JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 06 Issue 02
2024

Research Article
https://doi.org/10.56979/602/2024

Securing Cloud Environments: A Convolutional Neural Network (CNN) approach to Intrusion Detection System

Syed Younus Ali[1,2], Umer Farooq[2], Leena Anum[3], Natash Ali Mian[4], Muhammad Asim[5,6], and Tahir Alyas[2*]

[1]Department of Computer Science, Minhaj University, Lahore, 54000, Pakistan.
[2]Department of Computer Science, Lahore Garrison University, Lahore, 54000, Pakistan.
[3]Department of Management Sciences, Lahore Garrison University, Lahore, 54000, Pakistan.
[4]SCIT, Beaconhouse National University (BNU),Lahore, 54000, Pakistan.
[5]Department of Computer Science, National College of Business Administration & Economics Sub Campus Multan, 60000, Pakistan.
[6]Khawaja Fareed University of Engineering and Information Tehcnology, Rahim Yar Khan, 64200, Pakitan.
[*]Corresponding Author: Tahir Alyas. Email: tahiralyas@lgu.edu.pk

Received: January 19, 2024 **Accepted:** February 19, 2024 **Published:** March 01, 2024

Abstract: Cloud-computing has become an essential portion of recent IT structure, contribution scalable resources and on-demand services to users. However, the increasing reliance on cloud environments has raised concerns about security, especially with the rise of sophisticated cyber threats. Intrusion detection systems (IDS) play a crucial role in detecting and mitigating possible security breaches. In this studies proposes a approach to enhance intrusion detection in cloud computing through the CNN. This deep learning architecture adapted for the unique challenges in cloud computing security. Unlike traditional IDS methods that rely on rule-based or signature-based approaches, the CNN-based intrusion detection system presented in this research leverages the network's capability to automatically learn hierarchical features from raw data. This study is involves the collection of diverse and representative datasets from cloud environments, including normal network traffic and various types of attacks. The CNN is trained on these datasets to learn the inherent patterns of legitimate activities and deviations indicative of potential intrusions. The proposed system demonstrates its adaptability to evolving threats by continuously updating its knowledge through regular retraining with new data. The evaluation of the CNN-based intrusion detection system is conducted through comprehensive experiments, comparing its performance against traditional methods. The results indicate that the CNN-based approach outperforms conventional IDS techniques, demonstrating its potential as a robust and efficient solution for intrusion detection in cloud computing environments.

Keywords: CNN; Intrusion detection system; Intrusion; Machine learning; Cloud Computing.

1. Introduction

Cloud computing is inexpensive and offers a pay per use model, it is an emerging technology that is being attracts all stack holders. Cloud computing has revolutionized the way businesses and organizations manage their IT infrastructure and services. However, with the increased reliance on cloud environments, the need for secure environments, like instruction-free environments, has become paramount. IDS plays a crucial role in monitoring and analyzing network traffic, detecting potential threats, and promptly responding to security incidents within cloud infrastructures. By leveraging cloud-based IDS solutions, various types of cyber threats, including unauthorized access attempts, malware infections, and malicious activities. Cloud-based IDS systems offer scalability, flexibility, and real-time monitoring capabilities, making them well-suited for dynamic and distributed cloud environments. Furthermore, integrating Intrusion Detection Systems with advanced technologies like machine learning and artificial intelligence, such as

Convolutional Neural Networks (CNNs), can significantly improve threat detection accuracy and reduce false positives. CNNs excel in analyzing complex patterns and anomalies in network traffic, enabling more effective threat detection and response in cloud computing environments. [1].

One of the primary tools in the toolbox of security requirements for protecting computer systems and networks from malicious activity is the intrusion detection system (IDS). It is a piece of hardware or software that keeps an eye on host or system activity and network traffic to spot any malicious or policy-violating behavior within the system. They also report the system administrator with warning sirens and other information if such behaviour is found. But these warning notifications from an intrusion detection system (IDS) can be a false alarm and unrelated to the real intrusion that's affecting the system's functionality. System administrators resolve attacks by altering the affected system's content, configuration, or security environment [2].

The idea of computer security threat monitoring gave rise to intrusion detection systems. IDS is a defensive and proactive computer monitoring and defense solution that guards vital IT infrastructure from unauthorized activity. The IDS can be divided into the following three types: signature-based intrusion detection systems (IDSs), which match network traffic to a database of recognized signatures in order to identify patterns of malicious attempts; anomaly-based intrusion detection systems, which identify anomalous network patterns that cannot be connected to recognized signatures; for example, they raise an alert when excessive power is utilized suspiciously; Using its log file and application, behavior-based intrusion detection system (IDS) can identify abnormal activity. [3].

By closely examining each packet at the network and transport layers, the network intrusion detection system (NIDS) keeps an eye on and evaluates network traffic. It actively searches for any potentially troubling activity or network-based assaults, including port scans and Denial of Service (DoS) operations. When anomalous activity is detected in the network traffic, notifications may be sent to the system administrator immediately. Commonly used commercial intrusion detection systems (IDSs), such Snort, Tcpdump, and Natural Flight Recorder, are well known for their efficiency in medium-sized networks and their simplicity of use. They responded by putting out a creative strategy to deal with these difficulties. This method, which examines network data passing via virtual computers, has shown to significantly improve attack detection while successfully lessening the effects of assaults.

IDS keeps an eye on specific hosts or devices connected to a network by analysing events that happen on them and looking for changes in their activity. It examines all facets of a host's activity, including system calls, file-system updates, incoming and outgoing packets, and application logs. The system sends a warning to the administrator informing them to take precautions against possible harmful attacks if any unusual behaviour is found. HIDS is preferred over Network-based Intrusion Detection Systems (NIDS) in many industries. A particular HIDS model that detects intrusions by matching predetermined patterns with the system logs was created based on the study of Microsoft Windows XP log files [5]. Hybrid Intrusion Detection System (HIDS), alternatively referred to as Distributed IDS (DIDS), incorporates multiple detection methods or systems, such as Network-based IDS (NIDS) and Host-based IDS (HIDS). This system is specifically designed for deployment across extensive distributed networks like cloud computing, facilitating seamless communication among all entities. The interconnected hosts within the network gather system information and transform it into a standardized format before transmitting it to a central server or network monitor, as elucidated. This configuration ensures efficient data exchange and centralized monitoring across the network. [6].

The hypervisor that builds and manages virtual machines, which are abstractions that are accessible over a cloud network. It is also crucial to the operation management of every virtual machine instance. Virtual machines are primary targets for potential intruders, necessitating their protection. Therefore, hypervisor-based IDSs are established over the virtual network to secure both the virtual machines and the hypervisor. They offer a level of abstraction between the VM and host. Key components of IDS are positioned hypervisor to detect anomalous user activities through metadata analysis on the network. These components monitor and analyze communication between VMs and hypervisors[7].

2. Literature review

This section summarizes the information analyzed and discussed about cloud-based IDS. The comparison is structured thematically, based on the approaches used for IDS.. Different approaches have been

utilized, reflecting the diverse research in this area. Soft computing, or computational intelligence, is applied for solving challenging problems, including NP-complete problems. Intrusion detection is an NP-complete problem, challenging for humans with bounded rationality to make feasible. This study examines three main categories of soft computing: fuzzy logic, artificial neural networks, and machine learning. Tables 1 and 2 present a comprehensive study of the cloud-based IDS proposed along these three attributes.

Table 1. Summary of IDS and alogorithm

References	Algorithm Preferred	Description
Kozik et al., 2018	Extreme Learning Machine	The anomalies, such as DoS, have been detected by integrating traffic classification into edge devices with the machine learning solution.
Idhammad et al., 2018 [13]	Random Forest and Naive Bayes Classifiers	It offers a suggested methodology that makes use of machine learning classifiers to identify anomaly-based intrusions in five modules. It has compared its findings with those of other approaches using the CIDDS-001 dataset.
Gill, & Buyya, 2018 [14]	SVM	The secure resource management component of the cloud was the primary emphasis of this paper. It has suggested a self-defense strategy against assaults.
Aljawarneh et al., 2018 [15]	ML based hybrid classifiers	The complexity of computation and time has been greatly decreased.
Alzahrani, & Hong, 2018 [16]	Back Propagation Neural Networks	A hybrid model has been proposed to detect intrusions based on anomalies as well as signatures. It has detected DDoS attacks by utilizing the idea of an artificial neural network technique.
Modi et al., 2016 [17]	NN for anomaly-based detection	This study uses a hybrid network intrusion detection system via cloud to identify threats, such DDoS and DoS, etc. It has produced results in a remarkably short amount of detection time and increased overall detection rate.
Alfy, & Al-Obeidat, 2014 [18]	Fuzzy Classification method	On the basis of irregularities in the flow, this paper has identified intrusions. It used a greedy selection strategy to handle a variety of qualities before detecting attacks.
Ziong et al., 2018 [19]	SSN approach	This research has examined cloud network traffic according to its dynamic properties. It has, however, not identified any particular kind of attack and has instead regarded all attacks as anomalies.

Table 2. Summary of Attacks detected, Dataset used and Algorithm Preferred

Year	Attacks	Dataset	Algorithm	Ref. No
2021	generic, Brute-force, Analysis, backdoor, sql injection, DDoS	CICIDS 2017 dataset, CICIDS 2019 UNSW-NB15 dataset,	Classifier for Decision Jungles	[21]
2020	U2R, DoS, probe, R2L,	Dataset NSL-KDD	ANN hybridizes with FCM.	[22]

2020	DoS, U2R, R2L, probe	Dataset NSL-KDD	SVM and FCM are hybridized.	[23]
2019	Different Attacks	36 datasets of actual assaults gathered from the network traffic blog between 2014 and 2016 simplistic Bayes classifier	naïve Bayes classifier	[24]
2019	DoS	Dataset generated	The rule-based algorithm, scoring system, and ranking algorithm are used to classify assaults.	[25]
2019	U2R, R2L, DoS, probe	dataset NSL-KDD	PSO has been modified so that the particles' updated location is produced by appending the previous location to the new location. This lengthens the period of investigation. PSO is used with CS.	[26]
2019	Various attacks	generated Dataset	The authors employ VirtualBox, and Ubuntu 15 is used to generate datasets in conjunction with the Apache web server.	[27]
2019	U2R, DoS, probe, R2L	dataset NSl-KDD	Logistic regression	[28]
2019	backdoor, Analysis, generic, sql injection, DDoS, Brute-force	dataset CICIDS 2017	Bat algorithm	[29]
2019	U2R, DoS, probe, R2L, , sql injection, web attack.	CICIDS 2017 dataset	GA	[30]
2018	DDoS	CICIDS dataset	Time-sliding window algorithm	[31]

CNNs, a class of deep learning algorithms, have shown exceptional performance in areas requiring pattern recognition and feature extraction, such as image and speech recognition. The intricate patterns of network traffic and user behavior that characterize cyber intrusions. Unlike traditional IDS, which rely on predefined rules and signatures, CNN-based systems can learn and evolve, improving their detection capabilities over time [24].

The application of Convolutional Neural Networks in intrusion detection systems offers promising prospects for enhancing cloud security. By leveraging the advanced pattern recognition and feature extraction capabilities of CNNs, it is possible to develop more effective and adaptive security mechanisms for cloud environments. However, addressing the challenges related to data diversity, model scalability, and computational efficiency is crucial for realizing the full potential of CNN-based IDS in securing cloud

infrastructures[25].

3. Materials and Methods

The methodology for securing cloud environments using a Convolutional Neural Network (CNN) approach to Intrusion Detection System (IDS) involves several key steps. Firstly, data collection is essential, where network traffic data is gathered from various sources within the cloud environment. This data typically includes packet headers, payload information, and metadata related to network communications.

Next, data preprocessing is performed to prepare the collected data for analysis. This involves tasks such as data cleaning, normalization, feature extraction, and transformation to ensure consistency and compatibility with the CNN model. Feature selection may also be applied to identify relevant attributes that contribute to intrusion detection accuracy.

Supervised learning techniques are employed to train the CNN model subsequent to the preprocessing of the data. The CNN model is successfully trained to distinguish between normal and anomalous patterns using a labeled dataset that includes both examples of malicious and normal network traffic.

In the training phase, the CNN model picks up complex patterns and features from the network traffic data, which helps it identify abnormalities in the data that could be signs of security breaches or intrusions. Validation datasets measure the model's performance, and hyperparameters are adjusted to maximize detection accuracy and reduce false positives.

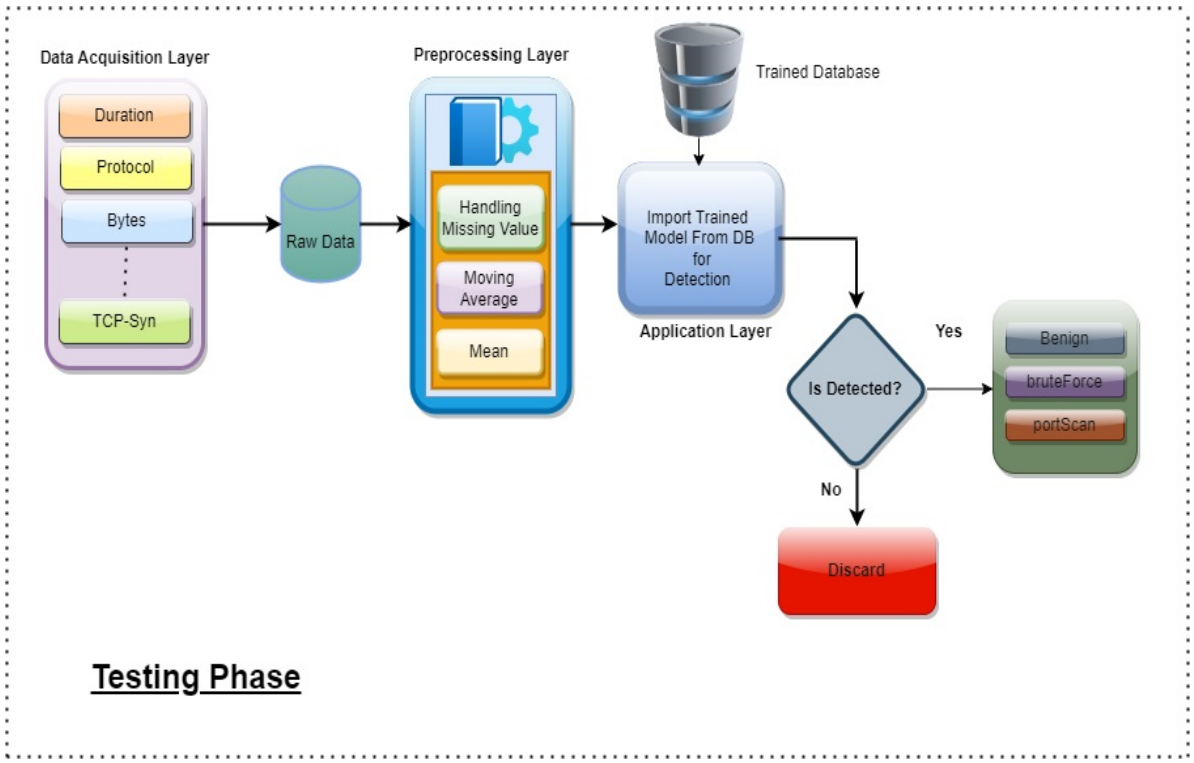

Figure 1. Proposed system diagram

During the testing phase, the system is experienced with a variety of normal and abnormal conditions. This permits the trained model to adapt to different types of behavior and improve its detection capabilities. Data Acquisition Layer In the data acquisition layer, raw data is collected from the target system. This raw data could include protocol bytes, bytes sent over TCP, or any other relevant information. The raw data is then used to detect anomalies in the system's behavior. The data acquisition layer also includes different parameters like duration, protocol, bytes, and TCP-Syn etc data. These metrics provide information about the time duration of an event, the type of protocol being used, the number of bytes transmitted, and the synchronization sequence used in the TCP handshake, respectively. Preprocessing Layer: In the preprocessing layer, raw data is preprocessed to make it suitable for analysis. This could involve handling missing-values, outliers, or noise in the data. The preprocessed data is then used as input for the trained model. Handling Missing Value: One approach to handling missing values in the preprocessing

layer is by using a moving average, mean, or other interpolation technique. These techniques help to fill in missing values and improve the accuracy of the trained model.

One way to handle missing values in the preprocessing layer is by using a moving-average. A moving-average is calculated by taking the average of a fixed number of data points (usually called the "window size") in a specified time period. For example, if there is a missing value in the duration column, a moving average can be calculated by averaging the values in the previous three time periods. By using a moving average to handle missing values, the system can maintain a smoother representation of the system's behavior over time. This can progress the accuracy of the trained model and reduce the likelihood of false positives.

Mean: In addition to a moving average, you can also use the mean of the previous data points to handle missing values. For example, if there is a missing value in the byte's column, the mean can be calculated by averaging the values in the previous three time periods. By using the mean to handle missing values, the system can maintain a more stable representation of the system's behavior over time. This can improve the accuracy of the trained model and reduce the likelihood of false positives. While both methods have their advantages, the choice between using a moving average or the mean to handle missing values will depend on the specific requirements and constraints of the project. Import Trained Model from DB: The trained model is imported from a database. This database could be a remote server or a local file system, depending on the specific implementation. If an anomaly is detected, the system takes appropriate action, such as discarding the data or initiating further investigation.

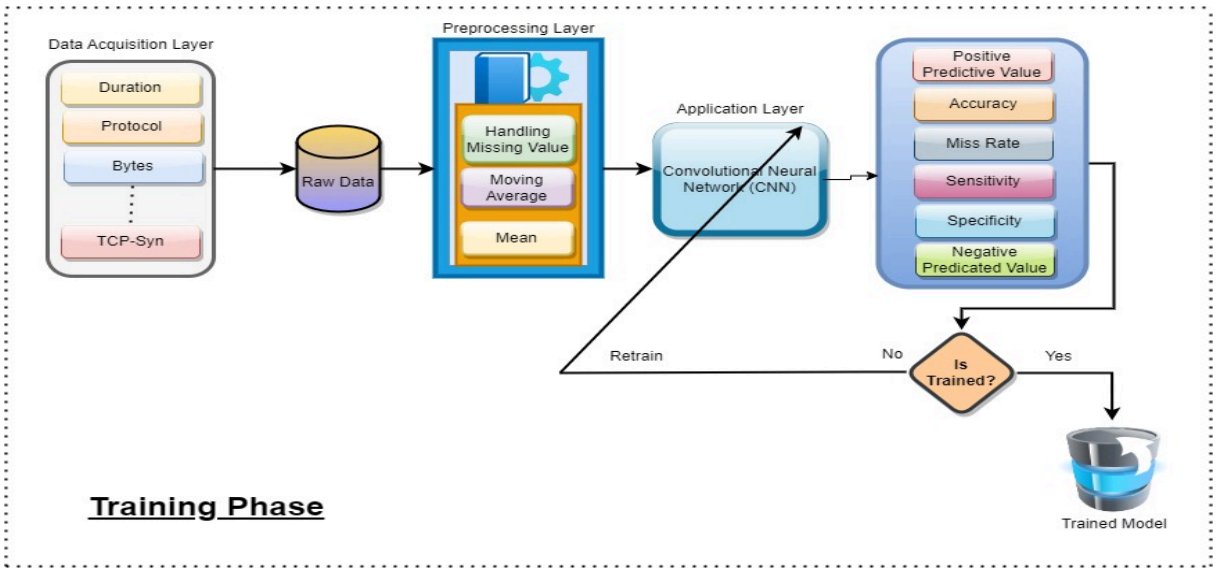

Figure 2. Training Phase

Training phase model provides a high-level overview of a Machine Learning model, which consists of multiple layers, each performing specific tasks. These layers are interconnected, and data flows through them. Here is an explanation of each point in the image:

Data Acquisition Layer: This layer collects raw data from various sources such as sensors, files, or databases. The duration of this layer determines how frequently data is collected.

Protocol: This indicates the type of communication protocol used for data transmission. Examples include TCP, UDP, and HTTP.

Bytes: This represents the size of the data transmitted, usually measured in bytes.

TCP-Syn: This refers to the synchronization (SYN) phase of the TCP protocol.

Raw Data: This is the initial form of the data, as received from the data source.

Training Phase: This is the process of feeding the raw data into the ML model.

Preprocessing Layer: This layer performs various preprocessing tasks on the raw data before it is fed into the model. Examples include handling missing values, scaling, and normalization. Handling Missing Value: This part of the preprocessing layer addresses the issue of missing data in the dataset.

Moving Average: This is a statistical method used to smooth out fluctuations in the data. It is often used as a preprocessing step before training the machine learning model.

Application Layer: This layer predicts new, unseen data using the taught machine learning model. Positive Predictive Value: The effectiveness of the machine learning model is assessed using this measure, also called accuracy. It is the proportion of accurate forecasts to all of the forecasts produced.

Miss Rate: It represents the proportion of misses, or false negatives, to all true negatives in the dataset.

Sensitivity: It is measured as the ratio of real positives in the dataset to the total number of true positives.

Specificity: The machine learning model's performance is assessed using this measure. The proportion of real negatives in the dataset is divided by the number of genuine negatives.

Negative Predicted Value: This metric, known as specificity, is used to evaluate the performance of the machine learning model. It is the ratio of the number of true negatives to the total number of negatives predicted by the model.

4.Results

This graph displays the features and their corresponding importance in a dataset or model, with a focus on network traffic analysis. The line graph represents each feature is importance, with values ranging from 0.20 to 0.35. The importance values indicate the relative significance of each feature in the dataset or model. The letter 'I' on the importance graph represents a breakpoint in the importance scale, making distinguishing features with similar importance easier.

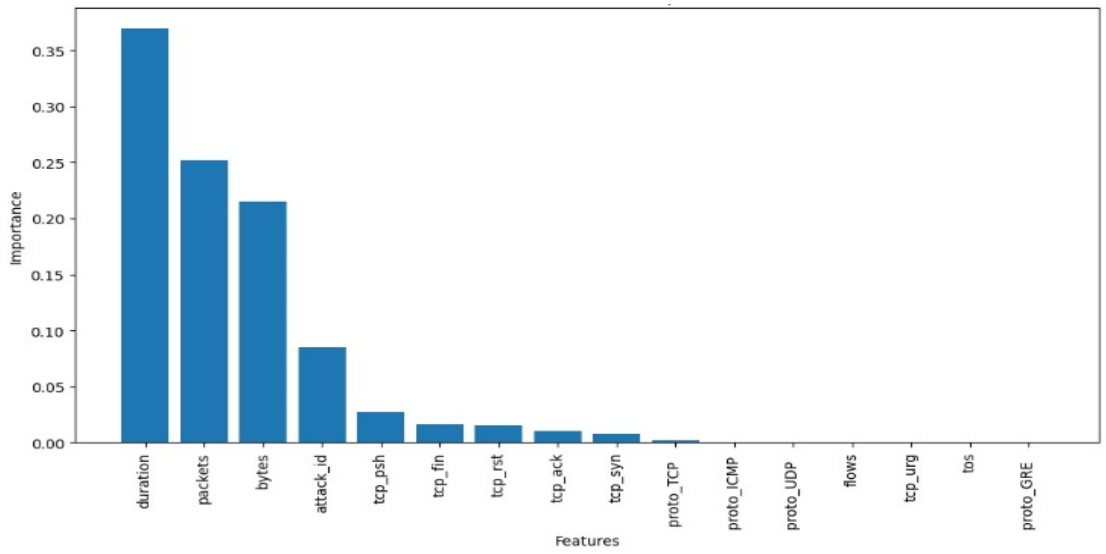

Figure 3. Features and their corresponding importance

Feature importance for a machine learning model, specifically a model used to detect network attacks. Feature importance is a way to rank a dataset's features (or variables) in order of their importance or relevance in making predictions with the model.

In this image, the feature importance is listed in descending order on the right side, with the most important feature at the top and the least important feature at the bottom. The feature names are listed in the first column, and the corresponding importance score is listed in the second column.

For example, the most important feature for this model is "duration", with a score of 0.369176. This means that the "duration" feature is the most useful for the model in making accurate predictions about network attacks. The next most important feature is "packets", with a score of 0.251674, followed by "bytes" with a score of 0.214719.

The importance of features can help us understand which features are most important for the model and how the model makes its predictions. This can be useful for a variety of purposes, such as selecting the most relevant features to include in the model, identifying potential sources of bias or error in the data, and gaining insights into the underlying patterns and relationships in the data.

Table 3. Feature Importance

Packet	Feature	Importance
1	Packets	0.251674
2	Bytes	0.214719
11	Attack_id	0.085197
6	Tcp_psh	0.027211
9	Tcp_fin	0.016209
7	Tcp_rst	0.015391
5	Tcp_ack	0.009944
8	Tcp_syn	0.008031
14	Proto_TCP	0.001692
13	Proto_ICMP	0.000580
15	Proto_UDP	0.000240
3	Flows	0.000000
4	Tcp_urg	0.000000
10	Tos	0.000000
12	Proto_GRE	0.000000

This graph, known as a Precision-Recall (PR) curve, is often used to assess how well binary classifiers work. The classifier has been used in this instance to distinguish between several categories of network data, including "attacker," "normal," "suspicious," "unknown," and "victim."

The proportion of real positive instances—that is, accurately recognized attackers, normal, etc.—out of all actual positive cases in the data is shown by the x-axis of the graph. The accuracy, or the percentage of actual positive instances among all cases the classifier predicted to be positive, is shown on the y-axis.

Plotting the accuracy and recall data for various categorization criteria results in the curve. For instance, the classifier may predict a large number of examples as affirmative at a low threshold, which might lead to a high recall but perhaps poor accuracy. The classifier becomes more conservative as the threshold is raised, which may result in a decrease in recall, fewer false positives, and an improvement in accuracy.

With the exception of the "unknown" class, which has a slightly lower precision of 0.98, we can observe in this graph that the classifier has extremely high recall and accuracy for all classes. This implies that the classifier has a low percentage of false positives and false negatives and is very accurate at distinguishing between the various kinds of network data.

All things considered, the PR curve is a helpful tool for assessing how well classifiers work, especially when the classes are unbalanced or the cost of false positives and false negatives is large. The classifier seems to be working quite well in this instance, with good recall and accuracy across all classes.

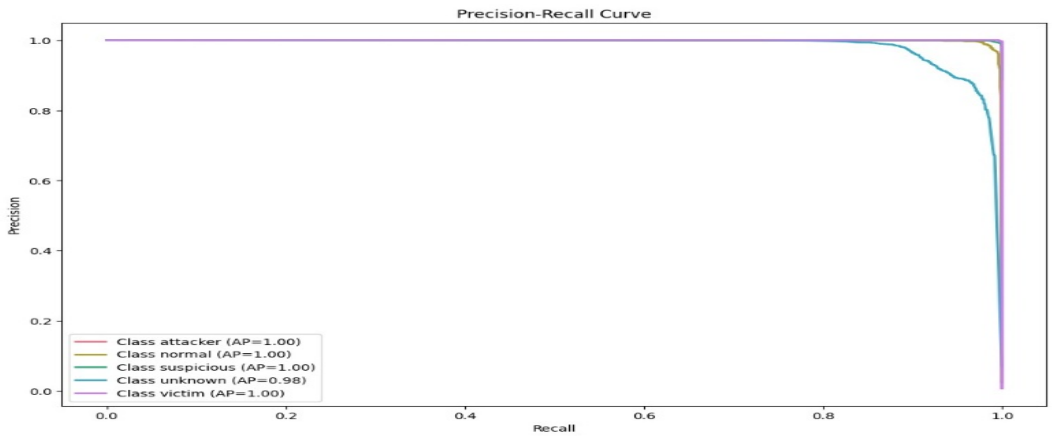

Figure 4. Precision-Recall (PR) curve

Actual values to the predicted values for a binary classification problem with four possible outcomes: "normal", "suspicious", "attacker", and "victim". The graph has two rows, with the actual values in the top row and the predicted values in the bottom row. The first column shows the "normal" class, and we can see that the actual values have some "normal" instances, while the predicted values also correctly predict some "normal" instances.

The second column shows the "attacker" class, and we can see that the actual values have some "attacker" instances, while the predicted values correctly predict some "attacker" instances, but also misclassify some "normal" instances as "attacker". The third column shows the "victim" class, and we can see that the actual values have some "victim" instances, while the predicted values correctly predict some "victim" instances, but also misclassify some "normal" instances as "victim". The fourth column shows the "suspicious" class, and we can see that the actual values have some "suspicious" instances, while the predicted values correctly predict some "suspicious" instances, but also misclassify some "normal", "attacker", and "victim" instances as "suspicious".

From the graph, we can see that the model has some difficulty distinguishing between the "normal", "attacker", and "victim" classes, but it performs better in identifying the "suspicious" class. Overall, the model's accuracy is moderate, but there is still room for improvement.

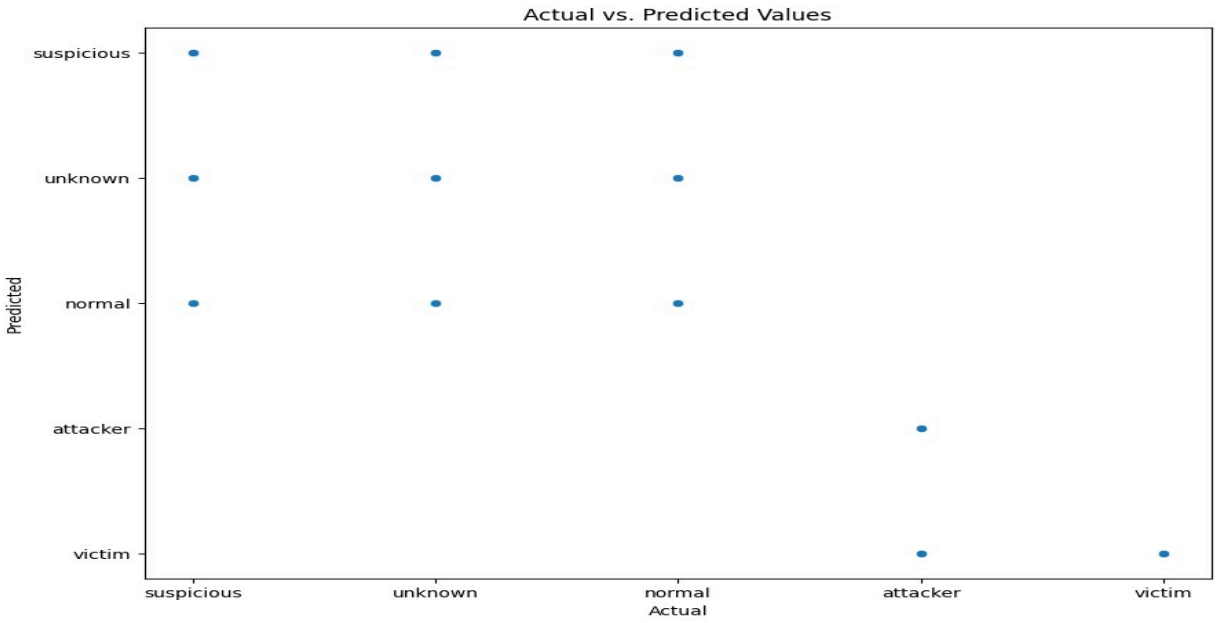

Figure 5. Actual Vs Predicted Values

a correlation matrix and some more relevant data for different network traffic aspects. The correlation coefficient is a statistical metric used to characterise the strength and direction of a link between two variables.

The variables in this correlation matrix are:

Duration: how long a link remains active. The quantity of packets transmitted or received during a connection is called a packet.

Bytes: the total amount of data transmitted and received across a connection

Flows: the quantity of distinct linkages the quantity of TCP packets with the URG flag set (tcp_urg).

TCP packet count with the ACK flag set is tcp_ack.

the quantity of TCP packets with the PSH flag set, or tcp_psh

the quantity of TCP packets with the RST flag set, or tcp_rst.

the quantity of TCP packets with the SYN flag set, or tcp_syn

TCP packet count with the FIN flag set is tcp_fin.

TOS: the IP header's Type of Service field

assault ID: a special identification for a particular kind of assault

Proto_GRE: the quantity of GRE protocol-using packets

Proto_ICMP: the quantity of ICMP-enabled packets

Proto_TCP: the quantity of TCP-enabled packets

Proto_UDP: the quantity of UDP-enabled packets

The correlation coefficients vary from -1.0 to 1.0. The two variables have a high positive association when the correlation coefficient is 1.0, which means that when one variable rises, the other variable also rises. Strong negative correlation is shown by a correlation value of -1.0, which means that as one variable rises, the other falls. There is no association between the two variables when the correlation coefficient is 0.

This indicates that the proportion of packets with these flags set tends to decrease with connection length. Additionally, there are significant negative correlations found between packets suggesting that the proportion of packets with these flags set tends to decline as the number of packets rises. The packets with these flags set tend to drop as flows grow, according to the somewhat negative correlations between flows and tcp. Weak negative correlations have been found between tos and tcp_ack and tcp_fin, suggesting that the amount of packets with these flags set tends to drop significantly as the TOS value rises.

Attack_id and tcp_urg, modest positive correlations, suggesting that the frequency of packets with these flags set tends to rise somewhat as the attack ID value grows. Proto_GRE has weak positive correlations with tcp_syn, and tcp_fin, suggesting that the proportion of packets with these flags set tends to rise somewhat as the number of GRE packets increases.

The number of packets with these flags set tends to increase slightly as the number of ICMP packets increases, according to weak positive correlations between proto_ICMP and tcp_urg, tcp_ack, tcp_psh, tcp_rst, tcp_syn, and tcp_fin. The number of packets with these flags set tends to increase as the number of TCP packets increases, according to the moderate positive correlations between proto_TCP and tcp_urg, tcp_ack, tcp_psh, tcp_rst, tcp_syn, and tcp_fin.

Proto_UDP has weak positive correlations with tcp_urg, tcp_ack, tcp_psh, tcp_rst, tcp_syn, and tcp_fin, suggesting that the proportion of packets with these flags set tends to rise somewhat as the number of UDP packets increases.

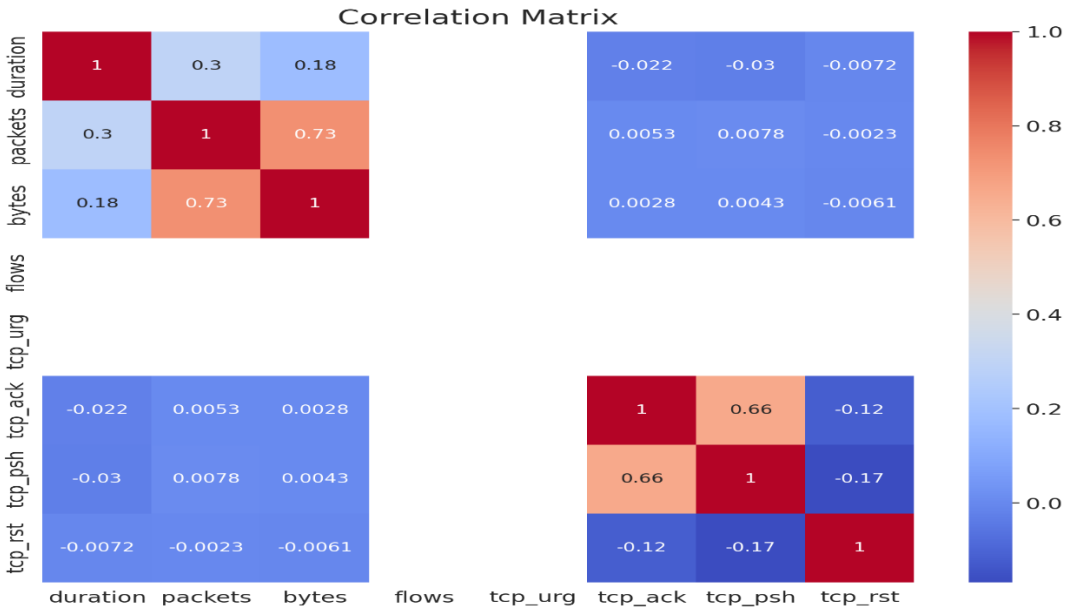

Figure 6. Correlation Matrix

Confusion Matrix is a visual representation of the performance of a classification model. It displays the accuracy of a model in predicting the actual class of an observation, given its predicted class.

The numbers in the provided matrix are as follows:

True Positives : 10

True Negatives : 36,114

False Positives : 222

False Negatives (FN): 19,327

In this case,

TPR = 10 / (10 + 19,327) = 0.000969678 (around 0.1%).

In this case,

TNR = 36,114 / (36,114 + 222) = 0.998342953 (around 99.8%).

False Positive Rate (FPR): It is the ratio of false positives to the sum of true negatives and false positives.

In this case,

FPR = 222 / (36,114 + 222) = 0.000585454 (around 0.06%).

False Negative Rate (FNR): It is also known as Miss Rate. It is the ratio of false negatives to the sum of false negatives and true positives.

In this case,

FNR = 19,327 / (19,327 + 10) = 0.999039679 (around 99.9%).

These metrics help in evaluating the performance of a classification model.

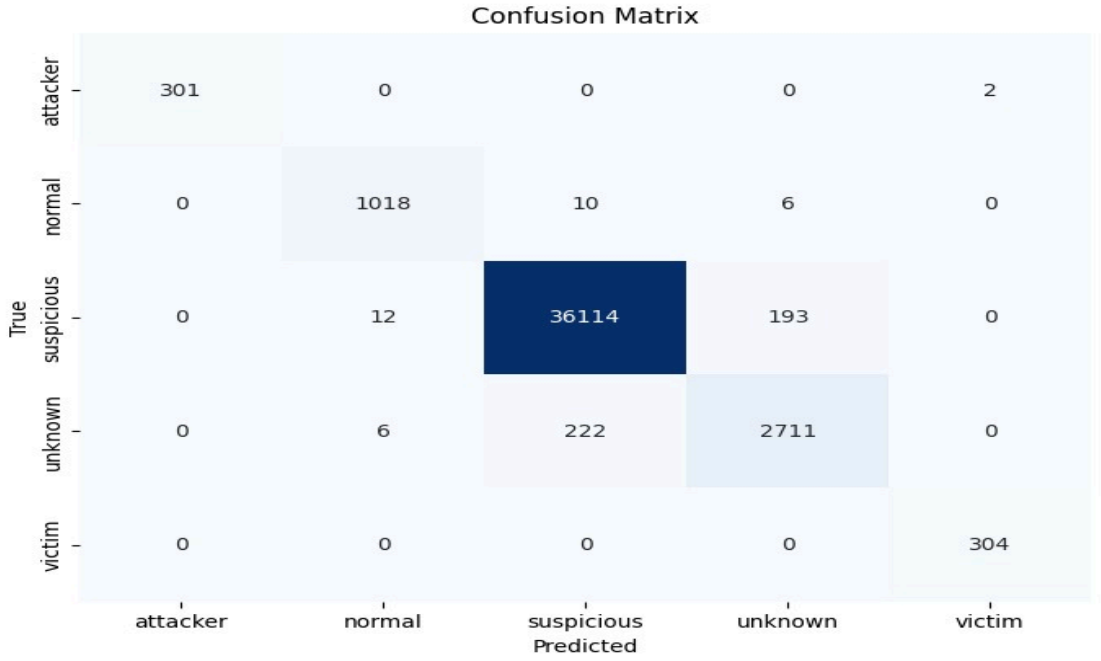

Figure 7. Confusion Matrix

Classification Report of a Machine Learning model. The report is a detailed analysis of the performance of the model, considering different types of activities. The categories in this report are attacker, normal, suspicious, unknown, victim, and other types.

The table in the report includes various metrics:

1. Precision matric
2. Recall (Sensitivity)
3. F1-Score:
4. Support:

The table also shows the Accuracy of the model, which is the ratio of correctly predicted instances to the total instances.

In addition to these metrics, the report provides average values:

1. Macro Average: It calculates the average of the metric values for each class and then takes the average of those.
2. Weighted Average: It calculates the average of the metric values for each class, giving more weight to the classes with higher number of instances.

In this case, the model's performance is very good, with high values for precision, recall, f1-score, and accuracy. The report also indicates that the model can handle various types of activities effectively.

Table 4. Classification deatils

	Precision	Recall	F1-score	Support
Attacker	1.00	0.99	1.00	303

Normal	0.98	0.98	0.98	1034
Suspicious	0.99	0.99	0.99	36319
Unknown	0.93	0.92	0.93	2939
Victim	0.99	1.00	1.00	304
Accuracy			0.99	40899
Macro Avg	0.98	0.98	0.98	40899
Weighted avg	0.99	0.99	0.99	40899

The heatmap shows occurrences of a projected class in each column and instances of an actual class in each row. The number of cases when the model predicted one class but the true label was the actual class is represented by the value at the intersection of the row and column. In this particular instance, the confusion matrix represents a model's performance on a multiclass classification issue. The horizontal axis represents the expected labels, while the vertical axis represents the genuine labels. The number of times the model successfully predicted the class is represented by the diagonal members of the matrix, while the number of times the model incorrectly predicted the off-diagonal components represent the class. Metrics like accuracy, precision, recall, and F1-score may be used to assess the model's performance. The model's accuracy in this instance was 98.90%, meaning that 98.90% of the cases were properly classified by the model.

5. Conclusions

By using CNNs, valuable characteristics may be extracted from network traffic data, which helps the model recognize unusual patterns that might be signs of security vulnerabilities. When compared to conventional techniques, the accuracy of the CNN-based intrusion detection system was higher. CNNs' capacity to automatically derive hierarchical representations from unprocessed data allows for more accurate threat identification, both known and unknown. Because cyber threats are always changing, intrusion detection systems must also be flexible. CNNs showed greater resistance against changing attack techniques because of their capacity to learn and change patterns over time. The CNN model addressed a prevalent issue in intrusion detection systems, which demonstrated a significant decrease in false positives. This enhancement helps lessen the workload on security teams and better uses available resources.

References

1. Iqbal, S., Kiah, L.M., Dhaghighi, B., Hussain, M., Khan, S., Khan, M.K. & Choo, K.R. (2016). On cloud security attacks: A taxonomy and intrusion detection and prevention as a service. International Journal of Network and Computer Applications, 74, 98-120.
2. Wasim Khan, M. H. (2022). An unsupervised deep learning ensemble model for anomaly detection in static attributed social networks. International Journal of Cognitive Computing in Engineering, 153-160.
3. Khan, W. & Haroon, M. (2022). An efficient framework for anomaly detection in attributed social networks. Int. J. Inf. Tecnol., 14, 3069– 3076.
4. N. Tabassum, A. Namoun, T. Alyas, A. Tufail, M. Taqi, and K. Kim,(2023) "applied sciences Classification of Bugs in Cloud Computing Applications Using Machine Learning Techniques.
5. Husain, Mohammad Salman & Haroon, Dr. Mohammad. (2020). An enriched information security framework from various attacks in the IoT. International Journal of Innovative Research in Computer Science & Technology (IJIRCST),8(3). Available at: SSRN: https://ssrn.com/abstract=3672418.
6. Zeeshan Ali Siddiqui & Mohd. Haroon. (2023). Research on significant factors affecting adoption of blockchain technology for enterprise distributed applications based on integrated MCDM FCEM-MULTIMOORA-FG method. Engineering Applications of Artificial Intelligence,118,105699.
7. Kumar, M., Hanumanthappa, M. & Kumar, T.V.S. (2012). Intrusion detection system using grid computing using Snort. International Conference on Computing, Communication and Applications, 1-6.
8. Kene, S.G. & Theng, D.P. (2015). A review on intrusion detection techniques for cloud computing and security challenges. IEEE Sponsored 2nd International Conference on Electronics and Communication Systems, pp. 227-232.
9. Ibrahim, D. (2016). An overview of soft computing. 12th International Conference on Application of Fuzzy Systems and Soft Computing, pp. 34-38.
10. T. Alyas, I. Javed, A. Namoun, A. Tufail, S. Alshmrany, and N. Tabassum (2022), "Live migration of virtual machines using a mamdani fuzzy inference system," Comput. Mater. Contin., vol. 71, no. 2, pp. 3019–3033
11. Idhammad, M., Afdel, K. & Belouch, M. (2018). Distributed intrusion detection system for cloud environments based on data mining techniques. The First International Conference on Intelligent Computing in Data Sciences, 35-41.
12. Gill, S.S. & Buyya, R. (2018). SECURE: Self-protection approach in cloud resource management. Journal of IEEE Cloud Computing, 5(1), 60-72.
13. Belouch, M., El Hadaj, S., & Idhammad, M. (2018). Performance evaluation of intrusion detection based on machine learning using Apache Spark. Procedia Computer Science, 127, 1-6.
14. Gill, S. S., & Buyya, R. (2018). A taxonomy and future directions for sustainable cloud computing: 360-degree view. ACM Computing Surveys (CSUR), 51(5), 1-33
15. Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. Journal of Computational Science, 25, 152-160.
16. Alzahrani, S., & Hong, L. (2018, July). Detection of distributed denial of service (DDoS) attacks using artificial intelligence on cloud. In 2018 IEEE World Congress on Services (SERVICES) (pp. 35-36). IEEE.
17. N. Tabassum, T. Alyas, M. Hamid, M. Saleem, S. Malik, and S. Binish Zahra (2022), "QoS Based Cloud Security Evaluation Using Neuro Fuzzy Model," Comput. Mater. Contin., vol. 70, no. 1, pp. 1127–1140.
18. El-Alfy, E. S. M., & Al-Obeidat, F. N. (2014). A multicriterion fuzzy classification method with greedy attribute selection for anomaly-based intrusion detection. Procedia Computer Science, 34, 55-62.
19. Xiong, W., Hu, H., Xiong, N., Yang, L. T., Peng, W. C., Wang, X., & Qu, Y. (2014). Anomaly secure detection methods by analyzing dynamic characteristics of the network traffic in cloud communications. Information Sciences, 258, 403-415.
20. https://www.kaggle.com/code/dhoogla/cidds-001-01-oner-0-5auroc
21. S. Rajagopal and P. P. Kundapur, "Towards effective network intrusion detection: from concept to creation on Azure cloud," IEEE Access, vol. 9, Article ID 19723, 2021.
22. S. Rajagopal and P. P. Kundapur, "Towards effective network intrusion detection: from concept to creation on Azure cloud," IEEE Access, vol. 9, Article ID 19723, 2021.
23. J. K. Samriya and N. Kumar, "A novel intrusion detection system using hybrid clustering-optimization approach in cloud computing," Materials Today, 2020.
24. N. Jaber and S. U. Rehman, "FCM–SVM based intrusion detection system for cloud computing," Cluster Computing, vol. 23, pp. 1–11, 2020.
25. R. Rajendran, S. V. N. Santhosh Kumar, Y. Palanichamy, and K. Arputharaj, "Detection of DoS attacks in cloud networks using intelligent rule based classification system," Cluster Computing, vol. 22, no. 1, pp. 423–434, 2019.
26. P. Ghosh, A. Karmakar, J. Sharma, and S. Phadikar, "CS-PSO based intrusion detection system in cloud environment," Emerging Technologies in Data Mining and Information Security, Springer, Berlin, Germany, pp. 261–269, 2019.
27. S. Malik, N. Tabassum, M. Saleem, T. Alyas, M. Hamid, and U. Farooq(2022.), "Cloud-IoT Integration: Cloud Service Framework for M2M Communication," Intell. Autom. Soft Comput., vol. 31, no. 1, pp. 471–480.
28. E. Besharati, M. Naderan, and E. Namjoo, "LR-HIDS: logistic regression host-based intrusion detection system for cloud environments," Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 9, pp. 3669–3692, 2019.

29. R. Patil, H. Dudeja, and C. Modi, "Designing an efficient security framework for detecting intrusions in virtual network of cloud computing," Computers & Security, vol. 85, pp. 402–422, 2019.
30. Z. Chiba, "Intelligent approach to build a Deep Neural Network based IDS for cloud environment using combination of machine learning algorithms," Computers & Security, vol. 86, pp. 219–317, 2019.
31. M. I. Sarwar, Q. Abbas, T. Alyas, A. Alzahrani, T. Alghamdi, and Y. Alsaawy (2023), "Digital Transformation of Public Sector Governance With IT Service Management–A Pilot Study," IEEE Access, vol. 11, no. January, pp. 6490–6512, doi: 10.1109/ACCESS.2023.3237550.