Enhancing Network QoS via Attack Classification Using Convolutional Recurrent Neural Networks

Jawad Alkenani*, Mohsen Nickray
Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran
E-mail: Jawadalkenani@sa-uc.edu.iq [1], m.nickray@qom.ac.ir [2]
*Coresponding author

Keywords convolutional neural networks, recurrent neural networks, attack class, anomaly detection

Received: November 21, 2024

Cyber-attacks and intrusions in networks refer to malicious activities that breach or damage data. These activities include direct attacks, such as denial-of-service (DoS) attacks, which overwhelm servers with requests to disrupt services. Intrusion involves unauthorized access to systems by exploiting security vulnerabilities. Malware threats like viruses and worms infect systems to steal information. Additionally, social engineering techniques deceive individuals into revealing sensitive information, while phishing relies on fake messages or websites to gather user data. To prevent these attacks, it is necessary to implement effective security strategies, such as knowing the attack class to protect the network and data. In this paper, ConvRNN (Convolutional Recurrent Neural Network) is used as a large-scale advanced model between Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to process data containing spatial and temporal information. In addition, ConvRNN generates magical features from data through convolutional layers and serial convolution by RNN, which creates the model's ability to understand complexity, especially in security and surveillance agreements. The simulation results show that the proposed model outperforms LSTM, including precision, recall, F1 score, ROC curve, TPR, FPR, FNR, and accuracy.

Povzetek: Prispevek predlaga izboljšano klasifikacijo omrežnih napadov z uporabo konvolucijskih rekurentnih nevronskih mrež, s poudarkom na izboljšanju kakovosti storitev (QoS) in odkrivanju varnostnih groženj.

1 Introduction

Cyber-attacks and intrusions pose significant threats to the integrity, confidentiality, and availability of data in networks. Various types of attacks exist, including denial-of-service (DoS) attacks, where malicious actors overwhelm a server with excessive requests, disrupting services for legitimate users. Intrusions involve unauthorized system access by exploiting security vulnerabilities, allowing attackers to steal, alter, or delete data. Malware is another major threat, encompassing different types of malicious software such as viruses, worms, ransomware, and Trojan horses. Viruses attach to legitimate files and spread when shared, while worms replicate themselves across networks independently. Ransomware encrypts files and demands payment for decryption, and Trojan horses disguise themselves as legitimate software to carry out hidden malicious activities [1].

Social engineering techniques manipulate individuals into revealing confidential information, often through impersonation or pretexting. Phishing involves fraudulent emails or websites that deceive users into providing sensitive information, with spear-phishing targeting specific individuals or organizations. Man-in-the-middle (MitM) attacks occur when an attacker intercepts communication between two parties, allowing data theft or manipulation, particularly in unsecured Wi-Fi networks [2]. To prevent these attacks, organizations can implement firewalls that filter incoming and outgoing traffic based on security rules, blocking potentially harmful traffic. Data encryption ensures that intercepted information remains unreadable without proper decryption keys. Regular software updates are crucial for patching vulnerabilities that attackers may exploit, while continuous network monitoring helps identify unusual activities that could indicate an attack. Educating employees about cybersecurity best practices and the importance of strong passwords can significantly reduce the risk of successful attacks. Multi-factor authentication (MFA) adds an extra layer of security by requiring additional verification steps beyond just a password. Finally, having a well-defined incident response plan ensures organizations can respond quickly and effectively to security breaches, minimizing potential damage. Understanding the various cyber-attack types and implementing comprehensive security measures are essential for protecting networks and sensitive information [3].

This group related to ConvRNN (Convolutional Recurrent Neural Network) includes a variety of applications that take advantage of its capabilities in processing data with temporal and spatial dimensions. In the field of real-time video analysis, ConvRNN can be used to detect abnormal control or suspicious activation, which contributes to the activation of security surveillance. ConvRNN is also relied upon in network detection guidance in networks, where it helps to analyze network traffic data and abnormal people that may indicate the most important features of the network. In addition, ConvRNN applications have become computer vision, such as object recognition and motion tracking, which enables later understanding of behavior in the view. There are also studies on the development of intelligent systems using ConvRNN, where the accuracy of detection and analysis in real-time is enhanced. Finally, ConvRNN is used in the analysis of complex temporal data, such as weather forecasting or financial feature analysis, which shows the potential to handle complexity in various fields [4].

ConvRNN combines convolutional layers with recurrent layers, making it effective for processing data with both spatial and temporal dimensions. This architecture can significantly enhance Quality of Service (QoS) across various applications. In video surveillance, ConvRNNs analyze video streams by capturing spatial features and temporal dependencies, improving real-time object detection and tracking. For traffic prediction, they forecast traffic patterns by analyzing spatial data from road networks alongside historical traffic conditions, helping manage traffic and reduce congestion [5]. In speech recognition, ConvRNNs improves the accuracy of systems by effectively processing audio signals and enhancing user experiences in applications like virtual assistants. For network traffic analysis, they predict network performance and detect anomalies by examining traffic patterns over time, optimizing bandwidth usage, and maintaining service quality. In healthcare monitoring, ConvRNNs analyze time-series data from sensors in wearable devices to track health metrics, improving the reliability and responsiveness of healthcare services. The benefits of using ConvRNNs include enhanced feature extraction, where convolutional layers excel at capturing spatial features while recurrent layers handle temporal dependencies, resulting in richer data representations. They also achieve higher accuracy in predictions and classifications by capturing both spatial and temporal dynamics. Additionally, their architecture is well-suited for real-time applications, providing timely responses in critical systems [6]. In summary, ConvRNNs play a crucial role in improving QoS across various domains by effectively integrating spatial and temporal information, leading to better performance and increased user satisfaction.

In this research, we propose how DL approaches have been used to improve QoS in IoT. According to the articles evaluated, QoS in IoT-based systems is violated when the security and privacy of the systems are jeopardized or when IoT resources are not adequately managed. As a result, the purpose of this study is to investigate how Deep Learning has been used to improve QoS in IoT by avoiding security and privacy breaches in IoT-based systems and assuring effective and efficient resource allocation and management.

The paper is structured as follows: Section 2 provides an overview of Quality of Service (QoS) in IoT and deep learning algorithms, focusing on techniques used to improve QoS in IoT. It discusses challenges like network congestion, delays, and the need for efficient data processing, and how deep learning can address these issues. Section 3 presents a proposal based on deep learning for improving QoS, highlighting its use in data processing and feature extraction to enhance performance, such as improving data throughput, reducing latency, and stabilizing the network. Section 4 discusses the model evaluation, including the parameters, dataset, and metrics used to assess performance, focusing on how QoS is measured and the metrics like speed, accuracy, and response time. Section 5 presents the results, comparing the proposed model with existing models, and discusses improvements in QoS such as better throughput and lower latency. The final section concludes the paper, by summarizing key findings, lessons learned, and suggesting areas for future research, along with recommendations for more effective application of deep learning to improve QoS in IoT networks.

2 Literature review

In the field of networking, deep learning is a powerful tool for improving service quality. This area relies on advanced techniques such as neural networks to analyze vast amounts of data related to traffic and network performance. By analyzing this data, patterns, and anomalies can be detected, which may indicate network issues or opportunities for performance enhancement. When it comes to traffic management, deep learning models can be used to predict congestion periods [7]. By processing historical data, these models can forecast times when there will be a spike in resource demand. Based on these predictions, resources can be dynamically redistributed to mitigate congestion and enhance network responsiveness. Additionally, deep learning can improve the overall performance of the network. By analyzing data traffic and prioritizing it, bandwidth can be managed more effectively. For instance, bandwidth can be allocated in a way that ensures critical or essential applications receive priority, thereby enhancing the overall user experience. Deep learning also plays a crucial role in threat detection [8]. By implementing deep learning algorithms, systems can recognize abnormal activities or suspicious behaviors that may indicate a breach or an attack. These capabilities help enhance network security and protect sensitive data.

Finally, deep learning significantly improves user experience. By analyzing user behaviors and interactions with the network, services can be fine-tuned and adjusted to better meet users' needs. This approach leads to increased customer satisfaction and loyalty, ultimately contributing to business success. Accordingly, deep learning provides powerful tools for enhancing service quality in networking, contributing to more efficient, secure operations and an improved user experience [8].

The methodology of related works, their performance, and outcomes have been compiled in Table 1 to provide a concise and organized summary of previous research. The table focuses on the methods employed and evaluation criteria.

Table 1: Summarization table on the related works

Ref	**Methodology**	**Performance/Results**
[9]	• **MAFENN**	The goal of the MAFENN algorithm and framework design is to improve the feedforward DL networks' or their variants' learning capabilities using straightforward data feedback. A multi-agent MAFENN-based equalizer (MAFENN-E) is created for wireless fading channels with inter-symbol interference (ISI) to confirm the viability of the MAFENN framework in wireless communications. Based on experimental findings, the SER performance of systems that use the quadrature phase shift keying (QPSK) modulation method.
[10]	• **PLC-READER**	PLC-READER, a memory attack detection and response framework for safe cyber-physical systems. PLC_READER comprises a fine-grained memory structure analysis approach to pinpoint the crucial memory data. According to experimental results, PLC-READER can identify all memory assaults with 100% accuracy and promptly carry out the necessary emergency measures.
[11]	• **OPTIMIST**	OPTIMIST, a transparent, distributed IDS that is well-positioned and capable of managing both high-rate and low-rate DDoS attacks. Numerous simulation and testbed experiments demonstrate that OPTIMIST is the most effective method for striking a balance between DDoS detection and energy overhead. To classify DDoS attacks in software-defined networking (SDN)-based Industrial Internet of Things (IIoT) networks.
[12]	• **CNN-LSTM**	offers a feature selection technique for identifying the most pertinent data characteristics using a hybrid convolutional neural network and long short-term memory (CNN-LSTM). The suggested model achieves a high accuracy of 99.50% with a time cost of 0.179 ms, according to performance findings.
[13]	• **CADeSH**	CADeSH is a two-step collaborative anomaly detection technique that first distinguishes between potentially harmful and benign traffic flows using an autoencoder. Only the rare flows are then analyzed using clustering, which determines whether they are malicious or benign. Eight IoT sensors spread across many networks provide 21 days of real-world traffic data to assess the approach. The findings of the experiment indicate an F1 score of 0.929, an FPR of 0.014, and a macro-average area under the precision-recall curve of 0.841.
[14]	• **TL**	Transfer learning (TL) is used to overcome the lack of labeled data and the dissimilarity of data characteristics for training in their collaborative learning framework for intrusion detection in IoT networks. The suggested framework can outperform the most advanced deep learning-based methods by over 40%, according to experiments conducted on current real-world cybersecurity datasets.
[15]	• **ViFLa**	ViFLa is updating DL-based models for traffic anomaly detection in IoT systems via machine unlearning, a method that rapidly updates machine-learning models without retraining. The technique, known as ViFLa, interprets each batch of training data as a virtual client in an FL framework and organizes them according to projected unlearning likelihood.
[16]	• **FL**	an intrusion detection method based on the semi-supervised FL scheme is proposed to address known FL issues, such as the privacy risk of having model parameters used to recover private data, the lack of independent and identically distributed private data, which hurts FL training, and the high communication overhead caused by the large model size, which impedes the solution's deployment.
[17]	• **Deep Learning Approach**	They proposed a unique anomaly detection strategy based on unsupervised deep learning techniques. The model compares the use of Restricted Boltzmann machines as generative energy-based models to autoencoders as non-probabilistic algorithms to determine if Deep Learning can detect anomalies. The simulation results indicate around 99% anomaly detection accuracy, ensuring QoS in IoT.
[18]	• **LSTM**	DL method for intrusion detection in IoT networks using bi-directional long short-term memory recurrent neural networks. Their study concentrated on the binary categorization of normal and attack behaviors using the Internet of Things network. With over 95% accuracy in attack detection and QoS in intrusion detection.
[19]	• **PAD**	The WMCA multi-channel face PAD database, which includes a variety of 2D and 3D assaults, is used to test the suggested solution. Additionally, we have conducted tests employing RGB channels only on the MLFP and SiW-M datasets. The usefulness of the suggested strategy is demonstrated by superior performance in invisible attack protocols. Publicly accessible software, data, and techniques are used to replicate the findings.
[20]	• **RNN**	Introduces a Time-Series-based Recurrent Neural Network (RNN) model, utilizing the LSTM network and applied to the CICDDoS2019 dataset. The proposed model outperforms previous benchmark models, achieving the highest performance with a one-layer LSTM network in a multiclass classification task. The one-layer LSTM model achieves an F1-Score of 0.980, Recall of 0.975, and Precision of 0.988.

3 Proposed method

DL offers great potential to enhance QoS in IoT networks and applications in the era of big data by enabling unique analytics. Different IoT networks require different QoS. However, maintaining QoS in IoT is a difficult task. Two areas need to be well handled to enforce QoS in IoTs: (1) Network and equipment security, which guarantees network resource security and privacy. (2) Verify the appropriate allocation and management of IoT network resources. Numerous facets of our daily existence, such as business, infrastructure, lifestyle, health, education, and the environment, might be revolutionized by IoT. Our lives depend on a few of these elements, therefore any decline in QoS might have catastrophic consequences. As a result, every issue that might jeopardize QoS must be addressed as soon as possible. IoT QoS breaches happen because of inadequate resource management or security flaws in IoT networks and systems. Optimization and heuristics are examples of traditional resource management techniques that are unable to effectively learn from data and behave appropriately in real time. For large and distributed IoT networks and applications, deep learning algorithms offer dynamic, intelligent decision-making and autonomous resource management.

Network traffic is a crucial component in today's network administration and management systems. Quality of Service (QoS) and network management both benefit from this information since the service being utilized directly affects the user's QoS needs. Because of the vast quantity and diversity of linked devices, Internet of Things (IoT) traffic will be difficult for existing network management and monitoring systems to handle. Figure 1 illustrates the proposed method of work.

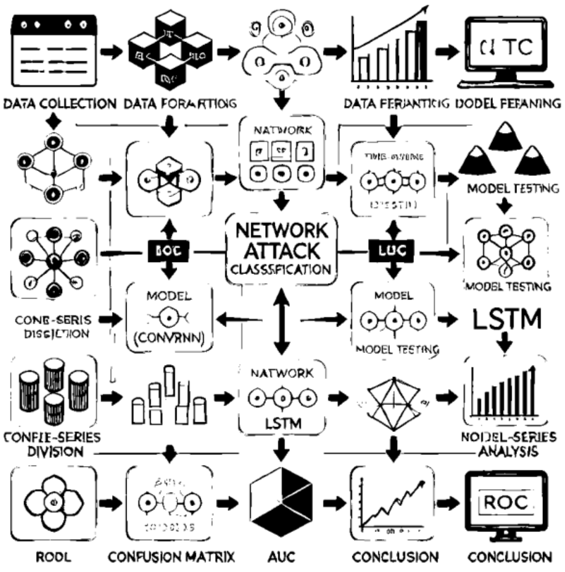

Figure 1: Proposed method of work

3.1 Preprocessing

Preprocessing is a crucial step in any machine learning project as it involves preparing raw data for use in models. The CICIDS2017 dataset was selected because it offers a diverse range of cyberattacks and realistic simulated network traffic with well-labeled data, making it highly suitable for training models to detect threats and analyze network behavior effectively. The case of the CICIDS2017 dataset, which contains network traffic data, includes several key steps. First, the data be loaded using the `pandas` library, which provides a flexible way to load data from CSV files into a DataFrame, a table-like structure. Once the data is loaded, it is important to explore the general structure of the dataset. This helps in understanding the columns, data types, and statistical measurements, and identifying any missing or incorrect data. After loading and inspecting the data, handling missing values becomes the next step. Some columns may contain missing values (like Nan), which need to be addressed. There are several ways to handle missing data, such as filling the missing values with the mean or median of the column or dropping the rows that contain missing values.

The next step is dealing with categorical columns (text-based data). The CICIDS2017 dataset may include columns that are textual, such as protocol names. These columns need to be converted into numerical values so that machine learning models can handle them. This can be achieved using One-Hot Encoding or Label Encoding. One-Hot Encoding converts categorical text columns into binary columns that represent each category as a 1 or 0, while) Label Encoding) transforms the text into numeric labels. Once the textual data is handled, the next step is to normalize the data. Normalization is the process of scaling data so that it falls within a certain range, such as 0 to 1. This step is especially important for models like LSTM or ConvRNN, where large values may negatively affect the model's learning efficiency. A common tool for this step is the MinMaxScaler, which scales the data to a range of 0-1.

After the data is normalized, it must be split into training and testing datasets. This step is essential to ensure that the model is trained and tested on different data to get an accurate evaluation. Typically, the data is split into 70% for training and 30% for testing.

Considering the size of the categories and percentages between the selected sample, the first two weeks of the OpenStack environment, and the full data, we count the number of items in each category in each group (the selected sample, the first two weeks, and the full data). The categories we use may include “Normal,” “Attacker,” “Victim,” “Suspicious,” and “Unknown,” or any other categories depending on the data you have.

3.2 ConvRNN

The ConvRNN model is a deep learning model that combines the strengths of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), allowing it to process data that has both spatial and temporal components. In this work, ConvRNN is used to analyze data that contains both time-dependent (temporal) and spatial features, such as network traffic data or cloud environment data, where patterns evolve. When using ConvRNN in this context, the convolutional layers are first used to extract spatial features from the input data. These convolutional layers apply convolutional filters to detect recurring patterns in the spatial structure of the data. For example, if the data represents network traffic, the convolutional layers help identify recurring patterns such as the protocols used or the amount of data flowing through the network. This helps the model understand the spatial relationships in the data.

Once the spatial features are extracted through CNN, these features are passed on to RNN layers, such as LSTM (Long Short-Term Memory). These recurrent layers are crucial for capturing the temporal dynamics of the data, meaning the relationships between events over time. The RNN layers allow the model to track how the spatial features evolve, making them capable of understanding how the patterns change over time. For example, if there is a cyber-attack on the network, the model can track sudden increases in traffic or abnormal changes in network behavior over time.

The integration of CNN and RNN in the ConvRNN model allows it to leverage the strengths of both types of networks. The CNN layers handle the spatial features of the data, while the RNN layers deal with the temporal dependencies. This combination makes ConvRNN particularly powerful for tasks that require both spatial and temporal understanding.

In this specific work, ConvRNN is applied to OpenStack data, which is a cloud environment that involves time-series data that changes continuously. The goal of using this model is to classify patterns in the OpenStack environment, such as cyber-attacks or abnormal activities within the network. By utilizing ConvRNN, the model can analyze the network traffic over time and identify patterns that indicate attacks or unusual behavior in the cloud environment.

The advantage of using ConvRNN is that it effectively combines the ability to process spatial data (which requires identifying patterns in the static features of the data) with the ability to handle temporal data (which involves understanding how patterns evolve). This makes the model capable of detecting attack classes that might appear as sudden changes in the network traffic patterns over time, such as a normal attack or an unusual surge in traffic. Overall, ConvRNN is a powerful model for handling data with both spatial and temporal components, making it ideal for applications such as attack detection in network environments like OpenStack, where patterns evolve dynamically over time.

4 Evaluation

The simulation for the proposed method of optimal spectrum and power allocation was conducted on a system equipped with an Intel Core™ i5 processor, 7th generation, running at a speed of 2.60 GHz. This processor has seven cores, providing efficient multi-tasking capabilities that help accelerate the complex calculations required for simulation. The system operates in a dual-boot configuration, allowing the user to switch between Windows 10 and Windows 8, which provides additional flexibility to choose the operating system best suited to the specific simulation and software requirements.

The system includes 1 GB of RAM, sufficient for running essential simulation tasks, although it may limit the handling of large datasets or intensive multi-processing operations. MATLAB 2020b was used to perform the simulations and conduct necessary analyses. MATLAB is one of the most widely used software packages in engineering and scientific fields, offering a powerful environment for data analysis, algorithm development, and executing experiments that require high computational accuracy and flexibility in handling various data types.

4.1 Evaluation parameters

A confusion matrix for a multi-class classification model, such as one with categories like Normal, Attacker, Victim, Suspicious, and Unknown, shows the performance of the model in classifying each category. True Positives (TP) represent cases correctly classified within their respective categories, such as when "Normal" cases are correctly identified as "Normal." False Positives (FP) refer to cases that were incorrectly classified as a particular category, like "Attacker" cases wrongly classified as "Normal." False Negatives (FN) occur when cases are incorrectly classified into a different category, such as "Normal" cases mistakenly classified as "Attacker." True Negatives (TN) represent all other cases that were correctly identified as not belonging to the target category. The confusion matrix provides valuable insight into the model’s accuracy for each class, revealing areas where errors occur and helping to assess and improve the model's performance, Table 2 shows the confusion matrix.

Table 2: Confusion matrix for five categories

	Predicted: Normal	Predicted: Attacker	Predicted: Victim	Predicted: Suspicious	Predicted: Unknown
Actual: Normal	TP	FP	FP	FP	FP
Actual: Attacker	FP	TP	FP	FP	FP
Actual: Victim	FP	FP	TP	FP	FP
Actual: Suspicious	FP	FP	FP	TP	FP
Actual: Unknown	FP	FP	FP	FP	TP

Here's an explanation of the key performance metrics used to evaluate a classification model, including their formulas and interpretations:

4.1.1 True positive rate (TPR)

Also known as Recall or Sensitivity, this metric measures the proportion of actual positive cases that are correctly identified by the model. It reflects how well the model can detect positive instances.

$$TPR = \frac{TP}{TP+FN} \quad (1)$$

4.1.2 False positive rate (FPR)

This metric measures the proportion of actual negative cases that are incorrectly classified as positive by the model. It shows the likelihood of a Type I error (incorrectly classifying a negative instance as positive).

$$FPR = \frac{FP}{FP+FN} \quad (2)$$

4.1.3 False negative rate (FNR)

This metric measures the proportion of actual positive cases that are incorrectly classified as negative. It shows the likelihood of a Type II error (incorrectly classifying a positive instance as negative).

$$FNR = \frac{FN}{TP+FN} \quad (3)$$

4.1.4 Classification rate (CR) or accuracy

Accuracy is the metric that measures the overall correctness of the model. It is calculated as the ratio of correctly predicted instances to the total number of instances.

CR (Accuracy) gives a general sense of how well the model is performing, but it does not always reflect the model's performance in classifying individual classes, especially in imbalanced datasets.

$$CR = Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

4.1.5 Receiver operating characteristic curve (ROC)

The ROC curve is a graphical representation that shows the tradeoff between the True Positive Rate (TPR) and False Positive Rate (FPR) at various thresholds. It helps evaluate the model's ability to distinguish between classes.

The X-axis represents the False Positive Rate (FPR).
The Y-axis represents the True Positive Rate (TPR).

These metrics help evaluate a model's performance in detail. TPR, FPR, and FNR provide insights into correct and incorrect classifications, while CR (Accuracy)

4.2 Research database

Child Table 3 is a dataset used for studying and analyzing network security attacks. It contains information that helps classify network activities and identify the type of attack or suspicious behavior. The table typically includes data such as the time of the event, source and destination IP addresses, the protocol used, the event label (such as "Normal," "Attacker," "Victim," or "Suspicious"), and the duration of the connection between devices. The CLDDS table is used to train artificial intelligence or machine learning models in Intrusion Detection Systems (IDS) and to analyze security patterns in networks.

External Server and OpenStack are two subfolders of the traffic folder. Several CSV files containing the collected ow-based network traffic in unidirectional NetFlow format may be found in these subfolders. These sub-folders le names are created as follows: Every file begins with CIDDS-001. Trac is identified as internal origin when it is logged in the OpenStack environment. The external server's trace is identified as having an external origin. Information about when the network traffic was recorded (week 1, week 2, week 3, and week 4) is provided in the last section (period). The CIDDS-001 data collection, which includes about 32 million flows, was collected over four weeks. In the OpenStack context, over 31 million flows were therefore recorded. At the remote server, about 0.7 million flows were recorded.

In this paper, we focus on Class labels (normal, attacker, victim, suspicious, and unknown) for the first and second weeks of OpenStack.

Table 3: Database specifications [21]

Number	**Description of the feature**	**Feature's name**
1	Source IP address	Src IP
2	Source port	Src Port
3	Destination IP address	Dest IP
4	Destination port	Dest Port
5	Transport protocol (eg, ICMP, TCP, or UDP)	Proto
6	The start time stream was first observed	Date first seen
7	Duration of flow	Duration
8	The number of bytes sent	Bytes
9	The number of packages sent	Packets
10	or append all TCP flags	Flags
11	Type of service	Tos
12	Not specified	Flows
13	Class label (normal, attacker, victim, suspicious and unknown)	Class
14	Attack type (PortScan, DoS, Bruteforce, PingScan)	AttackType
15	A unique attack ID allows attacks that belong to the same class to have the same attack ID	AttackID
16	Provides more information about configured attack parameters (eg, number of password-guessing attempts for SSH-Brute-Force attacks)	AttackDescription

4.3 Parameter setting

To design a ConvRNN model for attack classification using the CLDDS dataset, we need architecture that leverages convolutional layers for spatial feature extraction, followed by recurrent neural network (RNN) layers to capture temporal sequences, and finally, dense layers for final classification.

We start with one sequential convolutional layer, which is responsible for extracting the basic spatial features from the data. In the convolutional layer, we use 32 filters with a kernel size of (3x3). This layer serves as a foundation, capturing simple, fundamental features in the data such as repeated patterns. After this layer, we apply a Tanh activation function, commonly used in neural networks to introduce non-linearity, which allows the model to learn complex relationships in the data. Next, we add a Max Pooling layer with a (2x2) pool size, which reduces the data size and number of parameters, speeding up training and avoiding excessive complexity.

Once we have extracted the spatial features, we move on to the recurrent layers. Here, we use one LSTM layer to analyze the temporal sequences of the extracted features. RNN layers are highly suitable for tasks involving time sequences, like detecting attack classes. In the LSTM layer, we use 64 units (or cells), which are responsible for capturing the temporal information from the data. After this layer, we add a Dropout layer with a rate of 0.3 to prevent overfitting and improve the model's generalization.

After completing the recurrent layers, we pass the data to a two-hidden Dense layer with 64. This dense layer aggregates the features extracted from previous layers and prepares them for the final output layer. Finally, we add an output Dense layer with 5 units, representing the target classes: "normal," "attacker," "victim," "suspicious," and "unknown." We use the SoftMax activation function in this final layer to produce probabilities for each class, allowing the model to classify each sample based on the highest probability. Table 4 shows the architecture of the ConvRNN model for attack classification with its main details.

Table 4: ConvRNN model architecture for attack classification

Layer Type	Parameters	Purpose
Input Layer	-	Input shape: based on feature size and sequence length
Conv2D	Filters: 32, Kernel Size: (3,3), Activation: Tanh	Extract basic spatial features, capturing simple patterns
MaxPooling2D	Pool Size: (2,2)	Reduce feature size, parameters, and computational cost
LSTM	Units: 64	Capture temporal relationships within extracted features
Dropout	Rate: 0.3	Prevent overfitting and improve generalization
Dense (Hidden)	Units: 64	Aggregate features from previous layers for final output prep
Output Dense	Units: 5, Activation: Softmax	Produce class probabilities for 'normal', 'attacker', 'victim', 'suspicious', and 'unknown'

In the ConvRNN model, cross-entropy is used as both the objective and the loss function. Cross-entropy is commonly used for classification tasks because it measures the difference between the actual distribution of labels and the predicted distribution, helping to improve the model's prediction accuracy. The model is trained using the Adam optimizer, which is widely used in machine learning because it adapts the learning rate for each parameter individually, making the optimization process more efficient. The default learning rate for the Adam optimizer is set to 0.001, which works well for most tasks. The number of epochs is set to 50, meaning the model will perform 50 full passes over the training data. The batch size is set to 1024, meaning the model updates its weight based on 1024 samples in each iteration. This large batch size helps stabilize gradient estimation during training but requires significant memory resources.

Table 5 shows the training settings of the ConvRNN model.

Table 5: ConvRNN model training settings

Parameter	**Value**	**Description**
Loss Function	Cross-Entropy	Measures the difference between actual and predicted label distributions to improve classification accuracy
Optimizer	Adam	Adapts the learning rate for each parameter to improve optimization efficiency
Learning Rate	0.001	Default rate for the Adam optimizer, suitable for most tasks
Epochs	50	The model will perform 50 full passes over the training data
Batch Size	1024	The large batch size stabilizes gradient estimation, and requires more memory resources

5 Results

Examining the results of the trained models (ConvRNN and LSTM) in detail provides insights into why ConvRNN performed better in this case. First, the confusion matrix is a primary tool for understanding how well each model classified the data. In the case of ConvRNN, in Figure 2 the matrix shows that the model classified the categories more accurately, with values on the diagonal representing correct classifications and off-diagonal values indicating misclassifications. If ConvRNN has a higher number of correct classifications with fewer errors than LSTM, this indicates that ConvRNN was better at understanding and categorizing the data.

Next, the classification report, containing precision, recall, F1-score, and overall accuracy, provides a broader view of model performance. Precision reflects the model’s ability to classify positive samples accurately, while recall (sensitivity) measures the model's ability to detect positive samples. Here, if ConvRNN exhibits higher precision and recall, the model could classify samples accurately without confusing them with other classes. Additionally, the F1-score a harmonic means of precision and recall highlights the balance between these two metrics. If ConvRNN achieves higher F1 scores, it suggests that it balanced precision and recall more effectively in classification.

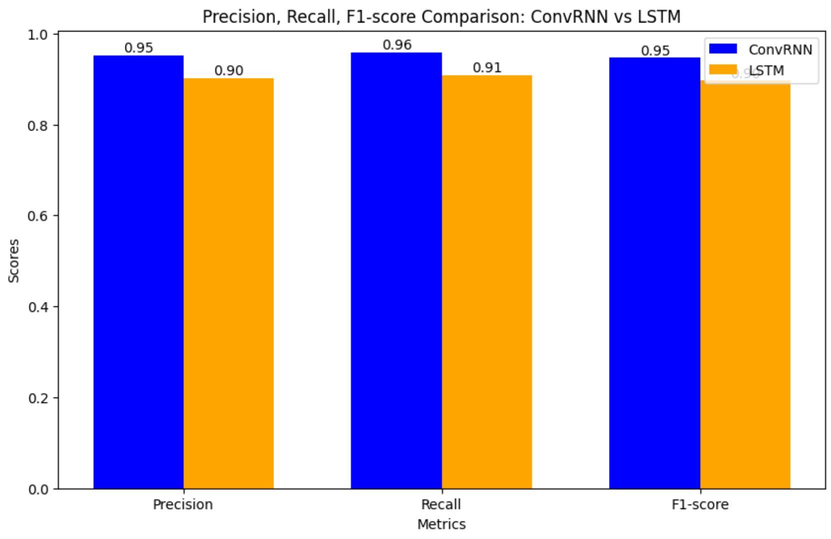

Figure 2: Comparison of the two models Precision, Recall, F1 Score

\

The ROC curve (Receiver Operating Characteristic), in Figure 3 is valuable for assessing how well each model distinguishes between different classes. TPR (True Positive Rate) and FPR (False Positive Rate) are used to plot this curve. If ConvRNN achieves a more favorable ROC curve compared to LSTM, it indicates that ConvRNN can distinguish between categories more accurately, as shown by a higher AUC (Area Under the Curve), which summarizes the model's overall classification capability.

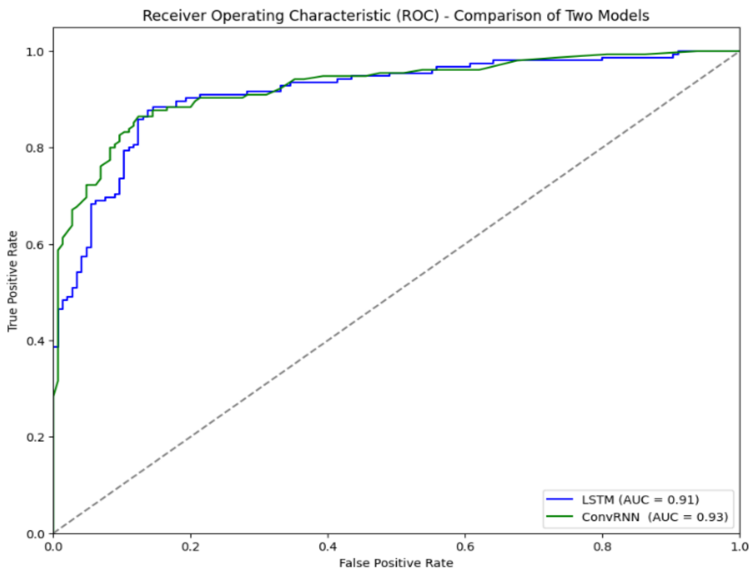

Figure 3: Comparison of the two models ROC Curve

The comparison in figure 4 of results between ConvRNN and LSTM for metrics such as True Positive Rate (TPR), False Positive Rate (FPR), False Negative Rate (FNR), and overall Accuracy demonstrates a clear advantage for the ConvRNN model. Regarding TPR, which measures the percentage of correctly identified positive cases, ConvRNN achieved a higher rate, indicating its efficiency in accurately detecting real attacks within the data compared to LSTM. This advantage is due to ConvRNN's ability to recognize complex patterns within network data, where its convolutional and recurrent layers enhance its sensitivity to true positive cases. For FPR, which measures the rate of negative cases incorrectly classified as positive, ConvRNN exhibited a lower rate than LSTM. This reduction in FPR signifies ConvRNN's capacity to minimize false alarms, thereby improving the classification accuracy and reliability of the model in monitoring network traffic. This is essential in reducing the likelihood of benign network activity being flagged as an attack, increasing the model's trustworthiness. Similarly, FNR, reflecting the number of true positive cases that went undetected, was also lower in ConvRNN compared to LSTM. This lower FNR highlights ConvRNN's superior ability to capture a wider range of attack patterns without overlooking them, further showcasing its strength in handling various attack scenarios within the data.

In terms of overall Accuracy, ConvRNN achieved significantly higher results than LSTM. This accuracy metric reflects the model's ability to correctly classify both positive and negative cases, and ConvRNN’s improvement in TPR while reducing FPR and FNR collectively contributed to this superior performance. Consequently, ConvRNN has proven to be a more reliable and effective model for network attack classification, offering high accuracy, reduced false alarms, and better detection of actual threats.

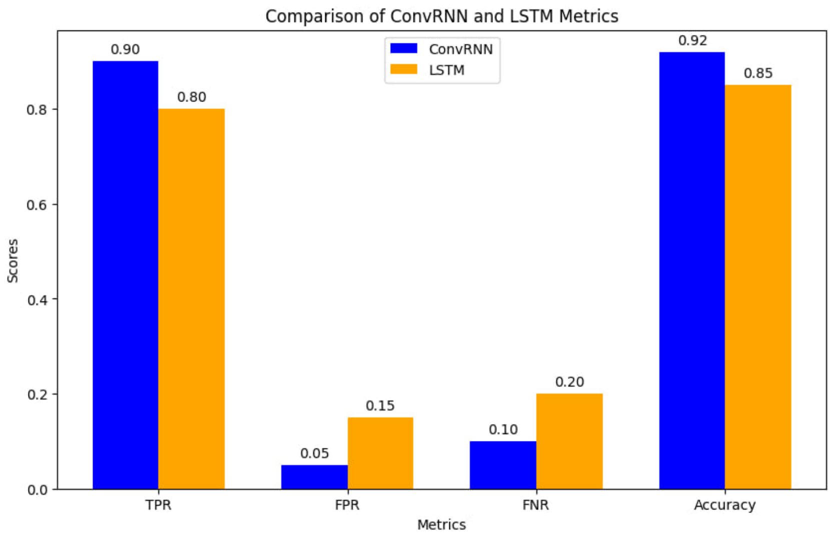

Figure 4: Comparison of the two TPR, FPR, FNR, Accuracy

The loss comparison between the ConvRNN and LSTM models reveals that ConvRNN consistently outperformed LSTM in both training and testing phases, as shown in figure 5. ConvRNN achieved lower training and testing loss, indicating that it was more effective at fitting the data and generalizing to new, unseen data. This lower loss demonstrates ConvRNN's ability to capture both spatial and temporal patterns in the network data, leading to more accurate predictions. In contrast, although LSTM showed some improvement in reducing loss during training, its performance on testing data was less robust, resulting in higher loss. This reinforces ConvRNN's superiority in minimizing errors and providing more reliable results in classifying network attack patterns.

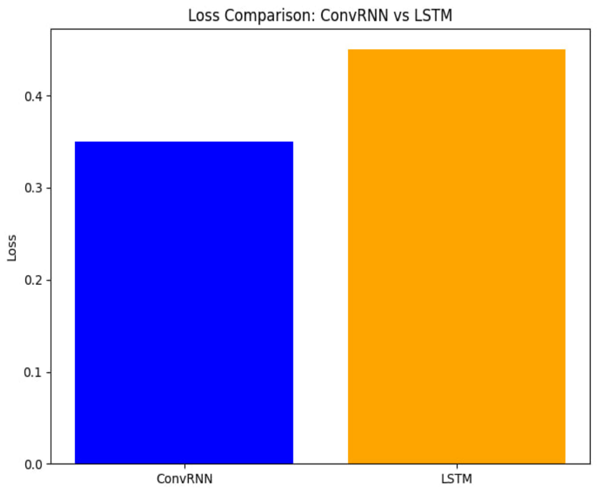

Figure 5: Loss comparison between the ConvRNN, LSTM

The comparison of training and prediction time between ConvRNN and LSTM revealed that ConvRNN, due to its more complex architecture, required slightly longer training times than LSTM. The ConvRNN model combines convolutional and recurrent layers, which demand more computational resources, thus increasing training time. However, regarding prediction time, both models performed similarly, with LSTM being marginally faster due to its simpler architecture. Despite the longer training time, ConvRNN demonstrated significantly higher classification accuracy, showing that the extra time spent on training was worthwhile for improved results in classifying network attacks.

Figure 6 shows the training time, and Figure 7 shows the prediction time.

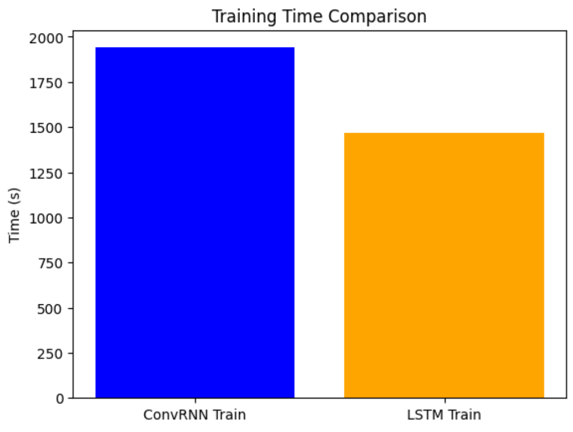

Figure 6 Training time time.

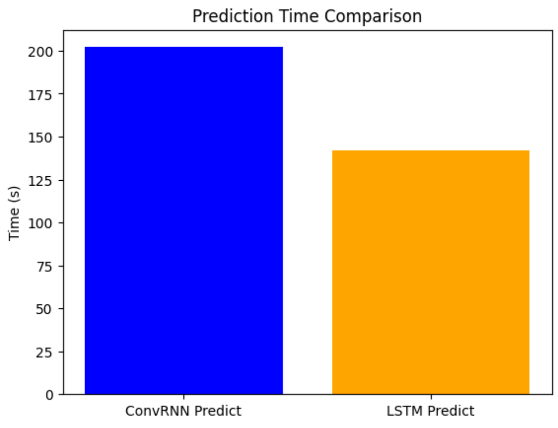

Figure 7: Prediction time.

In summary, ConvRNN outperformed LSTM due to its effective combination of convolutional and recurrent layers, which allow it to learn more quickly, identify local patterns accurately, and improve consistently across epochs. This combination makes ConvRNN especially well-suited to this task, allowing it to classify more accurately than LSTM in this context.

6 Conclusions

The ConvRNN model demonstrates superiority in classifying network attacks and improving service quality by effectively capturing spatial and temporal patterns. By integrating convolutional layers to extract spatial features and LSTM layers to process temporal sequences, ConvRNN excels in analyzing complex, multi-dimensional data. Although its intricate structure necessitates longer training times, the model achieves higher accuracy and surpasses LSTM in crucial performance metrics, such as true positive rate and error reduction. In contrast, LSTM, which focuses solely on temporal patterns, is less effective when dealing with data that incorporates spatial characteristics. Simulation results confirm that ConvRNN outperforms LSTM across various measures, including precision, recall, F1 score, ROC curve, true positive rate, false positive rate, false negative rate, and overall accuracy.

References

[1] Chaganti, Rajasekhar, et al. "A comprehensive review of denial-of-service attacks in the blockchain ecosystem and open challenges." *IEEE Access* 10 (2022): 96538-96555, doi: 10.1109/ACCESS.2022.3205019.

[2] Al-Shareeda, Mahmood A., et al. "Review of prevention schemes for man-in-the-middle (MITM) attack in vehicular ad hoc networks." *International Journal of Engineering and Management Research* 10 (2020), doi:10.31033/ijemr.10.3.23.

[3] Suleski, Tance, et al. "A review of multi-factor authentication in the Internet of Healthcare Things." *Digital health* 9 (2023),doi: 10.1177/20552076231177.

[4] Vazhenina, Daria, and Atsunori Kanemura. "Reducing the number of multiplications in convolutional recurrent neural networks (ConvRNNs)." *Advances in Artificial Intelligence: Selected Papers from the Annual Conference of Japanese Society of Artificial Intelligence (JSAI 2019) 33*. Springer International Publishing, 2020,doi: 10.1007/978-3-030-39878-1_5.

[5] Bodapati, Suraj, et al. "Comparison and analysis of RNN-LSTMs and CNNs for social reviews classification." *Advances in Applications of Data-Driven Computing* (2021): 49-59,doi: 10.1007/978-981-33-6919-1_4.

[6] Raza, Muhammad Raheel, Walayat Hussain, and José Maria Merigó. "Cloud sentiment accuracy comparison using RNN, LSTM and GRU." *2021 Innovations in intelligent systems and applications conference (ASYU)*. IEEE, 2021,doi: 10.1109/ASYU52992.2021.9599044.

[7] Sujanthi, S., and S. Nithya Kalyani. "SecDL: QoS-aware secure deep learning approach for dynamic cluster-based routing in WSN assisted IoT." *Wireless Personal Communications* 114.3 (2020): 2135-2169, doi: 10.1007/s11277-020-07469-x.

[8] Wu, Zheng, et al. "Online multimedia traffic classification from the QoS perspective using deep learning." *Computer Networks* 204 (2022): 108716,doi: 10.1016/j.comnet.2021.108716.

[9] Li, Yang, et al. "MAFENN: Multi-agent feedback enabled neural network for wireless channel equalization." *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021,doi: 10.1109/GLOBECOM46510.2021.9685522.

[10] Y. Geng et al., “Defending cyber–physical systems through reverse engineering-based memory sanity check,” IEEE Internet Things J., vol. 10, no. 10, pp. 8331–8347, 15 May 2023,doi: 10.1109/JIOT.2022.3200127.

[11] P. Bhale, D. R. Chowdhury, S. Biswas, and S. Nandi, “OPTIMIST: Lightweight and transparent IDS with optimum placement strategy to mitigate mixed-rate DDoS attacks in IoT networks,” IEEE Internet Things J., vol. 10, no. 10, pp. 8357–8370, 15 May 2023,doi: 10.1109/JIOT.2023.3234530.

[12] A. Zainudin, L. A. C. Ahakonye, R. Akter, D.-S. Kim, and J.-M. Lee, “An efficient hybrid-DNN for DDoS detection and classification in software-defined IIoT networks,” IEEE Internet Things J., vol. 10, no. 10, pp. 8491–8504, 15 May 2023,doi: 10.1109/JIOT.2022.3196942.

[13] Y. Meidan, D. Avraham, H. Libhaber, and A. Shabtai, “CADeSH: Collaborative anomaly detection for smart homes,” IEEE Internet Things J., vol. 10, no. 10, pp. 8514–8532, 15 May 2023,doi: 10.1109/JIOT.2022.3194813.

[14] T. V. Khoa et al., “Deep transfer learning: A novel collaborative learning model for cyberattack detection systems in IoT networks,” IEEE Internet Things J., vol. 10, no. 10, pp. 8578–8589, 15 May 2023,doi: 10.1109/JIOT.2022.3202029.

[15] J. Fan, K. Wu, Y. Zhou, Z. Zhao, and S. Huang, “Fast model update for IoT traffic anomaly detection with machine unlearning,” IEEE Internet Things J., vol. 10, no. 10, pp. 8590–8602, 15 May 2023,doi: 10.1109/JIOT.2022.3214840.

[16] R. Zhao, Y. Wang, Z. Xue, T. Ohtsuki, B. Adebisi, and G. Gui, “Semi-supervised federated-learning-based intrusion detection method for Internet of Things,” IEEE Internet Things J., vol. 10, no. 10, pp. 8645–8657, 15 May 2023,doi: 10.1109/JIOT.2022.3175918.

[17] Dawoud, A.; Sianaki, O.A.; Shahristani, S.; Raun, C. Internet of Things Intrusion Detection: A Deep Learning Approach. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020; pp. 1516–1522,doi: 10.1109/SSCI47803.2020.9308293.

[18] Roy, B.; Cheung, H. A Deep Learning Approach for Intrusion Detection in the Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network. In Proceedings of the 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), Sydney, NSW, Australia, 21–23 November 2018; pp. 1–6,doi: 10.1109/ATNAC.2018.8615294.

[19] George, Anjith, and Sébastien Marcel. "Learning one class representations for face presentation attack detection using multi-channel convolutional neural networks." *IEEE Transactions on Information Forensics and Security* 16 (2020): 361-375,doi: 10.1109/TIFS.2020.3013214.

[20] Gaur, Vimal, et al. "Multiclass classification for DDoS attacks using LSTM time-series model." (2022): 135-141,doi: 10.1049/icp.2022.0605.

[21] Ring, Markus, et al. "Technical Report CIDDS-001 data set." *J Inf Warfare* 13 (2017).