Volume 9, Issue 7, July – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24JUL1158

CNN-based Network Intrusion Detection and Classification Model for Cyber-Attacks

Uwadia Anthony. O
Department of Information and Communication Engineering
Elizade University, Ilara-mokin, Ondo State

Abstract:- A Convolution Neural Network (CNN)-based Network Intrusion Detection Model for Cyber-attacks is of great value in identifying and classifying attacks on any network. The Knowledge Discovery in Database Cup '99 dataset containing approximately 4,900,000 single connection vectors was divided into two phases; 75% of the total dataset was used during the learning process of the machine learning technique, while 25% was used on a fully trained model to validate and evaluate its performance. The model's performance indicated that it can detect and classify different classes of attacks with an accuracy of 98% with 20 epochs at a 0.001 learning rate using machine learning. The model loss for the training and validation was 7.48% and 7.98%, respectively, over 20 epochs, which implies that the model performed better on the training dataset. This study demonstrated that the convolutional Neural network-based Network Intrusion Detection and classification model shows high detection and low false negative rates. The CNN model offers a high detection rate and fidelity to unknown attacks, i.e., it can differentiate between already-seen attacks and new zero-day attacks. At the end of the experiment, the proposed approach *is suitable in modeling the network IDS for detecting intrusion attacks on computer networks thereby enabling a secured environment for the proper functioning of the system*

Keywords:- *Component; Network; Intrusion Detection System (IDS); Convolutional Neural Network(CNN); Artificial Neural Network(ANN); Machine Learning (ML)*

I. INTRODUCTION

The internet has become an unavoidable part of people's lives, permeating every sector of human activity. The importance of the internet is not lost to cybercriminals, who now prefer the cyber domain over the traditional world[14]. Cyber-attacks on computer systems have increased rapidly over the years. Any computer system connected to a network with open ports is at high risk of being accessed. With the increasing complexity and quantity of network packets transmitted in networks, the threat of cyberattacks is on the rise [12]. In the past, firewall systems provided a good enough level of security by following the defense-in-depth principle. However, with the advancement of cyber-attacks, the same level of security can't be ensured.

Due to the increasing number of network attacks, the Intrusion Detection System (IDS) has gained more importance and attention in practical applications for network security [24]. An IDS is considered the second line of defense used to prevent an intruder from accessing the network and protects the network's data, confidentiality, integrity, and availability[8]. The signature-based IDS or misuse detection methods can detect only known attacks and occur with the help of a database of attack signatures. However, the problem occurs with unknown attacks, i.e., zero-day attacks, because no signature is detected for the new attacks such as DDoS, phishing, and ransomware attacks. The consequence of a security breach might be rather severe, but large-scale cyber-attacks could be countered by effective Network Intrusion Detection Systems (NIDS) [18]. NIDS involves identifying unauthorised access to networks through the collection and analysis of network connections and is essential to securing and protecting the communication infrastructure. Network Intrusion detection systems have a wide range of applications, including fraud detection, network monitoring and security. In surveys, intrusion detection techniques have been identified as a significant field of research in ensuring network security.

Moreover, new techniques and methods are in high demand, among which the methods and techniques of ML and ANNs have been widely accepted. Due to the large volume of data, complexity and irregularity of the attacks, machine learning-based techniques can be adopted for an attack detection system. However, supervised learning increases the chances of a false positive rate during detection.

CNNs have become increasingly popular for network intrusion detection because of their major characteristics, such as automatic feature extraction and parameter sharing, that reduce computational complexity during training. In essence, CNN-based models have specific attributes that make them suitable. With CNNs showing good results in various pattern recognition problems, it is believed that this method - CNN can be used to classify different attacks. A CNN can learn and filter out irrelevant features, reducing the risk of false positive alerts [23]. CNNs are a segment of deep learning, and they are widely used in various fields ranging from computer vision [4] to object and pattern recognition.

In comparison to other machine learning models, Convolutional Neural Network (CNN)-based models are ideally used for detecting cyber-attacks by decomposing the input data using filters. This work aims to model a CNN-based Network intrusion detection and classification of cyber-attacks

In literature, several techniques for tackling NIDS have been presented in this respect. Still, many proposed identifying methods impose excessive computational complexity and require a significant computing capability to identify intrusions. To address this concern, the deep learning strategy is used in this study as it has lately earned enormous attention due to its flexibility, scalability, and potential to largely assign to (automatic) detection.

II. LITERATURE REVIEW

Network intrusion detection systems are the front-line defense for modern network security. IDSs play a pivotal role in the cyber security of organisational computer networks. Traditionally, Intrusion detection is performed in two stages: The first stage identifies threats, and the second stage creates rules for security devices, including generating rules for an IDS to identify possible threats and an IPS to save the organisation from possible threats based on these rules. The critical limitation of traditional IDS employing rule-based methods is invaluable to a precise measure [6]. The main aim of cyber security is to protect our systems, hardware, software and data from cyber-attacks that may compromise one or more of the following properties: Confidentiality, Integrity, Availability, and Authenticity[2]. Cyber-attacks can be divided into two categories: Passive attacks, which use monitoring or viewing techniques, whereas Active attacks, typically referred to as altering, modifying and destroying the data, can be achieved using various mechanisms.

In a rapidly evolving cybersecurity arms race, utilising the most advanced tools and techniques is critical to ensure timely and effective identification of unknown network attacks [23]. ML technology has transformed the Network Intrusion Detection System (NIDS) due to its ability to recognise unknown attacks effectively. Nowadays, most of the research on NIDS is based on the ML technology [25]. Various ML techniques are introduced for the detection of network attacks, including decision tree, support vector machine (SVM), genetic algorithms, ANN, k-nearest neighbours (KNN), and clustering [10]. ANN has received extraordinary attention among these ML techniques due to its effectiveness. State-of-the-art convolutional neural networks (CNNs) have produced numerous breakthrough results in computer vision, speech recognition, and natural language processing [26]. Machine learning-based IDSs can provide semiautomatic mechanisms, which are good learners and quick to secure organisational networks [24]. In recent years, deep learning-based models, specifically Convolutional Neural Networks (CNNs), have begun to demonstrate tremendous potential in tackling the problems of Network Intrusion Detection.

There has been an increase in the use of deep learning algorithms in developing a Network Intrusion Detection System (NIDS) to detect known and unknown network attacks. Many software systems and network environments have become more complex and prone to security breaches [8]. Several of these NIDS based on deep learning have demonstrated impressive detection accuracy by efficiently creating model representations from input data.

[11] proposed an intrusion detection model using AlexNet for multi-class classification of DoS and DDoS attacks. The model has 8(eight) layers with 4(four) convolutions and 3(three) fully connected layers. In their work, to balance the dataset, a Deep Convolutional Generative Adversarial Network(DCGAN) model was used to generate a fake real-time dataset in the same proportions as the actual real-time dataset where the MNIST data set was used. The Adam Search Optimization Algorithm was used to optimise the ANN parameters.

In 2023, [12] performed experiments on reduced feature sets (CCIDS 2018, UNSW 2016) and multiple supervised classification algorithms to assess the sensitivity and influence of individual features. Deep CNN and convolutional layers were not used in their work. They have only used 3(three) convolutional feature extractor layers and 2(two) fully connected layers with digital dropout and rectified linear activation functions in their model design.

In another study, [15] tried to achieve a more accurate network packet classification by chopping the time-stamped packet headers to fixed-term intervals. In their work, three ordinal interceptions were captured to detect network abnormities in the downstream traffic: Up-beam size, down-stream packet average length and the byte distribution range of the downstream packet.

This shows that studies have been conducted on network-based intrusion detection systems designed using machine-learning techniques [14]. Still, some of the reviewed literature revealed that incorporating CNN for experiments that involve large volumes of traffic, CNN also proved to be faster than most machine learning models [3]

III. METHODOLOGY

The subsections below cover the methodologies involved in the implementation of this study. Figure 1 shows the block diagram for the network Intrusion Detection Model.

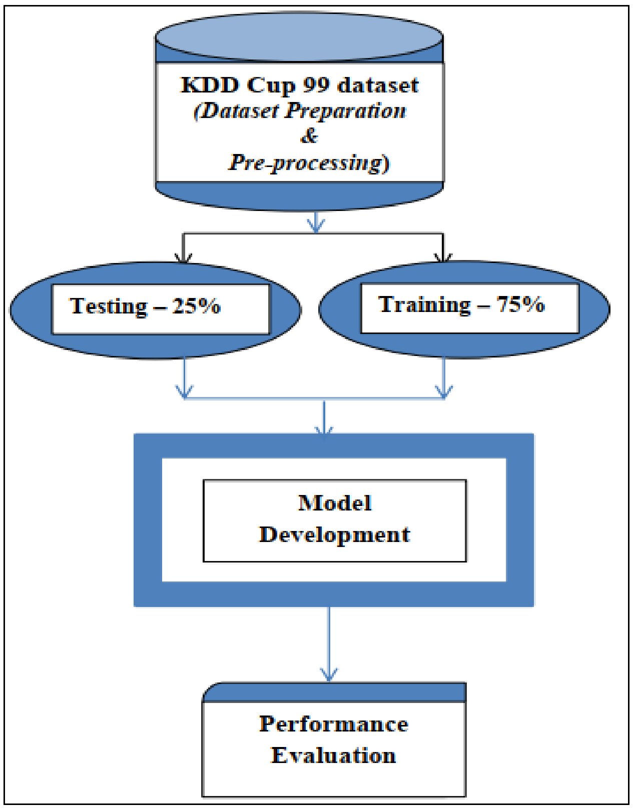


Fig 1: Block Diagram of the Network Intrusion Detection Model for Cyber-Attacks

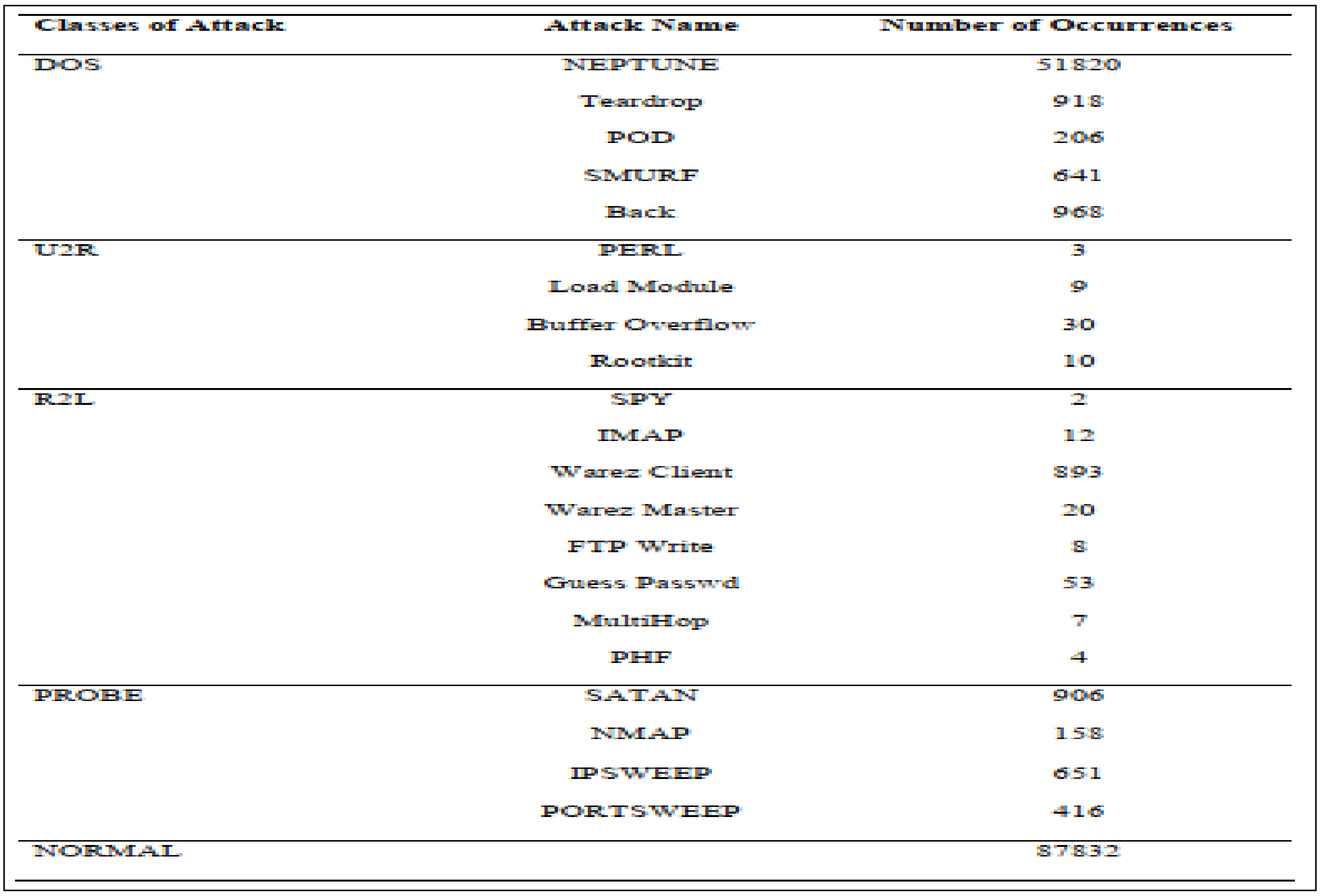
A. *Data Collection*

The first step in creating an intrusion detection system is to examine the dataset to detect intrusion attacks [7]. This stage is the Data Acquisition Process, which collects different connections.

The KDD Cup 99 dataset, based on the DARPA Intrusion Detection Evaluation dataset, was created at the MIT Lincoln Lab in an emulated network environment [16]. The dataset used in scenarios of this type typically consists of traffic to be analysed to detect an intrusive attack. It consists of approximately 4.9 million single connection vectors with 41 packet features, labelled as normal or an attack, with exactly one specific attack type gathered through an internet connection implemented based on TCP/IP connection settings [19].

The KDD Cup 99 Dataset, with an imbalanced class distribution favouring the most frequent class (DoS attacks), has been criticised for its deficiencies. Despite its shortcomings, it is widely used in Intrusion Detection Research Study and remains popular. The dataset includes 21 attacks grouped into four(4) categories: DoS, U2R, R2L and Probing, or as normal, each with distinct characteristics having different occurrences. 109,189 occurrences of records were extracted, as shown in Table 1

Table 1: Data Distribution of the Classes of Intrusion Attacks and Normal Connections



Classes of Attack	Attack Name	Number of Occurrences
DOS	NEPTUNE	51820
	Teardrop	918
	POD	206
	SMURF	641
	Back	968
U2R	PERL	3
	Load Module	9
	Buffer Overflow	30
	Rootkit	10
R2L	SPY	2
	IMAP	12
	Warez Client	893
	Warez Master	20
	FTP Write	8
	Guess Passwd	53
	MultiHop	7
	PHF	4
PROBE	SATAN	906
	NMAP	158
	IPSWEEP	651
	PORTSWEEP	416
NORMAL		87832

The fundamental features of any internet connection implemented based on the TCP/IP connection setting are shown in Table 2.

Table 2: Basic Features of TCP/IP Connection.

Features	Type	Description
Total duration of connections in second	continuous	Length (number of seconds) of the connection
Total number of bytes from sender to receiver	continuous	number of data bytes from source to destination
Total number of bytes from receiver to sender	continuous	number of data bytes from destination to source
Total number of wrong fragments	continuous	number of "wrong" fragments
Total number of urgent packets	continuous	number of urgent packets
Protocol type	discrete	type of the protocol e.g. TCP, UDP, etc.
Type of service	discrete	network service on the destination, e.g., HTTP, telnet, etc.
The status of the connection (normal or error)	discrete	normal or error status of the connection
Label (1) if the connection is established from to the same host. Otherwise label (0)	discrete	1 if successfully connected; 0 otherwise

Source: [1]

B. *Data Pre-Processing and Cleaning*

After collecting the dataset from trusted sources, the second stage is the Data pre-processing and cleaning stage, which is the importing procedure of the dataset, as shown in Figure 2 and implementing different statistical measurement values such as occurrence distribution, classes of attacks and percentage of occurrences.

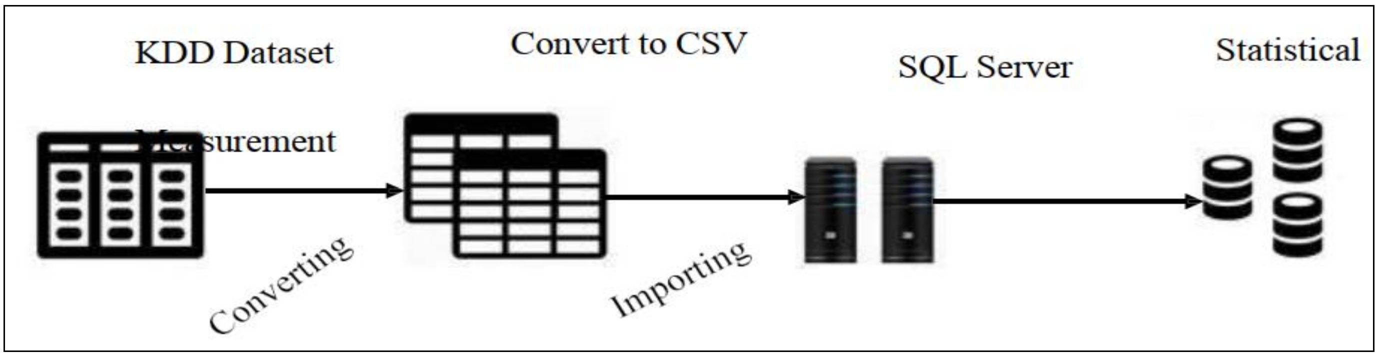


Fig 2: Importing Procedure of KDD Cup '99 Dataset

In the data preparation stage, data cleaning, which is crucial, involves removing null values and duplicate entries from the dataset and storing them in a pickle file for future use. The Characteristics of the KDD Cup '99 dataset, including data points, features and output labels, are counted and explored through Exploratory Data Analysis (EDA) using Python libraries like matplotlib, pandas and seaborn. Utility functions like Bi-variate and Univariate Analysis are employed to create plots showing the percentage distribution of the data points for each class and feature in the dataset. Data encoding converts categorical data into a numerical form for model training. Finally, the data is then organised into "x" (training variables) and "y" (expected outcomes), which represent the 21 different classes of attacks. This stage strips the datasets of errors, missing values, and unnecessary features. This stage usually involves decision-making before gleaning the information from the dataset.

C. *Dataset Split*

The dataset was split into two sections: a training set to train the CNN detection algorithm and a test set completely hidden from the training process. 75% of the dataset was used for training, while 25% was used to evaluate the model's performance, which was randomly selected with their statistics, as shown in table 3

Table 3: Dataset Percentage Splitting

Dataset	Training	Testing	Total
Percentage of the total dataset	75%	25%	100%
Number of images	109,189	36,397	145,586

D. Model Design and Development

In this research study, an interlinked multi-layered neural network was used. The five-layered neural network is structured with several trainable parameters, as shown in table 4.

The multiple layered neural networks earn over the input data using a selected kernel filter to extract important features seen as important in the model. The model training used the activation function to incorporate non-linearity effects into the developed model. Two activation (ReLU and softmax) functions were used in this research. These layered neural networks used the same activation function (ReLU), while the Softmax activation function was used for the last layer of the CNN, which is the denser layer (outer layer) due to its capability to handle the classification of multi-classification problems.

Table 4: Summary of the CNN Model Parameter

Layer(type)	Output Shape	Trainable Parameters
Dense(Dense)	None, 10	1,210
Dense_1(Dense)	None, 50	550
Dense_2(Dense)	None, 10	510
Dense_3	None, 1	11
Dense_4	None, 23	46
Total Trainable Parameters		2,327

E. Model Flowchart

The flowchart for the model developed in this study is shown in Figure 3.

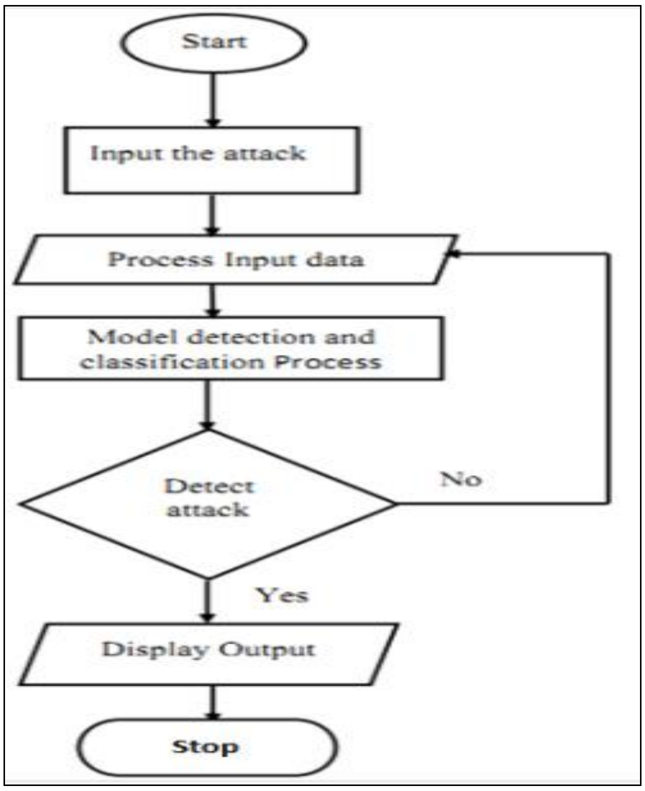


Fig 3: Flowchart of the Network Intrusion Detection and Classification Model

F. Model Implementation and Environment

This study was implemented in Python, using TensorFlow with Keras in the backend [9] on a core i3, 4GB DDR RAM, and a web IDE for colab with Windows 10 operating system. Machine learning libraries like Sci-kit learn were used, other libraries such as Pandas were used for data analysis and manipulation, NumPy was used for n-dimensional array support, and Matplot lib was used to produce graphs for the result.

G. Performance Evaluation

As an essential part of the process of research and development of the network-based intrusion detection and classification systems, performance metrics provide transformation in developing the accuracy, precision, recall, false positive rate, detection rate, and computational cost of machine learning-driven detection model [13]. Thus, these are used to model the system according to the specific requirement of the problem because all the performance matrix's computation depends on the type of classes present in it.

For the performance evaluation of the proposed model, statistical metrics were used, which include Detection Rate (DR), False Alarm Rate (FAR), Efficiency (Ef), Micro Precision, Micro Recall and Micro F1-score.

$$\text{Detection Rate}\,(DR) = \frac{TP}{(TP+FN)} \quad (1)$$

$$\text{False Alarm Rate}(FAR) = \frac{FP}{(TN+FP)} \quad (2)$$

$$\text{Efficiency}(Ef) = \frac{DR}{FAR} \quad (3)$$

- Precision (PN): the ratio of the number of cases the model could accurately predict the class of test data over the sum of correct predictions and those the model assumed to be correct but was wrong. [20] expressed precision mathematically as presented in equation 4

$$PN = \frac{TP}{TP+FP} \quad (4)$$

- Accuracy: is the ratio of accurately classified data items to the total number of observations. It is the closeness of a measured value to a known value. [20] expressed accuracy mathematically as presented in equation 5.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

- Recall (RL): This is also referred to as sensitivity. It is the percentage of cases that the model was correct in predicting over the sum of correct predictions and incorrectly classified cases. [20] expressed recall mathematically, as presented in equation 6

$$\text{RL} = \frac{TP}{(TP+FN)} \quad (6)$$

- PN and RL values exist between 0 and 1; the larger value indicates better performance.
- F-measure (F1-score): is calculated by PN and RL due to multi-class Classification problems. The harmonic mean of these two values forms the F1-score of the model and is a robust way to compare and evaluate the developed model. [20] expressed F1-score mathematically as presented in equation 7

$$\text{f1-score} = \frac{1}{\frac{1}{2}(\frac{1}{RL}+\frac{1}{PN})} = \frac{2RL*PN}{RL+PN} = 2*\left(\frac{1}{\frac{1}{precision}+\frac{1}{recall}}\right) \quad (7)$$

Where:

TP: True Positives is the number of legal packets identified correctly by the system and reach the destination safely.

TN: True Negatives is the number of packets dropped in the network before reaching the destination.

FP: False Positives is the number of attack packets that are forwarded to the destination falsely.

FN: False Negatives is the number of legal packets that are discarded falsely

If RL or PN has a small value, then the F1-score tends to have a small value, which means that the larger f1-score depends on the larger RL or PN value. This unique metric has a value between 0 and 1; a larger f1-score indicates better performance.

Generally, the F1-score summarises the precision and recall with an emphasis on the available threshold, which is important because it compares different prediction models on the problem with varied probabilities, decision thresholds, and constraints for collaborating the NIDS with the same deployment.

- Confusion Matrix**:** Confusion matrix is one of the most intuitive and descriptive metrics used to find the accuracy and correctness of a machine learning algorithm. The confusion matrix provides a more comprehensive evaluation of the performance of the NIDS by classifying samples based on their true and predicted labels [21]. From the confusion matrix, the true positive rate (TPR), false negative rate (FNR), false positive rate (FPR), and true negative rate (TNR) can be calculated to compare the performance of the NIDS on different classes of samples [14]. It is a visual representation of the model's performance, and its main usage is in classification problems where the output contains two or more classes. Precision, Accuracy, and Recall can be called through the confusion matrix for the model, which is depicted in figure 4, showing the values of TP, TN, FP and FN.

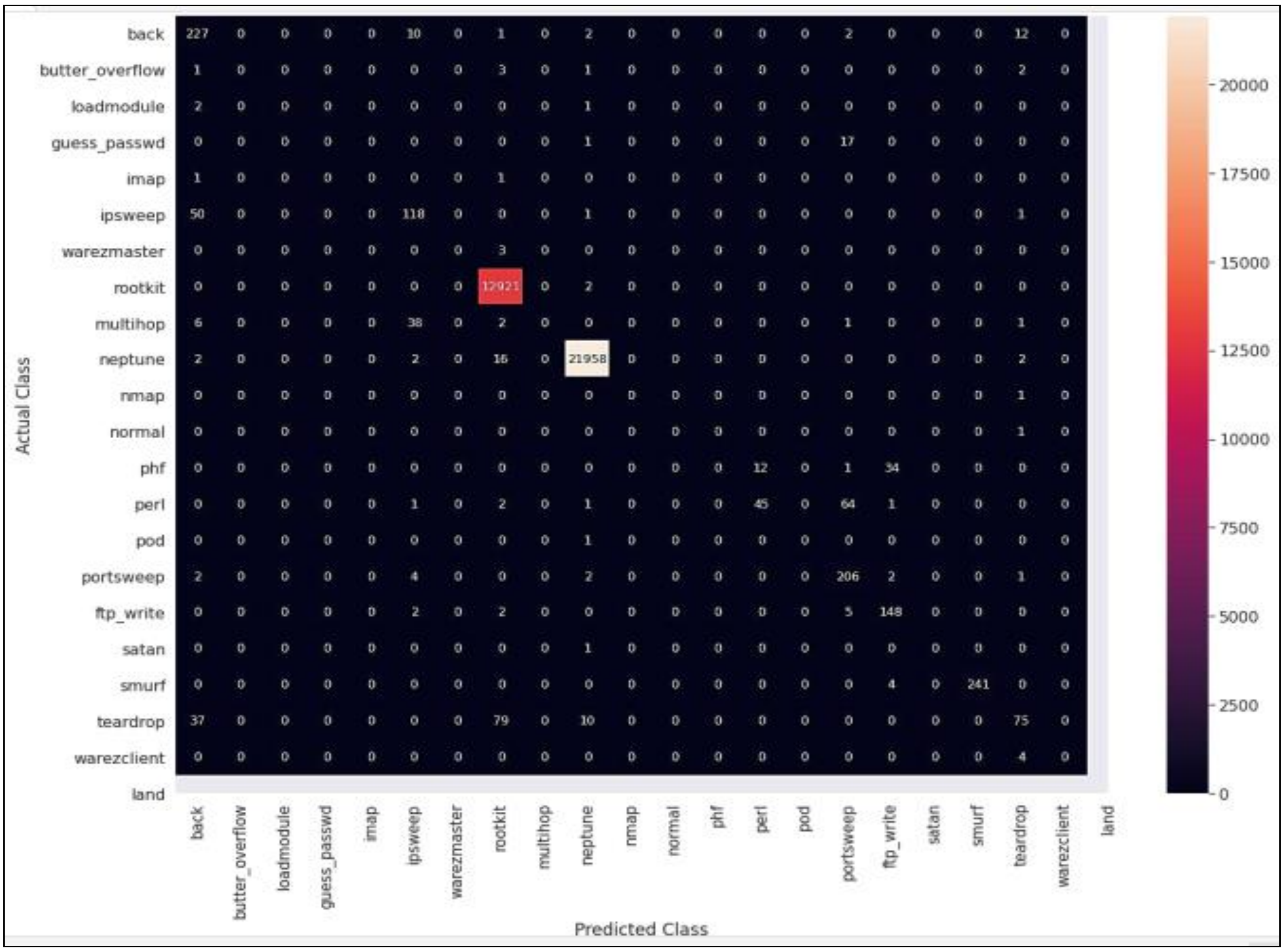


Fig. 4: Confusion Matrix for the Model

IV. RESULT AND DISCUSSION

The model loss curve of the trained model depicted in Figure 5 shows that the training and validation loss curves overlap. The graph shows that the model performed optimally on the trained and validation set, with a decreasing validation loss, until the 20th epoch. This investigated the model fitting and overfitting over the training dataset.

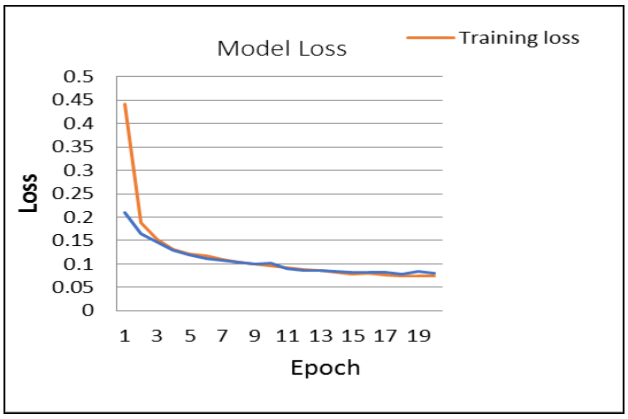
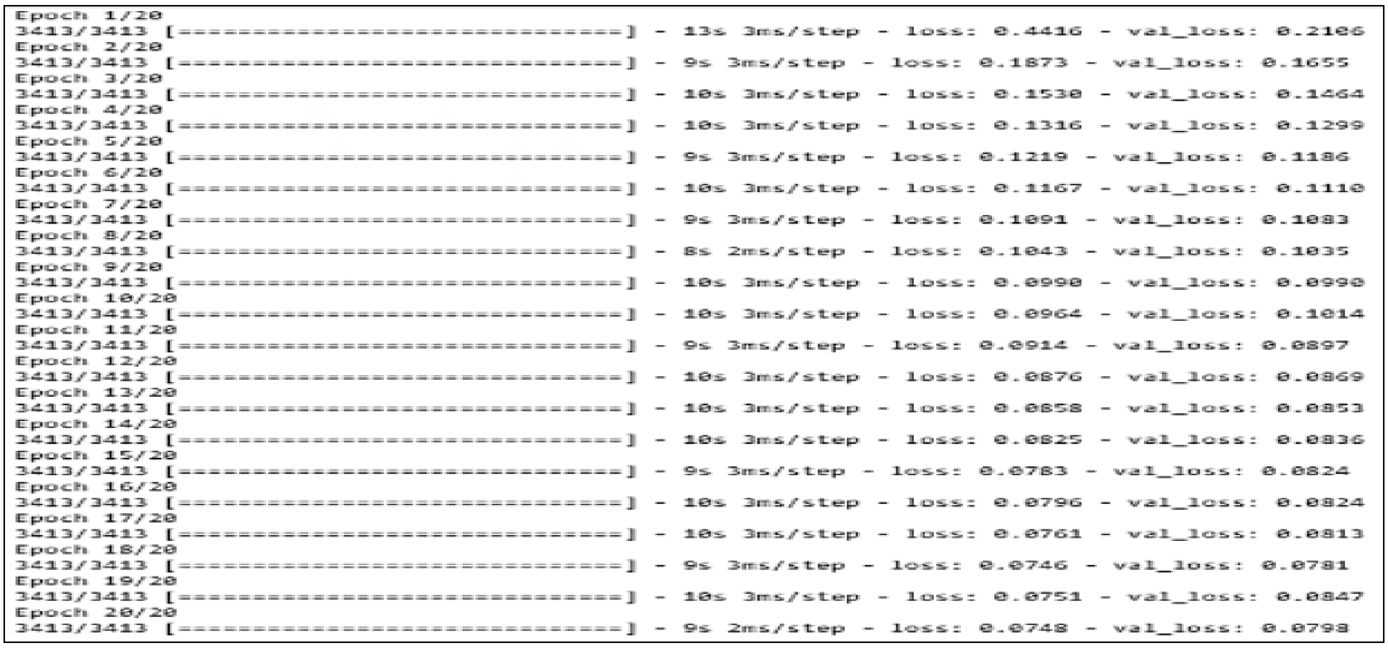


Fig 5: Model Loss Curve for the Trained Model

Figure 4 shows that the model has 446 misclassified attacks, which implies that the total false positive rate was 446 connections, meaning that approximately 1.2% of normal connections were classified into wrong categories and an average detection rate of approximately 98.8%.

The model loss obtained through certain iterations recorded a training loss of 7.48% and a validation loss of 7.98% over 20 epochs, as illustrated in Figure 6. The model learning rate starts decreasing right from the beginning of the training and slowly stabilises at the 12th epoch; the learning rate of 0.001 used with the batch size indicates a good model training process.



```
Epoch 1/20
3413/3413 [==============================] - 13s 3ms/step - loss: 0.4416 - val_loss: 0.2106
Epoch 2/20
3413/3413 [==============================] - 9s 3ms/step - loss: 0.1873 - val_loss: 0.1655
Epoch 3/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.1530 - val_loss: 0.1464
Epoch 4/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.1316 - val_loss: 0.1299
Epoch 5/20
3413/3413 [==============================] - 9s 3ms/step - loss: 0.1219 - val_loss: 0.1186
Epoch 6/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.1167 - val_loss: 0.1110
Epoch 7/20
3413/3413 [==============================] - 9s 3ms/step - loss: 0.1091 - val_loss: 0.1083
Epoch 8/20
3413/3413 [==============================] - 8s 2ms/step - loss: 0.1043 - val_loss: 0.1035
Epoch 9/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.0990 - val_loss: 0.0990
Epoch 10/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.0964 - val_loss: 0.1014
Epoch 11/20
3413/3413 [==============================] - 9s 3ms/step - loss: 0.0914 - val_loss: 0.0897
Epoch 12/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.0876 - val_loss: 0.0869
Epoch 13/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.0858 - val_loss: 0.0853
Epoch 14/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.0825 - val_loss: 0.0836
Epoch 15/20
3413/3413 [==============================] - 9s 3ms/step - loss: 0.0783 - val_loss: 0.0824
Epoch 16/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.0796 - val_loss: 0.0824
Epoch 17/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.0761 - val_loss: 0.0813
Epoch 18/20
3413/3413 [==============================] - 9s 3ms/step - loss: 0.0746 - val_loss: 0.0781
Epoch 19/20
3413/3413 [==============================] - 10s 3ms/step - loss: 0.0751 - val_loss: 0.0847
Epoch 20/20
3413/3413 [==============================] - 9s 2ms/step - loss: 0.0748 - val_loss: 0.0798
```

Fig 6: Snippet of Model Training Process with Epoch

A summary of the experimental hyperparameters used to train the CNN Model in this study is shown in table 5.

Table 5: Hyperparameters used for Training the CNN Model in this Study

Hyperparameter	Value
Epoch	50
Activation Function	ReLU, Softmax
Loss Function	Categorical Cross Entropy (CCE)
Optimization algorithm	ADAM
Learning rate	0.001
Patience	5
Verbose	1
Min_delta	1e-3

V. CONCLUSION

This study developed a CNN-based Network Intrusion detection/classification model for cyber-attacks to classify different cyber-attacks using the KDD Cup 99 dataset. After applying hyperparameter optimisation techniques, a single dataset – KDD Cup 99 dataset, which contains 21 classes of attacks was used to train the model.

The experimental result shows that the developed model achieved an accuracy of 98% over 20 epochs with a learning rate of 0.001, which accurately classified and detected 21 different classes of attacks.

The result shows that the hyperparameter and optimisation improved the overall performance of the developed model.

REFERENCES

[1]. M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, “Evaluation of machine learning algorithms for the intrusion detection system”, *In 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, 2017, 000277-000282
[2]. I. Al-Turaiki, and N. Altwaijry, “A Convolutional Neural Network for Improved Anomaly-Based Network Intrusion Detection”,2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8233218/
[3]. A. Andalib, and V. Vakili Tabataba, “An Autonomous Intrusion Detection System Using an Ensemble of Advanced Learners”, 2020, https://arxiv.org/pdf/2001.11936
[4]. B. Cao, C. Li, Y. Song, and X. Fan, “Network Intrusion Detection Technology Based on Convolutional Neural Network and BiGRU”, 2022, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9019421/
[5]. L. D’hooge, M. Verkerken, T. Wauters, F. De Turck, and B. Volckaert, “Investigating Generalised Performance of Data-Constrained Supervised Machine Learning Models on Novel, Related Samples in Intrusion Detection”, 2023, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9960990/
[6]. M. Dima Genemo, “Suspicious activity recognition for monitoring cheating in exams”, 2022, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8866922/

[7]. M. Gao, L.. Ma, H. Liu, Z. Zhang, Z. Ning, and J. Xu, " Malicious Network Traffic Detection Based on Deep Neural Networks and Association Analysis", 2020, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7085765/

[8]. A. Henry, S. Gautam, S. Khanna, K. Rabie, T. Shongwe, P. Bhattacharya, B. Sharma, and S. Chowdhury, "Composition of Hybrid Deep Learning Model and Feature Optimization for Intrusion Detection Syste", 2023, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9866711/

[9]. N. Ketkar, "Introduction to keras", *In Deep Learning with Python*, Apress, Berkeley, CA, 2017, pp. 99-111.

[10]. M. Kodys, Z. Lu, K. Wai Fok, and V. L. Thing, "Intrusion Detection in Internet of Things using Convolutional Neural Networks", 2022, https://arxiv.org/pdf/2211.10062

[11]. A. Kumar Silivery, and R. Mohan Rao Kovvur, "A model for multi-attack classification to improve intrusion detection performance using deep learning approaches", 2023, https://arxiv.org/pdf/2310.16380

[12]. A. Kumar Silivery, K. Ram Mohan Rao, and L. Suresh Kumar, "An Effective Deep Learning Based Multi-Class Classification of DoS and DDoS Attack Detection" 2023, https://arxiv.org/pdf/2308.08803

[13]. M. Mihailescu, D. Mihai, M. Carabas, M. Komisarek, M. Pawlicki, W. Hołubowicz, and R. Kozik, "The Proposition and Evaluation of the RoEduNet-SIMARGL2021 Network Intrusion Detection Dataset", 2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8272217/

[14]. X. H. Nguyen, X. D. Nguyen, H. H. Huynh, and K. H. Le, "Realguard: A Lightweight Network Intrusion Detection System for IoT Gateways" 2022, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8778231/

[15]. V. Ramanathan, K. Mahadevan, and S. Dua, "A Novel Supervised Deep Learning Solution to Detect Distributed Denial of Service (DDoS) attacks on Edge Systems using Convolutional Neural Networks (CNN) ", 2023, https://arxiv.org/pdf/2309.05646

[16]. M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of Network-Based Intrusion Detection Data Sets", *Computers & Security*, 2019, vol. 86, pp 147-167

[17]. A. A. Sayed, A. A. Taher Azar, A. Ella Hassanien, and S. El-Ola Hanafy, "Negative Selection Approach Application in Network Intrusion Detection Systems", 2014, https://arxiv.org/pdf/1403.2716

[18]. I. Shivhare, J. Purohit, V. Jogani, S. Attari, and D. Madhav Chandane, "Intrusion Detection: A Deep Learning Approach", 2023, https://arxiv.org/pdf/2306.07601

[19]. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", *In 2009 IEEE* symposium on computational intelligence for security and defense applications, 2009, pp. 1-6

[20]. M. Vakili, M. Ghamsari, and M. Rezaei, "Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification", *arXiv preprint arXiv:2001.09636*, 2020, pp. 1-13.

[21]. W. Wang, F. Harrou, B. Bouyeddou, S. M. Senouci, and Y. Sun, Y, "A stacked deep learning approach to cyber-attacks detection in industrial systems: application to power system and gas pipeline systems", 2022, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8490 44/

[22]. Z. Wang, F. A. Ghaleb, A. Zainal, M. Md Siraj, and X. Lu, "An efficient intrusion detection model based on convolutional spiking neural network, 2024, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10963367/

[23]. P. Wu, H. Guo, and R. Buckland, "A Transfer Learning Approach for Network Intrusion Detection",2019, https://arxiv.org/pdf/1909.02352

[24]. T. Ahmad, D. Truscan, J. Vain, and I. Porres, "Early Detection of Network Attacks Using Deep Learning, 2022, https://arxiv.org/pdf/2201.11628

[25]. O. Ceviz, P. Sadioglu, S. Sen, and V. G. Vassilakis, "A Novel Federated Learning-based Intrusion Detection System for Flying Ad Hoc Networks', 2023, https://arxiv.org/pdf/2312.04135

[26]. H. Dhillon and A. Haque, "Towards Network Traffic Monitoring Using Deep Transfer Learning", 2021, [https://arxiv.org/pdf/2101.00731