Babylonian Journal of Internet of Things
Vol.2024, **pp**. 60–69
DOI: https://doi.org/10.58496/BJIoT/2024/008; ISSN: 3006-1083
https://mesopotamian.press/journals/index.php/BJIoT

Research Article

A Real-Time Intrusion Detection System for DoS/DDoS Attack Classification in IoT Networks Using KNN-Neural Network Hybrid Technique

Aisha Ibrahim Gide [1], Abubakar Aminu Mu'azu [1*]

[1] *Department of Computer Science, Umaru Musa Yar'adua University Katsina ,Nigeria.*

ARTICLE INFO

Article History

Received 05 May 2024
Accepted 10 Jun 2024
Published 05 Jul 2024

Keywords

IoT security

real time DoS/DDoS attacks detection

Kth Nearest Neighbor (KNN)

dense neural networks

ABSTRACT

As more devices are connected to the Internet through the Internet of Things (IoT), there are huge security challenges. One of the major problems is Distributed Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. These attacks floods networks with useless traffic, disrupting IoT services. There is a need for better security measures to handle this. Intrusion Detection Systems (IDS) is used to find suspicious activities, but many of them can't keep up with new types of attacks in real time. This study focuses on creating an efficient real time hybrid framework that uses the Kth Nearest Neighbor (KNN) algorithm and dense neural networks. This proposed framework aims to identify and categorize DoS/DDoS attacks in real time through the utilization of a simulation model and MQTT-IoT-IDS2020 dataset and compared with existing frameworks, our proposed framework excels in accuracy, precision, and recall.

1. INTRODUCTION

The internet, once primarily a tool for academic research, has now become an essential service, comparable in significance to utilities like electricity, water, and gas. Wherever there's something valuable, there's also the presence of crime aiming to exploit that value through illicit use of technology or preventing others from accessing that resource. Because the internet is interconnected globally, it's susceptible to attacks from anywhere, making cybersecurity a critical concern. A significant number of cyber-attacks, encompassing DoS and DDoS attacks, are primarily executed by automated systems (Bots or Botnets) directed by human operators, utilizing multiple devices with access to internet [1].

Securing the Internet of Things (IoT) against potential cyberattacks is a highly intricate undertaking. Nevertheless, it proves to be somewhat manageable when examined in a structured manner. Each layer in this structure presents its unique challenges and vulnerabilities that need to be identified in order to ensure protection against a range of attacks. A network vulnerability is essentially a weakness within the network infrastructure that could potentially allow an intruder to determine the extent of their intrusion into the network's cybersecurity. This vulnerability is particularly perilous, as it could lead to an attack if it goes unnoticed or unaddressed. [2].

Intrusion detection serves a dual purpose by not only identifying successful intrusions but also monitoring attempts to breach security, supplying crucial information for timely countermeasures. The Intrusion Detection System (IDS) has emerged to examine key aspects of computer systems and networks, aiming to identify abnormal behaviors against system policies or signs of aggression within the network. Based on the monitored environment, IDS is categorized into two groups: Network-based IDS (NIDS) which examines network packets within specific network segments or devices to pinpoint any suspicious activity, while Host-based IDS (HIDS) monitors the behavior of hosts, including system calls to efficiently detect intrusions. Understanding the current technology of these systems is the initial step in designing an effective detection system for DoS/ DDoS attacks [3].

In recent years, machine learning is developing with incredible speed. Machine learning techniques have found extensive applications across different fields, including pattern recognition, natural language processing, and computational learning. Through these methods, computers acquire the capability to function without explicit programming, generating

*Corresponding author. Email: abubakar.muazu@umyu.edu.ng

algorithms that learn from data and autonomously make data-driven decisions or predictions. A variety of machine learning approaches, such as supervised learning, unsupervised learning, and reinforcement learning (RL), have been widely adopted to improve network security components like authentication, access control, antijamming offloading, and malware detection. [4].

In 2006, Hinton introduced a novel training approach known as layer-wise-greedy-learning, signifying the inception of deep learning methods. The fundamental concept behind layer-wise-greedy-learning is that unsupervised learning is conducted during network pre-training prior to the subsequent layer-by-layer training, some of the researchers that adopted this method are [5, 6 ,7] The widespread adoption of unsupervised learning is attributed to the pre-training process, where non-random initial values are assigned to the network. This pre-training step enhances the efficiency and effectiveness of the unsupervised learning approach. As a result, the training process can reach a more favorable local minimum, leading to a faster convergence rate.

[5] pointed out that Network Intrusion Detection Systems (NIDS) serve as classifiers, distinguishing between unauthorized or abnormal traffic and authorized or regular traffic. Several researchers, including [8],[9] have employed machine learning approaches to effectively categorize attack traffic, achieving high test accuracy. They utilized supervised methods such as k-nearest neighbors and random forest. Techniques like support vector machines (SVMs), naive Bayes, K-nearest neighbor (K-NN), neural networks (NNs), deep NNs (DNNs), and random forest are applied in supervised learning to label network traffic for constructing classification or regression models. On the other hand, unsupervised learning, which doesn't rely on labeled data, explores the similarities among unlabeled data to cluster them into different groups [10].

2. PROBLEM STATEMENT

Over the past decade, there has been substantial growth in the area of Internet of Things (IoT), representing a network of diverse interconnected devices over the internet. However, IoT networks exhibit vulnerabilities in terms of security, which malicious attackers exploits to compromise these devices. Security threats to IoT encompass various types, including replay attacks, brute force attacks, eavesdropping attacks, and Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks.

DoS/DDoS attacks are recognized as the common threats that significantly affect the security of IoT (Internet of Things) devices. The main goal of such attacks is to incapacitate targeted systems, making them unavailable to authorized users by deploying harmful malware. DoS seeks to disrupt services provided by IoT applications by overwhelming network resources with unnecessary traffic. On the other hand, DDoS occurs when the host server is overwhelmed with an extensive volume of unwarranted requests from geographically scattered zombie devices.

To counteract such threats, Intrusion Detection Systems (IDS) play a crucial role. IDS is a monitoring system designed to identify suspicious activities and generate alerts upon detection. These activities, referred to as intrusions, intend to gain unauthorized access to a computer system. IDS can be categorized into Network-based IDS (NIDS), which connects to one or more network segments, scrutinizing network traffic for malicious activities, and Host-based IDS (HIDS), which is linked to a specific computer device, monitoring malicious activities occurring within the system.

However, there are several researches based on network intrusion detection system that used different approaches such as machine learning algorithms, deep neural networks, ensemble methods and hybrid methods to classify multiple types of DoS/DDoS attacks but majority of the current solutions struggle to identify new forms of these attacks. This is due to some IDS are trained offline which are designed to identify and analyze attacks after they've occurred, offline intrusion detection system lacks the ability to actively prevent or mitigate ongoing attacks in real-time, which is a crucial aspect of modern cybersecurity, they rely heavily on historical data for pattern recognition. If the attackers alter their methods or the system's characteristics change, the IDS might not be as accurate. This means they cannot detect and respond to real-time or ongoing attacks, leaving systems vulnerable until the data is analyzed. Real-time attack detection allows for immediate identification of malicious activities as they occur which can help minimize the impact of an ongoing attack.

Conversely to this extend, this work aims to design an improved hybrid network intrusion detection and classification system using Kth Nearest Neighbor (KNN) algorithm and dense neural networks, which will detect DoS/DDoS attacks in real time and classify them.

3. PROPOSED METHODOLOGY

The first phase of proposed approach involves obtaining network traffic data. After acquiring the data, it is preprocessed to obtain the refined data. The final step involves training and testing the hybrid model using the preprocessed data to assess its performance in detecting patterns of DoS and DDoS attacks. Detailed explanations of each of these steps are provided in the subsequent subsections.

A. Data Acquisition

The initial step of the proposed methodology involves data acquisition, aiming to gather both regular and attack-related network traffic. Generating a substantial amount of normal and attack traffic in real-time necessitates significant network resources and diverse captures of network activity, making it a demanding task. Setting up a large-scale network also consumes considerable time and financial resources. However, to bypass this exhaustive procedure is by leveraging publicly accessible network traffic datasets. To guarantee the dataset's quality, our analysis concentrated on specific criteria:

- The dataset should incorporate real-time network traffic.
- It should be extensive and flexible.
- It must include recent occurrences of Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks.
- The dataset should cover various attack methodologies.

Based on these criteria, for this particular study, we opted for the MQTT-IoT-IDS2020 dataset. This dataset stands out due to its larger sample size compared to other network traffic datasets.

B. Pre-processing of the Dataset

This section discusses the techniques applied to ready the dataset for machine learning tasks, involving actions like data cleansing, normalization, and addressing class imbalances. Preparing data is pivotal to ensure machine learning models are prepared for use. Initially, the process involves merging separate files into a cohesive flow network at the Pandas script level. Pandas, a Python library aimed at dataset analysis and manipulation, is utilized for this task. Importing the data file into a Pandas data frame generates a CSV file that encompasses both binary and multi-class label attributes.

Data Cleansing involves removing duplicates by using the drop_duplicates() method in pandas to remove duplicate rows from the dataset. And also handling missing values by using the dropna() to remove rows with missing values. Data Normalization also involves the use StandardScaler from sklearn.preprocessing library to scale features so that they have a mean of and a standard deviation of 1.

Data cleansing and normalization have a significant impact on the overall performance of the model. Data cleansing removes errors and inconsistencies, improving the accuracy. Normalization scales and standardizes features, reducing overfitting and improving generalization. Cleaned and normalized data can help algorithms converge faster during training, leading to quicker development of the model.

To counter the class imbalance, Synthetic Minority Oversampling Technique (SMOTE) is used. This approach tackles scarcity of certain class instances by generating synthetic data points. It does so by creating new instances within the minority class, interpolating between existing ones, as described by (Munshi, 2022). The choice of SMOTE parameters can significantly impact the performance of the final model. Two key parameters that should be carefully considered are the sampling strategy and the k_neighbors parameter. The sampling strategy parameter determines how much synthetic data is generated for the minority class. Setting this value too high may lead to overfitting and poor generalization on unseen data, while setting it too low may not effectively address the imbalance issue. On the other hand, the k_neighbors parameter specifies the number of nearest neighbors which was 5 in our case, used to generate synthetic samples. Choosing a smaller value may result in noisy and less diverse synthetic samples, whereas a larger value might introduce more noise by creating unrealistic samples.

Evaluating the model's effectiveness involves dividing the dataset into training and testing sets, preserving a 70:30 ratio.

C. Flowchart of the proposed algorithm

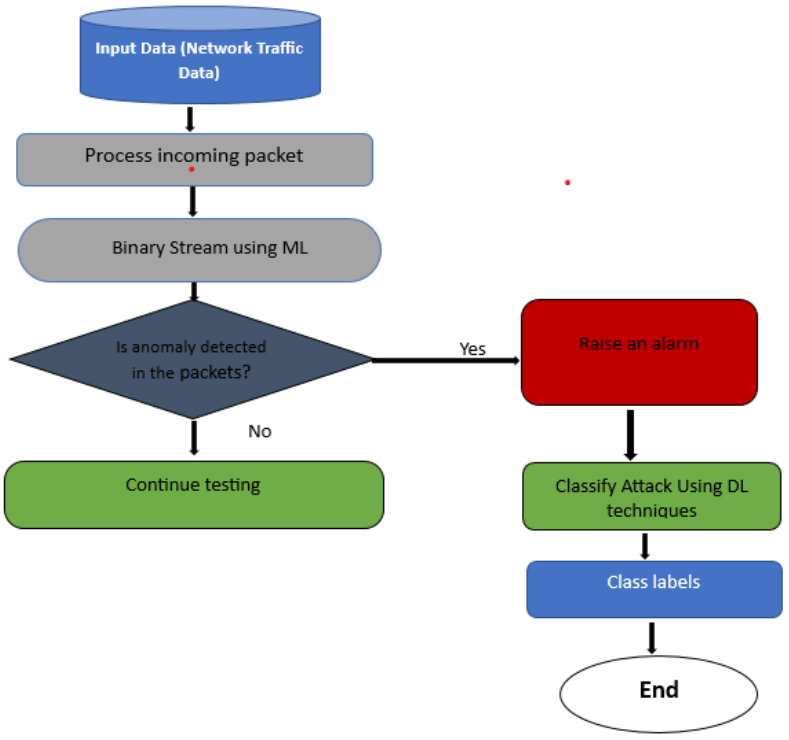

Fig.1. Flowchart of the proposed intrusion detection and classification algorithm

D. Realtime Attack Detection Module

In order to analyze the traffic patterns, similar to the approach outlined in [9] study, this research examines the training dataset D = {D1, . . . , Dn}. Standardizing each data dimension is carried out using either mean and standard deviation or upper and lower bounds to address variations across different dimensions. For each point P in the dataset D, the algorithm identifies the k-nearest neighbors (KNN) among the other training points, specifically the distance to the kth nearest neighbor s(Di), where i = 1, . . . , n. Training concludes by selecting the (1 − α)th percentile sk(P(D)) of {sk(D1), . . . , sk(Dn)}, where P = round[(1 − α)N], to detect statistical deviation at a significance level α, e.g., 0.05.

During the test phase, as observations arrive sequentially, the proposed Intrusion Detection System (IDS) computes the KNN distance sk(Dt) concerning the training points in D. The instantaneous attack evidence Δt is calculated as follows:

$$\Delta t = d[log\ sk\ (Dt) - log\ sk\ (D(P))] \quad (1)$$

where d is the number of data dimensions. This specific form of Δt ensures its asymptotic optimality in the mini-max sense. The specific form of Δt in Equation (1) is designed to ensure asymptotic optimality in the mini-max sense. This means that, as the number of observations increases and under certain assumptions, the algorithm's performance will approach the best possible performance regardless of the attack strategy used. The logarithmic transformation in Equation (1) helps stabilize and normalize the KNN distance sk(Dt). By taking the difference between log(sk(Dt)) and log(sk(D(P))), where D(P) represents a point in the training dataset D, Δt captures the deviation of sk(Dt) from its expected value based on historical data. There might be other formulations that could potentially improve detection accuracy, depending on specific requirements or characteristics of the dataset. However, it's important to note that this specific formulation has been chosen for its asymptotic optimality properties and is likely to perform well under a wide range of scenarios. The decision statistic Mt is given by:

$$Mt = max\{Mt - 1 + \Delta t, 0\}, M0 = 0 \quad (2)$$

Equation (2) triggers an alarm the first time Mt exceeds a threshold at a time T, given by:

$$T = min\{t: Mt \geq hd\} \quad (3)$$

The threshold hd in Equation (3) plays a crucial role in balancing detection delay and false alarm rate. The selection of hd depends on system requirements and trade-offs between these two factors.

Determining an appropriate threshold hd typically involves empirical evaluation or optimization techniques using historical data or simulated attack scenarios. One common approach is to use performance metrics such as receiver operating characteristic (ROC) curves or area under ROC curves (AUC), which measure both detection delay and false alarm rate across different values of hd. Varying hd can have implications for system performance:

- A higher threshold hd will reduce false alarms but might increase average detection delay because attacks need to cross a higher threshold before triggering an alarm.
- Conversely, a lower threshold hd will decrease average detection delay but might lead to more false alarms due to increased sensitivity.
- Finding an optimal balance depends on factors such as cost implications associated with false alarms versus missed detections, desired response time for attacks, available computational resources, etc.

Similarly, the choices of k (number of nearest neighbors) and α (significance level) determine a trade-off between robustness to nominal outliers (i.e., noise) and sensitivity to attacks. Smaller k and larger α enhance sensitivity to attacks, causing a quicker rise in Mt and faster detection, but they are less robust to statistical outliers, making them more prone to false alarms. Notably, the significance level α plays a peripheral role in the proposed IDS, not influencing attack decisions for each observation, unlike anomaly detection methods based on significance tests. α is initially set to a typical value such as 0.05, and then hd is chosen to meet a false alarm rate. The instantaneous attack evidence computed at each time t using (1) accumulates over time with (2) to make an attack decision. Therefore, determining hd requires careful consideration based on system-specific requirements and trade-offs between detection delays and false alarm rates.

The offline synchronization ensures that even during periods of connectivity loss or system downtime, data regarding potential threats is captured and stored. This resilience helps in maintaining a continuous analysis capability, reducing vulnerability to gaps in monitoring. By combining real-time detection with offline analysis, the system can refine its understanding of attack patterns. In essence, the combination of real-time detection and offline synchronization creates a more robust and adaptable security system, capable of continuous monitoring and learning, thereby fortifying the overall defense against potential threats.

Proposed Packet processing algorithm and Real-Time Attack Detection Algorithm

Packet processing algorithm

```
Input: a dataset X = [0...n-1] of packets
Output: an array of packets without duplicates
hashTable = new HashTable()
assign threshold for packet count = N
 for each packet in X do
   hashValue = hash(packet)
   reverseHashValue = hash(reverse(packet))
    if not (hashTable.contains(hashValue) or hashTable.contains(reverseHashValue)) then
      hashTable.insert(hashValue, packet)
      InitializeWeightsAndBiases()
      ComputeLabel(packet)
    end if
  end for
results = new array()
  for each entry in hashTable do
   results.append(entry.value)
   Backpropagate()
  end for
```

Real-Time Attack Detection Algorithm

Input:

- Training dataset X = {x1, x2, ..., xm}
- Number of layers L
- Real-time input data: x_realtime

Initialize: s = 0, t = 0

for n = 1 to N do

Partition training set into Dn M1 and DnM2

Determine Ln(α)

While Mt < hd do

t = t + 1

Get new data { Dnt } and compute{ Dnt }

Mnt = max { Mnt + Dnt , 0 }

Mt = ∑Nn=1 Mnt

if Mt >= hd then

DeclareAttack(T = t)

end if

end while

Offline Attack Classification Algorithm

for l ∈ [1, L] do:

Initialize parameters:

- Weight matrix: Wl = 0
- Bias vector: bl = 0
- Weight matrix for decoding: Wl = 0
- Bias vector for decoding: bl = 0

Encoding layer:

- Calculate encoding or hidden representation using equation (1):

hn = fθ(xn) = σ(Wxn + b)

hl = S (Wlxl1 + bl)

Decoding layer:

while not loss == stopping criteria do:

- Compute yi using equation (2):

yi = gθ(hn) = σ(Whn + b)

- Compute the loss function: binary cross-entropy
- Update layer parameters θ = {W, b}

end while

end for

Classifier:

- Dense neural network with Soft-max activation at the output layer

- Initialize (Wl+1, bl+1) at the supervised layer

- Calculate the labels for each sample xn of the training dataset X

- Apply batch normalization and dropout for speeding up the calculation

- Perform back-propagation in a supervised manner to tune parameters of all layers, loss function categorical cross-entropy

- Output the predicted attack class and detection status (e.g., normal or attack).

Output:

- Class labels for training data

- Attack class and detection for real-time input data

4. EXPERIMENTAL RESULTS

i. Training and validation accuracy

Similar to many other works conducted by using machine learning techniques for the same dataset, both binary and multiclass classification are used in this study. As stated above, binary classification is used in the detection model for predicting two outcomes. In this case, it is expected to predict whether the flow belongs to "Attack" or "Benign" class.

ii. Confusion Matrix for Binary Classification

Figure 2. shows the confusion matrix for binary classification. In binary classification, label 0 is considered as a malicious attack and label 1 as legitimate packet. Figure 2 shows the confusion matrix of the model obtained from the 3000 packets in the dataset, 50 attack flows are incorrectly reported as legitimate flows and 110 legitimate flows are falsely announced as a malicious attack, however, 1460 attack packets and 138 legitimate packets are correctly classified.

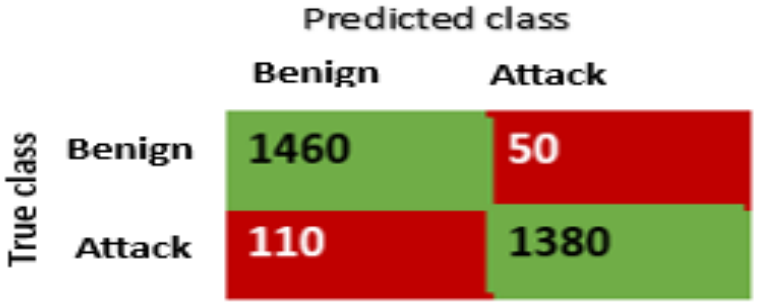

Fig.2. Confusion matrix of binary classified data

The confusion matrix in Figure 2., we can analyze the misclassifications as follows:

1. False Negatives (Misclassified Malicious Attacks): Number of misclassified attack flows is 50, these are instances where actual malicious attacks were incorrectly classified as legitimate flows. Characteristics contributing to these errors could include:

- Similar features or patterns between some attack flows and legitimate flows, making it difficult for the model to distinguish them.
- Unusual or stealthy attack techniques that deviate from typical patterns, causing them to be misinterpreted as legitimate.

2. False Positives (Misclassified Legitimate Packets): The number of falsely announced legitimate flows is 110, these are instances where actual legitimate packets were incorrectly classified as malicious attacks. Characteristics contributing to these errors could include:

- Anomalies in normal network traffic that resemble patterns seen in some types of attacks, confusing the model's classification process.

- Incomplete understanding of the differences between benign traffic and certain types of low-level attacks by the model.

To improve classification accuracy and reduce misclassifications, feature engineering can focus on extracting more discriminative features that differentiate between different classes more effectively and also by experimenting with alternative machine learning algorithms/approaches might provide better results for this particular dataset.

iii. ROC graph

ROC: ROC, or Receiver Operating Characteristics, is a curve that plots the true positive rate against the false positive rate of a model. The area under the curve (AUC) serves as a metric for assessing the performance of a classification model. AUC offers a comprehensive measure of performance across all possible classification thresholds. A threshold is a point along the graph line. ROC curves enable the comparison of multiple models. Models with curves closer to the top-left corner (having higher true positive rates and lower false positive rates) are generally considered better.

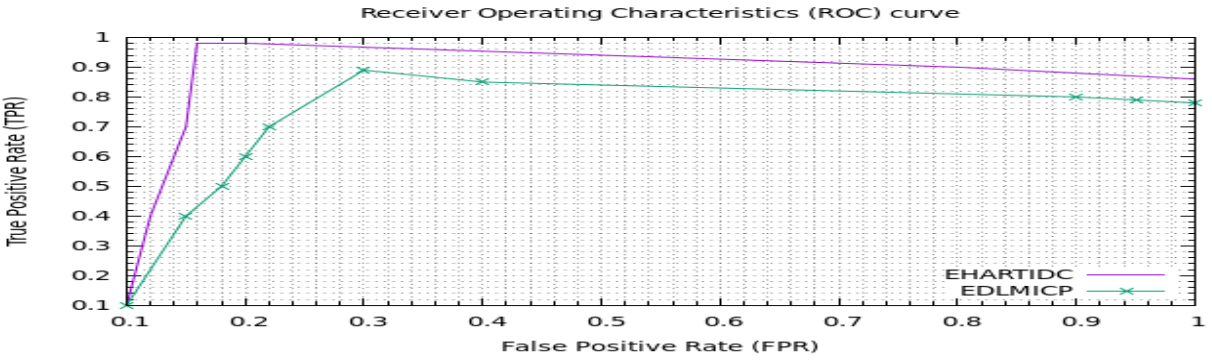

Fig.3. ROC curve

In figure 3 we can see that the proposed, did better than its counterpart, Efficient Deep Learning Model for Intrusion Classification and Prediction (EDLMICP) algorithm by covering more area therefore having better AUC since the more AUC, the better performance. It is interesting to note that our proposed EHARTIDC in comparison with the EDLMICP algorithm, the True positive rate values are mostly above 90%. The maximum TPR value obtained is 96.7%, this proves that the false positives and false negatives are minimized.

Comparing True Positive Rate (TPR) and False Positive Rate (FPR) across different subsets of the dataset, like various types of DoS/DDoS attacks such as bruteforce, flood, malformed, slowite, reveals how well the model detects specific attack categories. TPR, indicating the correct identification of attacks, may vary based on attack characteristics. Some attacks may be easier to detect, resulting in higher TPR, while others, especially stealthy ones, may lead to lower TPR. Conversely, FPR, representing benign traffic misclassified as attacks, may also fluctuate based on the similarity between benign traffic and attack patterns within each subset.

iv. Accuracy

The binary training is performed across 10 epochs set which is evaluated as the most optimum count for accuracy. Figure 4 below shows the accuracy of training and validation throughout the epochs.

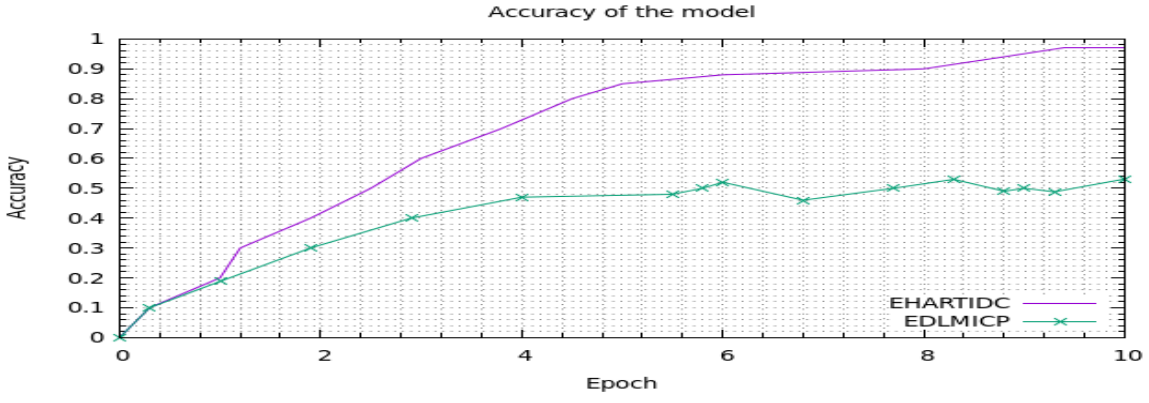

Fig.4. Accuracy of the model

For Binary Classification:

Accuracy = x 100 = = 95%

TPR = = =

TNR = =

FPR = =

FNR = =

Recall = = = 0.976

The overall precision surpasses that of analogous experiments carried out by [11], wherein the outcomes suggest that the suggested MQTT features exhibited effective differentiation between normal and attack traffic, leading to elevated detection rates and minimal false positives. Nevertheless, the MLP classifier attained only an 84% classification accuracy for the seven-class dataset when utilizing all the features.

5. CONCLUSIONS

The proposed model for real-time intrusion detection in IoT networks offers adaptability to changing network conditions and emerging attack types. It achieves this through real-time monitoring and analysis of network traffic patterns, dynamic threshold adjustment mechanisms, offline analysis and learning capabilities, a hybrid approach combining KNN algorithm and dense neural networks, and a feedback loop for continuous learning from detected attacks. These features collectively enable the model to promptly detect anomalies, adjust to evolving threats, and continuously improve its detection accuracy in dynamic IoT environments.

This research mainly assessed how well the proposed model performed when used to detect intrusion detection in IOT networks. The study centered on the widely adopted hybrid method, which has proven its capability in effectively addressing binary issues within the MQTT-IoT-IDS2020 dataset. The selected models specifically tackled binary challenges, exhibiting significant effectiveness in distinguishing between 'benign' and 'attack' instances, including various attack types such as MQTT_BF, Bruteforce, Dos, Flood, Malformed, Slowrite, and Sparta.

The strategies employed in the study demonstrated effectiveness in handling class imbalance present in the MQTT-IoT-IDS2020 dataset. The utilization of SMOTE for synthetic sample generation and the adjustment of class weights played a crucial role in achieving this balance, ultimately enhancing the model's performance. The study results underscored the importance of finely tuning hyperparameters to optimize model effectiveness. In dealing with a vast hyperparameter space, hyperparameter tuning proved to be a highly beneficial approach. Additionally, the detailed insights provided by the confusion matrices aided in understanding the areas where the models excelled or required improvement.

The proposed binary classification model showcased impressive outcomes, with AUC scores reaching up to 95.7, indicating an excellent ability to distinguish between classes. Furthermore, it demonstrated commendable accuracy, precision, and recall scores across both 'benign' and 'attack' categories, suggesting a balanced and efficient performance.

The proposed model for real-time intrusion detection in IoT networks requires computational resources for tasks such as KNN algorithm inference, dense neural network processing, and real-time monitoring of network traffic. In resource-constrained IoT environments, optimizations such as model trimming and distributed processing may be necessary to meet these requirements. Careful consideration of trade-offs between model complexity, detection accuracy, and resource constraints is essential for practical deployment in large-scale IoT networks.

While the study yielded largely positive results, it also highlighted areas that could benefit from further enhancement. Future endeavors should explore more sophisticated oversampling techniques or alternative methods for addressing class imbalance. Moreover, given the model's success in predicting certain attack types, a more in-depth investigation into the significance of these classifications could provide valuable insights

Funding

The author's paper makes it clear that the research was conducted without any financial assistance from institutions or sponsors.

Conflicts Of Interest

The author's paper explicitly states that there are no conflicts of interest to be disclosed.

Acknowledgements

The author expresses gratitude to the institution for the opportunities provided to present and share preliminary findings of this research.

References

[1] R. K. Mohammad Shurman, "DoS and DDoS Attack Detection Using Deep Learning and IDS," Sensor and Actuator Networks, vol. 2023, p. 7, 2020. Jordan: Jordan University of Science and Technology, Network Engineering and Security Department, Jordan. [Online]. Available: mdpi.com.

[2] M. Roopak, "Intrusion Detection System for IoT Networks for Detection of DDoS Attacks," 2021.

[3] B. Zarpelão, "A Survey of Intrusion Detection in Internet of Things," Journal of Network and Computer Applications, vol. 47, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2017.02.009.

[4] Z. W. Weibo Liua, "A survey of deep neural network architectures and their applications," Neurocomputing, vol. 234, pp. 11–26, 2017.

[5] Rezvy, "An efficient deep learning model for intrusion classification and prediction in 5G and IoT networks," in IEEE Conference Proceedings, 2019, pp. 6. DOI: 978-1-7281-1151-3/19/$31.00 ©2019 IEEE.

[6] Smys, "Hybrid Intrusion Detection System for Internet of Things (IoT)," Journal of ISMAC, vol. 2, no. 4, 2020.

[7] R. Kumar, "UIDS: a unified intrusion detection system for IoT environment," Evolutionary Intelligence, 2019. [Online]. Available: https://doi.org/10.1007/s12065-019-00291.

[8] S. Doshi, "Timely Detection and Mitigation of Stealthy DDoS Attacks via IoT Networks," arXiv, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2006.08064.

[9] Pathak, "Study on Decision Tree and KNN Algorithm for Intrusion Detection System," International Journal of Engineering Research and Technology (IJERT), vol. 9, no. 7, 2020.

[10] Xiao, "IoT Security Techniques Based on Machine Learning," Signal Processing and the Internet of Things, vol. 9, 2018.

[11] Z. B. Naeem Firdous Syed, "Denial of service attack detection through machine learning for the IoT," Journal of Information and Telecommunication, vol. 23, 2020. [Online]. Available: https://doi.org/10.1080/24751839.2020.1767484.