion detection system architecture include, data gathering sensors, detector engine and storage module.

### **Storage Module**

It contains the sets of IDS rules that the detector uses to compare the data it has received.

### **Data gathering sensors**

They are employed to keep an eye on specific behaviors or processes as well as the infrastructure used for data collecting. They create the main classification using the information gathered from their site.

### **Detector module**

The collected data is compared to the predetermined set of rules in this module. IDS detects a deviation from the normal state and sounds an alarm.

### **Action**

An alarm's sound causes a predetermined action to happen. Depending on the kind of notification, there might involve the IDS carrying out a predetermined task, such eliminating the malicious packets. There are instances when a passive approach is suitable, like recording the conduct and allowing the human element to choose the course of action.

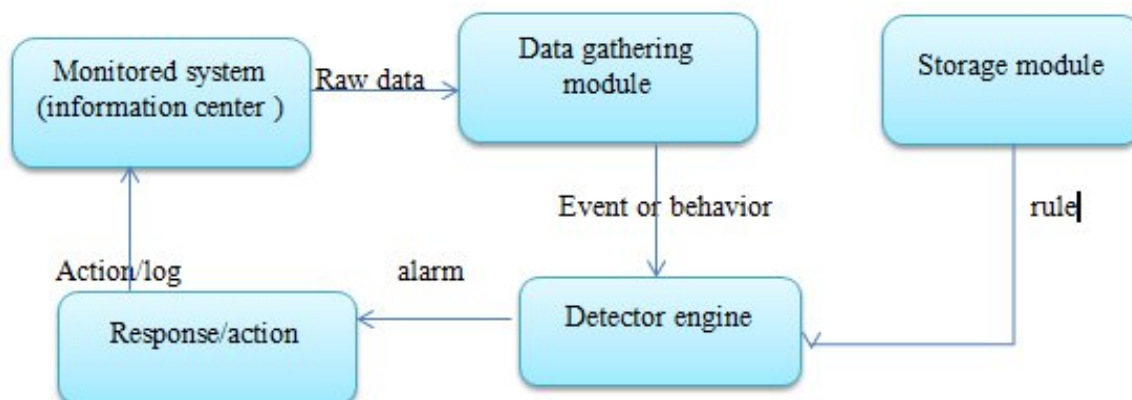


Figure 2.2 Intrusion Detection System procedure 1 (Bayar B. and Stamm M., 2017)

### **2.5.1. Impact of Network Intrusion**

Companies who experienced data breaches are faced more and more severe consequences. The main reason for this is the growing regulatory load brought on by informing people whose personal information has been compromised. Each influence has both domestically in the US and Canada and international wide (A. Daly, 2018).

Company that has different requirements for notice and penalties for businesses caused by data breach. In the event of a consumer data breach, businesses need to identify the jurisdictional regulatory body and the residential address of their customers (A. Makkar, 2018). Regulations outline the kinds of information that need to be reported after a breach, along with who needs to be notified, how, and whether or not to notify certain authorities. Breach of personal, financial, or health data typically carries notification requirements, though state-specific definitions differ. Businesses that operate internationally may have clients in several legal countries and be subject to a multitude of legal requirements. The expenses of this procedure, along with any associated lawsuits, fines, and possible damages reimbursement, may be sufficient to force some businesses out of business (M. Alazab, 2021).

### **2.6. Traditional Machine Learning Approaches**

The use of machine learning algorithms for intrusion detection is common. AI-based intrusion detection systems (IDS) continue to be a popular research field, with encouraging outcomes compared to SIDS constraints (InTech, 2010). Many scholars are still investigating algorithms and their combinations.

Huge number of scholars continue to study various algorithms and their combinations. Machine learning, statistics, and knowledge-based methods are used to construct a baseline that represents the typical behavior of the related system. Any break from this paradigm is considered an abnormality (or intrusion.) ([11] C.-J. Lin, 2012). To build a model that replicates baseline behavior, the model is trained to comprehend the typical traffic profile. The trained model is next tested using new data that is completely different from the data used in the training phase. Lastly, the overall performance of the model is evaluated by comparing the expected and actual results. ([ Daniel Jurafsky & James H. Martin, 2016).

By performing essential computations and choices, such as identifying the kind of packets in the flow, machine learning algorithms can By carrying out necessary calculations and decisions, like

determining the type of packets in the flow, machine learning algorithms can improve network security to enhance network security. The classification of machine learning algorithms will be discussed in the next section. This thesis explores deep learning techniques, a subset of machine learning, to find network intrusion detection systems. (Bertoli, et al., 2021). The significance of deep learning's capacity to learn intricate data representations has also increased recently. Through the modeling of high-intensity network data, this capability holds great promise for accurately identifying today's complex attacks.. (Javaid, Niyaz, Sun, & Alam, 2016).

The broad topic of machine learning is touched upon by many academic fields, such as information technology, statistics, probability, artificial intelligence, psychology, and neurobiology.. (Panwar, Negi, Panwar, & Raiwani, 2017). By creating a model that accurately represents a chosen dataset, machine learning may readily resolve problems. Statistics has been able to develop into a comprehensive discipline that generates fundamental statistical computational theories of the learning process because to machine learning, an advanced field that started with teaching computers to mimic the human brain. (Singh Panwar, Raiwani, & Panwar). The main goal of machine learning is to develop algorithms that let computers learn. One phase in the learning process is identifying data patterns or statistical regularities. Machine learning algorithms are developed in order to mimic how humans learn a task. (Osisanwo, 2017).

Since machine learning algorithms require an even distribution of classes, class imbalance is a prevalent problem, especially in network intrusion detection datasets. Zhang, Y. (2019). In practical applications where it's crucial to accurately identify minority classes (attacks, in our case), managing imbalance is essential. This issue can be addressed in a number of methods, such as random under sampling and oversampling.. (Halimaa, Sundarakantham, & Scholar, 2019).

### **2.6.1. Types of machine learning**

The field of machine learning is growing quickly and has permeated everyday life. ML is used, for instance, to detect persons in photos on social networking platforms like Facebook or to personalize ads on websites. The process by which machine learning algorithms generalize existing data is called supervised learning. There are two types of machine learning algorithms: supervised learning and unsupervised learning. ML algorithms come in two varieties: supervised learning and unsupervised learning. Enhancing a conventional machine learning model to make

it generate erroneous predictions is the aim of adversarial machine learning (AML). One way to achieve corruption is by model contamination or dataset alteration (Tianyu Gu, 2019).

### **2.6.2. Supervised machine learning algorithm**

For supervised algorithms, class data must be fully labeled. Training and testing datasets must be maintained separate in order to use supervised machine learning algorithms for network intrusion detection. The main objective is to use labeled data to train algorithms in order to create a model. The "unknown" in the test data is then predicted using the trained model.. (T. A. Tang, 2018). Supervised learning can only be used to detect known attacks; zero-day attacks are not detectable. (Tom Mitchell, 2015).

The goal of this classification is the creation of an SML. Its objective is to forecast discrete values that fall into a specific class. The results' value is discrete using labeled datasets, regression sort of SML that predicts continuous-valued outcomes for the most recent data that the algorithm has access to data. (Nasteski, 2017).

### **2.6.3. Unsupervised machine learning algorithms**

In unsupervised learning, clustering techniques are used to create a model from unlabeled data that can differentiate between network or host logs and malicious inputs. Unsupervised methods randomly analyze the data attributes based on their statistical characteristics and don't require any prior knowledge. (S. Eltanbouly and M. Bashendy, 2020). Providing evidence that unsupervised approaches do not necessitate the laborious data training stage. (D. Krotov and J. J. Hopfield, 2019).

It was researched on how intrusion detection systems truly cope with the concerns including real-time detection, low detection accuracy, and irregular detection rates. It was important to build hybrid machine learning algorithms to handle these problems (H. U. Dike, 2018). In order to improve system performance, hybrid intrusion detection approaches integrate supervised and unsupervised machine learning techniques, with the primary goal of reducing false-positive and false-negative alerts. (K. Kottmann, 2020).

### **2.6.4. Semi supervised learning ML algorithm**

Semi-supervised learning is a branch of machine learning that lies between supervised and unsupervised learning. is a method of developing a model that makes use of a lot of unlabeled

input and very little recognized data. (Berthelot, 2019). But the objective of semi-supervised learning is the same as that of supervised learning: to create a function that can reliably predict the output variable given the input variables.. (Chen K. &.-s.-s.-1., 2019). Unlike supervised learning, semi-supervised learning uses a dataset that includes both labeled and unlabeled data for training. Combining the two objectives is the purpose of the machine learning algorithm subsection known as semi-supervised learning. SSL aims to overcome the limitations of both supervised and unsupervised learning.

### **2.6.5 Artificial neural network (ANN)**

While decision trees and neural networks have been utilized extensively in IDS in the past time various researchers try to improve IDS by applying these two techniques to both SIDS and AIDS (Sharma & Kumar, 2016) (Song, 2018). A decision tree was used to identify different assaults, while a neural network was employed to improve the detection of known attacks. The three-layered structure made use of a neural network learning process called back propagation. Two models of network-based intrusion detection and prevention were run using distinct KDD 99 data sets (Elkan, 2016). DoS, probing, U2R, and R2L appended with one normal class were the four attack classes in the KDD 99 contest. Each of these classes was represented by five neurons on the output layer.

Additionally, following the experiment, the neural network performed better than previous studies, however it was unable to differentiate between R2L and U2R attacks, while the other two were identified (Abiodun & Arshad, 2018). Consequently, it is improved by using the decision tree.

## **2.8. Deep learning techniques for NIDS**

A subfield of machine learning and artificial intelligence is called deep learning. As was indicated in the part before this one, it began in the early days and has since spread globally. The development of artificial neural networks marked the beginning of training deeper neural networks. (Yang, 2020). This new method, known as deep learning, has exposed extremely boosting results in a number of fields. Anomaly detection is among the most popular uses of deep learning systems. As stated in (Vinayakumar, 2019), Because deep learning can learn hierarchical feature representations and sequential linkages of more complex datasets with the help of hidden layers, it is very effective for network intrusion detection. Several other fields,

such as image processing, machine translation, voice recognition, and natural language processing, also make use of deep learning.

Neural network architectures, which consist of several layers of neurons and are classified based on their architecture, learning method, and activation functions, are at the heart of deep learning. (Amini et al., 2006). A three layer fully linked neural network is depicted in Figure 2.9. The neurons in the input layer, on the left for instance are referred to as input neurons. The output layer, which is the layer on the right, is made up of just one output neuron. The hidden layer is the layer that connects the input and output layers. This architecture may be effective for relatively simple jobs, but in order to address more complicated problems and expedite the design process, we must break down the main problem into smaller ones. Deep Learning (DL) is derived from machine learning techniques that use a hierarchical series of successive layers of information processing stages to learn features or representations and classify patterns (S. Eltanbouly and M. Bashendy, 2020).

The capacity of deep learning approaches to learn the content of deep structures and produce the required output has demonstrated their superiority over shallow learning techniques. Deep learners are able to extract a better representation from feature sets than other shallow learners since they are made up of numerous hidden layers, which allow them to train a larger dataset more effectively. Comparing traditional machine learning techniques to deep learning techniques, some researchers' earlier efforts in prediction, classification, or clustering have not produced particularly impressive results. A research gap is the rise of cyber-attack cases in computer networks in spite of the application of deep learning techniques in intrusion decision systems. Furthermore, the DL hybrid-based approach to successful model creation has not been applied correctly. This research report offers a summary of the deep learning approach to network intrusion detection.

### **2.8.1. Deep learning architecture**

It is anticipated that the deep learning-based architecture will help overcome the challenges associated with developing an effective NIDS. Thanks to developments in deep learning algorithms, intrusion detection systems (IDS) have traditionally been able to attain a higher detection rate and a lower false alarm rate. Furthermore, there are three different kinds of DL architecture. (Y. Zhou, 2018). In particular, supervised and unsupervised learning form the basis

of hybrid deep learning architectures. Using generative learning or unsupervised deep learning is the most effective way to obtain features for the entire model. (A. Salih, 2021). Here, the number of unlabeled data points exceeds the number of labeled ones. Discriminative learning, another name for supervised deep learning, is mostly used for feature identification and classification, using labeled data for classification. (N. Hallett, 2020). Lastly, CNN and discriminative applications are used to train IDS utilizing a hybrid deep learning architecture that combines supervised and unsupervised learning approaches.

According to this study, we used the evaluation of DL architecture with numerous hidden layers and a subset of ML algorithm which has several features (M. Zebari, 2019). It is possible to think of DL as a key for feature selection and classification techniques and stated differently with a vast amount of IDS datasets that can be used by the DL architecture to discover deep patterns (D. Nehra, 2019).

Single-layer designs are usually characterized by a non-linear system that is controlled by linear data. Shallow architectures include logistic regression, support vector machines (SVM), single-layer Multiple Layer Perceptions (MLP), and others. According to various authors and scholar, deep learning has been more popular among researchers many recently for a number of reasons, including its adaptability to fresh issues, its ease of use in feature selection, and its capacity to handle large data spaces.

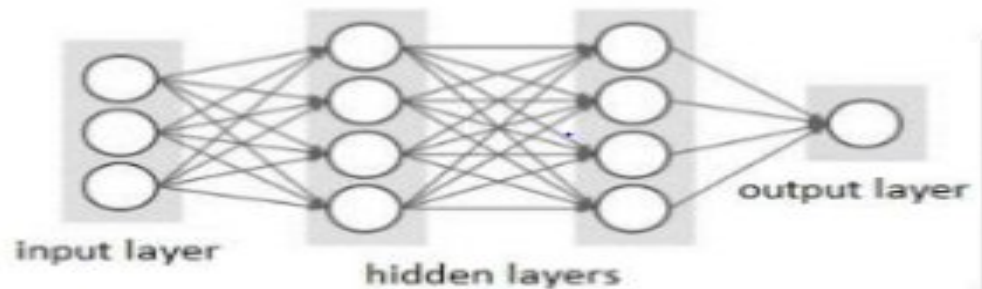


Figure2.3.Artifitil Neural network architecture 1(K. Bekshentayeva, 2019)

The most often utilized DL approaches in this research study are examined in detail. The previous scholar s and researcher discuss various techniques for network intrusion detection and their architecture, including Recurrent Neural Networks (RNN), Long Short-Term Memory-

Convolutional Neural Networks (CNN), Auto-Encoders (AE), Deep Belief Networks (DBN), and Recurrent Neural Networks (LSTM-RNN). (K. Bekshentayeva, 2019). To detect Distributed Denial of Service (DDoS) attacks on a network, researchers constructed an LSTM model using the Singular Value Decomposition (SVD) architecture.) (C. C. Ugwu, 2020). LSTM was used to identify traffic from DDoS attacks, and SVD by utilizing extracted features from network data. According to Loukas (G. Loukas, 2017), the LSTM techniques uses gates functions like, the input and output gates instead of activation functions to help improved the problem of gradient failing or exploding.

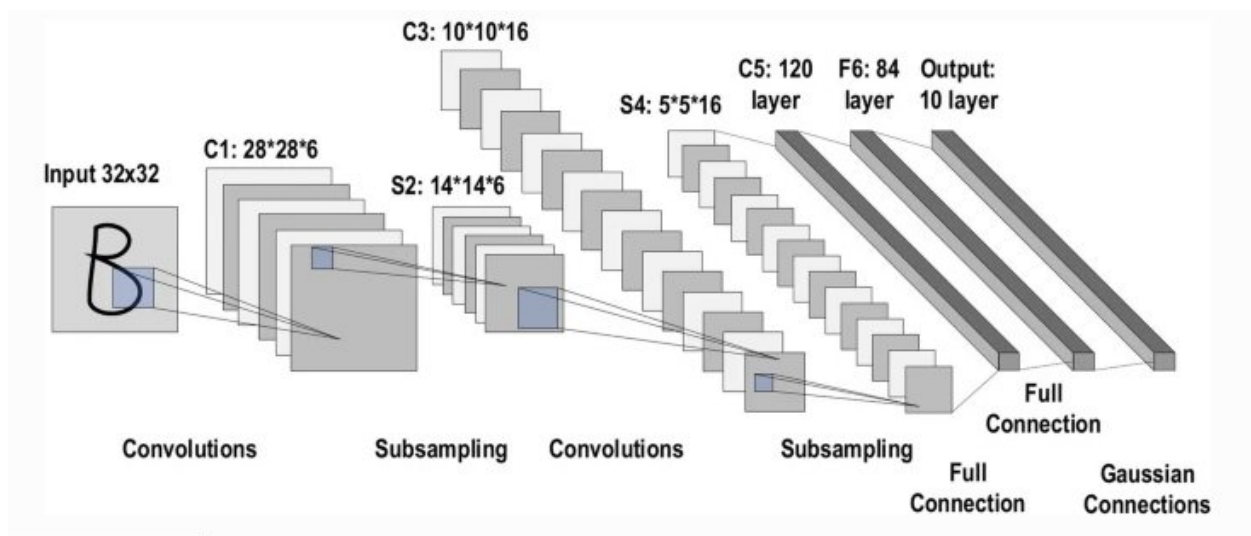


Figure2.4 deep learning layer architecture 1 (Lo WW & 1–5., 2019).

Intermediate or hidden layers are employed in multi-layered methods to create a deeper network. As we continue to add hidden layers, the architecture becomes more complex. Three hidden layers make up the deep neural network design, which is fully connected at the end, as seen in the image below.

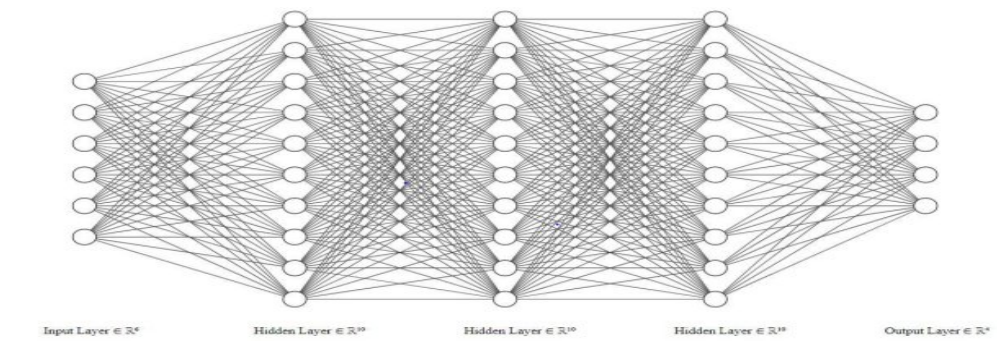


Figure2.5 Deep Neural Network with 3-Hidden layer architecture 1(Wang T & 6:324., 2020).

## 2.9. Types of deep learning algorithm

According to (Kwon, 2019), the three categories of deep learning approaches include, Deep networks for generative or unsupervised learning such as Deep Belief Network (DBN), Deep Boltzmann Machine (DBM), Restricted Boltzmann Machine (RBM), Auto encoders, and Deep auto encoders. Recurrent neural networks (RNN) and convolutional neural networks (CNN) are included in the second category of deep networks, supervised learning. hybrid deep network involves the third kind of deep network which consists of two or more deep learning algorithms.

### 2.9.1. Convolutional neural networks (CNN)

The most popular and widely used algorithm in deep learning techniques is CNN. When compared to its predecessors, CNN's primary advantage is its ability to automatically identify pertinent features without human oversight. (u J & 77:354–77., 2020). CNNs are extensively used in many different fields, including computer vision, and function similarly to a regular neural network. They are based on the neurons that are present in the brains of both humans and animals. The CNN specifically imitates the complex cell pattern that constitutes the visual cortex of a cat's brain. (Li HC & 20(21):6114., 2020). Unlike typical fully connected (FC) networks, CNNs utilize local connections and shared weights to completely utilize 2D input-data structures, including picture signals. This allows CNNs to fully optimize 2D input. This method uses a remarkably minimal number of parameters to speed up the network and simplify the training process. The local correlation present in the input, including local filters over the input, is thus spatially extracted by these cells. Specifically, these cells do not perceive the entire scene, only fragments of it. Many convolution layers come before sub-sampling (pooling) layers, while FC. An example of CNN architecture for IDS classification is shown in below figure Figure 6 Feature Extraction and & Classification using Convolutional Neural Network (Tian H & 22(5):1053–66., 2020).

In the CNN model, the input elements of each layer are arranged in three dimensions: height, width, and depth, or  $h \times w \times r$ , where the width equals the height ( $h$ ). The CNN architecture's channel number corresponds to the depth. (Yao G & 118:14–22, 2019). The depth ( $r$ ) of any RGB image, for instance, is three. Similar to the input data, each convolutional layer has a large number of kernels (filters), each of which is represented by  $k$  and has three

dimensions ( $n \times n \times q$ ). Thus, if  $q$  is equal to or less than  $m$ , then  $n$  should be smaller than  $r$ . (Shrestha A & 7:53040–65., 2019).

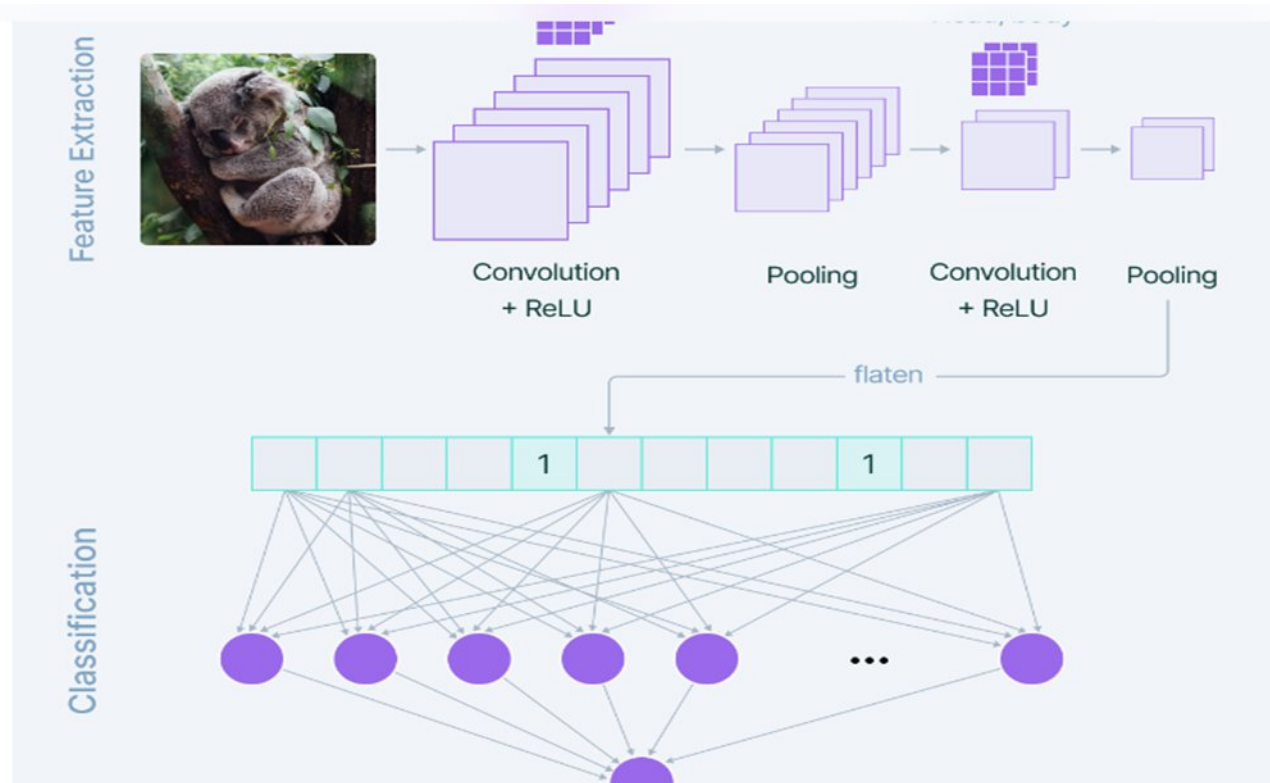


Figure 2.6. CNN layer architecture 1 (Qazi, Faheem, & Zia, 2023)

## .9.2. The role of Deep Learning for NIDS

In a similar vein, various machine learning methods make a decision after comparing new data and input to the training database. Deep learning, on the other hand, uses neural networks to make inferences on their own without the need of labeled training data. This method can be used by autonomous cars to distinguish between a signboard and a pedestrian. The output of one algorithm in a neural network depends on the outcome of another algorithm that uses algorithms that are next to each other in the network.. The system can therefore make decisions just like people do

Using conventional methods like machine learning and manual identification to extract characteristics from large datasets is difficult. However, deep learning techniques are more dependable and effective than machine learning techniques when used to large image datasets

because they can recall data from an infinite number of layers (IEEE & 1–2., 2018). One disadvantage of using deep learning architecture is the inclusion of redundant features.

### **2.9.2. Use of artificial neural network (ANN) in NIDS**

In a number of applications, ANNs have been created to produce mathematical models that increase agricultural computing capacity. Without prior knowledge of the functional links between variables, ANNs allow us to construct models based on the fundamental interactions between them. (<https://github.com/lanjelot103>, 2020). Conventional machine learning methods that use artificial neural networks (ANNs) have been widely employed to create models to forecast different network failures like viruses, hacker attacks, and denial-of-service attacks (DDOS), which reduce an organization's commercial significance and accessibility. (Leevy, 2020). Artificial neural network models offer a major advantage when applying mathematical reasoning to situations where the relationships between the dependent and independent variables are unknown since they are designed to resemble human thinking. Two ANNs were developed through a number of experiments to mimic serial hackers and the legitimacy of IDS across different system admin offices or sectors.

### **2.9.3. ANN architectural layers**

The Levenberg Marquardt ([10]C.Jirapummin, 2010) algorithm was used to train and validate two feed-forward back propagation having two output layers with adjustable weights and bias with three hidden layers, and an input layer (Explanation., 2020). Eight ANN neurons with the average values of eight known attack area replications made up the input layer. Both regular and some targeted network resources comprised the output layers. To find the optimal network design for each illness diagnosis system, several ANN architectural arrays with different numbers of neurons in the three hidden layers were examined. (Sherstinsky, 2020).

The Min-Max method was used to normalize the input and output data sets. Neuron numbers ranging from 5 to 10, 4 to 10, and 1 to 5 were selected at random in the three hidden layers to create different topology arrays (Vinayakumar, 2019). In the majority of cases, The neuron layers are determined by trial and error (Yu, 2018). Different architectures of artificial neural network arrays, resulting from these combinations were evaluated in the neural network toolbox. ANN consists of two layers of neural network architectures called forward and backward neural network (C. Xu, 2018).

#### 2.9.4. Feed forward neural network architecture

2.9.5. An input layer is the initial layer of a feed forward neural network. Hidden layers come next, and an output layer completes the feed forward neural network. Information should never reverse itself; it should only ever travel from the input layer to the output layer. (Alzubaidi L & 9(3):445., 2020).

#### 2.9.6. Feed-back neural network neural architecture

Backward feeding Neural networks can provide bidirectional signals by displaying network loops and sending information backward from the hidden and output layers. Until the neural network structure achieves the convergence state, the activation value changes instantaneously.

(Rubin J & 51(6):S18-21., 2018).

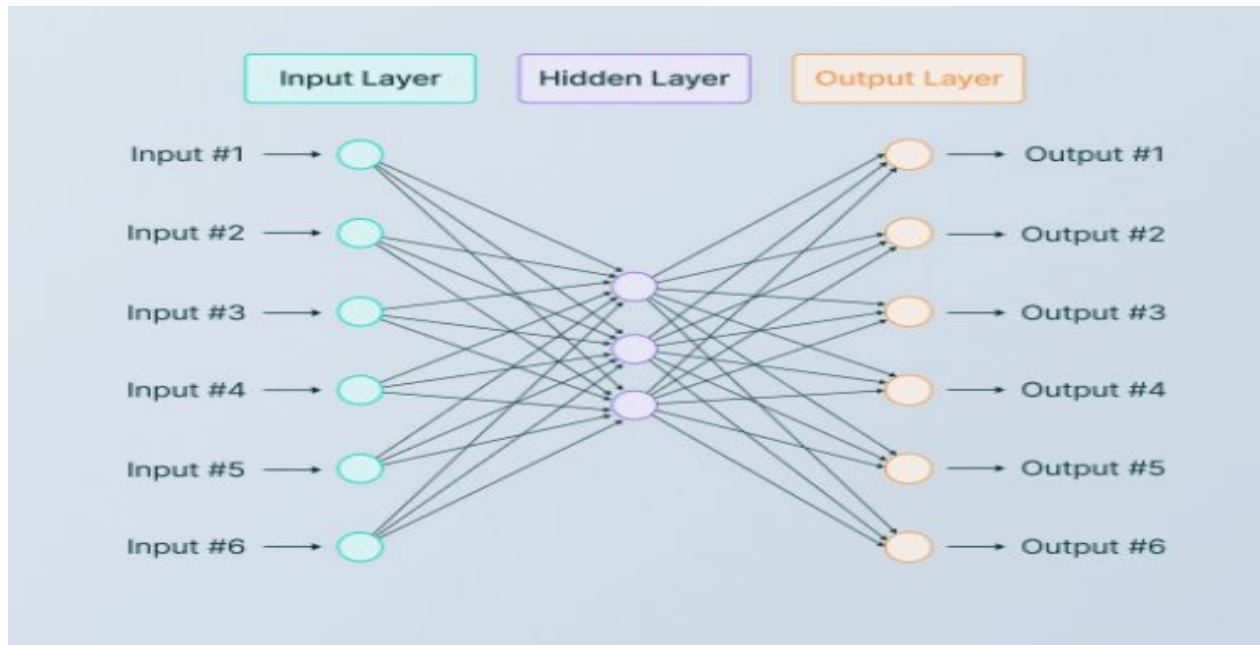


Figure 2.7 Artificial neural network 1 (Devaraj, 2012)

#### 2.9.7. Input layer

To calculate the number of neurons in the input layer, the neural network design receives a sample of the data set. The input layer is a neural network layer made up of passive nodes that transmit signals to the subsequent corresponding levels rather than helping with signal modification. (Survey., 2020) .

### **2.9.8. Hidden layer**

The neural network layer known as the "hidden layer" might have multiple layers based on the specific application you are working on and supply as many neurons as your application needs. The nodes of the hidden layer should be complicated as they interact with signal changes.

### **2.9.9. Output layer**

The number of neurons in the output layer of the neural network is equal to the number of values that the network produces in the ANN output layer topology. These layers' participating nodes are the ones that are still active (Drewek-Ossowicka, 2021).

### **2.9.10. Convolutional layer**

It moves through the input image in preset intervals, called strides, using a kernel or filter. To get the results you want, the length of your stride is essential. As the filter is applied, the dot product of the filter and the region of the picture it is applied to is computed. Next, the product matrix's total values are transferred to the appropriate location within the convolved feature map matrix. Consequently, the image's feature map's dimension is decreased. Different types of features are extracted from the data set using different types of filters, which vary in shape, color, and edge. One filter might, for instance, be responsible for feature extraction based on model training.

### **2.9.11. Padding**

Adding zeroes to the original data set boundary is a symmetrical method. This helps ensure that the feature map output is the appropriate size. When applying a filter to the input size, it is typically employed to prevent information loss.

### **2.9.12. Filter**

It is known as kernels. Different kernel variations exist. The convolutional output's depth is enhanced by each filter. For instance, the output depth will be three if three filters are used. The output of convolution is based on three constraints: stride, depth, and padding. We will have to adjust these variables to achieve the intended outcome.

### **2.9.13. Dimension**

CNN is lowering its parameters. During the convolution process, a filter matrix is convoluted across the input data set, resulting in a condensed dimension. Additionally, it deepens the data

set, making it possible to recognize and differentiate particular data sets. An RGB feature of  $7*7*3$  will be produced from a  $9*9$  RGB image with  $3*3$  kernels at stride 1 (per channel). The depth of the convolved image is represented by the third dimension..

#### **2.9.14. Pooling layer**

This layer operates steadily on a tiny kernel of the data size. After selecting the pixel with the highest intensity, the others were thrown away. The result will be a feature set matrix with reduced dimensions. As a result, the dataset has less redundant sparse cells that aren't used for classification. Ax pooling helps reduce the dimensionality of the network (or layer). The idea is that, at the expense of some information loss, the pixel with the most information can move closer to nearby or adjacent pixels.

#### **2.9.15. Activation Layer**

This layer is often activated using the ReLu, soft max, and sigmoid. Operation An activation function in ReLu that leaves positive values alone and sets all negative values to zero (Farah, 2018). The convolution and pooling layers come after this layer. They are performing on par with or better than humans in a number of fields, including computer vision. Although they seem intrigued, creating a CNN is a huge undertaking in and of itself. There is currently no established formula for CNN design. Many researchers, for example, have made broad recommendations. They do not, however, always hold because the goal is as dependent on data as it is on the algorithm.

#### **2.9.16. Dropout Layer**

In a fully connected network, the regularization layer is the dropout layer, which is usually deployed after the layer with neurons. (Gamage, 2020). In each iteration, some neurons' weights are arbitrarily decreased, allowing other neurons to acquire weight. Dropout rates between 20 and 50 percent are typically accepted. Using the dropout layer usually results in a 1–2% improvement in network performance. (Survey., 2020).

#### **2.9.17. Fully Connected Layer**

A fully connected layer is often built in a pyramidal fashion from top to bottom, with the number of parameters in each layer progressively increasing until the required number of classes is reached. Even while the network's learning capacity can be increased by adding additional

hidden units to the layer, the network's precision will eventually reach saturation. (Sherstinsky, 2020). Since choosing a unit is typically a matter of trial and error, there is no set technique. These are also stacked in the network's completely linked subset. Most studies are most effective when the number of units is a multiple of 64. If enough patterns are received by the network after the convolution layers are flattened, two or three layer networks are appropriate.

## **2.4. Deep learning based IDS**

CNN Computer vision uses Convolutional Neural Networks (CNN) extensively because of their ability to extract characteristics from images. (Jan Lansky, 2021) CNNs are also used for intrusion detection, where traffic data is represented by images that are passed to the CNN model. The literature study uses 1D CNN and 2D CNN algorithms with varying degrees of success depending on the size of the dataset. (Wei Wang, 2017). The System calls are user commands that are transmitted to the operating system kernel. These system calls can be analyzed by CNNs to find intrusions. The growing depth of CNNs as layer counts rise causes the vanishing gradient problem, in which a network's weight changes negligibly. The residual neural network (ResNet) (Kaiming He, 2016), model was developed to address this problem. It uses a skip connection to avoid diluting of the weights by skipping some layers. A popular method for categorizing network traffic is by using Res Net.

## **2.4. LSTM**

One type of neural network that can remember information is called a recurrent neural network (RNN). Long short-term memory is one type of RNN. However, compared to other RNNs, LSTMs are better at remembering long-term dependencies. LSTMs have a reputation for delivering exceptional results when working with time series data. (Pankaj Malhotra, 2015). An LSTM model was used by Loukas et al. (George Loukas, 2017) to forecast intrusion detection in office in real-time. Maya et al. (Shigeru Maya, 2019) Presented delayed LSTM, a model for anomaly detection in time-series data, in an effort to enhance the performance of LSTMs. The capacity to select the optimal model from a range of viable models is advantageous for this technique.

## **2.5. Combined Approach**

Zeng et al. (Yi Zeng, 2019) successfully processed traffic payload by using multiple deep learning approaches. The authors extracted local, time series, and text characteristics, respectively, by combining CNN, RNN, and auto encoder.

## **2.7. Generative Adversarial Networks**

Neural networks termed Generative Adversarial Networks, or GANs, pit two models against one another, referred to as a discriminator and a generator (Ian Goodfellow, 2014). While the discriminator learns to distinguish between actual and predicted label, the generator learns how to produce images that will trick the discriminator. As shown by Lee et al. (detection, 2021), GANs can be used to generate new samples of uncommon attack classes, thereby resolving the problem of data imbalance. With GANs, defense against adversarial attacks can also be accomplished. APE-GAN, a defense model used by Shen et al. (Shiwei Shen, 2017), preprocessed data before sending it to the intended classifier.

## **2.10. Review of related work**

Over the past few years, numerous researchers and academics have worked to create an intrusion detection system utilizing conventional machine learning methods. While the most popular machine learning approaches in previous research were J48, ANN, RF, SVM, k-NN, and Naïve Bayes, more recent studies have included deep learning techniques like CNN, LSTM, and RNN, which produce incredibly promising results when compared to the older techniques. A recent evaluation of neural network applications in intrusion detection systems revealed that several significant studies were contrasted according on the dataset(s), methods, and attack types looked at. (Drewek-Ossowicka et al., 2021).

(Moustafa & Slay (2015) Moustafa N, 2017) was developed a model concentrating on the UNSW-NB15 dataset's attack family classification. For feature selection, the study employed the Association Rule Mining approach. NB and the Expectation–Maximization (EM) method have been applied to categorization. However, as EM's accuracy was 58.88% and Naïve Bayes' was 78.06%, neither algorithm's accuracy in identifying uncommon attacks was appreciably higher.

(Moustafa & Slay (2016) Moustafa N & 25(1–3):18, 2019) completed their work in 2016 and selected features using the gain ratio and correlation coefficient. The UNSW-NB15 was then

subjected to five different classification algorithms: NB, DT, ANN, LR, and EM. According to the results, DT with a 15.75 False Alarm Rate (FAR) produced an accuracy of 85%. A subset of UNSW-NB15 was used in this study, although the detection accuracy was unsatisfactory.

(Kasongo & Sun (2020) Kasongo SM & 10.1186/s40537-020-00379-6., 2020) Presented the NIDS approach using five classification algorithms of LR, KNN, ANN, DT and SVM in

(Atefinia & Ahmadi, 2021) The researcher's main objective was to outperform the previous efforts in terms of precision rate. A multi-architectural modular deep neural network model, which combines four distinct designs, was suggested as a remedy for this cause. As a weighted averaging technique, an aggregator module was created that combines Deep Feed-Forward Neural Network (DFFNN), Stacked Restricted Boltzmann Machine (SRBM), Gated Recurrent Unit (GRU), and LSTM models to produce output for the complete network.

For brute force, DDoS, and DoS attack types, a multi-class classification yielded the best accuracy, precision, and recall rates, 93%. These results clearly point to a problem with overfitting. Furthermore, dataset subsets were used in all trials. His research did not include overall outcomes because they only included results for each of the six attack classes. Additionally, several measures for unbalanced datasets, including as G-mean score, balanced accuracy, ROC AUC, and PR AUC, were not covered by performance metrics. N/A values indicated unsatisfactory performance, but the explanation for their absence was not provided. However, his work was also limited by the lack of sufficient information regarding the hyperparameters utilized for each model, which has a significant impact on reproducibility.

In (Basnet et al., 2019), For the purposes of binary and multiclass classification, Multilayer Perceptron (MLP) models were developed. Comparing the performance of many deep learning frameworks, including Keras, TensorFlow, Theano, fast.ai, and PyTorch, was the main goal of this study. The study's output was compared using processing time and accuracy scores. However, this work was not without its limitations. There was insufficient information in this study regarding the hyperparameters that were employed to produce these outcomes. The usage of at least three concealed layers was the only answer. The reproducibility of this work is significantly impacted by this circumstance. Furthermore, the performance indicators explicitly employed to address the issue of class imbalance were not mentioned.

Ferrag et al. (2020) reported that a variety of deep learning techniques, including Deep Neural Network (DNN), Feed Forward Deep Neural Network (FFDNN), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Restricted Boltzmann Machine (RBM), Deep Belief Network (DBN), Deep Auto encoder (DA), Deep Migration Learning (DML), Self Taught Learning (STL), and Replicator Neural Network (Re NN), were employed for cyber security intrusion detection. According to the research by Ferrag et al., two real traffic datasets, CSECIC-IDS2018 and Bot-IoT dataset, were used to test seven deep learning models: recursive neural networks (RNN), deep neural networks (DNN), restricted Boltzmann machines (RBM), deep belief neural networks (DBNN), convolutional neural networks (CNNs), deep Boltzmann machines (DBM), and deep auto encoders. The four steps of the applied DL approaches are pre-processing, training, testing, and datasets. Just 5% of the total dataset was used for the research' training and testing phases. As a result, the CNN model achieved the maximum accuracy of the model for CSE-CIC-IDS2018, with 97.376%. Furthermore, the deep auto encoder model produced the maximum recall on this dataset, 98.18%..

**Kaleb ayele** (kaleb, 2021) conduct an anomaly intrusion detection system with GAN model. the research was used in his paper the recent existing GAN and WGAN which is one of GAN variants for anomaly intrusion detection using NSL KDD dataset. The Final evaluation result showed by keleb research was that, the accuracy in classifying normal and attack. The results of his model had shown on WGAN with accuracy of 89% prediction with default parameter and high prediction that performing with accuracy of 95.7% with optimized parameter.

The research conducted by **Aschalew Woldeyesus** (Aschalew, 2021) which is on title Hybrid Approach to Improve Accuracy of Intrusion Detection System in IoT Environment. In his research study the researcher focuses on developing intrusion detection and prevention system for Iot network .IT involves combining anomaly based and signature based modules. Signature based module is implemented using ip tables firewall and anomaly based module is implemented using deep learning algorithm. In his research, the researcher was not mention test performance matrix or parameters.

This study's primary flaw is that the model was trained using only 5% of the dataset. Furthermore, throughout the preparation step, the researcher was not given a thorough

explanation of the preprocessing techniques. Only detection rate and accuracy were used as performance metrics to assess the model. The other performance measures utilized to address the issue of class imbalance were not mentioned. Furthermore, the reproducibility of this work is impacted by the fact that the hyper parameters utilized for the multi-class classification were not shared.

Table 2.1. review of related work 1

| Author          | Title                                                                    | Publication year | Method used                                                                | Result                                                                                   | Research gap                                                                                                                                          |
|-----------------|--------------------------------------------------------------------------|------------------|----------------------------------------------------------------------------|------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Moustafa & Slay | classification of attack families available in the UNSW-NB15 dataset.    | 2017             | The study used the Association Rule Mining technique for feature selection | The accuracy of the model by two algorithm were 78.06% and 58.88%.                       | the accuracy of both algorithms for detecting rare attacks was not significant to detect an attacks                                                   |
| Kasongo & Sun   | Developed NIDS approach using five classification algorithms             | 2020             | LR, KNN, ANN, DT and SVM                                                   | The accuracy of KNN model result is 96.76% and multi class classification model is 82.6% | Although binary classification performed better an accuracy of 96.76% using the KNN classifier. But the other classifier low accuracy rate of 82.66%. |
| Ferrag et al    | Developing intrusion detection system using various deep learning method |                  | RNN, CNN, DBN and                                                          | The accuracy result was obtained using CSE-CIC-IDS2018                                   | The training and testing part of the studies were performed on only 5% of the entire                                                                  |

|                                        |                                                                                       |      |                                              |                                                                                    |                                                                                                                     |
|----------------------------------------|---------------------------------------------------------------------------------------|------|----------------------------------------------|------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
|                                        |                                                                                       | 2020 | RBM model                                    | data set is 97.376% with CNN model                                                 | dataset. The other gap of the research work is preprocessing method was not defined clearly.                        |
| Basnet et al                           | IDS using Multilayer Perceptron (MLP) model                                           | 2019 | multiclass and binary classification methods | The output of the study were based on accuracy and processing time with 91%        | In this study, there was not enough information about hyper parameters used to achieve this results .               |
| Kaleab ayele (kaleb, 2021)             | conduct an anomaly intrusion detection system                                         | 2021 | Apply and develop GAN model                  | The accuracy result of the model was 89%                                           | The research was utilized only one model to predict the IDS and the data set was out dated which is NSL KDD dataset |
| Aschalew Woldeyes us (Aschalew , 2021) | Hybrid Approach to Improve Accuracy of Intrusion Detection System in IoT Environment. | 2021 | Apply IOT method to conduct research         | The accuracy of the model was 0.99<br><br>The model was trained using IOT approach | Focused on only anomaly based and signature based IDS. The data set was only IPtable data set                       |

## **CHAPTER THREE**

### **RESEARCH METHOD**

#### **3.1. Introduction**

In this section, the research approach and method was described following the design science research methodology. The network intrusion detection method were also described in detail. The research approach and method, development tools and procedures were utilized to accomplish this research study and conduct the investigation under review of various literatures. This research study was conducted by reviewing the existing previous researches done on IDS from a variety of scholars and also using our skills. This thesis addresses every step of standard research methodology as well as from problem identification to solution discovered from implementation the research. Different parts of the proposed NIDS are described with their relevance and method to build the model. this component involves, research design, type of network attack, development of the model, the actual tools and techniques used , data preprocessing and , as well as the evaluation metrics of the model is also described.

The detection and classification of NIDS were based on several deep learning models like CNN, RNN, DNN and LSTM. On other hand, various data pre-processing task, NIDS structural model, the detection and classification techniques and also the evaluation metrics of the model were discussed in this chapter. The CSE-CIC-IDS2017 and CSE-CIC-IDS2018 dataset's were converted and normalized in accordance with data pre-processing algorithm in order to fit into NIDS model as vectors of fully numeric converted data. The Python development tools was used to create NIDS models based on CNN, LSTM, RNN and DNN models with Keras serving as the

deep learning framework and Tensor Flow as the backend frame work respectively. In the following section, we have discussed that, the research approaches and method that will be utilized in this research study.

### **3.2. Research Design**

The design science research methodology is conducted and used in this study. Design science is, at its core, a problem-solving paradigm (R. B. Onwuegbuzie and A. J. Johnson, 2004). It aims to improve human knowledge by developing innovative artifacts and making design science knowledge (DK) through innovative solutions for real-world problems (Lenzholder S & 54–64, 2017). It has recently become so popular for technological advancement and its penetration into all aspects of human activity. A business cannot function without some form of information technology.

The goal of design science is to create new and creative thing that are represented by structures, models, methods, and instantiation in order to push the limits of human and organizational capabilities. It seeks to produce knowledge about how objects can and should be constructed or ordered, typically through human action, in order to address organizational issues that have been discovered.

Among the subjects covered are how to model a business process, support information systems with organizational strategy, structure and construct database systems, and provide data analytics for efficient decision-making. Artifact development using known artifact development methods and techniques such as object-oriented methodology, agile methodology, and prototyping methodology are important activities in design science approaches (Jongbo, 2014). Identification of problems, system modeling, and artifact development are all part of development. The new artifact's relevance to solving problems in more cost-effective and efficient ways must be assessed. Artifact evaluation ensures that new artifacts are accepted by their intended users.

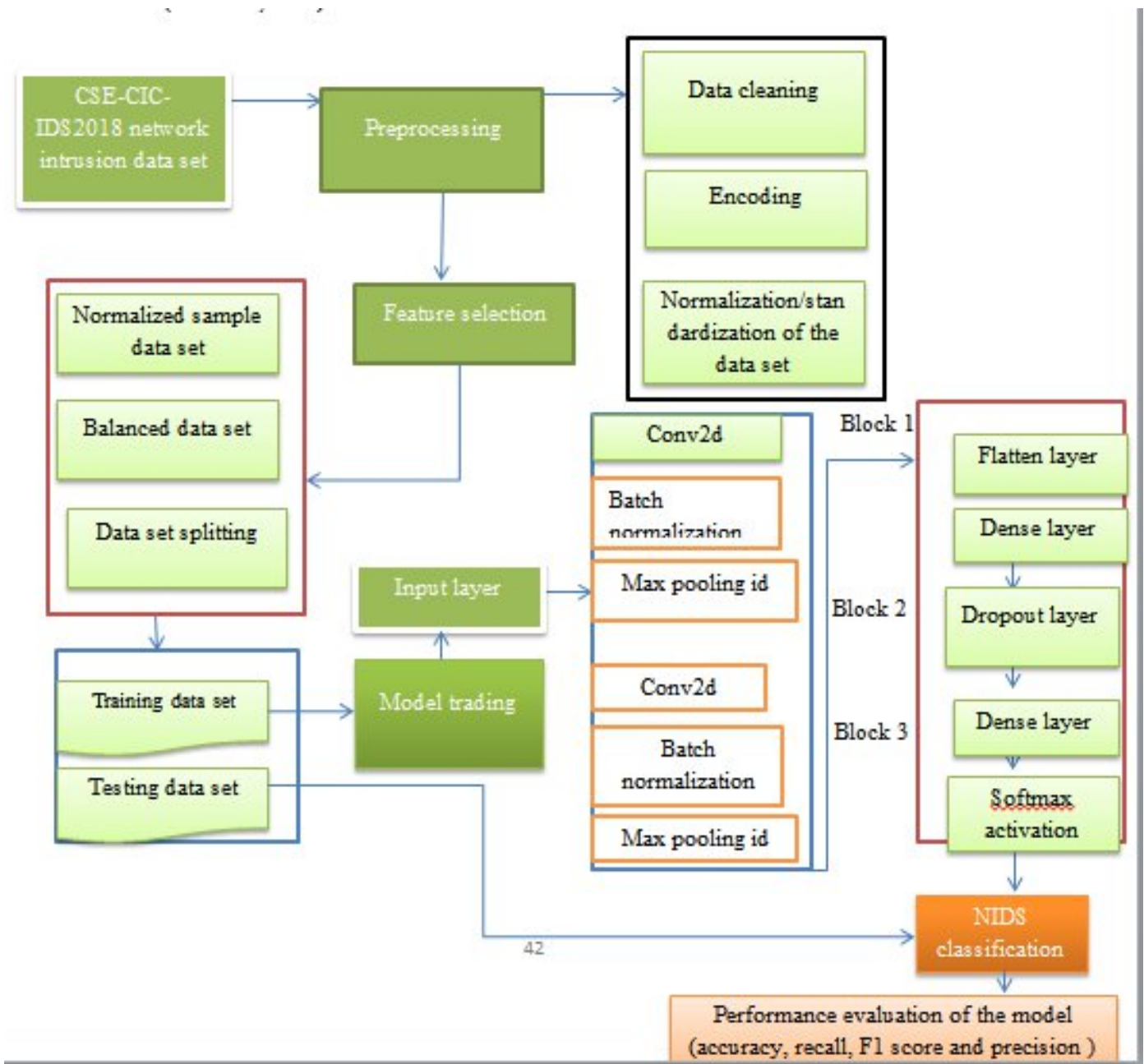
Convolution neural networks, deep neural networks, recurrence neural networks, and LSTM models were all used in this research study to create a deep learning artifact development model. It is classified as design science research as it identified a problem created artifacts, and evaluated their performance using a well-known artifact evaluation method. The research follows the steps below to detect and classify normal and attacked class of a network in a more efficient and effective manner (M. Mukherjee, 2016).

The second step is to establish goals that must be met in order to address the issue and narrow the scope of the investigation. Thirdly, information needs to be gathered from various datasets in order to look for trends that can separate known and unidentified assaults from regular data. The entire preprocessed data set is divided into training and testing sets during the train-test split step in order to assess the performance of the models. The training set is further divided into the real training and validation sets during the validation 30% is given to testing phase. The deep learning algorithm is taught using the training data on their default parameters during training with default parameters phase, producing a trained model. The performance of the trained model is subsequently assessed using the distinct preprocessed testing data. After that, we suggested a fix for the issue that was found in the literature review, using the CNN and LSTM algorithms. We used CNN and LSTM algorithms to construct and deploy anomalous intrusion detection models after determining the ideal hyper parameter requirements. Finally, we used test datasets

to evaluate the model using IDS measures, including accuracy, precision, recall, accuracy, and F1-score (M. Lucic, 2018).

Figure 3.1. Proposed architectural model of the this research study 1 (Qazi, Faheem, & Zia, 2023)

The study's suggested model uses the most recent CSE-CICIDS2018 intrusion dataset to create



an effective model of the convolutional neural network (CNN), DNN, RNN, and LSTM paradigms for network intrusion detection. The accompanying figure 3.1 represents the seven

primary steps that make up the suggested methodology for this research study .Figure 3.1. Proposed architectural model of the research (Qazi, Faheem, & Zia, 2023).

### **3.3. Approaches of the research**

The first step in the research process was gathering the dataset, which was used as the basis for training and evaluating the performance of deep learning models for network intrusion detection. After preprocessing the gathered dataset to exclude null or missing values, we carried out data normalization and label encoding in the subsequent stage. To effectively pick relevant features to train the deep learning models in the third stage, we employed a feature selection technique based on a given correlation coefficient matrix that maps. Before dividing the dataset into training and testing sets, we used SMOTE oversampling and under sampling to balance it in the fourth phase. We created and assessed deep learning models for NIDS cyber-attack and regular attack detection and classification in the fifth and sixth stages. Lastly, we assessed how well trained deep learning models performed and suggested the top-performing model for a successful network intrusion detection categorization.

A convolutional neural network (CNN) architecture with multiple layers and blocks is used in the suggested intrusion detection model for cloud environments in order to effectively detect intrusions. Activated by the ReLU function, the model's first three Conv2D layers have a kernel size of six and 64 filters each. After that, down sampling is carried out using MaxPooling1D layers, and effective training is completed utilizing batch normalization. The feature maps are transformed into a one-dimensional vector via a flatten layer after the convolutional and pooling layers. This vector is subsequently processed using ReLU activation and two dense layers that are fully connected, each consisting of 64 units. To avoid overfitting, a Dropout layer with a 0.5 dropout rate is included. The dense layer, the last output layer, is activated by the Soft max function to produce class probabilities.

The model uses the batch normalization, max Pooling, and Conv2D layers to iteratively extract hierarchical representations of the input data. The preprocessed data are split into equal sets for testing and training. The Dropout layer improves generalization, the first dense layer serves as an intermediate, and the dense layers that come after the Flatten layer combine the data gathered for classification. The final dense layer before the Soft max activation creates uncertainty in the form of probability distributions for every class. To assess the model's performance in the real

world and demonstrate its dependability and effectiveness in network intrusion detection for cloud environments, metrics like as accuracy, precision, recall, and F1-score are employed in model-training evaluation on a different dataset. Each of these stages is covered in further detail in the following subsections. Creating a network intrusion detection system by training a deep learning predictive model using the most up-to-date and precise dataset in light of upcoming cyber- attacks is a crucial part of a successful solution. As a result, the CSE-CIC-IDS2018 public dataset has been selected for use in this investigation..

In 2018, the Canadian Institute for Cyber Security (CIC) and the Communications Security Establishment (CSE) produced this dataset. This dataset, which had multiple real-world examples in diverse settings, focused mostly on network-based anomaly detection (2023). Throughout the testing phase for this thesis, seven distinct assault scenarios were carried out in addition to one benign or normal scenario.

### **3.4. Network Intrusion Data set**

Selecting a suitable dataset is a crucial task that is one of the most significant factors that comes with an intrusion detection system. Using outdated datasets may not produce the intended or objective findings because network attacks change over time (Ren, Jin, & OuYang, 2024). Based on this, we decided to use the CSE-CICIDS2018 dataset, one of the recently released IDS datasets created on the AWS (Amazon Web Services) cloud environment. Given that it contains both typical cloud network traffic and recent assault traffic, and that it is accessible to the general public. Seven different scenarios that simulate different types of network intrusions make up the CES-CICIDS2018 dataset, which is essential for assessing how well intrusion detection methods operate. Every scenario offers a variety of testing conditions by combining benign and cyber-attack instances, providing a diverse testing environment (Krishna, Lal, Mathewkutty, Jacob, & Hari, 2020). In order to ensure comprehensive evaluation and development of algorithms for network intrusion detection and classification, each scenario is designed to evaluate specific types of attacks and benign acts. This technique or approach increases the model's efficiency and accuracy and increases its capacity to manage intricate network threats. (Al-Fawa'Reh, Al-Fayoumi, Nashwan, & Fraihat, 2022).

The computer platform of Amazon was used to build the diversified environment. The entire infrastructure was split into victim and attacker organizations. There were fifty machines that

comprised the attacking group. There were thirty servers and four hundred machines in the affected organization. All of the machines' system logs and network traffic were gathered to obtain this dataset. As represented in figure 3.2, several operating systems like Windows, Mac, Ubuntu, 12.04, 16.04 and Windows Vista,7,8.1,10 which run different services are included in the environment to repeat all possible attack scenarios (Hagar & Gawali, 2022).

Any number of devices outside the target network was used to carry out each attack. Five distinct subnets that represented the various divisions and server rooms of the business were set up, imitating real-world network topologies. Table 3.1. shows how this dataset's network traffic is distributed. The attack scenarios correspond to each of the 14 attack types.

### **3.5. Preprocessing**

The preprocessing task enhances the efficiency of training and testing the model, which further improves the quality of the data and the output. An essential phase in the deep learning process is data preprocessing. The data preparation approaches can be applied to solve several difficulties simultaneously when working with a large dataset. In addition to irrelevant and categorical features, the dataset for this investigation includes null and outlier values and redundant records. As a result, we had to decide which samples were most pertinent by filtering and clearing the CES-CICIDS2018 dataset. The next steps are focused on label encoding, data standardization, and data cleaning procedures that we carried out as part of the dataset preprocessing task.

The fundamental stages of data cleaning and preparation involve identifying strategies for handling missing data fields, obtaining the information required to model or account for noise, eliminating noise or outliers as needed, and taking time sequence information and known changes into consideration. The process of getting the raw dataset ready for testing and analysis to get the recommended results is known as preprocessing. The propose classification system events and forecasting future events using historical observation data. Large amounts of log entries are frequently found in IDS datasets. However, many of the log entries contain redundant information because a single root cause may activate many components, which might result in distinct log messages for the same root cause. Yet, duplicate log entries are meaningless for deep learning as they just repeat the same information. To get rid of duplicate log events, we combined filtering and log message normalization. Preprocessing activities also aimed to prevent incomplete and missing values in the IDS data set.

Transformations used to normalize log messages can include the following steps: extracting CES-CICIDS2018 log files from zip files, eliminating rows of incomplete log records, Log files are parsed into an appropriate format for analysis, some missing values are removed, words are replaced with numbers, duplicate and redundant values are removed, text log files are exported into Excel format, string values are converted to integers, and the class values for all records are labeled with 1 for attack and 0 for normal (non-attack) values. The following attack terms, such as attribute value worm propagation attempt, network scan, and privacy breach, have been used to label these data. The terms listed above were regarded as attack traffic, whereas the remaining terms were regarded as typical network traffic. transforming an Excel log file into a SCV file.

### **3.5.1. Dataset Preparation**

A literature review reveals a new network intrusion detection evaluation dataset. The Institute of Cyber Security at the University of New Brunswick produced the CIC-IDS 2018 dataset. For the precise and efficient evaluation of deep learning-based network intrusion detection systems, a new benchmark dataset has been made available. This study uses the new CIC-IDS2018 benchmark dataset to assess the performance of NIDS models.

One of the newest datasets for network intrusion detection systems is CIC-IDS2018. Its availability as raw data in CSV files with flow-based features is one advantage. The CICIDS20178 dataset for NIDSs and IPSs was developed by the Canadian Institute for Cyber Security. Additionally, the raw network traffic data gathered by this dataset is divided into 10 CSV files daily, which can be accessed from the Amazon cloud. Over 80 features are extracted from network traffic data using CICFlowMeter-V3, a network traffic flow generator and analyzer. (Lashkari, 2020).

Canadian Institute of Cyber Security, no date) Malicious and normal communications can both be found in this file. Table 3.1 below shows that, the third column provides a more detailed description of the attack traffics. The dataset includes benign and ordinary traffic as well as 14 distinct types of intrusions. Two characteristics, or columns, titled Fwd Header Length, are included in the dataset. One of them must be removed because it is redundant. There are just 80 features left to look at after the duplicate characteristics are removed. According to the CICIDS-2018 data, data with a large class imbalance will negatively impact false alarm and detection accuracy.

Table 3.1. Attack Distribution of the model 1

| Attack type  | Attack description                                       | attack distribution in (%) |
|--------------|----------------------------------------------------------|----------------------------|
| Benign       | None                                                     | 80                         |
| DDOS         | DDoS attacks-LOICHTTP file, DDOS-LOIC-UDP and DDOS-HOIC  | 8.5                        |
| DOS          | DoS-Golden Eye, DoS-Slowloris,DoS SlowHTTPTest, DoS-Hulk | 4                          |
| Brute force  | FTP-Brute Force and SSH-Brute Force                      | 3.5                        |
| Bot net      | Bot                                                      | 2                          |
| Infiltration | Infiltration                                             | 1                          |
| Web attack   | Brute Force –Web, Brute Force –XSS and SQL Injection     | 1                          |
| <b>Total</b> | <b>14</b>                                                | <b>100%</b>                |

Table 3.2.represents that, the attack tools, attacked object and attacker target used to perform all kind of an attacks. (CSE-CIC-IDS2018 on AWS, n.d.)

Table 3.2. Attacks Environment of the NID 1

| Attack type         | Tools                                                         | Attacker   | Attacked object(victims)           |
|---------------------|---------------------------------------------------------------|------------|------------------------------------|
| Brute force attack  | FTP – Patator, SSH – Patator                                  | Kali Linux | Ubuntu 16.4 (Web Server)           |
| DoS attack          | Hulk, Goldeneye, Slowloris, and Slowhttpstest                 | Kali Linux | Ubuntu16.4 (Apache)                |
| DoS attack          | Heartleech                                                    | Kali Linux | Ubuntu 12.04(Open SSL)             |
| Web attack          | Damn Vulnerable Web App(DVWA) and In-house selenium framework | Kali Linux | Ubuntu 16.4 (Web Server)           |
| Infiltration attack | First level Dropbox download in a windows machine             | Kali Linux | Windows Vista and Macintosh        |
| Botnet attack       | Ares (developed by Python), remote                            | Kali Linux | Windows Vista, 7, 8.1, 10 (32-bit) |

|               |                                                           |            |                                                    |
|---------------|-----------------------------------------------------------|------------|----------------------------------------------------|
|               | shell, file upload/download, capturing                    |            | and 10 (64-bit)                                    |
| DDoS+PortScan | Low Orbit Ion Canon (LOIC)for UDP, TCP, or HTTP, requests | Kali Linux | Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit) |

### 3.5.2. Data Cleaning

Since the missing, null, and infinite values give the deep learning model inaccurate data and may reduce the model's accuracy, they were removed from the dataset during the data cleaning step. We therefore excluded samples that were invalid, irrelevant, or non-informative, including infinite and null values.

### 3.5.3. Label Encoding

To make categorical feature values fit into deep learning models that only take in numerical data, a technique called label encoding is applied. Numerous techniques are available for encoding features. We have used the most widely used technique among them, one-hot encoding, to translate the values of the categorical features into numerical values.

### 3.5.4. Data Standardization

One of the most crucial elements of preprocessing methods is data normalization. The process of rescaling feature values to guarantee that the mean value is 0 and the standard deviation is 1 is known as standardization. All of the attributes of the data are normalized using a single scale. In order to speed up learning when all characteristics have the same magnitude, deep learning models are typically trained using standardized data. After subtracting the average value of each feature in the dataset, each value is then divided by the standard deviation to begin the standardization process. This is how data are generally dispersed. The Z-score formula below is found in Equation (1), which is frequently used for standardization.  $Z = (x - \mu)/\sigma$  Equation (1) represents the standard equation for standardization, in which  $\sigma$  represents the data's standard deviation and  $\mu$  stands for the data's mean.

### 3.5.5. Fix Imbalance Class

It shows how many samples are there in each class in the CES-CICIDS2018 dataset. (Shelke, Deshmukh, & Shandilya, 2017) A number of classes sample distributions from the CES-CICIDS2018 dataset are shown. Given that the normal class comprises a significantly higher

percentage of samples than samples from other classes, it is evident that this dataset has a class imbalance problem.

To address this problem, we therefore had to initially lower the normal class samples by using an under-sampling technique. The other classes also have an imbalance problem, thus using the under-sampling strategy would not have entirely resolved the issue. Consequently, in order to address the issue of class imbalance, we first under-sampled the typical class traffic before applying the synthetic minority oversampling technique (SMOTE) (M. Mukherjee, 2016).

### **3.5.6. Dataset Splitting**

The data must be divided into a test set and a train set as 80:20 ratio in order to test and train deep learning models. This means that 20% of the dataset is used for testing, and 80% is chosen at random for training. Only the training set is exposed to the deep learning model during the training phase. The test set is not exposed to deep learning until the training phase is complete. In this way, we can confirm whether the model's training has an under- or over-fitting problem by analyzing the performance of the trained model on unknown data.

### **3.5.7. Feature Selection**

The CES-CICIDS2018 dataset comprises more than 80 features, as was previously mentioned. The deep learning model's performance is greatly impacted by the abundance of input features. The success of a deep learning model is largely dependent on feature selection (Jaw & Wang, 2021). An important factor in any intrusion detection system (IDS) approach's efficacy is feature selection. Its effects are several, one of which is that it minimizes computational costs by reducing model complexity. It also makes troubleshooting the model easier, which improves how the model interprets the learning outcomes. Additionally, it greatly shortens the time needed for model training and increases training accuracy.

Table3.3. Description of Network intrusion detection Features 1

| <b>CES-CICIDS 2018<br/>data set Feature<br/>name</b> | <b>Feature description</b>                                                 |
|------------------------------------------------------|----------------------------------------------------------------------------|
| Flow Duration                                        | Flow duration                                                              |
| Tot Fwd Pkts                                         | Total packets in the forward direction                                     |
| Tot Bwd Pkts                                         | Total packets in the backward direction                                    |
| TotLen Fwd Pkts                                      | Total size of packet in forward direction                                  |
| Infiltration attack                                  | First level Dropbox download in a windows machine                          |
| Fwd Pkt Len Max                                      | Maximum size of packet in forward direction                                |
| Fwd Pkt Len Min                                      | Minimum size of packet in forward direction                                |
| Fwd Pkt Len Mean                                     | Average size of packet in forward direction                                |
| Fwd Pkt Len Std                                      | Standard deviation size of packet in forward direction                     |
| Bwd Pkt Len Max                                      | Maximum size of packet in backward direction                               |
| Bwd Pkt Len Min                                      | Minimum size of packet in backward direction                               |
| Bwd Pkt Len Mean                                     | Mean size of packet in backward direction                                  |
| Bwd Pkt Len Std                                      | Standard deviation size of packet in backward direction                    |
| Flow Byts/s                                          | flow byte rate that is number of packets transferred per                   |
| Flow Pkts/s                                          | flow packets rate that is number of packets transferred per second         |
| Flow IAT Mean                                        | Average time between two flows                                             |
| Flow IAT Std                                         | Standard deviation time two flows                                          |
| Flow IAT Max                                         | Maximum time between two flows                                             |
| Flow IAT Min                                         | Minimum time between two flows                                             |
| Fwd IAT Tot                                          | Total time between two packets sent in the forward direction               |
| Fwd IAT Mean                                         | Mean time between two packets sent in the forward direction                |
| Fwd IAT Std                                          | Standard deviation time between two packets sent in the                    |
| Fwd IAT Max                                          | Maximum time between two packets sent in the forward direction             |
| Bwd IAT Mean                                         | Mean time between two packets sent in the backward direction               |
| Bwd IAT Std                                          | Standard deviation time between two packets sent in the backward direction |
| Bwd IAT Max                                          | Maximum time between two packets sent in the backward direction            |
| Bwd IAT Min                                          | Minimum time between two packets sent in the backward direction            |

|                  |                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------|
| Fwd PSH Flags    | Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)  |
| Bwd PSH Flags    | Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP) |
| Fwd URG Flags    | Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)  |
| Bwd URG Flags    | Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP) |
| Fwd Header Len   | Total bytes used for headers in the forward direction                                            |
| Bwd Header Len   | Total bytes used for headers in the backward direction                                           |
| Fwd Pkts/s       | Number of forward packets per second                                                             |
| Bwd Pkts/s       | Number of backward packets per second                                                            |
| Pkt Len Min      | Minimum length of a flow                                                                         |
| Pkt Len Max      | Maximum length of a flow                                                                         |
| Pkt Len Mean     | Mean length of a flow                                                                            |
| Pkt Len Std      | Standard deviation length of a flow                                                              |
| Pkt Len Var      | Minimum inter-arrival time of packet                                                             |
| FIN Flag Cnt     | Number of packets with FIN                                                                       |
| SYN Flag Cnt     | Number of packets with SYN                                                                       |
| RST Flag Cnt     | Number of packets with RST                                                                       |
| PSH Flag Cnt     | Number of packets with PUSH                                                                      |
| ACK Flag Cnt     | Number of packets with ACK                                                                       |
| URG Flag Cnt     | Number of packets with URG                                                                       |
| CWE Flag Count   | Number of packets with CWE                                                                       |
| ECE Flag Cnt     | Number of packets with ECE                                                                       |
| Down/Up Ratio    | Download and upload ratio                                                                        |
| Pkt Size Avg     | Average size of packet                                                                           |
| Fwd Seg Size Avg | Average size observed in the forward direction                                                   |
| Bwd Seg Size Avg | Average size observed in the backward direction                                                  |
| Fwd Byts/b Avg   | Average number of bytes bulk rate in the forward direction                                       |

|                            |                                                                                |
|----------------------------|--------------------------------------------------------------------------------|
| Fwd Pkts/b Avg             | Average number of packets bulk rate in the forward direction                   |
| Fwd Blk Rate Avg           | Average number of bulk rate in the forward direction                           |
| Bwd Byts/b Avg             | Average number of bytes bulk rate in the backward direction                    |
| Bwd Pkts/b Avg             | Average number of packets bulk rate in the backward direction                  |
| Bwd Blk Rate Avg           | Average number of bulk rate in the backward direction                          |
| Subflow Fwd Pkts           | The average number of packets in a sub flow in the forward direction           |
| Subflow Fwd Byts           | The average number of bytes in a sub flow in the forward direction             |
| Subflow Bwd Pkts           | The average number of packets in a sub flow in the backward direction          |
| Subflow Bwd Byts           | The average number of bytes in a sub flow in the backward direction            |
| Init Fwd Win Byts          | Number of bytes sent in initial window in the forward direction                |
| Init Bwd Win Byts          | # of bytes sent in initial window in the backward direction                    |
| Fwd Act Data Pkts          | # of packets with at least 1 byte of TCP data payload in the forward direction |
| Fwd Seg Size Min           | Minimum segment size observed in the forward direction                         |
| Active Mean                | Mean time a flow was active before becoming idle                               |
| Active Std                 | Standard deviation time a flow was active before becoming idle                 |
| Active Max                 | Maximum time a flow was active before becoming idle                            |
| Active Min                 | Minimum time a flow was active before becoming idle                            |
| Idle Mean                  | Mean time a flow was idle before becoming active                               |
| Idle Std                   | Standard deviation time a flow was idle before becoming active                 |
| Idle Max                   | Maximum time a flow was idle before becoming active                            |
| Idle Min                   | Minimum time a flow was idle before becoming active                            |
| <b>Total features NIDS</b> | <b>80</b>                                                                      |

### 3.6. Types of attack in CIC-IDS2018 Dataset

Any network traffic resulting from an attack with the goal of causing damage or penetration into a computer system or network is considered malicious network traffic. These classes of malicious attack traffic make up the datasets chosen for this investigation.

#### 3.6.1. Benign or normal

It offers an accurate background traffic that is produced by the Benign Profile method. The B-Profile is responsible for profiling abstract human interaction behavior and generating realistic,

secure background network traffic. It is built using a variety of network protocols with various topologies and the abstract features of malevolent and human behavior..

### **3.6.2. Brute Force Attack**

In addition to breaking passwords, the primary network assault should be utilized to uncover secret pages and data within an online application. (Yang, 2020). Usually, software tools are used in password-breaking attacks to guess different password combinations repeatedly. (Morris, 2018). The FTP-brute force and SSH brute force branded attacks were executed using the versatile brute force tool patator. Getting illegal access to personal, business, system, and network accounts is the main objective of a brute force attack. To do this, attackers generally utilize automated tools like Patator, a multithreaded Python script that has the "ssh login" and "ftp login" modules. These programs use a trial-and-error approach by utilizing a large number of user names, passwords, encryption keys, login credentials and their possible combinations. According to the authors of (Sharafaldin, 2018) the CICIDS2017 dataset, which shares many of the same attack types as CSE-CIC IDS2018, report that follows a features that are the most useful in identifying the sort of brute force attack. Additionally, keep in mind that as TCP is the foundation of both FTP and SSH, all of the extracted network flow properties are tied to it. The investigation was conducted using the network intrusion detection dataset from 2018 to 2019, which only includes brute force assaults and benign traffic. The following terminology were used in this attack: TotLen Fwd Pkts, ACK Flag Cnt, Fwd PSH Flags, SYN Flag Cnt, Init Fwd Win Byts, Subflow Fwd Byts, and Fwd Pkts/s feature.

### **3.6.3. DoS based attacks**

A range of attack tools were used to carry out branded attacks. Stopping a resource, like a website, application, or server, from carrying out its intended tasks is the aim of a denial of service (DoS) attack. (I. Sharafaldin, 2019). The victim service can stop being available to authorized users if too many requests are made within a short period of time, and challenging it (DDoS Evaluation Dataset (CIC-DDoS2019), 2019). There are two kinds of Dos attacks types considered in this research study which are called DoS attacks-Golden eye and DoS attacks hulk (A. Y. Nur, 2021).

### **3.6.4. DoS attacks golden eye type**

The attack vectors called HTTP Keep Alive plus no caches are employed by the HTTP DoS test tool in golden Eye Dos attack types. Testing a website's vulnerability to DoS assaults is its primary usage. With the target website, this utility first establishes a complete TCP connection. It then makes repeated attempts to maintain the connection by sending requests at regular, long-term intervals. When a server is augmented by many attacking machines, commonly referred to as botnets, the maximum number of concurrent connections is rapidly reached, rendering the server incapable of processing requests from active users.

### **3.6.5. DoS attacks hulk type**

Python was used to create the DDoS assault tool hulk flood. The objective is to deplete the target's resources by producing distinct and disguised requests to one or more URLs from numerous attack machines. This tool's ability to generate a different pattern for every request sets it apart from other HTTP Flood utilities. In this manner, getting around intrusion detection and prevention systems is made considerably simpler.

### **3.6.6. DoS Attacks lowering HTTP Test**

To exploit slow HTTP DoS vulnerabilities, a tool called Slow HTTP test was created to simulate several Layer of DoS attacks. The typical method for carrying out slow HTTP assaults is to send a Web server HTTP requests slowly (Zavrak, 2020). Servers typically wait for data transfers to be completed if requests are incomplete or the transfer rate is too sluggish. Legitimate users' inbound requests can no longer to be answered at this period since the server's concurrent connection pool has reached its maximum limit. So, it requires fewer resources than other DoS attacks, this kind of attack is significantly easier to carry out. Many related attacks can also be carried out using this attack type, such as the Apache Range Header attack, which drains the pool of concurrent connections, the Slow Loris, Slow http post, and the slow read attack (based on the TCP persist timer exploit), which causes the server to use a lot of memory and CPU.

In this technique, the attacker uses the HTTP GET request to get more connections than the server is allowed to have. By allowing the attacker to establish several HTTP connections to the server, this renders requests from genuine clients to the server inoperable.

### **3.6.7.DoS attacks slowloris**

The Slowloris Perl-based application is used in this denial-of-service attack. Initially, a full TCP connection is made to the target server. (Frazão, 2019)The server gets legitimate, partially formed HTTP requests at predetermined intervals to maintain the connection and keep the sockets active. As a result, no new connections can be made and inbound requests requesting new open sockets will be rejected (Wang Z. J., 2021). The features that are most useful for identifying DoS attacks are listed above. The analysis is completed by merging the datasets from Friday, February 16, 2018, and Thursday, February 15, 2018, which solely contain benign traffic and DoS attacks.

### **3.6.8.Brute force web**

This types of attack is specially, a written piece of code leveraging the internal Selenium framework is used to carry out this attack. Damn Vulnerable Web App (DVWA), the victim's system, has multiple vulnerabilities. Typically, brute force attacks are carried out with automated tools and pre-made or self-prepared dictionaries that include word lists of known pages, username/password pairings, and other information. Because of accounts with weak password rules and username/password combinations, they are very good at hacking into networks that are vulnerable. are performed in order to gain access to secret pages or material within a web application. By sending HTTP GET or POST queries to the server with a word list of known pages, the attacker can easily access the available pages on the victim server. This category includes the following types of attacks.

### **3.6.9.Brute force XSS type**

Similar to brute force web attacks, this attack also uses automated code written in the Selenium framework on the DVWA target machine. This attack is carried out by inserting malicious code, usually client-side scripts, onto the victim website. Attackers are able to get beyond security measures and private data stored by users' browsers for use on the compromised website in this manner. When user input is not properly vetted or encoded, these attacks may succeed.

### **3.6.10. SQL Injection types**

Additionally, DVWA is the target of this attack, which involves a weak PHP or MySQL web application. This attack aims to extract sensitive data by manipulating the database through SQL

command injection. When the website lacks input validation or accesses the database against the least privilege concept and these attacks may succeed. The analysis is carried out by merging the datasets from Friday, February 23, 2021, and Thursday, February 30, 2023, which solely contain benign traffic and Web attacks. The features listed below have been shown to be the most effective in identifying DoS attack.

### **3.6.11. Infiltration**

To carry out this attack, a weak application is used. For instance, Adobe Reader version 9 is being used maliciously. The infected file, ready to be opened by the susceptible application, is sent by the attacker to the victim via email. In the event that the exploitation step is successful, the victim's computer is anticipated to have a backdoor installed, joining the botnet and connecting to Command and Control (C2C) servers in order to receive commands from the attacker and transmit private data (such as credentials, sensitive files, and the results of network reconnaissance to the attacker. One of the scenarios in our dataset is where a victim downloads a malicious file from Drop box, as described in (CSE-CIC-IDS2018 on AWS, n.d.). Subsequently, an insecure program opens the file, enabling it to search the network and gather private data. By combining the information from Wednesday, February 28, 2021, and Thursday, March 01, 2022, which only contain infiltration and benign traffic used in this study. The infiltration attack type include the following feature.

### **3.6.12. DDOS attack types**

Distributed denial of service (DDoS) assaults are carried out using the low orbit Ion Cannon (LOIC) open source tool. The primary aim of the LOIC tool are DDoS and DoS attack execution and network stress testing. This malware floods the target services with phony TCP, UDP, or HTTP packets, creating enormous volumes of network traffic. Depleting the victim's network or application resources is the aim of any DDoS attack. For these types of attacks to be this successful, multiple computers must be used.

### **3.6.13. DDOS-HOIC types**

To replace LOIC, the High Orbit Ion Cannon (HOIC) is being developed. It is also a popular open-source tool for denial of service attacks and network stress testing. Four distinct attack computers utilize the HOIC program to create this dataset using DDoS attacks. The top features that influence the detection ratio are those that are driven below. The study is carried out by

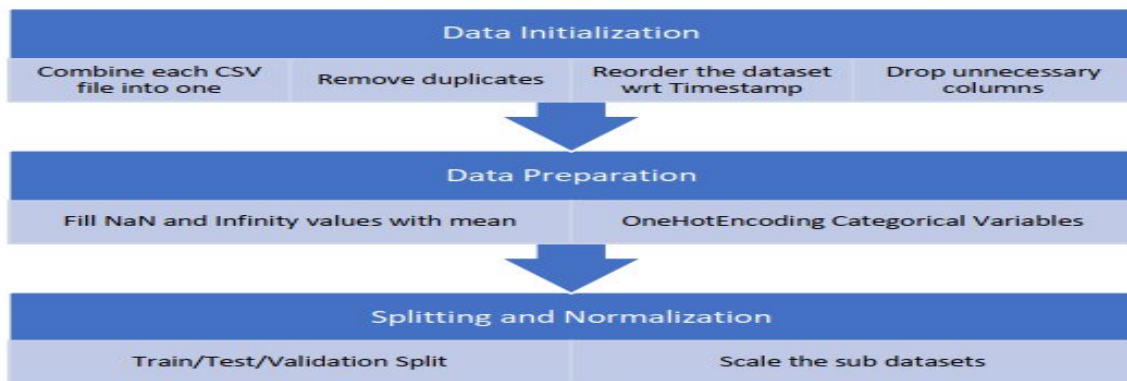
merging the datasets from Tuesday, February 21, 2021, and Friday, February 25, 2021, which solely comprise DDoS attacks and benign traffic. This type of attack contains the following important features .

### 3.7. Botnet attack types

An attacker can remotely manipulate a network known as a botnet. It consists of multiple compromised computers under the attacker's control that have been infected with bot malware. An attacker can send commands to individual bot machines via the botnet, and the attacker can receive sensitive data from the bot machines. In our instance, Zeus is a Trojan horse malware package that is employed to facilitate botnet connectivity. Zeus has the ability to capture sensitive data from the victim by using form-grabbing and man in the browser keystroke tracking. The target computers are also infected by the Ares botnet in addition to Zeus. There are two programs which make up Ares, called Python remote access tool and agent program or a command and control server. In addition to opening remote shells, it may download and upload files, take screenshots, log keystrokes, and remain persistent on the victim's computer. Generally , this types of attack comprise the following features.

### 3.8. Preprocessing phase

The aim of this thesis is using a multi class classification technique in order to develop a model



**Figure 24: Data Preprocessing Methodology**

that improve existing network intrusion detection systems in predicting benign traffic with 14 different attack types. The main procedures used in the research study are proposed here. In this research study, three primary phases of the preprocessing task were identified including, data initialization, data Preparation, and split or normalization. In this stage, the process procedure were represented in Figure 3.5

Several separate CSV files from several days were combined into a single file to create the dataset. Most of the related work focused on specific attack types seen in individual CSV files. But because the LSTM model is a deep learning algorithm, it also considers historical data during training. Furthermore, by absorbing as many samples as possible and understanding the regular traffic behavior, this is more rational and compatible with the characteristics of real-world traffic, particularly when it comes to classifying benign traffic on all days and identifying anomalous activity. Because they depend on the prior traffic, other attack types including brute force, botnet, infiltration, DoS and DDoS, and others are also taught using the entire preprocessed dataset. Duplicate rows were eliminated once the dataset was concatenated into a single CSV file. The dataset is then rearranged in accordance with the timestamp feature. As previously stated, the LSTM model heavily relies on historical data. We feed the model batches during model training, iterating through the batches for every epoch. We need to examine the traffic quickly in order to identify the attacker's actions more accurately.

The columns Src IP, Src Port, and Dst IP were removed for forecasts data set that were not dependent on specific socket information. So, Dst Port is useful for identifying particular assaults including, web attacks, application layer DoS attacks, FTP and SSH brute force attacks respectively. In addition to the previously mentioned problem, several columns were found to have insufficient influence on the classification. For instance, it is found that the values in the Bwd PSH Flags and Bwd URG Flags columns remain constant across the sequence of the dataset. Furthermore, using "Flow ID" as a feature for attack detection is useless because it is a unique variable created for every flow. In the preceding stage, the dataset is rearranged using the "Timestamp" feature. However, this column is also removed prior to training since we do not want our model to identify benign and attack traffic based on their timing. Consequently, the ultimate feature size is reduced from 86 to 78.

The purpose of the data initialization phase is to get the data ready for the splitting and normalizing phase. As the columns for Flow Byts/s and Flow Pkts/s infinity values were found and Nan values were used in place of these values, and the dataset was then divided into two NumPy arrays designated as X and y based on the independent and dependent variables. Only the labeled column in our dataset which shows the various attack kinds, is selected as dependent. The resulting shapes are (16537250, 76) for the X and 15677150, for the y respectively. The mean value of the associated column was then filled in for the Nan values of X using the Simple

imputer from scikit-learn. In essence, Simple Imputer modifies the dataset by filling in the missing values in accordance with the specified plan. There are four possible method used in this process including mean, median, most frequent, and constant method. For this study, we employed the mean method as default technique.

For examination of our dataset, we find that every independent variable was represented numerically. The dependent variables, however, were string and category in nature.

All values should be translated to numerical form prior to the training phase, as the machine learning the model requires numerical values. As a result, the Labeled column, y, is converted using a categorical encoder into a numeric array. In binary classification method, all columns of benign traffic is converted into 0.0 values whereas all attack traffic is converted into 1.0 values in classification matrix. By using multi class classification mode, we apply one hot encoding method to employee the model. The hot encoder method encodes categorical features as a numeric value. Using this approach, the data is fit into one-hot numeric values containing 0's and 1's which is different from other method as the labeled encoder LSTM, CNN, and DNN are examples of deep learning models that typically assign more weight and features to higher values during model training. This method misleads the model to capture the relationships while making incorrect estimations. Due to the multiclass classification of various attack types, which gives each category similar weight in terms of intrusion detection system, this scenario records higher performance in our study. This results in a new shape of the y array and one of the hot encodings utilizing (16765150, 15), with a distinct data set record for every network assault label. The scikit-learn train-test-split ratio is used as a starting point for the splitting and normalizing procedure. Twenty percent of the entire dataset is the test data set ratio. With the same ratio, the remaining 80% of the data set is divided into train and validation. The total data set of network intrusion dataset ratios are obtained and used as follow.

Table3.4. attack and Normal class distribution of data set 1

| <b>Network attacks class</b> | <b>Training set</b> | <b>Testing set</b> | <b>Validation set</b> | <b>Total dataset</b> |
|------------------------------|---------------------|--------------------|-----------------------|----------------------|
| Benign or normal traffic     | 75%(858443)         | 15%(2682916)       | 10%(2146601)          | 13,413,990           |
| Total attack data set        | 75%(1448903)        | 15% (452514)       | 10% (361743)          | 2,263,160            |

|                                   |         |         |         |           |
|-----------------------------------|---------|---------|---------|-----------|
| Benign class                      | 8584473 | 2682916 | 2146601 | 2,263,160 |
| Bot net attack class              | 178520  | 55866   | 44198   | 278584    |
| Brute Force –Web                  | 398     | 121     | 92      | 611       |
| Brute Force –XSS                  | 148     | 39      | 43      | 230       |
| DDOS attack HOIC                  | 416771  | 130521  | 104454  | 651746    |
| DDoS attacks-LOIC UDP             | 1107    | 345     | 278     | 1730      |
| DDoS attacks-LOIC-HTTP            | 368509  | 115302  | 92348   | 576159    |
| DoS attacks golden Eye            | 26565   | 8211    | 6628    | 41404     |
| DoS attacks-Hulk                  | 264264  | 82378   | 65519   | 412161    |
| <i>DoS attacks Slow HTTP Test</i> | 3881    | 1202    | 967     | 6050      |
| DoS attacks-Slowloris             | 6422    | 1976    | 1612    | 10010     |
| FTP-brute force                   | 12589   | 3979    | 3172    | 19740     |
| Infiltration attack class         | 103656  | 32189   | 26021   | 161866    |
| SQL Injection attack class        | 59      | 18      | 10      | 87        |
| SSH-brute force                   | 66014   | 20367   | 16401   | 102782    |

As Table 3.3. Above represents that, the attack and normal class distribution among train, test and validation datasets for the multi class classification model. The sub data sets are scaled between 0 and 1 for normalization once splitting is finished. The min max scale from scikit learn is used to do this. This scale keeps the original distribution's shape without distorting the original data, which is why it was chosen. Min max scale only has to use the translation formulas represented in below algorithm to scale and translate each feature in accordance with the specified range of 0 and 1 by default.

### 3.9. Preparation of the data set

A survey of the literature reveals a new network intrusion detection evaluation dataset. This CIC-IDS 2018 dataset was developed by the Institute of Cyber Security at the University of New Brunswick. The purpose of this new benchmark dataset is to accurately and efficiently evaluate network intrusion detection systems that rely on deep learning. In this study, the new CIC-IDS2018 benchmark dataset is used to assess the performance of NIDS devices.

The study of previous datasets, such KDD-Cup99 and NSL-KDD, exposed flaws that make the way for the development of more recent datasets that fixed the issues identified. For network intrusion detection system, one of the most recent datasets is CIC-IDS2018 ((CSECIC-IDS2018), 2022). It has the advantage of being offered in CSV files with flow-based features as raw data. A dataset for NIDSs and IPSs called CICIDS20178 was produced by the Canadian Institute for Cyber security. Additionally, this dataset offers gathered unprocessed network traffic data that is divided daily across ten CSV files that may be downloaded from the Amazon cloud. CICFlowMeter-V3 is a network traffic flow generator and analyzer that is used to extract over 80 attributes from network traffic data. (Edeh D. N., 2021) (Canadian Cyber Security Institute, undated).

The ISCX Group's CICIDS-2018 dataset, which comprises deep Learning CSV data, was used in this research work (Hagar & Gawali, 2022). Each comma separated value (CSV) file in deep Learning CSV is comprised of eight traffic monitoring sessions. Malicious and normal communications can both be found in this file. Table 3.1 below shows that, the third column provides a more detailed description of the attack traffics. The dataset includes benign and ordinary traffic as well as 14 distinct types of intrusions. Two characteristics, or columns, titled "Fwd Header Length," are included in the dataset. One of them must be removed because it is redundant. There are just 77 features left to look at after the duplicate characteristics are removed. According to the CICIDS-2018 data, data with a large class imbalance will negatively impact false alarm and detection accuracy. A new labeling attack traffic is established as mentioned in Table 3.4 by using the solutions proposed by (Al-Fawa'Reh, Al-Fayoumi, Nashwan, & Fraihat, 2022).

### **3.10. Data set**

One of the most important aspects of an intrusion detection system is the essential core of choosing an appropriate dataset. Using out-of-date datasets could not yield the desired or impartial results because network assaults evolve over time . As a result, we choose to use the CSE-CICIDS2018 dataset, which was produced on the AWS (Amazon Web Services) cloud environment and is among the most current IDS datasets to be published. (Farhan, Maolood, & Hassan, 2020). Because it contains both typical cloud network traffic and recent assault traffic,

and because it is publicly available data set , that we have used for implementation of our research

It is An essential tool for assessing the efficacy of intrusion detection models, as the CES-CICIDS2018 dataset, which comprises seven unique scenarios that repeat different types of network intrusion. Table 3.4 below illustrates that, how each scenario provides a varied testing environment by combining instances of both benign and malicious scenarios. A high-quality dataset is necessary for the implementation of IDS with deep learning in order to train and test an algorithm that enhances the real-world environment. The Canadian Institute of Cyber Security provides the standard data set CICIDS2018, which is used for training and testing in this study. This dataset ensures better performance because the test and train sets are not duplicated. Because the dataset only contains a respectable number of traffic records, the experiment can be run on the complete dataset rather than just a subset of it.

Table 3.5. the portion of data set for network attack class 1

| No | Types of attack               | Scenarios              | Size of the data set |
|----|-------------------------------|------------------------|----------------------|
| 1  | Brute Force Attack Simulation | Benign                 | 667,626 instances    |
|    |                               | FTP Brute Force        | 193,360 instances    |
|    |                               | SSH Brute Force        | 187,589 instances    |
| 2  | DoS Attack Types              | Benign                 | 996,077 instances    |
|    |                               | DoS Golden Eye         | 41,508 instances     |
|    |                               | DoS Slowloris          | 10,990 instances     |
| 3  | DDoS LOIC-HTTP                | Benign                 | 7,372,557 instances  |
|    |                               | DDoS attacks-LOIC-HTTP | 576,191 instances    |
| 4  | Advanced DDoS Techniques      | Benign                 | 686,012 instances 8  |
|    |                               | DDOS attack-LOIC-UDP   | 360,833 instance     |
|    |                               | DDOS attack-HOIC       | 1730 instances       |
| 5  | Web Application Attack        | Benign traffic         | 1,048,213 instance   |
|    |                               | Brute Force-Web attack | 1,048,213and 249     |

|   |                      |                 |                     |
|---|----------------------|-----------------|---------------------|
|   |                      |                 | instance            |
|   |                      | Brute Force-XSS | 79 instance         |
|   |                      | SQL Injection   | 34 for each attack  |
| 6 | Infiltration Tactics | Benign:         | 1,048,009 instances |
|   |                      | Infiltration    | 151 instances       |
| 7 | Botnet Detection     | Benign          | 762,384 instances   |
| 8 | 1107                 | Boot            | 286,191 instances   |

We have created a model to evaluate certain attack types and benign traffic in each scenario, ensuring the thorough evaluation and improvement of detection algorithms. This systematic approach improves the model's performance and accuracy, enabling it to handle both simple and complicated network threats.

~~On the other hand, the convolutional neural network (CNN) is a type of deep learning model that is used for image classification and object detection. It is a type of neural network that is designed to process data that has a grid-like topology, such as images. The CNN is composed of several layers, including an input layer, one or more convolutional layers, and a fully connected layer. The convolutional layers are responsible for extracting features from the input data, while the fully connected layer is responsible for classifying the features. The CNN is trained using a large dataset of images, and it learns to recognize patterns in the data. Once trained, the CNN can be used to classify new images or to detect objects in images. The CNN is a powerful tool for image classification and object detection, and it has been widely used in many applications, including medical image analysis, facial recognition, and autonomous driving.~~

Convolutional network or convolutional neural network or CNN is an extension to traditional feed forward networks (FFN) in the context of inspiring the biological factors. This category includes research works which have based their intrusion detection systems on convolutional neural networks. It involves 8 layers of the neural networks and used the pooling layer among them. Moreover, they used a learning rate of 0.001 and achieved high Accuracy rate. we employed data to image conversion methods and convolution architecture to identify infiltration, it only offered an accuracy rate of efficiently.

The main convolution layer component in the CNN architecture is the convolutional layer. In order to extract features from data sets, the convolution layer trains IDS data features from the input data while maintaining the relationship between features. The CNN divides the result by the entire number of pixels, enhances the result, and then places the filter value on the map. The process is then repeated for the other filters once the feature has been moved to each other's location in the image and the matrix output has been obtained. Through this layer, the filter is applied to every possible location on the data set.

### 3.12. Convolutional Neural Network(CNN)

CNN is a kind of neural network that needs to identify relevant feature representations of the input data. The first layers in this types of architecture are often a group of convolutional feature

extraction that are applied in combination with a picture after it has been passed through several learnable filters. The filters that are employed function as a sliding window that covers the entire input data set. The overlapping distance is known as the stride, and the final outputs are called feature maps. Numerous convolution kernels make up each CNN layer, which is utilized to produce unique feature maps. On the feature map of the next layer, a neuron is related to areas of neighboring neurons. To generate the feature map, the kernel is shared by all input spatial positions. Following convolution and pooling layers, the classification is completed by one or more completely linked layers. The convolutional operation between input feature maps and a convolutional layer in the CNN architecture is given by the following formula.

### 3.12.1. Network Architecture

The CNN-MCL layer is used to create a CNN that can distinguish between the typical and anomalous flows detected by a NIDS. Figure 3.5. below shows that, the recommended architecture's general layout and down to the smallest detail in each layer. This architecture can discover new connections between feature maps of deeper layers by extracting a higher level of representation of the previously learnt normal/abnormal properties. The last convolution layer's output is flattened before being sent to the classification block, which is made up of a fully linked layer and a Soft max. . The various layers that make up the CNN's architecture are described in detail, along with the recommended architecture.

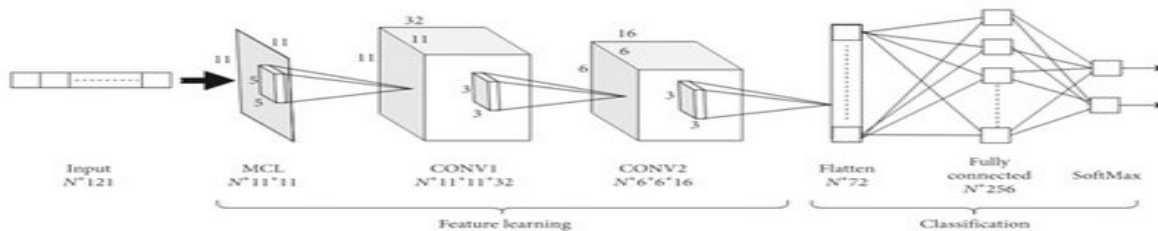


Figure 3.2.CNN Network architecture 1 (Gamage, 2020)

### 3.12.2. Input layer

The selected feature is represented as a unit in the CNN model's input layer. The relu activation function is used by the input layer. While fully connected networks employed an even denser network with only 256 units and 0.1 dropout rate to connect to the final layer, CNNs used layers with 128 and 256 filter units. The output layer has a soft max activation, whilst each

convolutional neural layer uses a relu activation function. One can calculate the categorical cross-entropy loss function and obtain the sample loss by using Equation 3.2 for the CNN.

$$loss = - \sum_{i=1}^{output-size} yi * log i$$

The number of  $i^{th}$  scalar values in the model output is represented along with the scalar value and  $output = size$  is the output target value.

$$loss = - \frac{1}{outputsize} \sum_{i=1}^{outputsize} yi * log yi + (1-yi) * log (1-yi)$$

Since the DNN model uses the soft max activation function in the output layer, the derivative of the loss function with respect to each weight and value in the training set may be computed using Equation 3.3. Equation 3.3 below illustrates how the relu activation function is used for the remaining hidden layers.

$$ReLU = \max (0, x)$$

The input value is returned by the relu function if it is positive, and 0 otherwise. This enhances the model's capacity to learn intricate correlations between the input features and the target variable by introducing non-linearity. Adam is an adaptive learning rate optimization algorithm that is applied in this model. Adam enhances the training process by determining unique learning rates for every neural network parameter, in contrast to conventional gradient descent. The gradient's first and second momentum are estimated, and the learning rate is adjusted for each weight.

### 3.12.3. Max pooling

This CNN architectural model uses 2 stride and  $3 \times 3$  max-pooling layer. The sliding window's local neighborhood has a maximum value in the max-pooling layer. With less dimensionality in the feature maps, there is less chance of over fitting and less computing expense for training. This is the goal of this layer. A high-dimensional feature map volume is specifically provided by the set of parallel convolutional layer. Thus, while preserving the most representative features, pooling layers aid in subsampling and accuracy enhancement. The two pooling layers used in this design have decreased the feature map dimensions from  $11 \times 11 \times 32$  and  $6 \times 6 \times 16$  to  $6 \times 6 \times 16$  and  $3 \times 3 \times 8$ , respectively. The association between feature maps—which are a linear

combination of features across channels in the same area is then discovered using this kind of mapping form. Then,  $3 \times 3 \times 8$  is resized to 72 layer 4 outputs.

A new connection between these feature maps is taught after learning local spatial association in a receptive field (local region/patch convolved with a filter) inside the same feature map. This is done in order to develop the previously learned hierarchical features.

#### **3.12.4. Reshaping Layer**

The flow shape of the input vector in NIDS is typically indicated by a  $N \times V$  or  $N \times 1 \times V$  vector, where  $V$  is the number of features that characterize the flow and  $N$  is the batch size. As in earlier work, the input shape is transformed from 1D to 2D. The first layer of this architecture has zero values for the remaining values since it can reorganize the input vector into an  $11 \times 11$  2D matrix and has a feature count of less than 121 for the input vector size. Layer 1 receives a  $N \times 11 \times 11$  pattern. We selected  $11 \times 11$  patches since the feature number in the tested dataset is greater than 100 but less than 121. This architecture can work with other sizes, like  $n \times m$ , but the test and train feature arrangements have to match.

#### **3.12.5. Dense Layer**

Depending on the quantity of dataset attributes, the values of the thick layers in the suggested model vary for every dataset. For instance, the first layer of the CISCIS2018 dataset has 77 variables available. One of the neural network's most well-tested functions is the activation function (Ren S & 39(6):1137–49., 2017). The following formula defines the non-linear ReLU function called ReLU function.

$ReLU(x) = \max(0, x)$  where  $x$  is the input.

#### **3.12.6. Drop out layer**

The drop out layer in the proposed model is thought to come after each dense layer, with values ranging from 0.15 to 0.5. This layer unintentionally removes and releases neurons, preventing the network from over fitting and allowing the network to retain data and not be disturbed in predicting the testing data. The training period was extended by using dropout layers to prevent over fitting (Yamashita R & 9:611–29., 2018). Throughout the training phase, it randomly sets some input units to 0 to prevent over fitting.  $1/(1 - \text{rate})$  is used to scale inputs that are not set to

0 such that the overall sum remains constant. By lessening the reliance of the neurons that follow it on them, dropout layers aim to reduce the possibility of over fitting.

### 3.12.7. Fully connected (FC) layer

In contrast to a dense layer, a completely connected layer is one that is connected to every neuron in the layer before it. It uses an activation function, like Soft Max or Tanh, as stated in equation 3.4.and 3.5 to classify the training inputs from the preceding layers according to their suitability.

$SoftMax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$  and  $Tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$ , where  $x$  is the input.

Each input and output in one or more dense layers—also referred to as fully linked layers—are connected by the weight during model training. The final convolution or pooling layer's output feature maps, which are usually flattened or transformed into a one-dimensional (1D) array of numbers (or vectors), are linked to these layers. Through a subset of fully connected layers, the characteristics of convolution and down sampling layers are communicated to the network's final outputs, such as the probabilities for each class in classification tasks. There are usually as many output nodes as classes in the final fully connected layer.

### 3.12.8. Output layer

In our suggested system, the output layer is the last stage of deep learning. The output layer uses the Soft max function to analyze the logits from the Dense Layer and produce a probability distribution for each of the possible output classes. Our model's main goal is to classify security data into classes of benign and malicious network traffic, and the Soft max function is perfect for managing multiclass classification systems.

### 3.12.9. Activation function .

Since they introduce non-linearity and make it possible to simulate intricate data interactions, activation functions are crucial to neural network architecture. One important factor that directly affects the CNN and LSTM models' ability to learn and represent information is the choice of activation functions. The information below lists the precise activation functions that were employed in our investigation. Usually, a convolutional layer is followed by an activation function, which is a nonlinear mapping. After that, each value in each convolutional block's feature maps receives an application of this function. There are numerous kinds of activation

functions. The ReLU activation function has been successfully implemented in computer vision applications.

In CNN layers, the Rectified Linear Unit (ReLU) activation function is selected. ReLU is a popular activation function in deep learning models that is known for being straight forward and efficient. The definition of ReLU is represented mathematically as shown in equation below .

$$\text{ReLU}(x) = \max(0, x)$$

where the activation function's input is represented by the symbol  $x$ . ReLU creates non-linearity by limiting the input to positive values only, i.e., thresh holding it at zero.

The LSTM layers are activated using the Sigmoid function. Recurrent neural networks (RNNs) commonly adopt sigmoid, a sigmoidal activation function, since it can compress input values into the range  $[0, 1]$ . It can be used to express probabilities or binary choices because of this feature. In the above equation, the definition of a sigmoid function is demonstrated mathematically.

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$$

where  $x$  stands for the activation function's input. The LSTM layers can control information flow by gating methods thanks to the distinctive S-shaped curve of sigmoid, which makes it easier to model sequential dependencies. Our CNN and LSTM model's activation functions are strategically chosen depending on how well they capture and represent the underlying data patterns, which improves the model's learning ability and performance in threat detection tasks.

To achieve optimal performance and generalization, the combination of both CNN and LSTM model are trained through a rigorous process of parameter optimization and algorithmic selection. The particular training specifications used in our research study are described in the sections below.

### **3.12. 10. Optimization Algorithm**

Stochastic Gradient Descent (SGD) is the algorithm used for optimization. SGD iteratively updates the model parameters by changing them in a way that minimizes the loss using the

gradient of the loss function with respect to the parameters. n. The SGD update rule can be found mathematically as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} \mathcal{L}(\theta_t)$$

where  $\eta$  represents the learning rate indicating the gradient of the loss function with respect to the parameters, and  $\nabla_{\theta_t}$  represents the model parameters at iteration  $\theta_t$ .

### 3.13. Learning Rate Scheduler

Using a learning rate scheduler, the learning rate is adaptively altered during model training. The learning rate scheduler automatically adjusts the learning rate according to preset parameters, including validation performance or epochs. By eliminating oscillations and enhancing convergence, this adaptive correction helps with model training.

#### 3.13.1. Dropout Regularization

The model design incorporates dropout regularization to improve generalization and avoid over fitting. By randomly deactivating a portion of the network's neurons during training, dropout lessens dependence on particular characteristics or pathways. The dropout operation's mathematical expression is as follows.

$$y = \frac{1}{1-p} \cdot x$$

If  $x$  is the dropout layer's input,  $y$  is its output, and  $p$  is the dropout probability, which is the proportion of neurons to deactivate. Together, these training guidelines guarantee the performance and generalization capabilities of the CNN and LSTM models, enabling them to effectively recognize and examine complex patterns in the NID data set while lowering the risk of over fitting. Model quantization is the critical process of changing a model's weights and activations from a higher precision format to a lower precision one. This enhancement is necessary to optimize the model for deployment in resource optimization environments, which are common in NID scenarios. The mathematical foundations of quantization are captured by this equation, as seen below.

$$Q(v) = \text{round}\left(\frac{v}{s}\right) + z$$

where  $z$  is the zero-point that represents the actual value zero,  $s$  is the scale factor, and  $v$  is the original value. The accuracy reduction process, which is crucial to increasing computer effectiveness, is highlighted in this equation. The efficiency and generalization capabilities of the CNN and LSTM model are enhanced by combining these training criteria, which reduces the likelihood of overfitting and enables it to effectively detect and analyze challenging patterns in data. Taking a care of the computational limitations that are common in IDS model is necessary before deploying an advanced model for CNN and LSTM model training used for network security. This section describes how we improved our model using sophisticated optimization methods called model quantization and pruning that were intended to increase efficiency without appreciably lowering performance.

### 3.13.2. Model quantization

Model quantization is the crucial process of changing a model's weights and activation functions from a higher precision format to a lower precision format. Deep learning applications frequently involve resource utilization conditions, which require the model to be optimized for deployment. To accomplish so, this modification is essential. The mathematical concept of quantization is encapsulated in the following formula.

$$Q(v) = \text{round}\left(\frac{v}{s}\right) + z$$

The scale factor is denoted by  $s$ , the original value by  $v$ , and the zero-point, which represents the real value of zero, by  $z$ . In order to increase computational efficiency, this formula highlights the precision reduction procedure.

### Regularization techniques

Regularization, also known as Lasso, or Least Absolute Shrinkage and Selection Operator, is a kind of linear regression that adds a penalty term to the loss function based on the absolute size of the coefficients itself. Known as L1 regularization, this penalty term can be computed using equation 3.5 below.

$$L_1 = \lambda \sum_{i=1}^n |w_i|$$

In this case,  $w$  is the regression model's coefficients, and  $\mu$  is a hyperparameter that controls the strength of the regularization.

### **3.14. Classification of network attacks**

The CNN architecture is made up of convolution, pooling, and fully connected layers. One or more completely linked layers, a pooling layer, and numerous convolution layers are layered, and recurrence is a well-known design feature. (Krizhevsky A., 2017). The process that transforms input data into output data across these levels is called forward propagation. demonstrates how a neural network classifier with a Soft Max activation function in the output layer is used to categorize the output features that the convolutional layers have learned. When all neurons are fully connected to a set of probability values, the last layer learns features where each neuron's output equals 1. These abnormal flows can be identified by choosing the intricate operation associated with the Soft Max layer neuron that has the highest activity level. More specifically, there are 256 neurons in this fully linked layer. This layer also finds new relationships between CNN's deepest convolutional features. Network traffic is categorized as either benign or attacked. After the model training procedure is over, the predictive model can identify one label for normal network traffic and ten labels for attacked network traffic.

#### **3.14.1. Binary VS Multiclass classification model**

The same decisions are taken to compute the output neuron's activation as in the hidden layer neurons in order to conduct binary classification. In contrast to the activation of the hidden unit, alternative decisions are made if multiclass categorization is necessary. provides a standard illustration of a neural network for multiclass classification. It has several output units, each of which stands for a different output class. The chance of each output unit's corresponding input class occurring is indicated. Multiclass classification makes use of the soft max activation function. When there are several output units, a vector of output values is represented by  $o(a)$  vector .

The sum of the numerators is the denominator, therefore normalizing this vector guarantees that the sum of all output probabilities is always 1. A number's exponent must be bigger than zero for this function to be strictly positive. By determining which class has the highest estimated

probability from the output vector  $o(a)$ , we may anticipate the real class of an input  $x$ . The input vector  $a_n$  is the pre-activation value of the neural network's last layer.

### **3.15. Model Training**

Using the CSE-CICIDS2018 dataset, the proposed model demonstrates how to successfully apply the convolutional neural network (CNN) paradigm for intrusion detection. (Gawali and Hagar, 2022). The proposed intrusion detection approach consists of seven major phases. Following the completion of all preprocessing and feature selection steps, we separated the dataset into a training and test set. Next, in order to identify intrusions in the cloud environment, we fed the CNN blocks model the training data. The preprocessed features were fed into the deep learning model so that it could be trained and tested. The preprocessed data or features were fed to the model during the training phase, and it began training using these features. Better outcomes were achieved by testing the model to evaluate its performance following the training phase. In certain cases, adjusting the hyper parameters was necessary to achieve the intended outcomes. The initial step was gathering the dataset, which served as the basis for training and evaluating the effectiveness of deep learning models for cloud intrusion detection. Prior to label encoding and data normalization, the gathered dataset was preprocessed in the second phase to exclude null or missing values. To effectively choose relevant attributes to train the deep learning models, we used a feature selection method based on a Pearson correlation coefficient matrix heat map in the third phase. SMOTE oversampling and under sampling were used to balance the dataset before it was divided into training and testing sets in the fourth stage.

We trained and evaluated deep learning models for cyberattack detection and classification in the fifth and sixth stages. Ultimately, we assessed the effectiveness of deep learning models that had been trained and suggested the most effective model for effective intrusion detection in cloud environments. To effectively detect intrusions, the suggested intrusion detection model for cloud settings employs a convolutional neural network (CNN) architecture with several layers and blocks. The ReLU function activates the first three Conv1D layers of the model, which have a kernel size of 6 and 64 filters. Next, batch normalization is used for efficient training, and max pooling 1D layers is used for down sampling. (Edeh, 2022).

For the network intrusion detection system that is part of the proposed study, we have employed a number of different methodologies. To train NIDS the data set, we used a variety of deep

learning (DL) models. The results of the different deep learning models are compared and contrasted once the classification process is finished utilizing deep learning model methodologies. (/master/NSL-KDD, 2019). Deep learning network architectures like CNN, RNN, DNN, and LSTM architectural models can be used to tackle the complexity of the current classifiers. Furthermore, due to their high accuracy and improved performance, deep learning-based network intrusion detection might be very important in network intrusion detection systems. CNN and RNN were therefore chosen for this study due to their features and effectiveness in categorizing objects that may be utilized in the process of distinguishing between malicious and authorized communications. In contrast, RNN, DNN, and LSTM models were chosen due to their ability to recall past occurrences, which could be crucial for categorizing the impacts of different (M. U. Oney and S. Peker, 2019).

This study's primary objective is to compare the effectiveness of using CNN and RNN models in NIDS. An experimental model will be used in the research study to evaluate the suggested study's effectiveness in a network intrusion detection system. The CNN algorithm was used to investigate and implement network intrusion detection using convolutional neural networks (CNN). (W. Lin, 2018). Using CNN model, a behavior based classifier learning model has been created for research purposes. The RNN model then models a series of packets, producing the micro flow properties (/Anihilakos/RNN-LSTM-Network-Intrusion, 2019). The suggested model uses feature-engineered raw data as input. Ultimately, a soft max classification is used to determine whether or not a micro flow is harmful.

### **3.16. Model Performance evaluation**

Following the training phase, we assessed the models' performance using a number of indicators. The model's performance was evaluated using classification problem accuracy, precision, recall, and F1-score. All trained models were given access to the test set, and we assessed each model's performance separately using the previously mentioned parameters.. (Farhan, Maolood, & Hassan, 2020). Following the computation of the CNN-based model architectural performance score, we evaluated the performance of the proposed model in comparison to previous studies. The prediction model's performance was categorized using the confusion matrix. Furthermore, it was discovered that the performance metrics for binary classification were accuracy (Acc), precision (Pre), recall (Rec), and F1 score (F1), whereas the performance indicators for

multiclassification were recall and false positive rate (FPR). The computation of each performance indicator is explained in detail below. (Kunang, Nurmaini, Stiawan, & Suprpto, 2018).

The accuracy and F1 score are used in this experiment to assess the model's performance. The model is trained using 80 percent of the data, with the remaining 5 percent going to validation and 1 percent going to testing. The classification metrics used to evaluate the model performance in this proposed thesis study include accuracy (CA), precision, recall, and F1 scores. Despite being difficult to understand and use, the notion of entropy is fortunately simple.

## Accuracy

The false alarm rate and detection rate were calculated and arranged to evaluate the model's performance. Table 3.5 displays the classification of the data. True positives (TPs) happen when actual attack data are classified as attacks, whereas false positives (FPs) happen when attack data are classified as normal. Likewise, false negative (FN) indicates that actual attack data were predicted to be normal, while true negative (TN) indicates that normal data was classified as normal.

Table 3.6 evaluation matrix of the model 1

|                     | Attack traffic | Normal traffic |
|---------------------|----------------|----------------|
| Actual attack class | TP             | TN             |
| Normal class        | FP             | FN             |

When data are accurately classified, it means that real attack data are labeled as attacks and normal data are classed as normal. This is known as accuracy.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

The detection rate is the frequency at which actual attack data are mistakenly categorized as attacks, as shown in equation 3.6 below.

$$\text{Detection rate} = \frac{TP}{TP + FN}$$

False alarms rates are the bases for which a regular data are wrongly classified as an attacks type as expressed by the following equation.

$$\text{False alarm rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

### **Precision:**

It is the proportion of true positives (TP), which is calculated by summing up all relevant classes, or false positives plus truly positives. You can use the following formula to express it.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

### **Recall**

It is the percentage of genuine positives in the total of True Positives and False Negatives. The equation of this performance measure is described using diagram below.

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

### **F1 score:**

The experiment will employ the F1 score due to the dataset's imbalance. The F1 score is calculated using the harmonic mean of precision and recall. The diagram below serves as an example of it.

$$\text{F1 Score} = \frac{2 * \text{precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

**A confusion matrix:** - The performance of an NID predictive model is similarly described by this evaluation metric.

## **3.17 .Development tools**

Different resources are required to do the implementation of this research work, which are categorized as hard ware and software resource. The software tools needed for this study includes Anaconda, matiolib, Keras, tensor flow , Visio 2010 and Python 3.6 version . Python is an on line Google cloud storage and access platform that support and contains all python

language libraries to implement machine learning application efficiently. Google colab support GPU and TPU video graphics that provides an appropriate processing system. The implementation of this research is done using the python 3.8 version of the Jupiter notebook since it has the core library called integrated development environment.

## **CHAPTER FOUR**

### **RESULT AND DISCUSION**

#### **4.1. Introduction**

The performance of the suggested NID model was examined using a series of experiments and discussions conducted on the CICIDS2018 datasets. The experimental work was done to validate CNN, RNN, and LSTM models and compare their performance with other deep learning models will presented in this chapter. Once the deep learning architectures were implemented using open-source code that we had significantly altered to meet our study's requirements, we conducted several tests with CNN, RNN-LSTM, and DNN to monitor and evaluate their performance.

The datasets we used for our experiments were first compiled in this part. Then, we go to the performance evaluation metrics of our proposed model and provide specifics about our

experimental setup. Lastly, we present our suggested model's experimental results and compare them with those of related studies.

## 4.2. Datasets

There are numerous datasets for intrusion detection assessments that include both normal and various network traffic information. We have used two NIDS datasets which are selected for Network intrusion detection including CIC-IDS2018, and CIC-DDoS2019 respectively. The CIC-IDS2017 dataset, developed by the Canadian Institute for Cyber Security (CIC), serves as the first evaluation dataset for our project. (Intrusion Detection Evaluation Dataset (CIC-IDS2017), 2022). The most recent network assaults and accurate benign traffic are included in our dataset. It includes a variety of attack types and protocol types. In order to evaluate our model, we use the CIC-IDS2018 dataset. The CIC-IDS2018 dataset covers the three years data of NID from July 5, 2022, July 9, 2024, and contains both typical normal attack and diverse attack data. Only on Monday was the innocuous data set acquired, but the assault data set was captured on earlier days.

The following network attacks were conducted: botnet, brute force FTP, brute force SSH, DDOS attack Goldeneye, web attack, infiltration, doS, DDoS slow HTTP test, doS attacks-Slowloris, DDoS attack-Hulk, brute force-XSS, DDOS-HOIC UDP, DDOS-LOIC HTP, boot net attack, SSH brute force attack, and SQL injection network attack. In this study, we tested our model solely using the attack dataset mentioned above, which is derived from the CIC-IDS-2018 dataset's Monday through Friday sample data, by counting the daily traffic record.

The CIC-IDS-2019 (DDoS Evaluation Dataset (CIC-DDoS2019), 2019), which was released by the CIC to address certain inherent issues with the CIC-IDS-2018 dataset . It is the second dataset that was employed in our model evaluation (A. Y. Nur, 2021). The dataset was extracted directly with help of TCP/IP connections and comprises eight features including connection period, protocol type, and accumulated traffic characteristic in each interval. The most current evaluation dataset we used for our model training is the Canadian Institute for Cyber security's (CIC) newly published CIC-DDoS2019 (Leevy, 2020).

| No | Network attack Classes | Training set(80%) | Validation set (10%) | Test set (10%) |
|----|------------------------|-------------------|----------------------|----------------|
|----|------------------------|-------------------|----------------------|----------------|

|    |                         |        |        |       |
|----|-------------------------|--------|--------|-------|
| 1  | Benign                  | 85844  | 8542   | 4473  |
| 2  | Bot net                 | 13520  | 12331  | 10232 |
| 3  | Brute Force –Web        | 16771  | 8542   | 6521  |
| 4  | Brute Force –XSS        | 18771  | 10321  | 7211  |
| 5  | DDOS attack-HOIC        | 12722  | 6211   | 2342  |
| 6  | DDoS attacks-LOICUDP    | 3881   | 601    | 621   |
| 7  | DoS attacks-LOICHTTP    | 26565  | 12322  | 5112  |
| 8  | DoS attacks-Golden Eye  | 12589  | 5511   | 2121  |
| 9  | DoS attacks-Hulk        | 46772  | 20121  | 13122 |
| 10 | DoS attacksSlowHTTPTest | 1121   | 212    | 211   |
| 11 | DoS attacks-Slowloris   | 41771  | 10112  | 8011  |
| 12 | FTP-BruteForce          | 64264  | 21212  | 12343 |
| 13 | Infiltration            | 68509  | 23112  | 11214 |
| 14 | SQL Injection           | 6422   | 2111   | 210   |
| 15 | SSH-Brute force         | 66014  | 26411  | 14322 |
|    | Total                   | 485536 | 167672 | 98066 |

Table 4.1. network attacks classes with data set distribution 1

The collection includes real-world data and a variety of contemporary and benign network assaults. At the application layer, numerous new attacks have been carried out utilizing TCP/UDP-based protocols, and a new taxonomy of reflection-based and exploitation-based assaults has been introduced. The dataset consists of 14 attacks that were gathered over a few days for testing and training purposes. Infiltration, SQL injection, SSHbruteforce assaults, Benign, Bot net, Brute Force – Web, DDOS attack-HOIC, DDoS attacks-LOICUDP, DoS attacks-LOICHTTP, DoS attacks-Golden Eye, DoS attacks-Hulk, DoS attacks SlowHTTPTest, DoS attacks Slowloris, FTP-BruteForce, and Infiltration are some of these attacks. Using the HTTP, FTP, SSH, and email protocols, the Profile system has been used to build users' abstract behavior.

There are fourteen different attacks in the testing data set. The dataset contained the distribution of the different attacks. The data collection was obtained using CICFlow Meter tools. For this investigation, we successfully extracted over 80 features from the CIC-IDS 2018 dataset. The

dataset can be found on the websites of the Canadian Institute for Cyber Security in both flow and PCAP file formats.

### 4.3.Pre-processing steps

Prior to training the IDS model, the following pre-processing steps were performed on the datasets we chose. This paper focuses on a binary classification problem for anomaly detection, classifying each observation as either normal or assault. Three types of socket information are included in the CIC-IDS2018 and CIC-IDS2019 databases: flow ID, source/destination port, and source/destination IP. To solve the over fitting issue, we eliminated socket-involved features from the data samples because they can differ from network to network. The completed dataset includes the CIC-IDS2018 and CIC-IDS2019 traffic labels together with 80 unique features.

The CIC-IDS2018 dataset's categorical features includes protocol type, services, and flag, were transformed into numerical features via CNN model. As stated above in dataset features, the protocols Flow Duration , UDP, and Tot Fwd Pkts have been mapped to (1,0,0), (0,1,0), and (0,0,1), respectively. Similarly, numerical features have been translated to the " Fwd Pkts " feature, which has 20 standards, and the services feature, which has 77 attribute. As a result, 80 numerical features are ultimately created from the initial 80 features.

### 4. 4. Performance measure of the model

The model's performance evaluation is pertinent to the dataset and our performance criterion. Determining whether there is a severe class imbalance problem in the dataset begins with looking at the distribution of the dataset, which is displayed as percentages in Figure 4.1.

$$\text{Imbalance ratio} = \frac{\text{majority class instance}}{\text{Minority class instance}}$$

The imbalance ratio of the majority class VS rest (Benign or various Attacks ) in this research study was seen and calculated as shown in below formula .

$$\text{Imbalance ratio} = \frac{445654}{122343} = 3.642$$

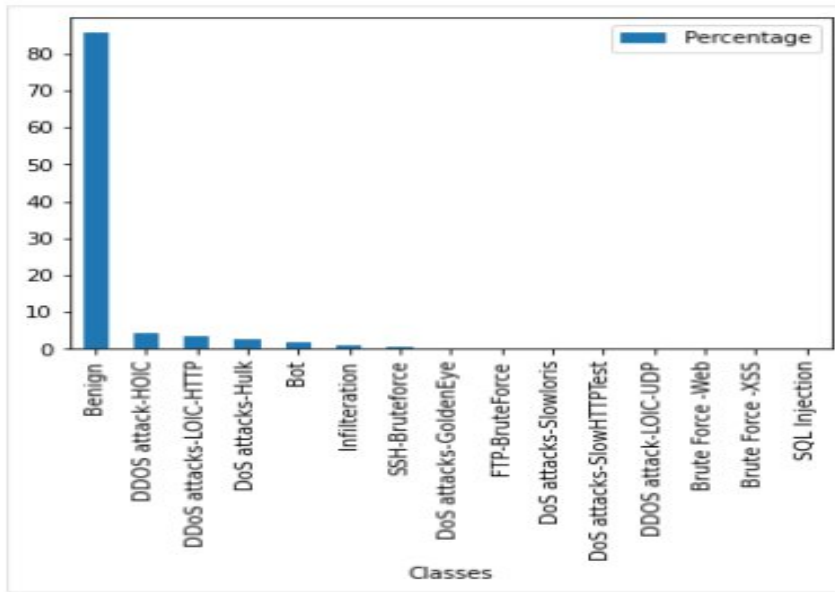


Figure 4.1 Class Distribution Percentage for attack and benign traffic 1

The sensitivity ratio and False Negatives were also assessed as more crucial for prediction, so Prediction Accuracy is designated as the most appropriate evaluation metric for this proposed study. As illustrated in the above figure, we used Soft max as an activation function in the output layer, and when we predict the model output, we get the probabilities of high accuracy prediction. In terms of network intrusion detection, we want to predict the attack Class Labels from the probabilities.

The outcomes of employing CNN for binary and multi-class classifications were described in depth in this chapter. Depending on the kind of classification, different models with varying activation functions, loss functions, and unit sizes were employed. Additionally, a number of methods, such as CNN models trained with custom class weights, LSTM, and SMOTE oversampling, were employed to improve the multi-class classification's performance measure by addressing the dataset's imbalance problem. Furthermore, the impact of various methods on the outcomes was noted and discussed in this section.

Learning rate, loss function, activation function, dropout, and batch size were among the many values explored for the layers of LSTM and CNN models. These models were constructed utilizing a range of hyper parameters until the most ideal results were obtained for both binary

and multi-class classification. Table 4.2 presents the hyper parameters that were studied together with the related outcomes that the models generated.

#### 4.4.1. Hyper parameters of the model

Table 4.2. hyper tune of the training model 1

| Classifiers | Output layers | Hidden Layers | Learning Rate | Epoch count no | Loss function             | Batch Size | Drop out | Output Activation Function | Optimizers | Accuracy | Precision | F1 score | Recall |
|-------------|---------------|---------------|---------------|----------------|---------------------------|------------|----------|----------------------------|------------|----------|-----------|----------|--------|
| Binary      | 2             | 2             | 0.0005        | 10             | Mse                       | 64         | 0.2      | Relu                       | adam       | 0.949    | 0.924     | 0.934    | 0.949  |
| Binary      | 2             | 2             | 0.0005        | 20             | Mse                       | 64         | 0.2      | Relu                       | Adam       | 0.979    | 0.995     | 0.958    | 0.924  |
| Binary      | 1             | 4             | 0.001         | 22             | Binary cross entropy      | 128        | 0.02     | Sigmoid                    | Adam       | 0.979    | 0.988     | 0.947    | 0.911  |
| Multiclass  | 1             | 2             | 0.0005        | 10             | Mse                       | 65         | 0.2      | Relu                       | Adam       | 0.894    | 0.807     | 0.848    | 0.894  |
| Multiclass  | 1             | 4             | 0.0005        | 30             | Mse                       | 512        | 0.2      | Relu                       | Adam       | 0.977    | 0.974     | 0.972    | 0.977  |
| Multiclass  | 1             | 4             | 0.0005        | 40             | Mse                       | 128        | 0.2      | Relu                       | Adam       | 0.967    | 0.942     | 0.954    | 0.967  |
| Multiclass  | 1             | 4             | 0.0005        | 50             | Mse                       | 128        | 0.2      | Softmax                    | Adam       | 0.975    | 0.966     | 0.965    | 0.975  |
| Multiclass  | 1             | 4             | 0.0002        | 40             | Categorical Cross entropy | 1024       | 0.1      | softmax                    | Adam       | 0.983    | 0.978     | 0.978    | 0.983  |
| Multiclass  | 1             | 4             | 0.00          | 60             | Categorical               | 102        | 0.5      | softmax                    | Adam       | 0.983    | 0.968     | 0.973    | 0.978  |

|            |   |   |            |    |                                 |          |     |         |      |       |       |           |           |
|------------|---|---|------------|----|---------------------------------|----------|-----|---------|------|-------|-------|-----------|-----------|
|            |   |   | 02         |    | Cross<br>Entropy                | 4        |     |         |      |       |       |           |           |
| Multiclass | 1 | 4 | 0.00<br>01 | 80 | Categorical<br>Crossentropy     | 102<br>4 | 0.2 | softmax | Adam | 0.978 | 0.973 | 0.9<br>78 | 0.9<br>83 |
| Multiclass | 1 | 4 | 0.00<br>02 | 50 | Categorical<br>Cross<br>Entropy | 102<br>4 | 0.2 | softmax | Adam | 0.983 | 0.973 | 0.9<br>77 | 0.9<br>82 |
| Multiclass | 1 | 4 | 0.00<br>02 | 50 | Categorical<br>Cross<br>Entropy | 102<br>4 | 0.2 | softmax | Adam | 0.982 | 0.965 | 0.9<br>69 | 0.9<br>69 |
| Multiclass | 1 | 4 | 0.00<br>02 | 50 | Categorical<br>Cross<br>Entropy | 102<br>4 | 0.2 | softmax | Adam | 0.969 | 0.807 | 0.8<br>48 | 0.8<br>07 |
| Multiclass | 1 | 2 | 0.00<br>02 | 40 | Categorical<br>cross<br>Entropy | 102<br>4 | 0.2 | softmax | Adam | 0.979 | 0.969 | 0.9<br>74 | 0.9<br>69 |

We tried with various output layer values for both binary and multi-class classifiers; however, the model performance rating remained same when the total sum of the output layers was increased. Additionally, the model demonstrated a mismatching model complexity in our dataset when we attempted to reduce the output layers, resulting in a higher performance matrix. Consequently, we have chosen one output layer, or "1." Additionally, we observed that the model's performance grew proportionately once we raised the number of hidden layers from two to four. For the model training, we have therefore determined that four hidden layers are the greatest fit.

For both binary and multiclass classifiers, we attempted to employ the mse and cross entropy loss functions. Based on the obtained results, the binary and categorical cross-entropies were determined to be the best fits for the model.

We used have dropout each hidden layer to make the suggested model more generalizable. Higher rates are measured to cause our network to under-learn. Thus, we have chosen values that are less than 0.5. We have therefore experimented with different dropout values, such as 0.1, 0.2, and 0.5, respectively. 0.2 was chosen as the model's best match among these values. We used a variety of output activation functions in this study, including softmax, sigmoid, and relu. We select classification over regression because the focus of this research study was on network intrusion detection and caseation. Therefore, because they can forecast the likelihood of the outcome, we favored sigmoid and softmax. Furthermore, by employing these functions, we were able to increase our productivity.

We experimented with several options for batch size and learning rate. In terms of classification time and stateless CNN, the optimum performance is achieved with a batch size of 1024 and a learning rate of 0.0002, which is adjusted based on batch size. About 934 seconds were spent on categorization with these batch size values.

#### 4.5. Binary Classification

First, binary classification is tested for the suggested dataset, following earlier research that used the machine learning method for similar datasets. Binary classification is used to categorize two outcomes, as the name implies. The goal of our research project is to forecast if the flow will go to the Attack or Benign class. As seen in figures 4.2 and 4.3, the preprocessing procedures and model creation are carried out. Twenty epochs are used in binary training, which is thought to be the most optimal number for training accuracy and loss. The model's training accuracy and training loss over epochs are displayed in the figure below.

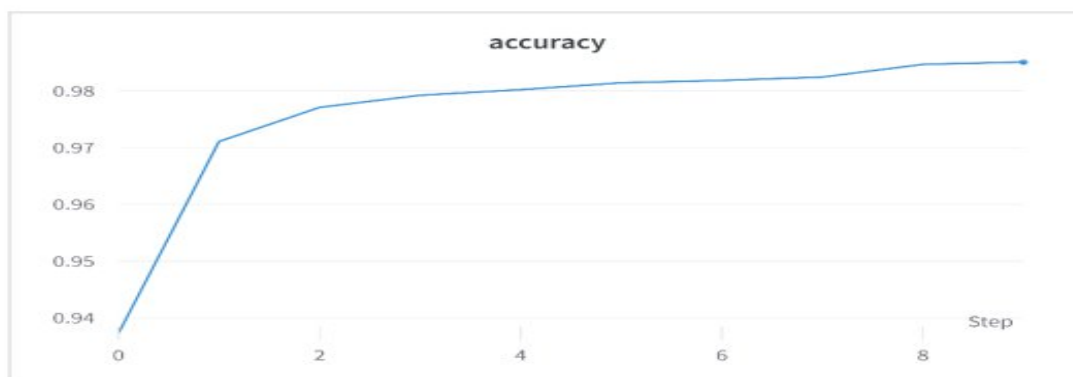


Figure 4.2.Training Accuracy with Binary Classification model 1

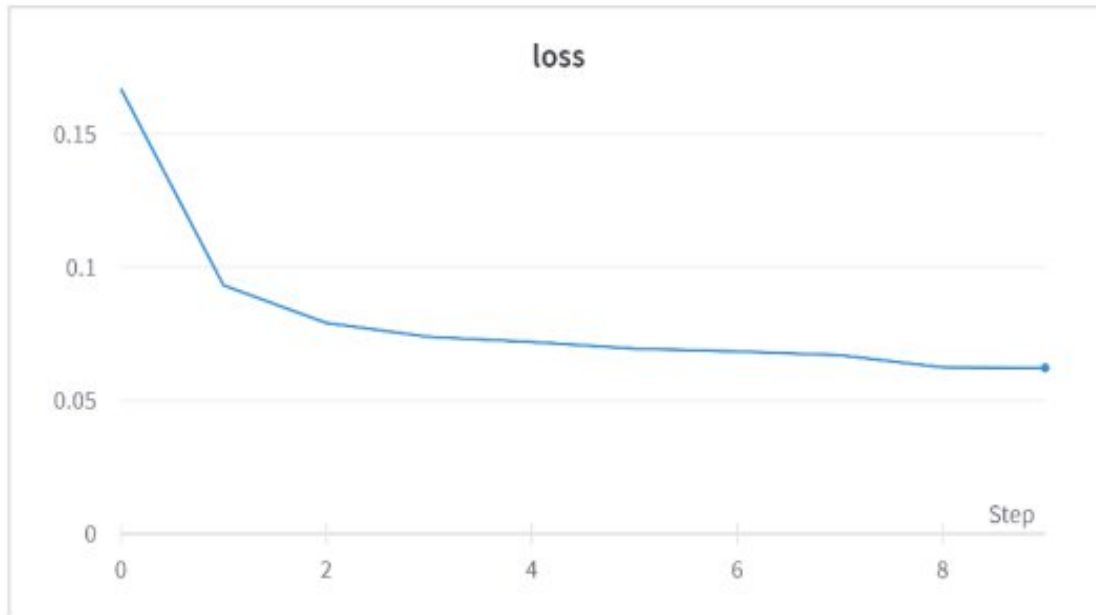


Figure 4.3. Training Loss with Binary Classification 1

The whole performance results of the model training was represented in Table 4.3. below.

Table 4.3. the whole Performance Results of the model with Binary Classification 1

| Evaluation parameters | Obtained result |
|-----------------------|-----------------|
| Accuracy              | 0.985           |
| Precision             | 0.985           |
| F1score               | 0.985           |
| Recall                | 0.985           |
| Balanced accuracy     | 0.959           |

Figure 4.4 below shows the confusion matrix for the model. Because benign traffic dominates the sample, the TN ratio is the largest of the other metrics. Because typical signature-based IDS

have significant FP rates, an FP ratio of only 0.40 percent is quite encouraging.



Figure 4.4.confusion matrix of the model using binary classification 1

The classification report from the model is shown in Figure 4.4. The precision, recall, and F1-Score mean values for each label are now averaged with the macro avg value which does not give any label a weight. The value weighted avg shows the average support weighted accuracy, recall, and F1 score for each label.

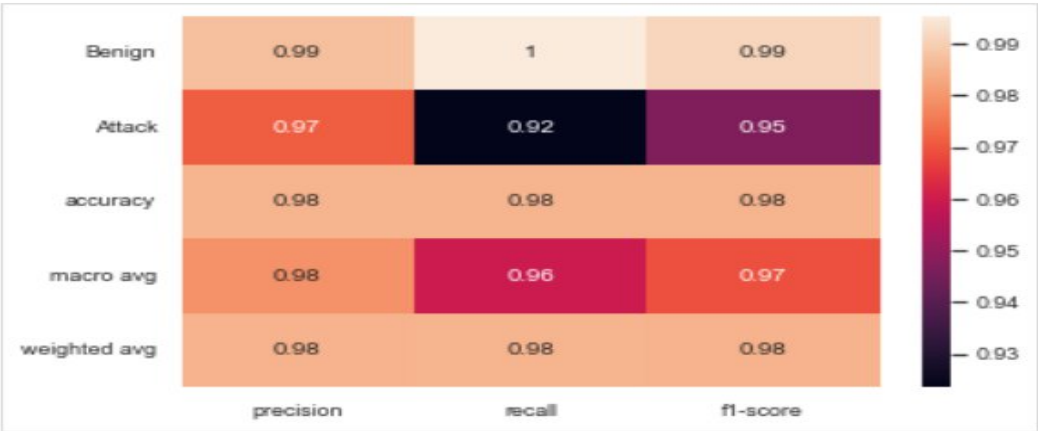


Figure 4.5.classification report using binary classification method 1

By comparing the total accuracy to similar experiments, it is higher. For instance, the greatest accuracy achieved by Ferrag et al. (2020) when using LSTM with various parameters for binary

classification of the same dataset, CSE-CIC-IDS2018, was 97.310%. Both the benign and attack traffics' FPR vs. TPR ratios are over the threshold line (FPR=TPR) when we look at the ROC Curve and AUC values displayed in Figure 35. It is a typical outcome of a competent classifier.

#### 4.6. Multi class Classification

This prediction's goal was to develop a multi-class classification using an LSTM model, based on assault kinds. The 40 epoch set and the 5 epoch training loss value allowed callback are used for multi-class training. While the training loss began to rise, the training ended at the 30th epoch. The training accuracy and training loss over epochs are shown in Figures 4.6 and 4.7 below.



Figure 4.6 Training Accuracy with Multi-class Classification 1



Figure 4.7. Training Loss Binary Classification 1

We provide the model's overall performance results as shown in Table 4.5. The F1 Score, precision, recall, and accuracy are nearly identical for binary classification. Class imbalance

deterministic metrics, such as G-Mean and balanced accuracy are significantly smaller than binary classification. G-Mean is zero primarily because some minority classes have zero recall value. Additionally, the balanced accuracy is lower than that of binary classification. The justification for this is that it gives FNs precedence over normal accuracy. In the case of multi-class classification, the FN values of each class are also calculated using the remaining FNs. Low recall values in several minority classes are the cause of this result.

Table 4.5. Performance Results of the model with Multi class Classification 1

| <b>Evaluation parameters</b> | <b>Obtained result</b> |
|------------------------------|------------------------|
| Accuracy                     | 0.986                  |
| Precision                    | 0.986                  |
| F1score                      | 0.981                  |
| Recall                       | 0.981                  |
| Balanced accuracy            | 0.630                  |

The model with multiclass classification provides a confusion matrix for each class based on one-vs-rest, as shown in figure 4.6. Dark blue is used to indicate the performance statistic with the greatest value, as seen in the figure. Likewise, in an unbalanced dataset where the majority class is Benign, the assault traffic's TN values (minority class) dominate other metrics. The model is unable to learn Brute Force Web, Brute Force XSS, and SQL Injection attacks since its TP values are all identical to zero. These attacks all have the trait of having the dataset's lowest proportion. Our suggested approach divides attacks such as Brute Force Web, Brute Force XSS, and SQL Injection into two categories: DDoS Attacks, LOIC HTTP Attacks, and Benign.

Additionally, the precision and recall value are poorer due to the high FN value of the infiltration attack. Despite having the sixth-order proportion across all of our model's datasets, our suggested model was unable to accurately detect this type of attack. Infiltration traffic is predicted by the model to be either benign, bot, HTTP DoS/DDoS, or FTP/SSH brute force.

Table 4.6 Confusion Matrix Reference Table

| <b>Actual/Predicted</b> | <b>Predicted Yess</b> | <b>Predicted NO</b> |
|-------------------------|-----------------------|---------------------|
| Actual: No              | TN                    | FP                  |
| Actual: yes             | FN                    | TP                  |

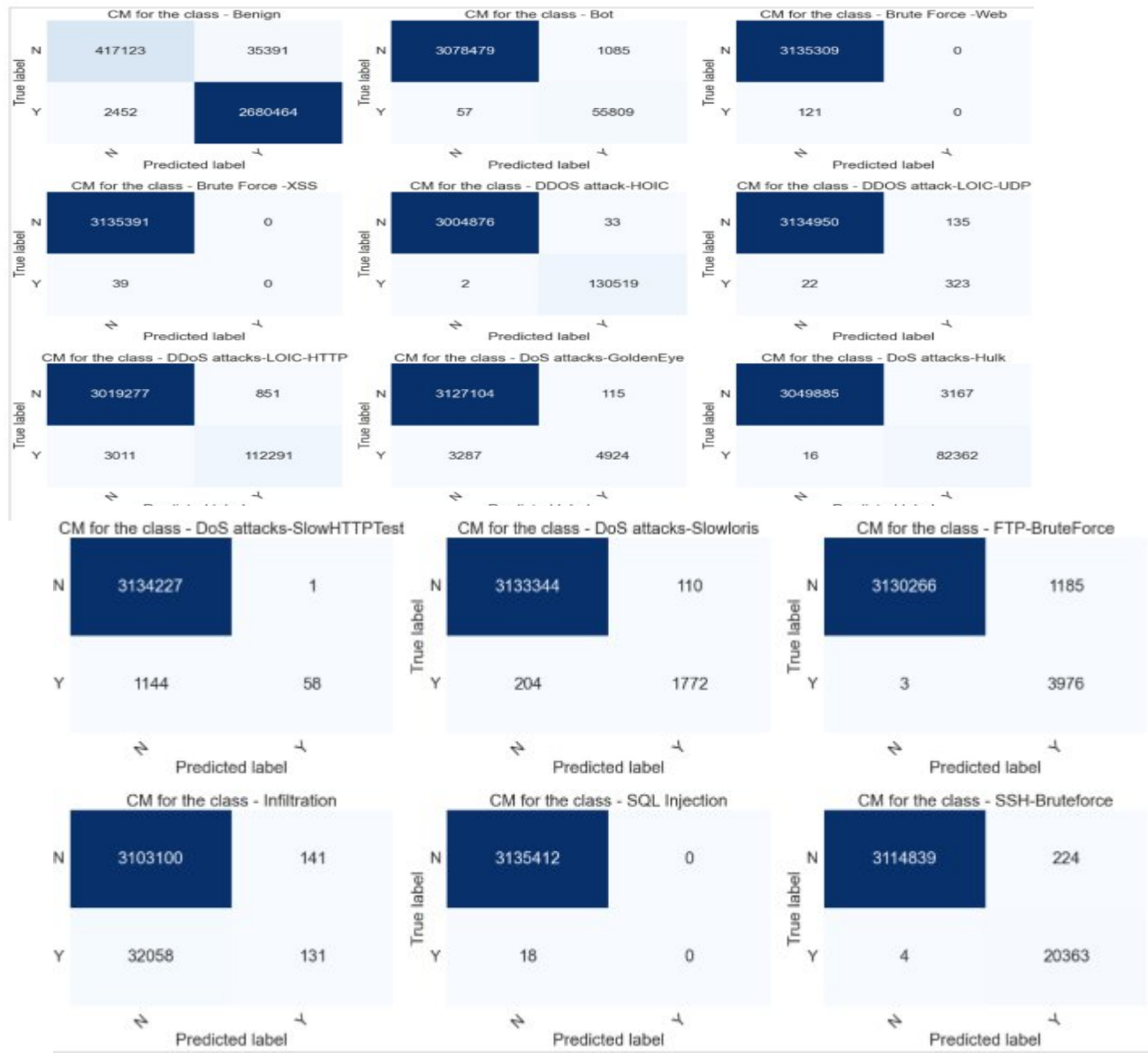


Figure 4.8. Confusion Matrix of proposed model with Multi-class Classification 1

The classification report for the model with multiclass classification was displayed using Figure 4.9. For the multiclass classification problem, the micro average views compute the overall result by averaging the TP, TN, FN, and FP of each label (Gamage, 2020). For datasets that are unbalanced, it is deceptive. The macro average number provides a more deterministic way to assess our model's overall performance. When determining the micro average values, it takes into account the weighted average of all labels.

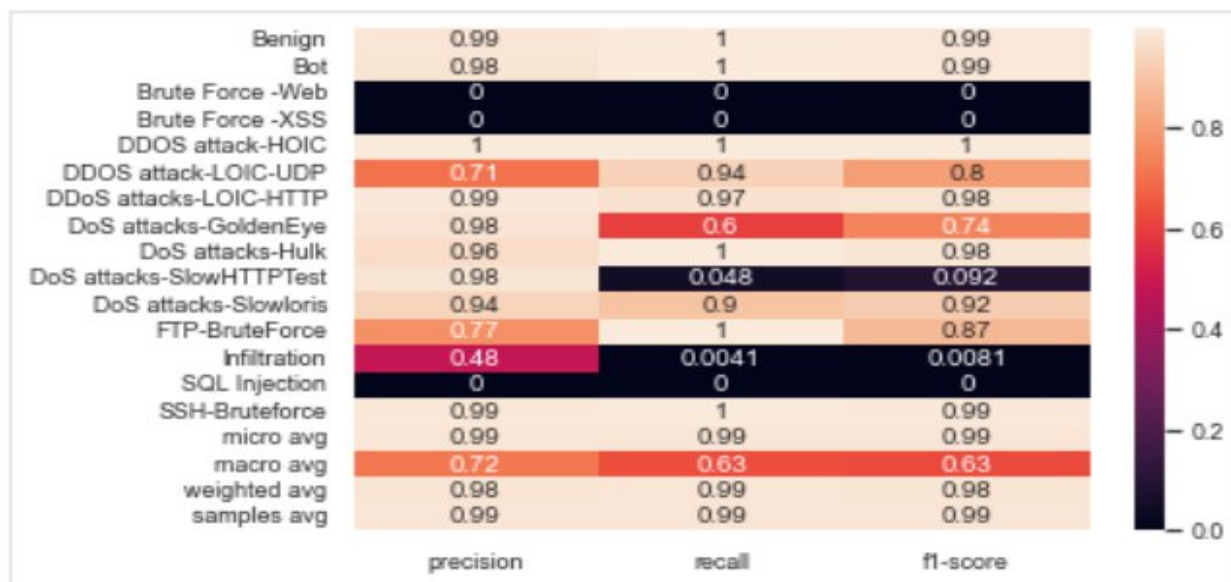


Figure 4.9 Classification Report with Multi-class Classification 1

Each class's ROC-AUC curve was depicted in Figure 4.9. However, all of the classes are displayed as being above the threshold, and its precision and recall ratings are low. Given that TPR and FPR primarily concentrate on the majority class, this scenario is misleading for imbalanced multi-class categorization.

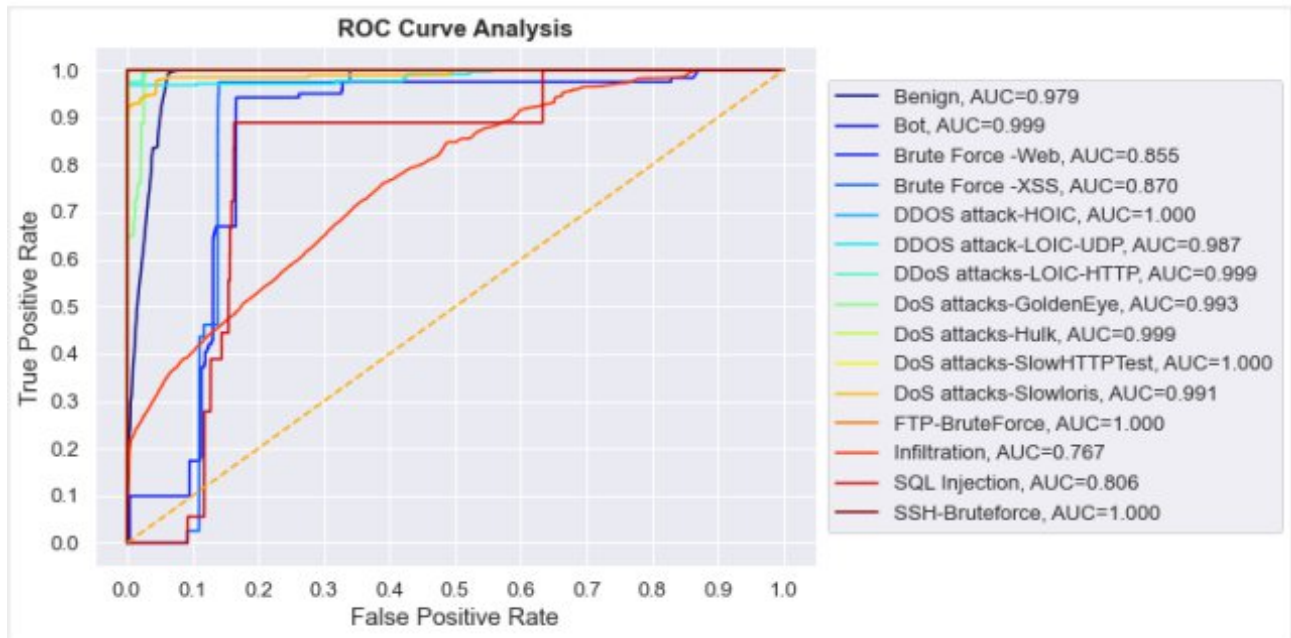


Figure 4.10. Precision Recall Curve with Multi-class Classification 1

Since our model works with precision and recall values, respectively, the Precision-Recall Curve for each class is provided, which yields more accurate results.

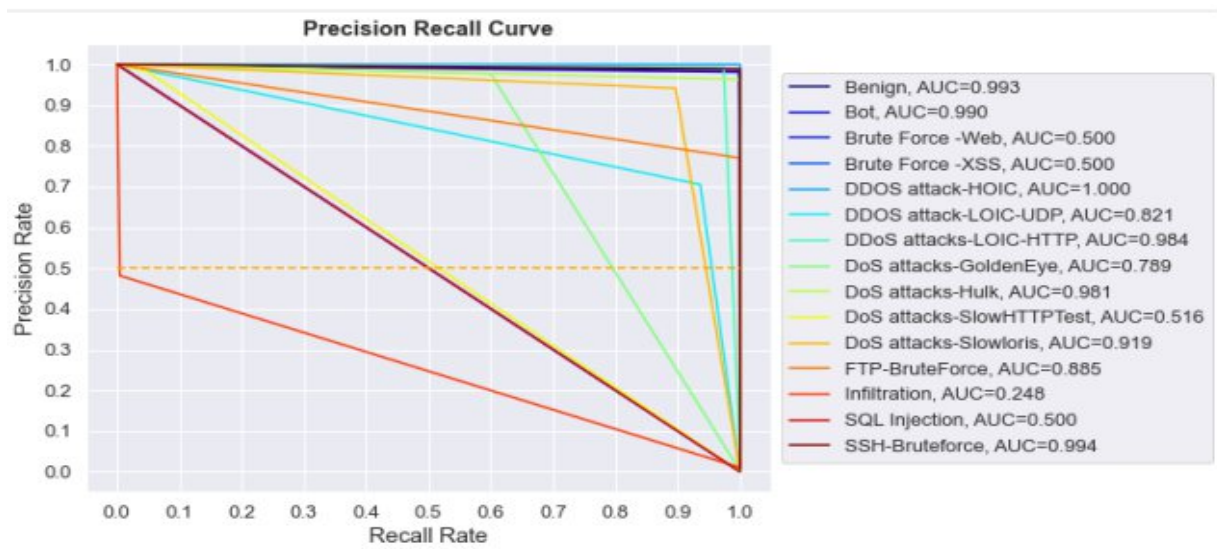


Figure 4.11 with Multi-class Classification 1

## **4.7. Model Training using Custom Class Weights**

The main advantage of this approach is that it doesn't require changing the IDS dataset's distribution. It is evident that the benefits and drawbacks of using SMOTE approaches to increase FP rates with TPs were discussed (Chawla, 2012). Here, we refer to it as the outcome of producing artificial inputs that don't quite mimic the actual situation. Thus, we looked into a different set of findings. With its default design, our suggested model tends to give each class equal weight regardless of the distribution, just as other machine learning models.

### **4.7.1. The sampling method of imbalanced Classification**

Random over-sampling and random under-sampling were the first sampling techniques used for unbalanced categorization. Random over-sampling seeks to increase the proportion of minority class samples in order to provide a more balanced dataset. The minority class samples were chosen at random and reproduced using the over-sampling technique in order to reach the target rate. By lowering the rate of majority class samples, random under sampling seeks to create a balanced distribution in the dataset. On the other hand, the majority class samples utilized in the under sampling technique are selected at random and removed from the dataset until the required condition is met.

These procedures are conducted with two primary goals in mind. One of them is for oversampling at random. Replications of the same samples should be included in the dataset since the minority class samples are duplicated. A serious over fitting issue could arise from the model memorizing these values and being less sensitive to cases with minimal fluctuations. A random under sampling technique is used for the other agreements.

Based on the imbalance ratio of the dataset, the majority class samples are removed. Since benign traffic accounts for 83% of the dataset in our example, it is necessary to delete the high value of the majority samples when working with a dataset that has a very high imbalance ratio. This could result in a substantial loss of data, making it impossible for our model to learn correctly and lowering the classifier's performance.

Due of these bottlenecks, various sampling techniques were implemented. All of these techniques are designed to address the issues of over fitting for oversampling and under sampling, which results in the loss of valuable information. Even though new techniques for

under-sampling have been created, we are all concerned about losing important data. We therefore made the decision to concentrate more on over-sampling strategies.

Next, the usefulness of the Synthetic Minority Over-sampling Technique (SMOTE) for our problem is evaluated. It was this strategy that was first introduced. The main difference with this approach is that, rather than replicating instances, it generates synthetic ones.

Network traffic was predicted using a variety of techniques. For all techniques, we first check the dataset using only attack traffic. Then, using the one-vs-rest method, we add the benign traffic after determining the weights for each attack type. The majority's class weight

The overall performance of the benign class is highly deterministic. When we assign minority classes considerably bigger weights in an attempt to bring the majority and minority class weights closer to one another, we observe that our model loses its ability to recognize the majority class, which lowers accuracy.

#### 4.8. Experiment I : Majority Class Referenced

The experiment's goal is to identify the four classes for which, out of all attack classes, this model has reflected the majority class's worse performance. The class distribution following the use of SMOTE oversampling for the training dataset is shown in Table 4.7 below. Ignoring benign traffic makes the DDoS attacks-LOIC-HTTP attack type the most prevalent class. The oversampling of the four problematic class types is done to get the same train samples as the majority class.

Table 4.6. network traffic dataset distribution After SMOTE #1 analysis [1](#)

| Classes                  | Train         |
|--------------------------|---------------|
| Benign                   | 8584473       |
| Bot                      | 178520        |
| Brute Force -Web         | 416771        |
| Brute Force -XSS         | 416771        |
| <b>DDOS attack-HOIC</b>  | <b>416771</b> |
| DDoS attacks-LOIC UDP    | 1107          |
| DDoS attacks-LOIC HTTP   | 368509        |
| DoS attacks-GoldenEye    | 26565         |
| DoS attacks-Hulk         | 264264        |
| DoS attacks SlowHTTPTest | 3881          |
| DoS attacks-Slowloris    | 6422          |
| FTP-BruteForce           | 12589         |
| Infiltration             | 416771        |

|                |        |
|----------------|--------|
| SQL Injection  | 416771 |
| SSH-Bruteforce | 66014  |

By employing this technique, the train dataset's class imbalance ratio which was based on benign vs. attack traffic was decreased from 5.92 to 2.85. With a batch size of 1024 and a learning rate of 0.0002, the training dataset comprised 50 epochs. Similarly, the LSTM model for multi-class classification was used. The model's training accuracy and loss are shown in Figures 4.9 and 4.10.

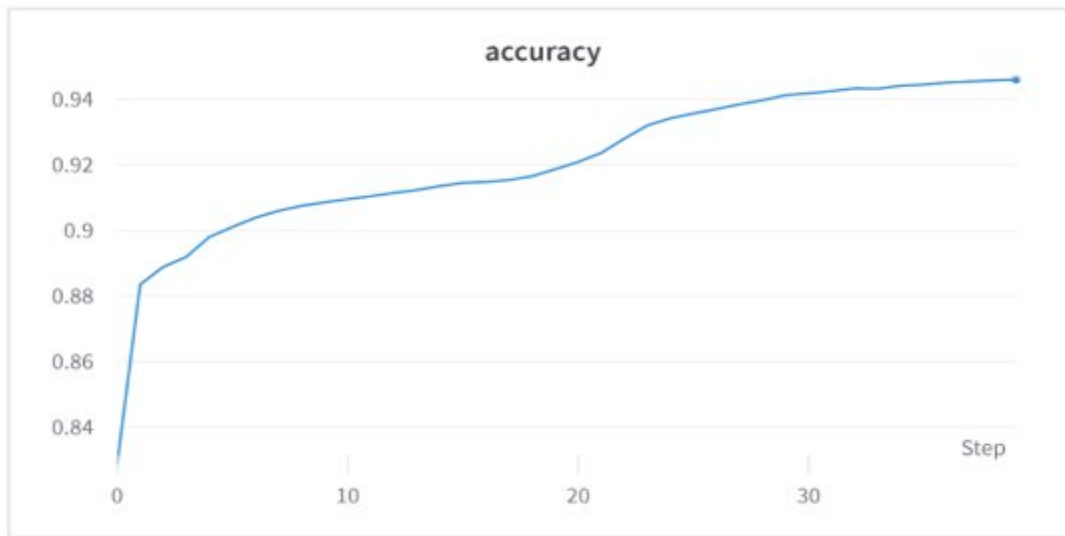


Figure 4.12. Training Accuracy for SMOTE analysis 1

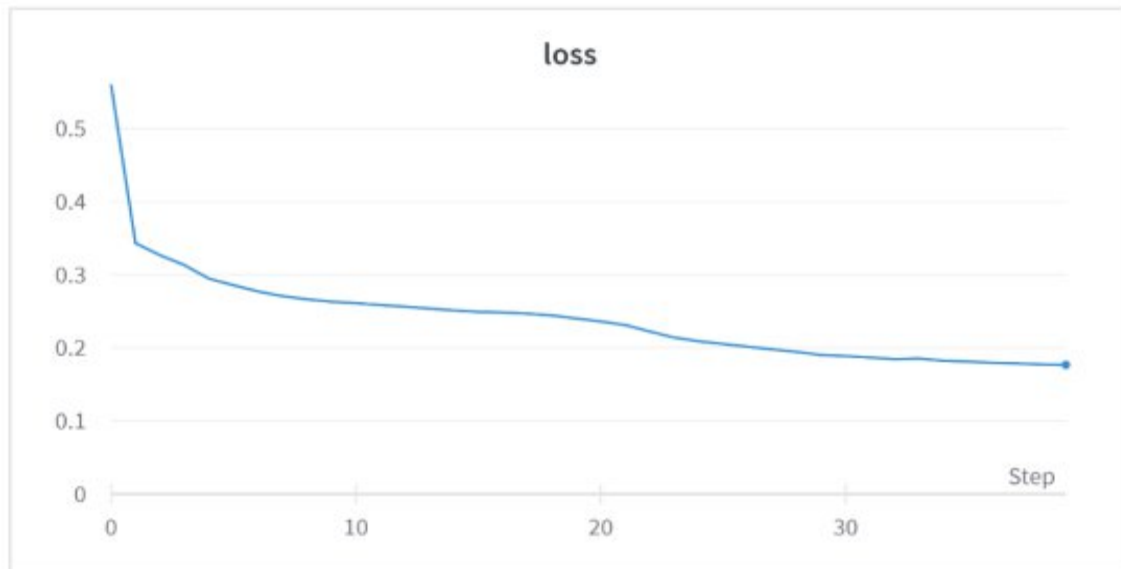


Figure 4.13. Training Loss for SMOTE analysis 1

Table 4.11 displays the model's overall performance results .Overall accuracy decreased slightly as compared to the multi-class classification findings, however whole recall, precision, and F1 Score increased. As we increase the minority class samples, the dominant rate of benign traffic has dropped, hence this accuracy drop is expected. Additionally, the g-mean values and balanced accuracy have both increased, indicating more balanced classification.

Table 4.7. Overall Performance Results after SMOTE 1 analysis 1

| Performance metric | Result |
|--------------------|--------|
| Accuracy           | 0.985  |
| Recall             | 0.985  |
| Precision          | 0.983  |
| F1 Score           | 0.983  |
| G-Mean             | 0.689  |
| Balanced Accuracy  | 0.823  |

Figure 4.17 displays the model's confusion matrix for each class based on one-versus-other comparisons. We find that there is an improvement for identifying and anticipating these attacks after looking into the Brute Force Web, Brute Force XSS, Infiltration, and SQL Injection classes. In particular, after each of those attacks, the true positive rates have dramatically increased.

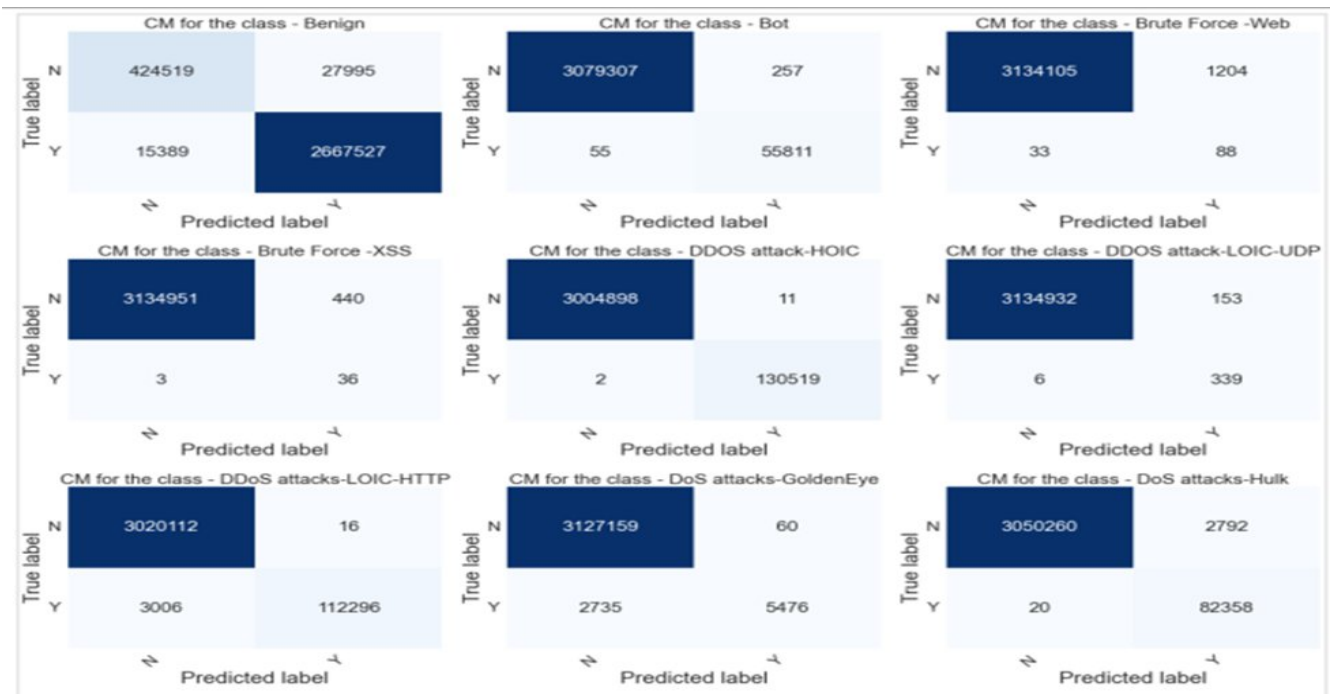


Figure 4.14 Confusion Matrix of the model after SMOTE 1 analysis 1

Figure 4.19 below displays the model's categorization report. According to multi-class classification, we have observed an improvement in the macro average value using this method, which is the intended outcome of a more balanced dataset. It is observed that the Precision value for the over-sampled classes does not grow as much as the Recall value. FPs are hence deterministic in terms of precision value. In light of this, we deduce that oversampling led to a greater number of false positives. Additionally, whereas Web attacks have seen a notable boost in performance, penetration has not seen the same gain. Despite improvements in precision and recall rates, they are still insufficient for classifying other attacks.

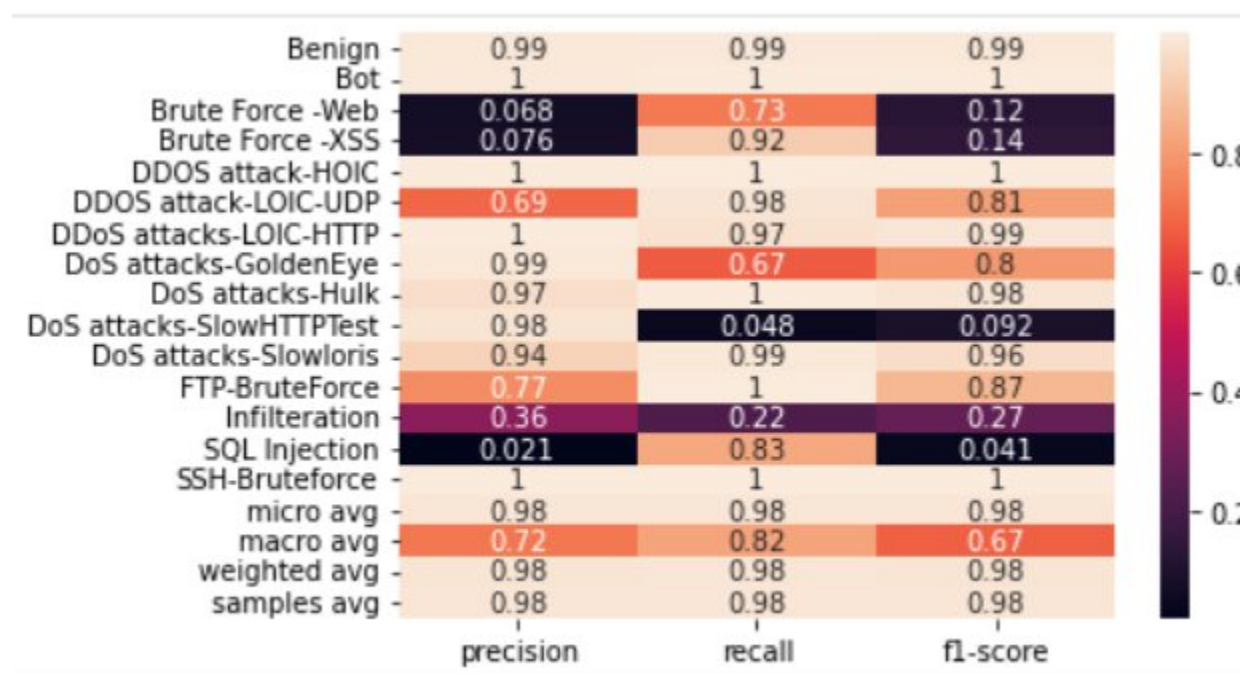


Figure 4.15. Classification Report for SMOTE 1analysis 1

In Figure 4.15, the ROC AUC Curve was displayed. In accordance with standard multi-class categorization, the AUC values have increased.

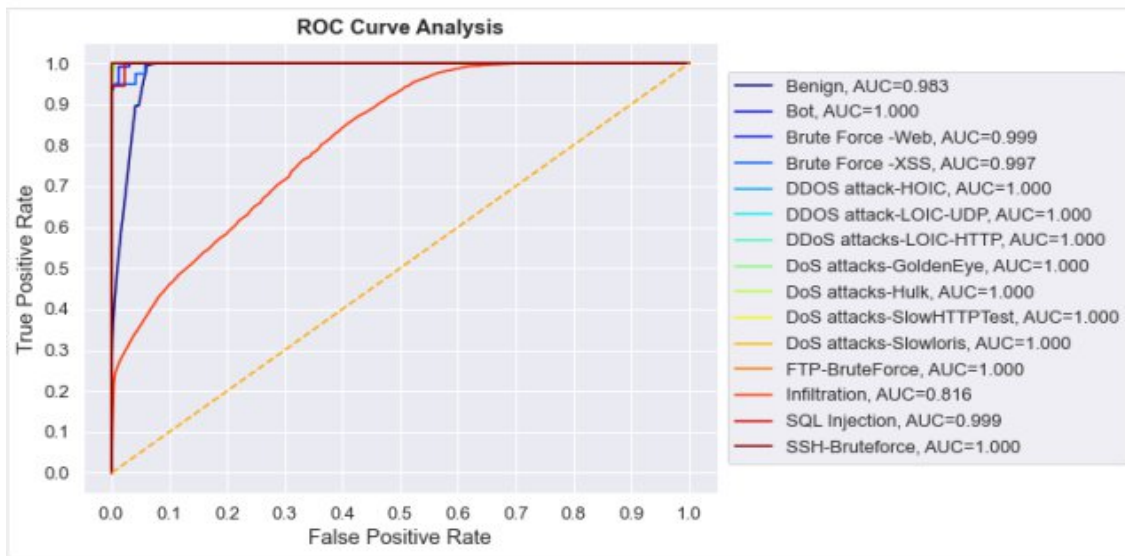


Figure 4.16. ROC AUC Curve after SMOTE #1 analysis 1

The figure 4.21 below displayed the PR-AUC Curve. FP rates increased as a result, although PR-AUC values did not improve in comparison to multi-class classification.

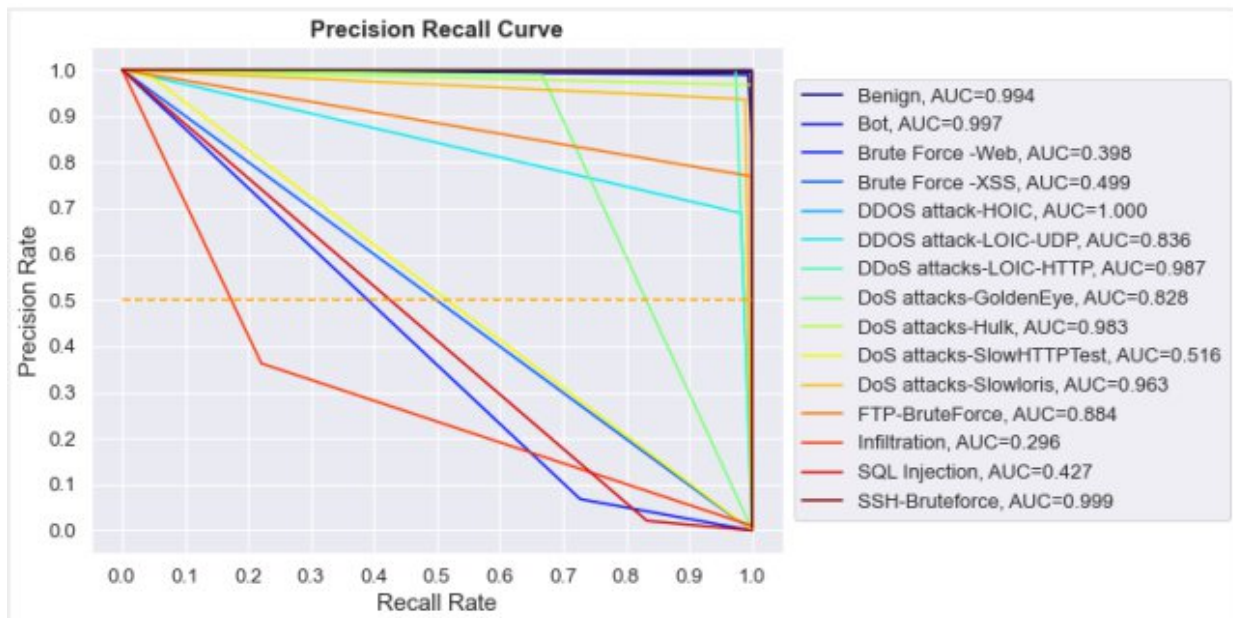


Figure 4.17. Precision Recall Curve after SMOTE 1 analysis 1

## 4.9.Experiment II using SMOTE with Class Weights method

Creating the four classes for which this model represented the majority class's worse performance across all attack class types was the main goal of this experiment. Because our custom class weight solution uses SMOTE 1 with Majority Class Referenced. Using SMOTE 1, we expand the four problematic attack class samples. Next, we choose our customized approach as the updated dataset and do a class weight. The model was trained over 60 epochs with a batch size of 1024 and a learning rate of 0.0002. In a similar manner, the multi-class classification LSTM model is used. In Figures 4.22 and 4.23, the model's training accuracy and loss are displayed.

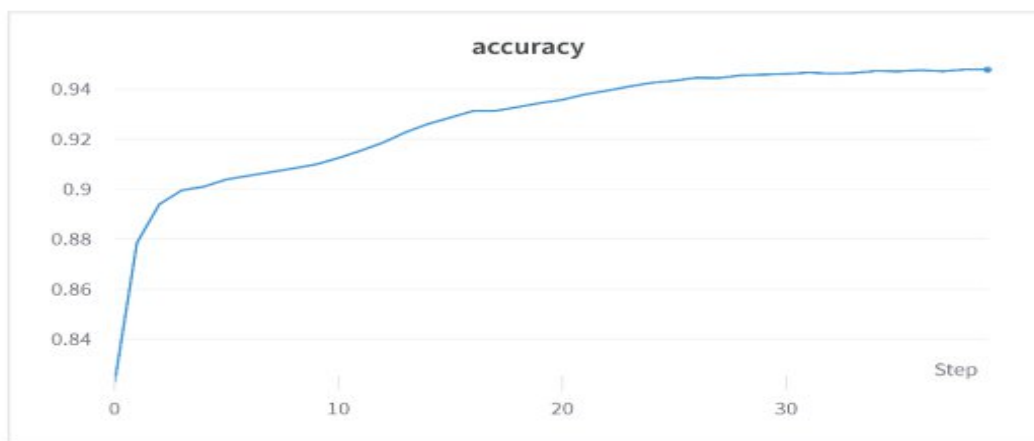


Figure 4.18. Training Accuracy using SMOTE 2 analysis 1

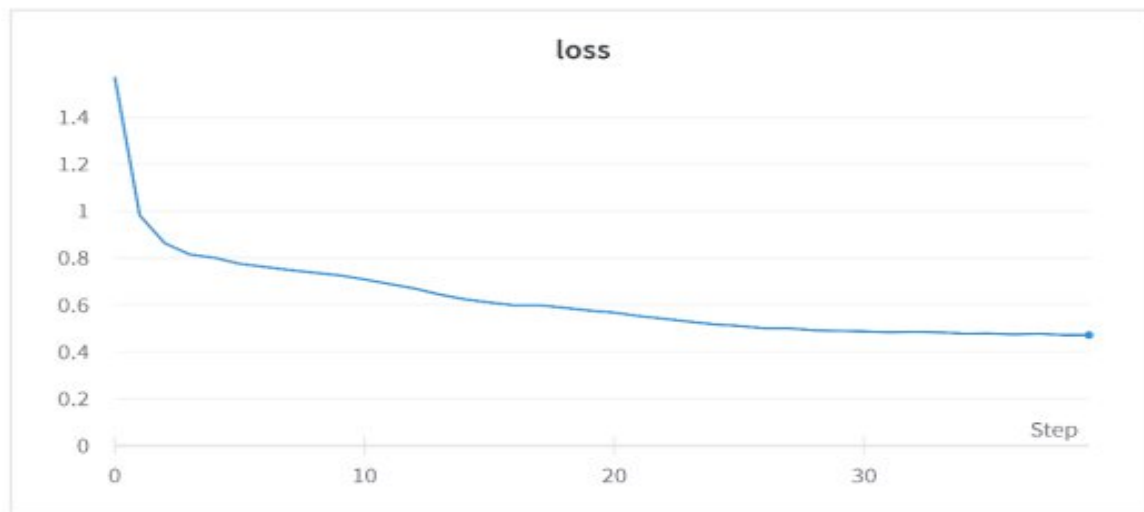


Figure 4.19. Training Loss using SMOTE #2 analysis 1

The model's overall performance outcomes were displayed in Table 1.17. Overall precision and F1 Score have increased as compared to the multi-class classification findings, despite a little decline in overall accuracy and recall. Furthermore, multi-class classification with SMOTE 1 techniques as well as multi-class classification alone have improved balanced accuracy and g-mean values. This suggests a classification that is more equitable.

Table 4.8 The whole Performance Results of the model with SMOTE 2

| Performance evaluation Metric | Result |
|-------------------------------|--------|
| Accuracy                      | 0.985  |
| Recall                        | 0.985  |
| Precision                     | 0.983  |
| F1 Score                      | 0.983  |
| G-Mean                        | 0.683  |
| Balanced Accuracy             | 0.816  |

The model's confusion matrix for each class based on one-vs-rest was displayed using Figure 4.20. For Brute Force Web, the FN ratio has gone up when compared to SMOTE 1. Additionally, there is a notable decline in the FP values of Brute Force Web and SQL Injection. The TP rate of infiltration has also increased. Our analysis revealed an improvement in the detection of Brute Force Web, Brute Force XSS, Infiltration, and SQL Injection attacks. For each of these attacks, the true positive rates have increased significantly. Figure 4.24: Confusion Matrix using SMOTE 2 analysis

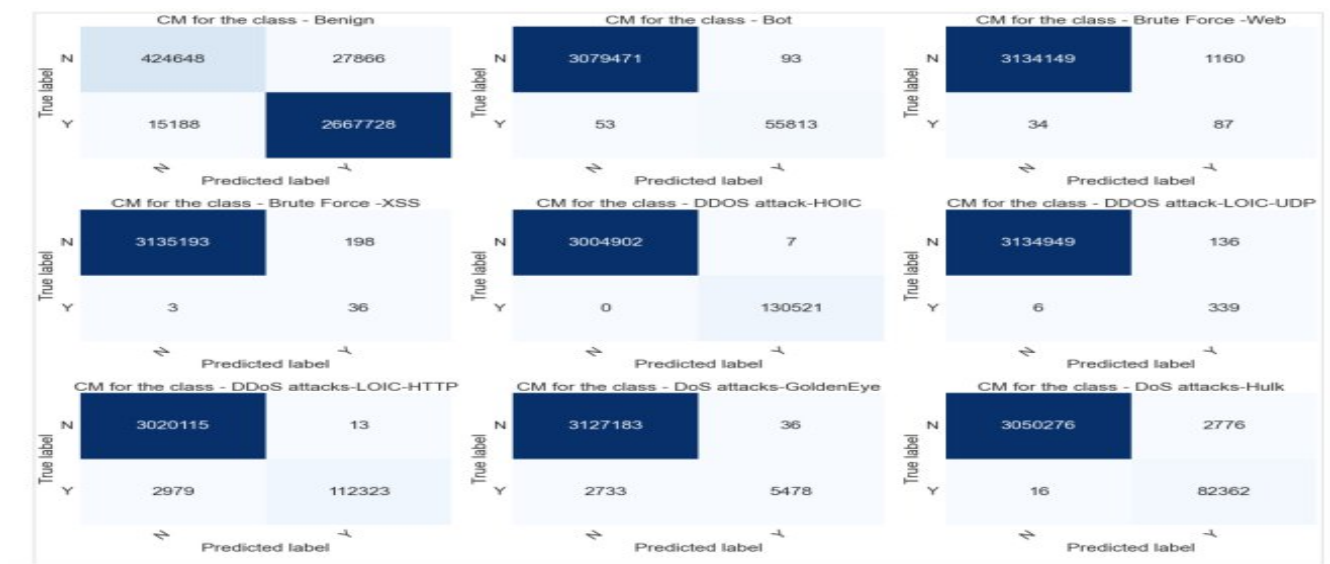


Figure4.20. classification report of the model

Figure 4.20 shows the classification report from the model. SMOTE 1 shows that the macro average precision, recall, and F1 Score values have increased, which is the desired result for more balanced training.

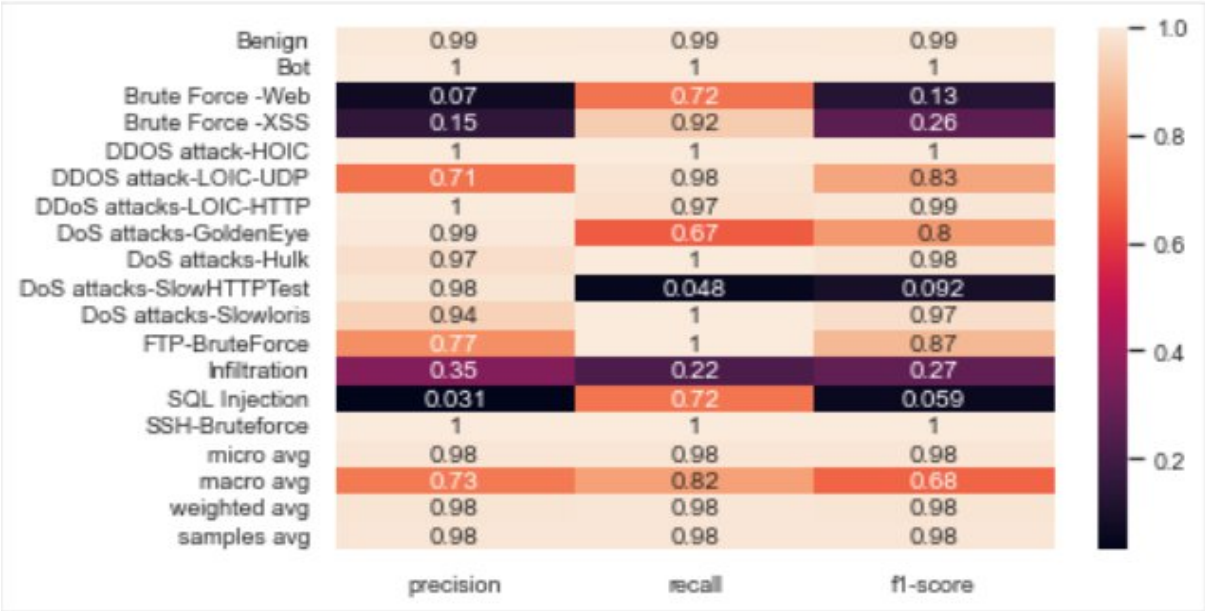


Figure 4.21 Classification Matrix of the model with SMOTE 2

Figure 4.21 is a representation of the ROC AUC Curve. Similar to SMOTE 1, the ROC AUC values have increased.

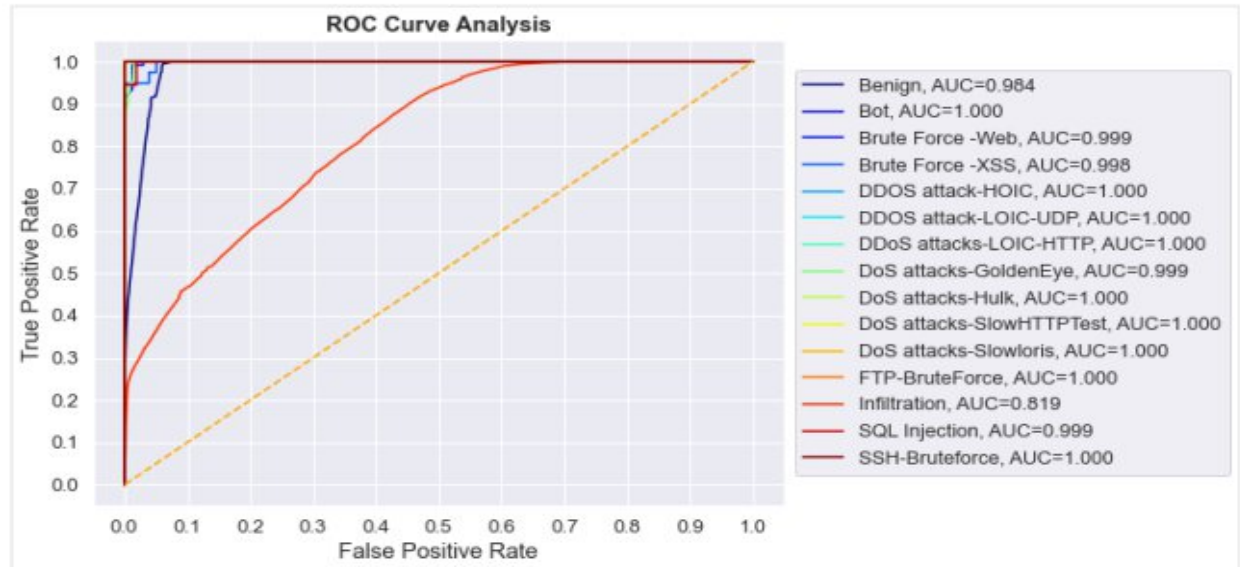


Figure 4.22. ROC AUC Curve with SMOTE 2

The Figure 4.22 shows the PR-AUC Curve. The Brute Force XSS PR AUC value has significantly increased. It is now above this average, whereas in SMOTE 1, it was below the 0.5 criterion. As with SMOTE1, PR-AUC values did not improve when FP rates increased in comparison to multi-class categorization.

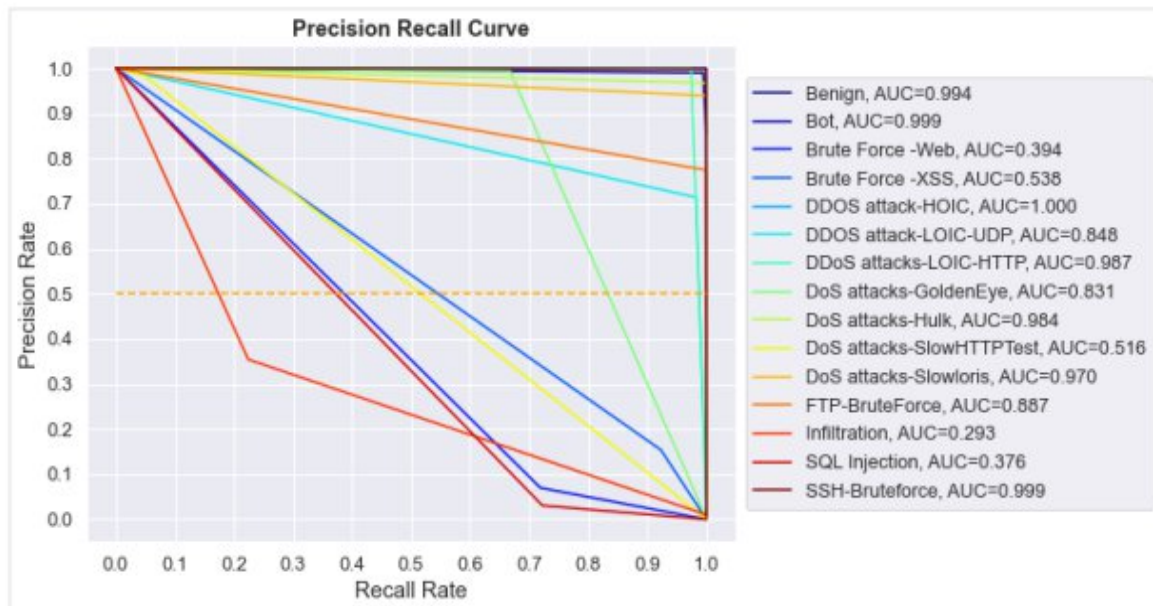


Figure 4.23 Precision Recall Curve with SMOTE 2 1

## **CHAPTER FIVE**

### **CONCLUSION AND FUTURE WORK**

#### **5.1. Conclusion**

In order to evaluate the efficacy of a deep learning technique for network intrusion detection, we examined network traffic data in this thesis. We build a deep learning model using real-world-relevant samples from the CSE-CIC-IDS2018 dataset. We have investigated how using only network flows, rather than raw traffic or deep packet inspection, affects intrusion detection. The encouraging results demonstrated that network flow analysis can identify a specific volume of attack traffic. This is very helpful to use in real-world situations because, according to raw traffic analysis, it uses less resources. With SSH traffic in particular and most web traffic encrypted, deep packet inspection is impossible if analysis is not done inline. The usage of network flow analysis increases in these situations.

We have employed LSTM and CNN in a number of ways. After feature engineering, we completed the LSTM model by choosing the optimal optimizer, loss functions, and activation functions based on our dataset and prediction objective. Utilizing the LSTM's time series analysis capabilities, we fitted the processed data into our model. According to binary classification results, our model can distinguish between malicious and benign traffic. But this paradigm isn't applicable to real-world situations. We investigate for several ways of the model to address the issue of class imbalance. Then we have used both of them (SMOTE and class-weight adjustment) and contrasted the improvement in the outcomes. Applying both tactics yields the most effective approach, according to the results of multiple trials.

The majority of the attacks that were added to our dataset could be detected by the deep learning model. Nevertheless, despite a number of attempts to address the unbalanced nature of our dataset, the detection performance of "Infiltration" traffic has not improved enough. Our approach tends to categorize infiltration traffic as benign or other attack types since it is more sophisticated than the other categories.

The primary cause, we deduce, is that we are not searching for specific patterns in the traffic's payload. For instance, if we could have examined the traffic's payload, we would have identified that the user was downloading a dubious PDF file from Drop Box. We are only attempting to

examine features that were taken from network flow, hence these are insufficient to identify this type of communication.

Using LSTM, we have achieved very encouraging results. Utilizing not-so-small batch sizes for our stateless LSTM and rearranging the dataset based on timestamp data allowed us to obtain relevant findings.

Generally we conclude that there is no difference in overall performance between the two classification systems. However, multiclass classification is thought to be more successful in real-world situations when it is necessary to identify particular types of attacks and create defenses against them.

## **5.2. Contribution of the research**

The majority of the research use a subset of the dataset for their experiments, like binary classification or train their model using only the distinct CSV files from the dataset that contain only benign traffic and particular types of attacks. But this wasn't know things actually work in the real world. The attacks don't happen one after the other. In order for the model to understand user behavior on a daily or weekly basis, innocuous traffic from all days is also included. Additionally, training our model with the entire dataset enables it to learn to distinguish between different kinds of attacks. The proposed research method is considered to be the most effective for network intrusion detection when we examine the results and compare them with previous research and scholar. With multi-class classification, however, we observe that our model continues to perform inadequately for detecting specific classes as a result of applying multiple class imbalance solutions. Their sample percentage in the dataset is significantly lower than the others, which is a common characteristic of these classes.

The imbalance ratio is so great that, whether or not we create synthetic data, we move away from the original samples while attempting to create mode data. It has been established that this may account for the outcomes. Additionally, it is observed that altering the class weights during learning has an impact on other classes as well because doing so reduces the weight of other classes, and the distribution that results does not accurately represent the situation in the real world.

Contrary to the previous conclusions, we have observed that the majority of pertinent publications did not provide comprehensive details about the preprocessing and the hyper

parameters used in their experiments. The reproducibility of these studies is significantly impacted by previous study. As a result, we explained our pre-processing and hyper parameter tuning methodology in this thesis. Furthermore, a significant problem that plagues the majority of earlier efforts is the measures employed to assess the model's success. Accuracy alone is insufficient for the evaluation because our dataset is extremely unbalanced. In order to provide more realistic results for our instance, we added further measures such balanced accuracy, g-mean, PR-AUC, and ROC-AUC.

### **5.3. Future work**

A more modern and worthwhile method, such Attention Mechanism can be used as a future endeavor to improve performance in addition to employing class imbalance solution, according to an evaluation of our model. Additionally, CIC Flow Meter can be expanded to include minority classes by feeding it with captures that you have created by yourself.

We know that our model has performed better than many of the earlier research, so we may advance and make improvements. A comparison with the conventional CSE-IDS 2028 under actual traffic data was anticipated while the model's performance was being improved. Therefore, the Splutter can be linked with the CIC Flow Meter to transform raw traffic into flow-based extracted features. Then, using its flow preprocessing features, the model's prediction and detection performance with Snort detection was compared, allowing us to test this model in a variety of scenarios.

Another concern that future researchers should consider is how well this proposed model adapts to the actual scenario. Since this is a new field of study, the feature work of this research can look into how this model integrates with signature-based IDS. The feature research should use an ensemble learning approach to better detect and predict network intrusion with multiple deep learning models. So the feature research must develop an ensemble deep learning approach to enhance IDS.

## References

- "corporatefinanceinstitute.com, ". (2020).  
"corporatefinanceinstitute.com,"corporatefinanceinstitute,5/10/2020.[Online].
- "www.saedsayad.com, ". (2022). "www.saedsayad.com,"www.saedsayad.com,[Online].Available:.
- [10]C.Jirapummin, N. a. (2010). [10]C.Jirapummin, N.Wattanapongsakorn and P.Kanthamanon,.
- [11] C.-J. Lin, R. C.-s. (2012). [11] C.-J. Lin, R. C. Weng, S. S. Keerthi. (2008) Trust region Newton method for large-scale logistic regression. In Journal of Machine Learning Research, vol. 9. .
- ] A. A. Aburomman and M. B. I. Reaz, ". s. (2018). ] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid.
- ] Daniel Jurafsky & James H. Martin, (. S. (2016). ] Daniel Jurafsky & James H. Martin, (2016) Speech and Language Processing.
- ] P. Vimalachandran, H. L. (2020).
- IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (n.d.). Available online: <https://www.unb.ca/cic/datasets/ids-2018.html> (accessed on 6 December 2023). (2023).
- 2023),. I. 2.-2. (2018). IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (n.d.). Available online: <https://www.unb.ca/cic/datasets/ids-2018.html> (accessed on 6 December 2023).
- A. Daly, T. i. (2018). A. Daly, The introduction of data breach notification legislation in australia: A comparative.
- A. Khraisat, I. G. (2019). A. Khraisat, I. Gondal , P. Vamplew and J. Kamruzza, "Survey of intrusion detection.
- A. M. Z. a. M. J. A. A. Qais Saif Qassim, ". S. (2016). A. M. Z. a. M. J. A. A. Qais Saif Qassim, "Qais Saif Qassim, Abdullah Mohd Zin, and.
- A. Makkar, N. K. (2018). A. Makkar, N. Kumar, A. Y. Zomaya, S. Dhiman, Spami: A cognitive spam protector for.
- A. Mouhammd, .. A.-N. (2016). A. Mouhammd , . A.-N. Ghazi, H. Ahmad and M. Mohammad, "Detecting distributed.
- A. Salih, Y. A.—6. (2021). A. Salih, Y. Ameen, R. Zeebaree, A. Sadeeq, F. Kak, N. Omar, M.Ibrahim, M. Yasin, N. Rashid, and S. Ageed, "Deep LearningApproaches for Intrusion Detection," Asian Journal of Research inComputer Science, pp. 50—64, 2021.
- A. Yadav, ". I. (2020). A. Yadav, "resources.infosecinstitute.com," INFOSEC, 4 August 2020. [Online].
- Abiodun, O. I., & Arshad, H. (2018). Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H.

- Alok Sharma, E. V. (2019). Alok Sharma, Edwin Vans, Daichi Shigemizu, Keith A Boroevich, and Tatsuhiko.
- Alzubaidi L, A.-S. O., & 9(3):445. (2020). Alzubaidi L, Al-Shamma O, Fadhel MA, Farhan L, Zhang J, Duan Y. Optimizing the performance of breast cancer classification by employing the same domain transfer learning from hybrid deep convolutional neural network model. *Electronics*. 2020;9(3):445.
- Anderson, J. C., Report, T., & James P. Anderson Company: Philadelphia, P. U. (1980). Anderson, J.P. Computer Security Threat Monitoring and Surveillance; Technical Report; James P. Anderson Company: Philadelphia, PA, USA, 1980.
- B.-H. Chen, J.-L. Y.-s.-3. (2019). B.-H. Chen, J.-L. Yin, and Y. Li, "Image noise removing using semi-supervised learning on big image data," in *Multimedia Big Data (BigMM)*, 2017 IEEE Third International Conference on. IEEE, 2017, pp. 338–345. <https://doi.org/10.1109/>.
- Bagui, S., Kalaimannan, E., Bagui, S., Nandi, D., & Pinto, A. U.-a.-N. (2019). Bagui, S.; Kalaimannan, E.; Bagui, S.; Nandi, D.; Pinto, A. Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset. *Secur. Priv.* 2019, 2, e91.
- Belavagi, M., & Muniyal, B. P.-1. (2016). Belavagi, M.C.; Muniyal, B. Performance evaluation of supervised machine learning algorithms for intrusion detection. *Procedia Comput. Sci.* 2016, 89, 117–123. [.
- Belouch, M., El Hadaj, S., & Idhammad, M. P.-6. (2018). Belouch, M.; El Hadaj, S.; Idhammad, M. Performance evaluation of intrusion detection based on machine learning using Apache Spark. *Procedia Comput. Sci.* 2018, 127, 1–6.
- Berthelot, D. C.-s. (2019). Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., & Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. .
- Bertoli, G., Júnior, L., Saotome, O., Dos Santos, A., Verri, F., Marcondes, C., et al. (2021). Bertoli, G.D.C.; Júnior, L.A.P.; Saotome, O.; Dos Santos, A.L.; Verri, F.A.N.; Marcondes, C.A.C.; Barbieri, S.; Rodrigues, M.S.; De Oliveira, J.M.P. An end-to-end framework for machine learning-based network intrusion detection system. *IEEE Access* 2021, 9.
- Bhavani TT, R. M., & 637–643., 2. p. (2020). Bhavani TT, Rao MK, Reddy AM. Network intrusion detection system using random forest and decision tree machine learning techniques. In: *First international conference on sustainable technologies for computational intelligence*, Springer; 2020. pp 637–643.
- Bhuyan, M., Bhattacharyya, D., Kalita, J. N., & Springer: Berlin/Heidelberg, G. 2. (2017). Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. *Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools*; Springer: Berlin/Heidelberg, Germany, 2017.
- Buczak, A. L.-1. (2012). Buczak, A. L., & Guven, E. (2016). "A survey of data mining and machine learning methods for cyber security intrusion detection." *IEEE Communications Surveys & Tutorials*, 18(2), 1153-1176.
- C. C. Ugwu, O. O.—1. (2020). C. C. Ugwu, O. Obe, S. Popoola, O. Adetunmbi, "A Distributed Denial of Service Attack Detection System using Long Short Term Memory with Singular Value Decomposition," 2020 IEEE 2nd International Conference on Cyberspac (CYBER NIGERIA), 2021,

pp. 112—118.

- C. Team, ". 1. (2022). C. Team, "corporatefinanceinstitute.com," 13 Decemebr 2022. [Online]. Available:.
- C. Xu, J. S.—4. (2018). C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection systemusing a deep neural network with gated recurrent units," IEEE Access,vol. 6, 48697—48707, 2018.
- CFI Team, ". c. (2019). CFI Team, "corporatefinanceinstitute.com," corporatefinanceinstitute, 13 December.
- Chen, H., Jiang, B., Ding, S., & Huang, B. D.-d.-s.—1. (2023). Chen, H.; Jiang, B.; Ding, S.X.; Huang, B. Data-driven fault diagnosis for traction systems in high-speed trains: A survey, challenges, and perspectives. IEEE Trans. Intell. Transp. Syst. 2020, 23, 1700–1716.
- Chen, K. &-s.-s.—1. (2019). Chen, K., & Wang, S. (2011). Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(1), 129–143.
- Cuelogic Technologies, ". C. (2022). Cuelogic Technologies, "medium.com," Cuelogic Technologies, 13 May 2019. [Online].
- D. Krotov and J. J. Hopfield, ". I.—7. (2019). D. Krotov and J. J. Hopfield, "Unsupervised learning by competinghidden units," Proceedings of the National Academy of Sciences,vol. 116, no. 16, pp. 7723–7731, 2019.
- D. Nehra, V. M.-s.-e.—3. (2019). D. Nehra,V. Mangat, and K. Kumar, "A Deep Learning Approach forNetwork Intrusion Detection Using Non-symmetric Auto-encoder,"In Intelligent Computing and Communication Systems, Springer,Singapore, pp. 371—382, 2021.
- Das S, G. P.—8. (2023). Das S, Gangwani P, Upadhyay H. Integration of machine learning with cybersecurity: applications and challenges. Artificial intelligence in cyber security: theories and applications. 2023. pp 67–81.
- detection, J. L.-b. (2021). JooHwa Lee and KeeHyun Park. Gan-based imbalanced data intrusion detection.
- Devaraj, D. R. (2012). Devaraj, D., R.UmaRani and J. P. Roselyn. 2007. Artificial.
- Drewek-Ossowicka, A. P. (2021). Drewek-Ossowicka, A., Pietrolaj, M., & Rumiński, J. (2021). A survey of neural.
- Elhanashi, A., Gasmi, K., Begni, A., Dini, P., Zheng, Q., Saponara, S. M.-B.-C.-I., et al. (2023). Elhanashi, A.; Gasmi, K.; Begni, A.; Dini, P.; Zheng, Q.; Saponara, S. Machine Learning Techniques for Anomaly-Based Detection System on CSE-CIC-IDS2018 Dataset. In Applications in Electronics Pervading Industry, Environment and Society: APPLEPIES 2022; S.
- Elkan, C. (. (2016). Elkan, C. (2000). Results of the KDD'99 classifier learning. Acm Sigkdd Explorations Newsletter,.
- Explanation., R. D. (2020). Rian Dolphin. (2020). LSTM Networks | A Detailed Explanation.
- Farah, N. A. (2018). Farah, N., Avishek, Md., Muhammad, F., Rahman, A., Rafni, M., & Md., D. (2015).

feature, S. N. (n.d.).

feature, S. N. (2018). Soosan Naderi Mighan and Mohsen Kahani. Deep learning based latent feature.

Ferrag, M. A. (2019). Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning.

G. Loukas, T. V.-b.-p.—3. (2017). G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," IEEE Access, vol. 6, pp. 3491–3508, 2017.

Gamage, S. &. (2020). Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion.

Garcia-Teodoro P, D.-V. J.-F.-b.-l., & 28:18–28. (2019). Garcia-Teodoro P, Diaz-Verdejo J, Macia'-Fernandez G, et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. Comput Secur. 2009;28:18–28.

George Loukas, T. V. (2017). George Loukas, Tuan Vuong, Ryan Heartfield, Georgia Sakellari, Yonpil Yoon,.

global-digital-report, ". (2022). global-digital-report, "wearesocial.com," 2018. [Online]. Available:.

Gu J, W. Z., & 77:354–77. (2018). Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, et al. Recent advances in convolutional neural networks. Pattern Recogn. 2018;77:354–77.

H. Altwaijry and S. Algarny, ". b. (2020). H. Altwaijry and S. Algarny, "Bayesian based intrusion detection system," Journal.

H. Altwaijry, ". b. (2013). H. Altwaijry, "Bayesian based intrusion detection system," in IAENG Transactions on Engineering Technologies:.

H. Liu and B. Lang, ". L. (2019). H. Liu and B. Lang, "Machine Learning and Deep Learning Methods for Intrusion.

H. U. Dike, Y. Z.-3. (2018). H. U. Dike, Y. Zhou, K. K. Deveerasetty, and Q. Wu, "Unsupervised learning based on artificial neural network: A review," in 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS). IEEE, 2018, pp. 322–327.

Halimaa, A., Sundarakantham, K. M.-2., & 916–920, p. (2019). Halimaa, A.; Sundarakantham, K. Machine learning based intrusion detection system. In Proceedings of the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 23–25 April 2019; pp. 916–920.

<https://github.com/lanjelot103>, L. (. (2020). Lanjelot. (2021). Patator Github Page. <https://github.com/lanjelot103>.

Ian Goodfellow, J. P.-A. (2014). Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David WardeFarley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems, 27, 2014.

IEEE, Z. Z., & 1–2., 2. p. (2018). Zhang Z. Improved Adam optimizer for deep neural networks. In: 2018

- IEEE/ACM 26th international symposium on quality of service (IWQoS). IEEE; 2018. p. 1–2.
- INSA, ". I. (2022). INSA, "esega.com," INSA, 10 2019 Accessed 28 01 2022. [Online]. Available:.
- InTech, T. O.-T. (2010). Taiwo Oladipupo Ayodele. (2010) New Advances in Machine Learning - Types of Machine Learning Algorithms. InTech.
- J. B. PhD, ". I. (2022). J. B. PhD, "machine learning mastery," 27 April 2021. [Online]. Available:.
- Jan Lansky, S. A. (2021). Jan Lansky, Saqib Ali, Mokhtar Mohammadi, Mohammed Kamal Majeed,.
- Javaid, A., Niyaz, Q., Sun, W., & Alam, M. A.-I. (2016). Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (formerly BIONETICS), New York, NY.
- Jazi, H. G.-b.-3. (2017). Jazi, H.H., Gonzalez, H., Stakhanova, N., Ghorbani, A.A.: Detecting http-based application layer dos attacks on web servers in the presence of sampling. *Comput. Netw.* 121, 25–36 (2017).
- Jongbo, O. C.-9. (2014). Jongbo, O. C. (2014). The role of research design in a purpose driven enquiry. *Review of Public Administration and Management*, 3(6), 87 - 94.
- K. Bekshentayeva, M. C. (2019). K. Bekshentayeva, M. Canute, Y. Kim, D. Lee, and A. Wong, "Network Intrusion Detection Using Various Deep Learning Approaches," BC V5A, 33rd Conference on Neural Information Processing Systems, Vancouver, Canada, 2019.
- K. Kottmann, P. H.-p. (2020). K. Kottmann, P. Huembeli, M. Lewenstein, and A. Acín, "Unsupervised phase discovery with deep anomaly detection," *Physical Review Letters*, vol. 125, no. 17, p. 170603, 2020.
- K. Yamanishi, J. i.-I.-3. (2017). K. Yamanishi, J. ichi Takeuchi, G. J. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proc. of the 6th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 320–324, Au.
- Kaiming He, X. Z. (2016). Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual.
- Kamarudin, M., Maple, C., Watson, T., & Safa, N. A.-b.-2. (2017). Kamarudin, M.H.; Maple, C.; Watson, T.; Safa, N.S. A logitboost-based algorithm for detecting known and unknown web attacks. *IEEE Access* 2017, 5, 26190–26200.
- Kasongo & Sun (2020) Kasongo SM, S. Y.-N., & 10.1186/s40537-020-00379-6., 7. d. (2020). Kasongo & Sun (2020) Kasongo SM, Sun Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. *Journal of Big Data*. 2020;7(1):38367. doi: 10.1186/s40537-020-00379-6. .
- Kasongo SM, S. Y.-N., & 7(1):1–20. (2020). Kasongo SM, Sun Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. *J Big Data*. 2020;7(1):1–20.
- Kasongo, S., & Sun, Y. P.-N. (2020). Kasongo, S.M.; Sun, Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. *J. Big Data* 2020.

- Khraisat, A. G. (2019). Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion.
- Kipf, T. N.-s. (2016). Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks.
- Kwon, D. K. (2019). Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2019). A survey of deep.
- Leevy, J. L. (2018). Leevy, J. L., & Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion.
- Lenzholder S, D. I., & 54–64, 2. p. (2017). Lenzholder S, Duchhart I, van den Brink A. The relationship between research design. In: van den Brink A, Bruns D, Tobi H, Bell S, editors. Research in Landscape Architecture: Methods and Methodology. New York: Routledge; 2017. pp. 54–64.
- Li HC, D. Z.-c., & 20(21):6114. (2020). Li HC, Deng ZY, Chiang HH. Lightweight and resource-constrained learning network for face recognition with performance optimization. Sensors. 2020;20(21):6114.
- Lin, P. Y. (2019). Lin, P., Ye, K., & Xu, C. Z. (2019). Dynamic network anomaly detection system by.
- Lo WW, Y. X., & 1–5., 2. p. (2019). Lo WW, Yang X, Wang Y. An xception convolutional neural network for malware classification with transfer learning. In: 2019 10th IFIP international conference on new technologies, mobility and security (NTMS). IEEE; 2019. p. 1–5.
- M. Alazab, S.-H. H. (2021). M. Alazab, S.-H. Hong, J. Ng, Louder bark with no bite: Privacy protection through the.
- M. Al-Qatf, Y. L.-H.-S. (2018). M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep Learning
- M. K. Khaleel, M. A. (2019). M. K. Khaleel, M. A. Ismail, U. Yunan, and S. Kasim, "Review on intrusion detection system based on the goal.
- M. Lucic, K. K.-s. (2018). M. Lucic, K. Kurach, M. Michalski, O. Bousquet and S. Gelly, "Are gans created equal? a large-scale.
- M. Mukherjee, "-O. A. (2016). M. Mukherjee, "Object-Oriented Analysis and Design,".
- M. S. a. Anuradha., ". I. (2012). M. S. a. Anuradha., "Network Intrusion Detection System for Denial of Service Attack.
- M. Zebari, R. Z.-S.-S.—1. (2019). M. Zebari, R. Zeebaree, and M. Yasin, "Real Time Video StreamingFrom Multi-Source Using Client-Server for Video Distribution," In2019 4th Scientific International Conference, Najaf (SICN), 2019. pp.109—114 .
- Mezina, A., Burget, R., & Travieso-González, C. N.-N.–1. (202). Mezina, A.; Burget, R.; Travieso-González, C.M. Network Anomaly Detection with Temporal Convolutional Network and U-Net model. IEEE Access 2021, 9, 143608–143622.
- Michie, D., Spiegelhalter, D., Taylor, C. M., Ellis Horwood Series in Artificial Intelligence: New York, N. U., & 13, V. (1994). Michie, D.; Spiegelhalter, D.J.; Taylor, C. Machine Learning, Neurall and Statistical Classification; Ellis Horwood Series in Artificial Intelligence: New York, NY, USA, 1994; Volume 13.

- Mikhail, F. & b.b.d. (2019). Mikhail, Fossaceca & Iammartino (2019) Mikhail JW, Fossaceca JM, Iammartino R. A semi-boosted nested model with sensitivity-based weighted binarization for multi-domain network intrusion detection. *ACM Transactions on Intelligent Systems and Technology*. 2.
- Mishra, P. V. (2019). Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2019). A detailed.
- Mishra, S., Datta-Gupta, A. C.—M., Mishra, S. D.-G., Elsevier: Amsterdam, T. N., & 97–118., p. (2018). Mishra, S.; Datta-Gupta, A. Chapter 5—Multivariate Data Analysis. In *Applied Statistical Modeling and Data Analytics*; Mishra, S., Datta-Gupta, A., Eds.; Elsevier: Amsterdam, The Netherlands, 2018; pp. 97–118.
- Moustafa & Slay (2016) Moustafa N, S. J.-N., & 25(1–3):18. (2019). Moustafa & Slay (2016) Moustafa N, Slay J. The evaluation of network anomaly detection systems : statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*. 2016;25(1–3):18.
- Moustafa N, S. J.-N., & 25:18–31. (2016). Moustafa N, Slay J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*. 2016 Apr 4;25:18–31.
- Munish S. and Anuradha., ". I. (2011). Munish S. and Anuradha., "Network Intrusion Detection System for Denial of Service.
- Murphy, K. P. (2020). Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press.
- N. Hallett, K. Y.-s.—7. (2020). N. Hallett, K. Yi, J. Dick, C. Hodge, G. Sutton, and G. Wang, "Deep learning based unsupervised and semi-supervised classification for keratoconus," In *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1—7.
- Nasteski, V. A.–6. (2017). Nasteski, V., An overview of the supervised machine learning methods. *Horizons. B.*, 4, 51–62, 2017, 10.20544/HORIZONS.B.04.1.17.P05.
- Okey, O., Maidin, S., Adasme, P., Lopes Rosa, R., Saadi, M., Carrillo Melgarejo, D., et al. (2022). Okey, O.D.; Maidin, S.S.; Adasme, P.; Lopes Rosa, R.; Saadi, M.; Carrillo Melgarejo, D.; Zegarra Rodríguez, D. BoostedEnML: Efficient technique for detecting cyberattacks in IoT systems using boosted ensemble machine learning. *Sensors* 2022, 22, 7409.
- Osisanwo, F. A.–1. (2017). Osisanwo, F.Y., Akinsola, J.E.T., Awodele, O., Hinmikaiye, J.O., Olakanmi, O., Akinjobi, J., Supervised Machine Learning Algorithms: Classification and Comparison. *Int. J. Comput. Trends Technol. (IJCTT)*, 48, 3, 128–138, June 2017.
- Pankaj Malhotra, L. V. (2015). Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. Long.
- Panwar, S., Negi, P., Panwar, L., & Raiwani, Y. I.–2.–2. (2019). Panwar, S.S.; Negi, P.S.; Panwar, L.S.; Raiwani, Y. Implementation of machine learning algorithms on CICIDS-2017 dataset for intrusion detection using WEKA. *Int. J. Recent Technol. Eng. Regul. Issue* 2019, 8, 2195–2207.
- Patil, N. K. (2021). Patil, N.V., Krishna, C.R., Kumar, K.: Distributed frameworks for detecting distributed denial of service attacks: a comprehensive review, challenges and future directions. *Concurr.*

- Comput. Pract. Exp. 33, 6197 (2021).
- Pisner, D., Schnyer, D. S., Elsevier: Amsterdam, T. N., & 101–121., p. (2020). Pisner, D.A.; Schnyer, D.M. Support vector machine. In Machine Learning; Elsevier: Amsterdam, The Netherlands, 2020; pp. 101–121.
- Prasad, R. &, & Rohokale, V. (. (2019). Prasad, R., & Rohokale, V. (2019). Artificial Intelligence and machine learning in cyber.
- Qazi, E., Faheem, M., & Zia, T. H.-L.-B. (2023). Qazi, E.U.H.; Faheem, M.H.; Zia, T. HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System. Appl. Sci. 2023, 13, 4921.
- R. Abdulhammed, H. M. (2019). R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features Dimensionality Reduction.
- R. B. Onwuegbuzie and A. J. Johnson, ". M. (2004). R. B. Onwuegbuzie and A. J. Johnson, "Mixed Methods.
- R. Panigrahi and S. Borah, ". d. (2020). R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing.
- Rubin J, P. S.-I., & 51(6):S18-21. (2018). Rubin J, Parvaneh S, Rahman A, Conroy B, Babaeizadeh S. Densely connected convolutional networks for detection of atrial fibrillation from short single-lead ECG recordings. J Electrocardiol. 2018;51(6):S18-21.
- S. Cooper, ". w. (2022). S. Cooper, "www.comparitech.com," www.comparitech, 4 November 2022. [Online].
- S. Eltanbouly and M. Bashendy, ". L. (2020). S. Eltanbouly and M. Bashendy, "Machine Learning Techniques for Network Anomaly,".
- S. M. H. Bamakan, H. W. (2019). S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, "An effective intrusion detection framework based on.
- S. N. Abd and H. R. Ibraheem, "-S. M. (2020). S. N. Abd and H. R. Ibraheem, "Rao-SVM Machine Learning Algorithm for Intrusion Detection System," Iraqi.
- S. T. F. Al-Janabi and H. A. Saeed, ". N. (2011). S. T. F. Al-Janabi and H. A. Saeed, "A Neural Network Based Anomaly Intrusion Detection System,".
- S.Defense, ". A. (2022). S.Defense,"SPADE,"[Online]. Available: <http://www.silicondefense.com/software/spice>.
- Saranya, T., Sridevi, S., Deisy, C., Chung, T., & Khan, M. P.–1. (2020). Saranya, T.; Sridevi, S.; Deisy, C.; Chung, T.D.; Khan, M.A. Performance analysis of machine learning algorithms in intrusion detection system: A review. Procedia Comput. Sci. 2020, 171, 1251–1260.
- Selvakumar K, S. L.-b., & 35(1):87-107. (2019). Selvakumar K, Sairamesh L, Kannan A. Wise intrusion detection system using fuzzy rough set-based feature extraction and classification algorithms. International Journal of Operational Research.2019;35(1):87-107.
- Sharma, R. S. (2020). Sharma, R., Sharma, K., Khanna, A., Study of Supervised Learning and Unsupervised

- Learning. Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET), 8, VI, June 2020.
- Sherstinsky, A. (. (2020). Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long.
- Shigeru Maya, K. U. (2019). Shigeru Maya, Ken Ueno, and Takeichiro Nishikawa. dlstm: a new approach for.
- Shiwei Shen, G. J.-g. (2017). Shiwei Shen, Guoqing Jin, Ke Gao, and Yongdong Zhang. Ape-gan: Adversarial.
- Shrestha A, M. A., & 7:53040–65. (2019). Shrestha A, Mahmood A. Review of deep learning algorithms and architectures. IEEE Access. 2019;7:53040–65.
- Singh Panwar, S., Raiwani, Y., & Panwar, L. E.-2. (n.d.). Singh Panwar, S.; Raiwani, Y.; Panwar, L.S. Evaluation of Network Intrusion Detection with Features Selection and Machine Learning Algorithms on CICIDS-2017 Dataset. In Proceedings of the International Conference on Advances in Engineering Science Managem.
- Singla, R. K.-b. (2018). Singla, R. K., Guleria, A., & Sharma, R. (2018). An overview of flow-based anomaly.
- Song, Y. &. (2018). Song, Y., & Lu, Y. (2015, April 25). Decision tree methods: Applications for classification and.
- Springer, L. Z. (2017). Li Z et al (2017) Intrusion detection using convolutional neuralnetworks for representation learning. In: International conferenceon neural information processing. Springer.
- Survey., I. R. (2020). Ian Reynolds. (2020). 2020 SANS Network Visibility and Threat Detection Survey.
- T. A. Tang, L. M.-b.-2. (2018). T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in sdn-based networks," in 20184th IEEE Conference on Network Softwarization and Workshops (NetSoft), Jun. 2018, pp. 202–206. DO I: 10.1.
- Tian H, C. S., & 22(5):1053–66. (2020). Tian H, Chen SC, Shyu ML. Evolutionary programming based deep learning feature selection and network construction for visual data classification. Inf Syst Front. 2020;22(5):1053–66.
- Tianyu Gu, B. D.-G. (2019). Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying.
- Tom Mitchell, M. H. (2015). Tom Mitchell, McGraw Hill (2015) Machine Learning.
- u J, W. Z., & 77:354–77. (2020). u J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, et al. Recent advances in convolutional neural networks. Pattern Recogn. 2018;77:354–77.
- V. Kanade, ". s. (2022). V. Kanade, "www.spiceworks.com," spiceworks, 24 march 2021. [Online]. Available:.
- Vinayakumar, R. A.-N. (2019). Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., &.
- W. Brown and . L. Stallings, C. S. (2018). W. Brown and . L. Stallings , Computer Security, Principles and

Practice, 4 th ed.

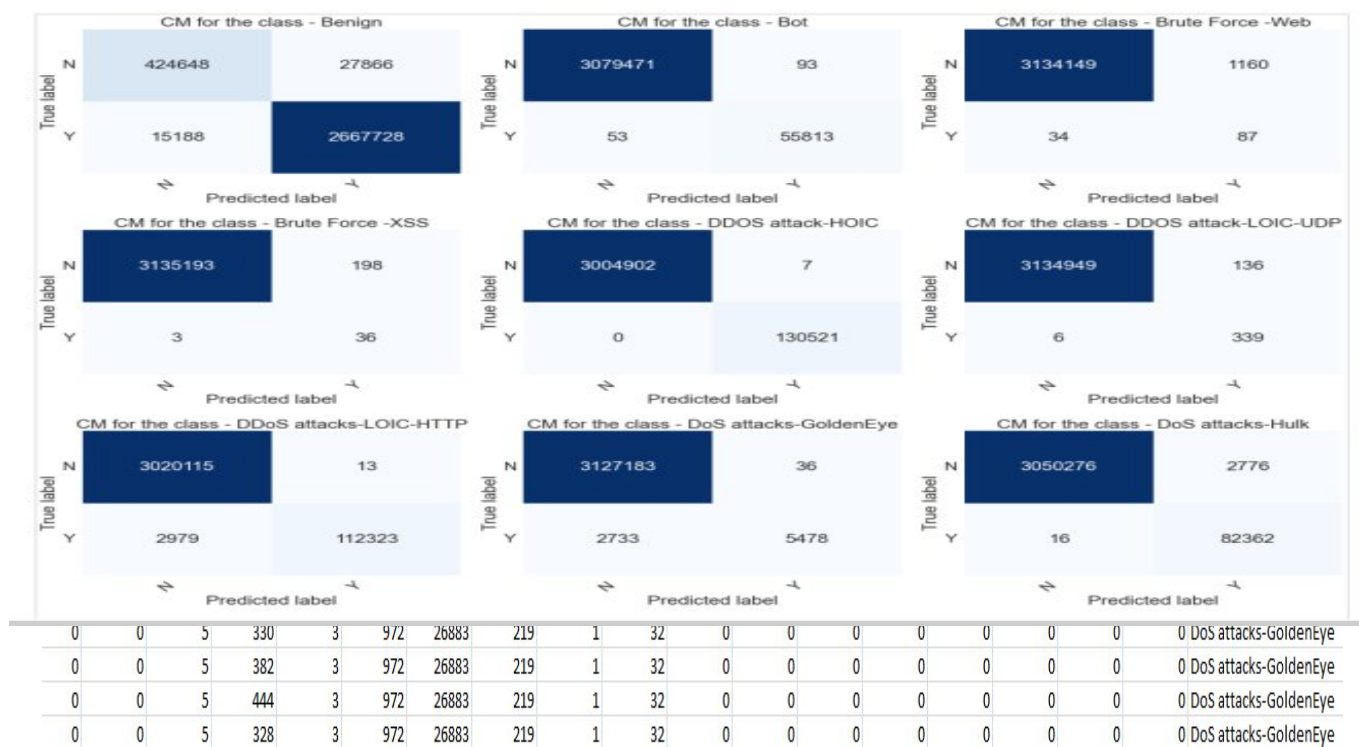
- W. Meng, W. T. (2018). W. Meng, W. T. Elmar , Qingju Wang,, Yu Wang and h. Jinguang, "When intrusion.
- Wang T, L. C., & 6:324. (2020). Wang T, Lu C, Yang M, Hong F, Liu C. A hybrid method for heartbeat classification via convolutional neural networks, multilayer perceptrons and focal loss. *PeerJ Comput Sci.* 2020;6:324.
- Wang, S. J. (2009). Wang, S. J., & Ye, N. (2008). "Constructing intrusion detection systems using data mining techniques." *International Journal of Information and Computer Security*.
- Wei Wang, M. Z. (2017). Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. Malware traffic classification using convolutional neural network for representation.
- X. Geerinck, ". 3. (2020). X. Geerinck, "medium.com," 3 jan 2020. [Online]. Available: <https://medium.com/>.
- Y. P. Reddy, P. V.-s.-l.-5. (2016). Y. P. Reddy, P. Viswanath, and B. E. Reddy, "Semi-supervised single-link clustering method," in *Computational Intelligence and Computing Research (ICCIC)*, 2016 IEEE International Conference on. IEEE, 2016, pp. 1–5. .
- Y. Zhang, X. C. (2019). Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network Intrusion Detection: Based on Deep Hierarchical.
- Y. Zhou, M. H.-l.—2. (2018). Y. Zhou, M. Han, L. Liu, J. S. He, and Y. Wang, "Deep learning approach for cyberattack detection," In *IEEE INFOCOM 2018-IEEE Conference, on Computer Communications Workshops, (INFOCOMWKSHPs)*, 2018, pp. 262—267. .
- Yang, S. T. (2020). Yang, S., Tan, M., Xia, S., & Liu, F. (2020). A method of intrusion detection based on.
- Yao G, L. T.-n.-n.-b., & 118:14–22. (2019). Yao G, Lei T, Zhong J. A review of convolutional-neural-network-based action recognition. *Pattern Recogn Lett.* 2019;118:14–22.
- Yi Zeng, H. G. (2019). Yi Zeng, Huaxi Gu, Wenting Wei, and Yantao Guo. deep–full–range: a deep.
- Yu, Y. L.-B.-L. (2018). Yu, Y., Liu, G., Yan, H., Li, H., & Guan, H. (2018). Attention-Based Bi-LSTM Model.
- Zha, L., Liao, R., Liu, J., Xie, X., Tian, E., & Cao, J. D.-t.-1. (2021). Zha, L.; Liao, R.; Liu, J.; Xie, X.; Tian, E.; Cao, J. Dynamic event-triggered output feedback control for networked systems subject to multiple cyber attacks. *IEEE Trans. Cybern.* 2021, 52, 13800–13808.
- Zhang, X., Chen, J., Zhou, Y., Han, L., & Lin, J. A.-l.-b.-9. (2019). Zhang, X.; Chen, J.; Zhou, Y.; Han, L.; Lin, J. A multiple-layer representation learning model for network-based attack detection. *IEEE Access* 2019, 7, 91992–92008.

## Appendix

| Bwd Bys/ | Bwd Pkts/ | Bwd Blk R | Subflow F | Subflow F | Subflow B | Subflow B | Init Fwd V | Init Bwd V | Fwd Act D | Fwd Seg S | Active Me | Active Std | Active Ma | Active Mii | Idle Mean | Idle Std | Idle Max | Idle Min | Label  |
|----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|------------|-----------|-----------|-----------|------------|-----------|------------|-----------|----------|----------|----------|--------|
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320579  | 704.2784 | 56321077 | 56320081 | Benign |
| 0        | 0         | 0         | 14        | 2168      | 12        | 2993      | 29200      | 232        | 8         | 32        | 1024353   | 649038.8   | 1601183   | 321569     | 11431221  | 3644991  | 15617415 | 8960247  | Benign |
| 0        | 0         | 0         | 2         | 64        | 0         | 0         | 244        | -1         | 0         | 32        | 0         | 0          | 0         | 0          | 0         | 0        | 0        | 0        | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320352  | 366.9884 | 56320611 | 56320092 | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320437  | 719.8347 | 56320946 | 56319928 | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320473  | 221.3244 | 56320629 | 56320316 | Benign |
| 0        | 0         | 0         | 3         | 182       | 4         | 935       | 29200      | 219        | 1         | 32        | 0         | 0          | 0         | 0          | 0         | 0        | 0        | 0        | Benign |
| 0        | 0         | 0         | 2         | 0         | 0         | 0         | 243        | -1         | 0         | 32        | 0         | 0          | 0         | 0          | 0         | 0        | 0        | 0        | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320814  | 53.03301 | 56320851 | 56320776 | Benign |
| 0        | 0         | 0         | 17        | 1280      | 12        | 2273      | 29200      | 232        | 8         | 32        | 2726234   | 0          | 2726234   | 2726234    | 17163940  | 0        | 17163940 | 17163940 | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320333  | 719.1276 | 56320841 | 56319824 | Benign |
| 0        | 0         | 0         | 57        | 4560      | 66        | 1728      | 253        | 256        | 56        | 20        | 0         | 0          | 0         | 0          | 0         | 0        | 0        | 0        | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320140  | 56.56854 | 56320180 | 56320100 | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320427  | 48.79037 | 56320461 | 56320392 | Benign |
| 0        | 0         | 0         | 5         | 137       | 3         | 442       | 29200      | 219        | 1         | 32        | 0         | 0          | 0         | 0          | 0         | 0        | 0        | 0        | Benign |
| 0        | 0         | 0         | 12        | 1184      | 8         | 2145      | 29200      | 232        | 6         | 32        | 566916.5  | 491680.3   | 914587    | 219246     | 13023255  | 7197923  | 18112955 | 7933555  | Benign |
| 0        | 0         | 0         | 3         | 64        | 0         | 0         | 244        | -1         | 1         | 32        | 0         | 0          | 0         | 0          | 0         | 0        | 0        | 0        | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320587  | 50.91169 | 56320623 | 56320551 | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320641  | 2.828427 | 56320643 | 56320639 | Benign |
| 0        | 0         | 0         | 3         | 0         | 0         | 0         | -1         | -1         | 0         | 0         | 0         | 0          | 0         | 0          | 56320749  | 91.21677 | 56320813 | 56320684 | Benign |

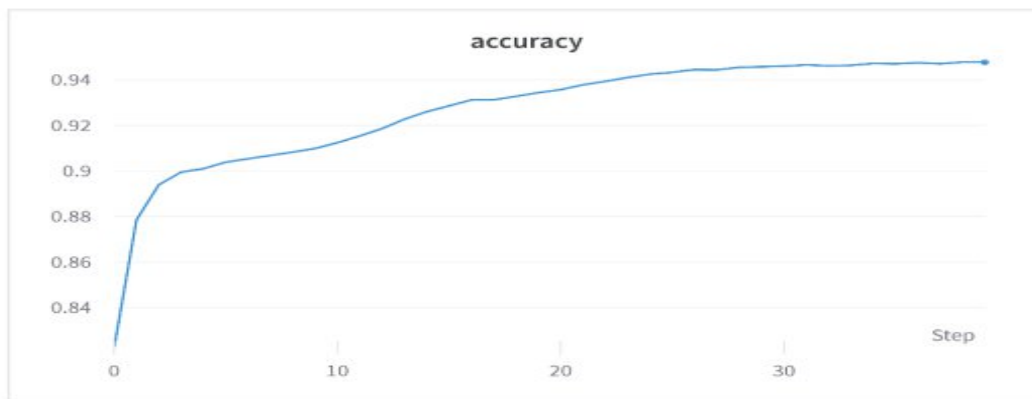
IDS data set with benign traffic

IDS data set with DOS golden eye traffic

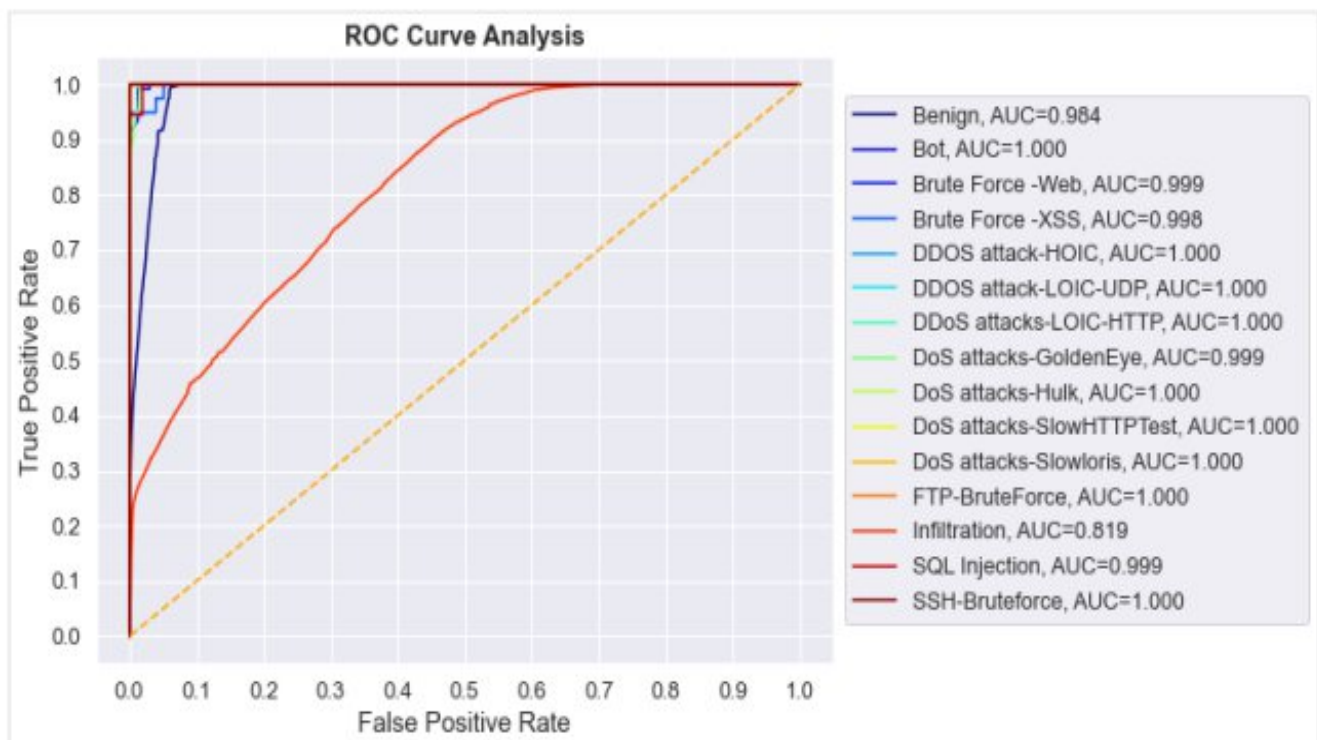


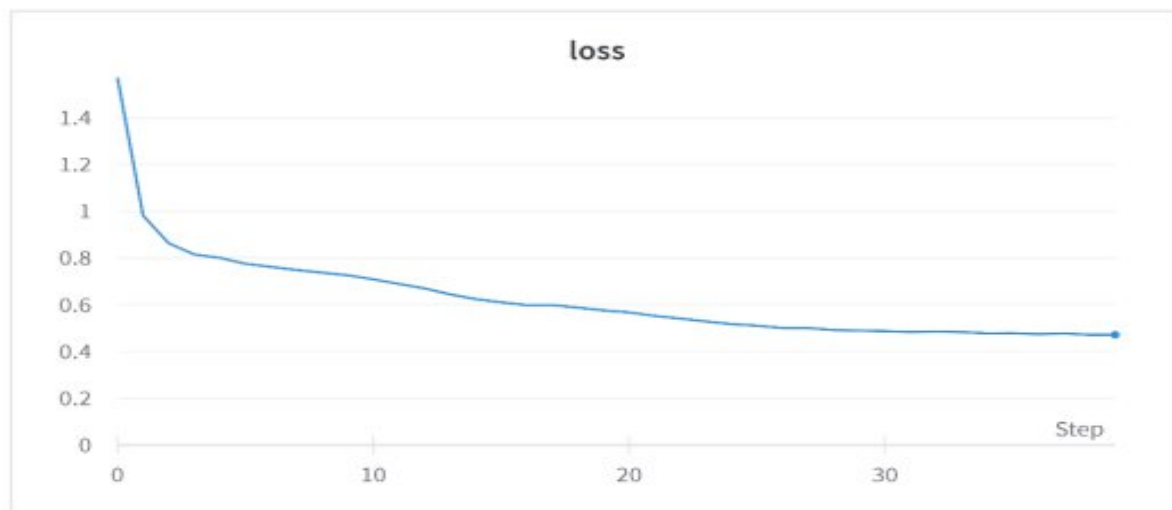
Confusion matrix using SMOTE analysis

Validation accuracy of ROC matrices evaluation



Training accuracy of the model





Training loose of the model