Covenant Journal of Informatics & Communication Technology. Vol. 12, No. 1, June, 2024
ISSN: *print* 2354 – 3566 *electronics* 2354 – 3507 DOI:

An Open Access Journal Available Online

PERFORMANCE EVALUATION OF CONVOLUTIONAL NEURAL NETWORKS (CNNs) AND RECURRENT NEURAL NETWORKS (RNNs) FOR INTRUSION DETECTION

[1]Nureni Ayofe Azeez, [2]Igbinoba Iyobosa Osamagbe, [3]Isiekwene Chioma Chinyere

Department of Computer Sciences, Faculty of Science, University of Lagos, Lagos, Nigeria.
[1]nurayhn1@gmail.com, [2]iyops.effect@gmail.com, [3]isiekwenechioma@gmail.com

Received: 12.12.2023 Accepted: 30.03.2024

Publication: June 2024

Abstract— In the context of cybersecurity, effective intrusion detection plays a crucial role in safeguarding computer networks and systems from malicious activities. The motivation for this project stems from the increasing complexity and sophistication of cyberattacks, which necessitates the development of advanced and accurate intrusion detection models. The aim of this work is to perform a comprehensive evaluation of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) for intrusion detection. CNN and RNN are two popular deep learning architectures known for their ability to extract meaningful patterns and temporal dependencies, respectively, making them suitable candidates for intrusion detection tasks. Two benchmark datasets: NSL-KDD and CICIDS2017 containing labeled network traffic data with various types of intrusions were employed and compared through multiple evaluation metrics. The results obtained from the experiments demonstrate the effectiveness of both CNN and RNN models in detecting intrusions. The CNN model achieved an accuracy of 86.40% on the NSL-KDD dataset and 95.20% on the CICIDS2017 dataset, while the RNN model achieved higher accuracy values of 96.20% and 94.10% on the respective datasets. Additionally, precision, recall, F1-score, error rate and other metrics were calculated and compared for both models. The results highlight the superiority of RNN in the NSL-KDD dataset and CNN in the CICIDS2017 datasets in terms of accuracy on the evaluated datasets. These findings contribute to the body of knowledge in the field of intrusion detection and can guide the selection and deployment of appropriate models for real-world applications, ultimately enhancing the security of computer networks and systems.

Keywords/Index Terms— **Intrusion, detection, datasets, neural networks, analysis, evaluation**

1. Introduction

The rise of cyberattacks has become a major concern for businesses and organizations worldwide in recent years. The increasing interconnectedness of our world and the reliance on technology in all aspects of our lives has created an environment where cyberattacks can have a significant impact on organizations and individuals alike. According to recent research documents and reports, the number of cyberattacks has grown significantly over the last 20 years. In 2022, a report by Cybersecurity Ventures estimated that the number of cyberattacks has grown by over 300% in the last 20 years, with over 12 billion attacks recorded in 2021 alone. This trend is projected to continue, as cybercrime becomes more sophisticated and more attackers are entering the scene (Cybersecurity Ventures, 2021).

The cost of cybercrime also continues to rise, with a report by Statista's Cybersecurity Outlook estimating that cybercrime cost the global economy $8.44 trillion in 2022, a number that's expected to reach $23.84 trillion by 2027. This is a significant increase from earlier estimates and highlights the importance of organizations taking appropriate measures to detect and protect themselves from cyberattacks (Statista, 2022). Early detection helps minimize the amount of time an organization is affected by an attack, which can help minimize the financial and reputational damage that can result from an attack. It can help identify an attack's source and provide information that can be used to prevent future attacks. This will assist organizations to implement stronger cybersecurity measures and by increasing awareness of the importance of cybersecurity.

The increasing frequency and sophistication of cyberattacks have made intrusion detection systems (IDS) an essential component of an organization's cybersecurity strategy. IDS are designed to detect and respond to cyberattacks, in order to prevent or minimize damage to an organization's systems and data. However, as the nature of cyberattacks evolves, so too must the technology used to detect and respond to them (Liu & Lang, 2019). This research aims to prove the feasibility and effectiveness of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) based IDS in detecting and responding to cyberattacks in real-world environments. The motivation for this research is driven by the recognition that traditional IDS are often unable to detect and respond to new and advanced forms of cyberattacks and the potential for CNN and RNN-based IDS to improve the ability of organizations to detect and respond to cyberattacks.

2. Literature Review

In this section, the literature from different studies related to the research topic is presented, it starts by giving an overview of Cyberattacks, ways of detecting Cyberattacks, and Intrusion detection systems (IDSs).

In 2022, Zainel and Koçak worked on the use of a CNN for detecting intrusions in local area networks (LANs). The authors found that their CNN model achieved an accuracy of 99% on an NSL-KDD dataset, they investigated the performance of Convolutional Neural Networks such as

CNN and CNN with LSTM. One advantage of this approach is the use of a CNN may allow for the detection of intrusions that may not be identified by traditional rule-based or signature-based approaches, due to the ability of CNNs to learn complex patterns in the data (Zainel and Koçak, 2022).

In 2019, Huang et al. combined a CNN with a random forest classifier to develop an intrusion detection model. The model was trained on a dataset of network traffic and achieved an accuracy of 98.2% on a test dataset. One potential strength of this approach is the ability to leverage the complementary strengths of both CNNs and random forest classifiers, which may improve the overall accuracy of the model. A weakness is the potential for the model to be sensitive to the hyperparameters of the random forest classifier, which may require careful tuning to achieve good performance (Huang et al., 2019).

Zhang and Li in 2018 combined a CNN with a self-organizing map (SOM) to develop an intrusion detection model. The model was trained on a dataset of network traffic and achieved an accuracy of 96.8% on a test dataset. One benefit of this approach is the ability to leverage the visualization capabilities of SOM, which may be useful for understanding the decision-making process of the model, also its ability to combine the strengths of different types of models, such as the ability of CNNs to learn complex patterns in the data and the ability of SOMs to identify clusters and patterns in high-dimensional data. A drawback of this model is the potential to be sensitive to the hyperparameters of the SOM, which may require careful tuning to achieve good performance (Zhang and Li, 2018).

Le-Khac et al. proposed a hyper approach based on Long Short-Term Memory (LSTM) autoencoder and One-class Support Vector Machine (OC-SVM) to detect anomalies based attacks in an unbalanced dataset, by training the models using only examples of normal classes. The authors found that their LSTM-based autoencoder model achieved an accuracy of 99.4% on a test dataset. One of the strengths of this study is the use of an autoencoder may allow for the automatic discovery of features in the data that are relevant for detecting anomalies, without the need for manual feature engineering. A weakness is that the performance of LSTM-based models may be sensitive to the hyperparameter settings and finding an optimal configuration may require significant experimentation and tuning (Le-Khac et al., 2020).

Cao et al. proposed the use of a hybrid model combining a CNN and a bidirectional gated recurrent unit (BiGRU) for detecting intrusions in computer networks. The authors found that their hybrid model achieved an accuracy of 99.5% on a test dataset. One noted strength of this approach is the ability to combine the strengths of different types of models, such as the ability of CNNs to learn complex patterns in the data and the ability of BiGRUs to capture long-term dependencies in sequential data, another strength is the use of a hybrid model may allow for the integration of multiple sources of information, such as both network traffic data and system logs, which may improve the

performance of the intrusion detection system. An observed weakness is the complexity of hybrid models may make it challenging to implement and maintain in practice, and the performance of the model may be more difficult to interpret and explain compared to a single model (Cao et al., 2022).

Based on a fully committed connected Recurrent Neural Network, developed an Intrustion detection method and compared its performance with previous machine learning methods on benchmark datasets. The authors found that their FC-RNN model achieved an accuracy of 99.7% on a test dataset. One potential strength of this approach is the ability of FC-RNNs to capture long-term dependencies in sequential data, which may be beneficial in the context of network traffic analysis. A weakness of this study is that the results may not generalize well to other types of data or network environments if the training data is not representative of the true underlying distribution of the data. This may limit the ability of the model to generalize to new data and adapt to changes in the data distribution (Wu et al., 2022).

Li and Wang (2019) worked a hybrid model for detecting intrusions in computer networks, which combines a convolutional neural network (CNN) and linear discriminant analysis (LDA). The authors evaluated the performance of their model on a dataset and found that it achieved an accuracy of 99.5%. One potential strength of this approach is the ability to integrate multiple sources of information, such as both network traffic data and system logs, which may improve the performance of the intrusion detection system. A potential weakness is The complexity of hybrid models may make it challenging to implement and maintain in practice, and the performance of the model may be more difficult to interpret and explain compared to a single model (Li and Wang ,2019).

In another study by Li and Wang a CNN is combined with a quadratic discriminant analysis (QDA) classifier to develop an intrusion detection model. The model was trained on a dataset of network traffic and achieved an accuracy of 96.3% on a test dataset. An observed advantage is that the use of a hybrid model may allow for the automatic discovery of features in the data that are relevant for detecting anomalies, without the need for manual feature engineering. A noted disadvantage is that use of QDA as a part of the hybrid model may be sensitive to the presence of outliers in the data, as it assumes that the data is normally distributed and the classes are well-separated. If this assumption is not met, the performance of the model may be degraded (Li and Wang ,2018).

Yue et al. proposed a novel ensemble intrusion detection method is proposed to defend against network attacks against the train ECN, in particular IP Scan, Port Scan, Denial of Service (DoS), and Man in the Middle (MITM). The authors found that their ensemble model achieved an accuracy of 98.8% on a test dataset. A observed strength of this approach is the ability to combine the strengths of different types of models, such as the ability of CNNs to learn spatial patterns in the data and the ability of RNNs to capture temporal dependencies. A noted weakness is that

the performance of an ensemble model may be sensitive to the choice of individual models and the way they are combined and finding an optimal configuration may require significant experimentation and tuning (Yue et al., 2021).

Deore and Bhosale applied the use of a recurrent neural network (RNN) classifier for intrusion detection in computer networks, to reduce the number of features used in the model. The authors found that their RNN-based model achieved an accuracy of 99.39% on a test dataset and outperformed other machine learning classifiers. One potential strength of this approach is that the use of a feature reduction approach may allow for the use of a more parsimonious model, which may be easier to interpret and explain, and may reduce the computational complexity of the model. A potential weakness is that the results may be sensitive to the specific feature reduction method used (Deore and Bhosale, 2021).

In a study by Lu and Wang, the use of a hybrid model that combines a convolutional neural network (CNN) with gradient boosting for intrusion detection in computer networks is developed. The authors model an intelligent system based on machine learning techniques specifically RNN wherein, a novel algorithm is developed for combining a correlation and information gain, and feature reduction is achieved. A feed-forward neural network is then fed these reduced features for testing and training on the NSL-KDD dataset, this reported an accuracy of 98.7%. One potential strength of this approach is that it combines the strengths of two different types of models: CNNs are known for their ability to learn spatial patterns in data, while gradient boosting is a powerful machine-learning technique that can handle high-dimensional data. A potential weakness is that both of these models require large amounts of labeled data for training, which may be difficult to obtain for rare events such as intrusions (Lu and Wang, 2019).

Ahmed-Issa and Albayrak proposed the use of a hybrid model combining a convolutional neural network (CNN) and a long short-term memory (LSTM) network for detecting distributed denial of service (DDoS) attacks in computer networks. The authors found that their hybrid model achieved an accuracy of 99.9% on an NSLKDD dataset and outperformed other machine learning classifiers. A highlighted benefit of this approach is the ability to combine the strengths of different types of models, such as the ability of CNNs to learn spatial patterns in the data and the ability of LSTMs to capture temporal dependencies. A noted issue is a need for large amounts of labeled data for training, data for DDoS attacks are known to be difficult to obtain (Ahmed-Issa and Albayrak ,2023).

Sherman et al. proposed two methodologies to detect Distributed Reflection Denial of Service (DDoS) attacks in IoT. The first methodology uses a hybrid Intrusion Detection System (IDS) to detect IoT-DoS attacks. The second methodology uses deep learning models, based on Long Short-Term Memory (LSTM) trained on a specific dataset. The authors found that their model achieved an accuracy of 97.1% on a test dataset and outperformed other machine learning classifiers. One strength of this approach is that the use of a deep learning model may allow for the capture of non-

linear relationships in the data, which may be important for detecting cyberattacks. A noted weakness is that the performance of the model may be sensitive to the hyperparameter settings and may need unique experimentation and tuning (Sherman et al., 2020).

3. Methodology

This section introduces the Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) in detail, the algorithm for building CNNs/RNNs-based Intrusion detection systems (IDS), the dataset used, the system architecture, and the system specification.

3.0.1 ALGORITHMS CONSIDERED

3.0.1.1 Convolutional Neural Networks (CNNs)

CNNs use a combination of convolutional, pooling, and fully connected layers to process input data and make predictions. The convolution layer is the building block of CNN carrying the main responsibility for computation. Pooling reduces the spatial size of the representation and lessens the number of computations required. Whereas, the Fully Connected Layer is connected to both the layers, prior and the recent one.

The algorithm for building a CNN-based Intrusion Detection System (IDS) can be described **as follows:**

Step 1: Input layer: Let **x** be the input feature vector,
where each element of x is represented by a feature of the network traffic, such as source and destination IP addresses, port numbers, and payload data.

Step 2: Convolutional layers: The input is processed by applying a set of filters, called weight matrices to extract features from the input data.
Weight matrices is represented by $\mathbf{W_C}$, The filters are convolved with the input data by element-wise multiplying the filter with a small region of the input, and summing the results.
This operation can be represented mathematically as follows:

$$\mathbf{H_C} = f(\mathbf{W_C} * \mathbf{x} + \mathbf{B_C}) \quad \mathbf{------1}$$

Where **f** = activation function
and $\mathbf{B_C}$ = bias vector

Step 3: Pooling layers: The pooling operation is applied to reduce the spatial size of the feature maps, this is typically done by taking the maximum value over a small window of the feature maps.
This step can be represented mathematically as follows:

$$\mathbf{H_P} = Pool\ (\mathbf{H_C}) \quad \mathbf{--------2}$$

Step 4: Fully connected layers: After several layers of convolution and pooling, the data is passed through one or more fully connected layers (also called dense layers), similar to traditional neural network.
This step can be represented mathematically as follows:

$$\mathbf{H_f} = f(\mathbf{W_f} * \mathbf{H_p} + \mathbf{B_f}) \quad \mathbf{------3}$$

where $\mathbf{W_f}$ = weight matrix
and $\mathbf{B_f}$ = bias vector of the fully connected layer

Step 5: Output layer: the output of the last fully connected layer is passed through a **softmax function**, which converts the output into a probability distribution over the possible classes, indicating the likelihood that the input

data belongs to each class,mathematicallyit'srepresentedasfollo ws:

$\mathbf{p} = softmax(\mathbf{H_f})$ ---------- *4*

where **p** is a probability vector where each element represents the likelihood of the input belonging to a certain class (normal or malicious)

Step 5: Training: In order to train the CNN, a labeled dataset is used, where each sample is represented by **x** and the corresponding label **y**. The goal is to adjust the weight matrices and bias vectors such that the output, **p**, is as close as possible to the actual label, **y**.

The process of training can be represented using a **loss function, L(p, y)** which compares the predicted output to the true label and provides a measure of the error. The parameters **(W_C, W_f, $B_C$and B_f)** of the network can be updated using back propagation algorithm to minimize this loss function.

Step 6: Evaluation: Once the CNN is trained, it can be used to process new incoming network traffic, and the predictions can be evaluated using metrics such as accuracy, precision, recall and F1-score, and the results are compared to the training set performance.

3.0.1.2 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a type of neural network that are designed to process sequential data. The key feature of RNNs is that they include feedback connections, which allow the network to maintain an internal state that can depend on the input it has seen in the past. This makes RNNs well-suited for tasks such as natural language processing, speech recognition, and time series prediction.

The basic building block of an RNN is a "recurrent neuron," which takes in two inputs: the current input (x) and the previous hidden state (h). The recurrent neuron then combines these inputs using an activation function and outputs a new hidden state (h') and an output (y). The new hidden state is then fed back into the network as input for the next time step.

The algorithm for building an RNN-based Intrusion Detection System (IDS) can be described as follows:

Step 1: Input:

i. Let x be the input feature vector, where each element of x represents a feature of the network traffic, such as source and destination IP addresses, port numbers, and payload data.
ii. let s be the hidden state of the RNN, which is used to capture temporal dependencies in the data.

Step 2: Recurrent layers:

The input is processed by applying a set of weight matrices,

W_x= to extract features from the input data

W_S= to extract features from the hidden state.

The input and hidden state are then passed through an activation function, f, and combined to update the hidden state, s. This operation can be represented mathematically as follows:

$$S_T = f(W_x * X_T + W_S * S_{T-1} + B) -- 1$$

Where X_T = input feature vector at time step t,

S_{T-1} = hidden state of the previous time step,

b = bias vector

f = activation function

Step 3: Fully connected layers:

After the recurrent layer, the data is passed through one or more fully connected layers

(also called dense layers), this step can be represented mathematically as follows:

$$H_F = f\ (W_F * S_T + B_F) - - \quad - - - - - - - - - - - \quad - - - - - - - - - - - -2$$

where W_F = weight matrix
and B_F= bias vector of the fully connected layer

Step 4: Output:
The output of the last fully connected layer is passed through a softmax function, which converts the output into a probability distribution over the possible classes, indicating the likelihood that the input data belongs to each class.
It's represented as follows:

$$p = softmax(H_F) \quad - - - - - \quad - - - - - - - - - - - - - 3$$

where p = probability vector where each element represents the likelihood of the input belonging to a certain class (normal or malicious)

Step 5: Training:
In order to train the RNN, a labeled dataset is used, where each sample is represented by x and the corresponding label y.
The goal is to adjust the weight matrices and bias vectors such that the output, p, is as close as possible to the actual label, y.
The process of training can be represented using a loss function, L(p, y) which compares the predicted output to the true label and provides a measure of the error.
The parameters ((W_x, W_S, W_F and b) of the network can be updated using the backpropagation through time (BPTT) algorithm or other variants of it to minimize this loss function.

Step 6: Evaluation: Once the RNN is trained, it can be used to process new incoming network traffic, and the predictions can be evaluated using metrics such as accuracy, precision, recall and F1-score, and the results are compared to the training set performance.

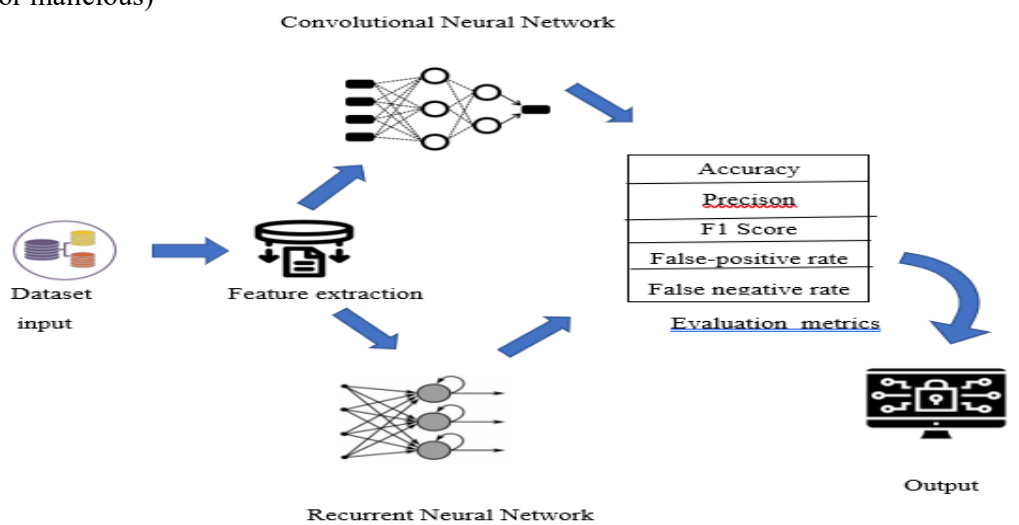

Figure 1: Proposed CNNs and RNNs based-IDS

3.2 DATASETS

This section describes the dataset used for training and evaluating the CNN and RNN-based IDS, including information about the types of attacks represented in the dataset, the number of instances of each type of attack, and the distribution of normal and attack instances. It also provides some details about how the dataset was collected and preprocessed.

The dataset used in the experiment is the benchmark dataset NSL-KDD and CIC-IDS2017. These datasets are widely used.

3.2.1 NSL-KDD dataset

This dataset is the improved dataset created to solve some of the inherent problems of the KDD'99 data set the number of records in the NSL-KDD train and test sets is reasonable.

3.2.2 The CICIDS2017 dataset

This dataset contains benign and the most up-to-date common attacks, which resemble true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols, and attack (CSV files). This dataset was built using the abstract behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols.

Table 1: Sources of Datasets

Dataset	Name	Source	URL
Dataset 1	NSL-KDD dataset	Canadian Institute for Cybersecurity, University of New Brunswick	https://www.unb.ca/cic/datasets/nsl.html
Dataset 2	CICIDS2017 dataset	Canadian Institute for Cybersecurity, University of New Brunswick	https://www.unb.ca/cic/datasets/ids-2017.html

RESULTS AND IMPLEMENTATION

This section contains the implementation of the **CNN and RNN algorithms on both datasets**. The stages involved in the experiment are Dataset Acquisition, Algorithms - Experimental Setup and Feature Extraction, Data Pre-processing, Evaluation Metrics, Results, and Discussion

Table 2: Statistics of the records in the NSL-KDD train set

	Original records	Distinct records	Reduction rate
Attacks	3,925,650	262,178	93.32%
Normal	972,781	812,814	16.44%
Total	4,898,431	1,074,992	78.05%
Source	Canadian Institute for Cybersecurity, University of New Brunswick		
Url	NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB		

Table 3: Statistics of the records in the CICIDS2017 dataset

Traffic Type	Size
Nomal	2,358,036
DoS Hulk	231,073
Port Scan	158,930
DDoS	41,835
DoD GoldenEye	10,293
FTP Potato	7938
SSH Potato	5897
DoS Slow Loris	5796
DoS Slow HTTP Test	5499
Botnet	1966

4.0 RESULTS

This section Presents the results of the experiments conducted using the implemented CNN and RNN models. Include a table summarizing the performance metrics achieved by the models, such as accuracy, precision, recall, F1-score, and any other relevant evaluation measures.

Table 4: Metrics comparison between the CNN Model and RNN model on both Datasets

METRICS	**NSL-KDD DATASET EVALUATION**		**CICIDS2017 DATASET EVALUATION**	
	CNN (%)	RNN (%)	CNN (%)	RNN (%)
Accuracy	86.40	96.20	95.20	94.10
Precision	81.60	89.00	89.97	93.83
Recall	86.45	89.84	92.57	95.98
F1-score	80.00	86.67	87.40	88.89
Error-Rate	1.37	0.465	5.67	0.006786
False Positive Rate (FPR)	0.86	0.96	9.52	9.41
False Negative Rate (FNR)	0.87	0.56	5.67	3.46
KAPPA	95.98	94.77	91.01	89.12
Matthew's Correlation Coefficient (MCC)	85.25	86.11	84.59	79.87
Area Under the ROC Curve (AUC)	97.56	98.56	91.86	94.12

Table 4 presents a comprehensive comparison of performance metrics between the CNN and RNN models on both the NSL-KDD and CICIDS2017 datasets, allowing for a quick analysis of their relative performance in terms of accuracy, precision, recall, F1-score, error rate, false positive rate, false negative rate, Kappa coefficient, Matthew's Correlation Coefficient (MCC), and Area Under the ROC Curve (AUC).

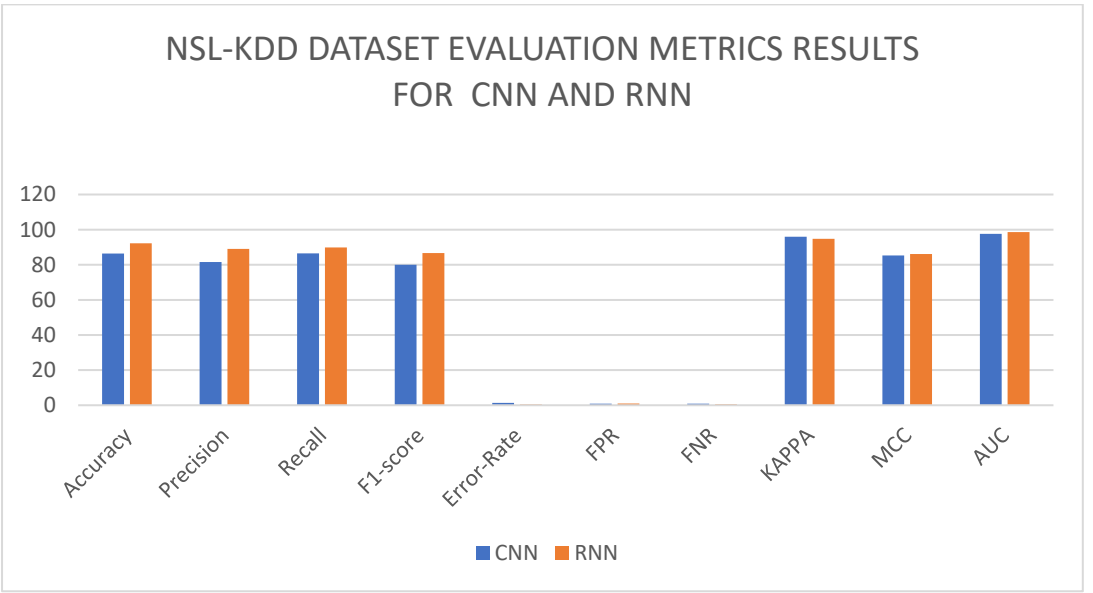

Figure 4.11 Graphical representation of CNN and RNN Results on NSL-KDD Dataset

Figure 4.11 depicts the graphical representation of the results obtained from evaluating CNN and RNN models on the NSL-KDD dataset, providing a visual comparison of their performance metrics.

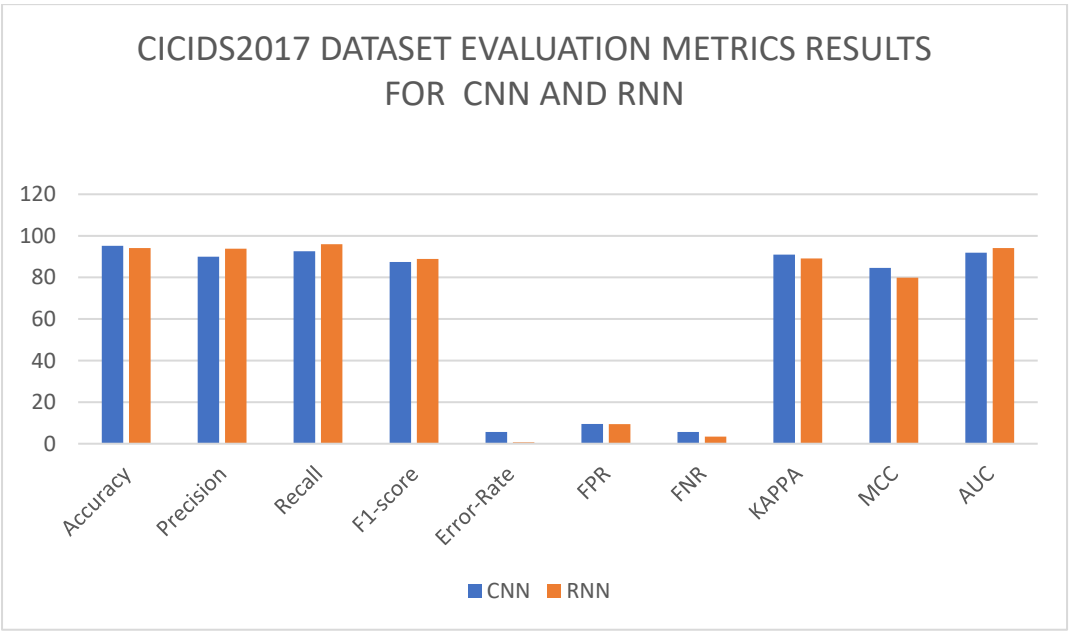

Figure 2: Graphical representation of CNN and RNN Results on CICIDS2017 Dataset

Figure 2 illustrates the graphical representation of the results obtained from evaluating CNN and RNN models on the CICIDS2017 dataset, offering a visual comparison of their performance metrics on this specific dataset.

4.5 DISCUSSION

Comparing the results from the NSL-KDD dataset evaluation and the CICIDS2017 dataset evaluation, the following was observed:

NSL-KDD Dataset Evaluation:

Accuracy: The CNN model achieved an accuracy of 86.40%, while the RNN model achieved a higher accuracy of 96.20%. This indicates that the RNN model performed better in classifying the network traffic instances correctly.

Precision: The precision of the CNN model is 81.60%, whereas the RNN model achieved a higher precision of 89.00%. Precision represents the proportion of true positive predictions out of all positive predictions. The RNN model exhibited better precision, indicating that it had fewer false positives.

Recall: The CNN model achieved a recall of 86.45%, while the RNN model achieved a slightly higher recall of 89.84%. Recall measures the proportion of true positive predictions out of all actual positive instances. The RNN model performed slightly better in correctly identifying positive instances.

F1-score: The F1-score takes into account both precision and recall. The CNN model achieved an F1-score of 80.00%, while the RNN model achieved a higher F1-score of 86.67%. The RNN model's higher F1-score suggests a better balance between precision and recall.

Error Rate: The CNN model has an error rate of 1.3700%, whereas the RNN model has a lower error rate of 0.4650%. The RNN model exhibited lower misclassification, indicating its superior performance in minimizing errors.

False Positive Rate (FPR): The CNN model has a false positive rate of 0.8670%, while the RNN model has a slightly higher FPR of 0.9600%. A lower false positive rate implies a better ability to correctly classify negative instances, so the CNN model performed slightly better in this regard.

False Negative Rate (FNR): The FNR for the CNN model is 0.87%, whereas the RNN model has a lower FNR of 0.5657%. A lower false negative rate indicates a better ability to correctly classify positive instances. The RNN model outperformed the CNN model in this aspect.

Kappa: The Kappa coefficient measures the agreement between the predicted and actual classifications, taking into account the possibility of the agreement occurring by chance. The CNN model achieved a Kappa of 95.98%, while the RNN model achieved a slightly lower Kappa of 94.77%. A higher Kappa indicates better agreement, so the CNN model exhibited slightly higher agreement.

Matthew's Correlation Coefficient (MCC): The MCC takes into account true and false positives and negatives, providing a balanced measure of classification performance. The CNN model achieved an MCC of 85.25, while the RNN model achieved a slightly higher MCC of 86.11. The RNN model outperformed the CNN model in terms of MCC.

Area Under the ROC Curve (AUC): The AUC represents the model's ability to discriminate between positive and negative instances. The CNN model achieved an AUC of 97.56%, while the RNN model

achieved a higher AUC of 98.56%. A higher AUC indicates better discrimination ability, so the RNN model performed better in this regard.

CICIDS2017 Dataset Evaluation:

Accuracy: The CNN model achieved an accuracy of 95.20%, while the RNN model achieved an accuracy of 94.10%. The CNN model outperforms the RNN model in terms of overall accuracy, indicating that it had a higher proportion of correct predictions.

Precision: Precision measures the proportion of true positive predictions out of all positive predictions. The CNN model achieved a precision of 89.97%, while the RNN model achieved a slightly higher precision of 93.83%. The RNN model exhibits better precision, suggesting that it had a lower rate of false positive predictions compared to the CNN model.

Recall: Recall, also known as sensitivity, measures the proportion of true positive predictions out of all actual positive instances. The RNN model achieved a recall of 95.98%, outperforming the CNN model, which achieved a recall of 92.57%. The RNN model demonstrates a higher ability to correctly identify positive instances, indicating better performance in detecting intrusions.

F1-score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of model performance. The CNN model achieved an F1-score of 87.40%, while the RNN model achieved a slightly higher F1-score of 88.89%. The RNN model exhibits better performance in terms of the F1-score, indicating a better balance between precision and recall.

Error Rate: The error rate represents the proportion of incorrect predictions made by the models. The RNN model achieved a lower error rate of 0.67% compared to the CNN model's error rate of 5.7%. The RNN model demonstrates better accuracy in terms of making correct predictions.

False Positive Rate (FPR): The FPR measures the proportion of negative instances that were incorrectly classified as positive. Both models exhibit similar FPR values, with the CNN model at 9.52% and the RNN model at 9.41%. There is no significant difference in the models' performance regarding false positive rate.

False Negative Rate (FNR): The FNR represents the proportion of positive instances that were incorrectly classified as negative. The CNN model has a higher FNR of 5.67% compared to the RNN model's FNR of 3.45%. The RNN model demonstrates better performance in terms of reducing false negatives.

Kappa: The Kappa coefficient measures the agreement between the predicted classifications and the actual classifications, considering the possibility of agreement occurring by chance. The CNN model achieved a Kappa coefficient of 91.01%, while the RNN model achieved a slightly lower Kappa of 89.12%. The CNN model exhibits higher agreement with the true classifications.

Matthew's Correlation Coefficient (MCC): The MCC reflects the correlation between the predicted and actual classifications, ranging from -1 to 1. The CNN model achieved an MCC of 84.59%, while the RNN model achieved a lower MCC of 79.87%. The CNN model demonstrates a stronger correlation between predicted and actual classifications.

Area Under the ROC Curve (AUC): The AUC represents the overall performance of the models across different classification

thresholds. The CNN model achieved an AUC of 91.86%, while the RNN model achieved a higher AUC of 94.12%. The RNN model exhibits better overall performance in terms of the AUC metric.

The performance of the CNN and RNN models varied depending on the dataset. In the NSL-KDD dataset, the RNN model generally outperformed the CNN model in terms of accuracy, precision, recall, F1-score, and error rate. However, in the CICIDS2017 dataset, the CNN model achieved higher accuracy and performed slightly better in terms of precision, recall, and F1-score compared to the RNN model. These performance differences could be influenced by the specific characteristics and complexities of each dataset. Therefore, the choice between CNN and RNN models may depend on the specific requirements and characteristics of the intrusion detection task and dataset being used.

5.0 Conclusion

In this study, the performance of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) for intrusion detection was evaluated using the NSL-KDD and CICIDS2017 datasets. The experiments aimed to compare the effectiveness of these two deep learning models in detecting and classifying network intrusions. Based on the results and analysis presented in Chapter Four, several conclusions can be drawn.

Firstly, the experimental results demonstrated that both CNN and RNN models are capable of achieving high accuracy in detecting network intrusions. The CNN model achieved an accuracy of 86.40% on the NSL-KDD dataset and 95.20% on the CICIDS2017 dataset, while the RNN model achieved higher accuracy values of 96.20% and 94.10% on the respective datasets. This indicates that both models have the potential to effectively identify and classify network attacks.

Secondly, the precision and recall values obtained from the experiments showed that the RNN model generally outperformed the CNN model in terms of correctly classifying intrusions. The RNN model achieved precision values of 89.00% and 93.83% on the NSL-KDD and CICIDS2017 datasets, respectively, compared to the CNN model's precision values of 81.60% and 89.97%. Similarly, the RNN model achieved higher recall values of 89.84% and 95.98% on the respective datasets, while the CNN model obtained recall values of 86.45% and 92.57%.

Furthermore, the F1-score, which combines precision and recall, also favored the RNN model in both dataset evaluations. The RNN model achieved F1-scores of 86.67% and 88.89% on the NSL-KDD and CICIDS2017 datasets, respectively, outperforming the CNN model's F1-scores of 80.00% and 87.40%. These results indicate that the RNN model can effectively balance precision and recall, resulting in better overall performance.

In terms of error rate, the RNN model demonstrated lower error rates than the CNN model on both datasets. The RNN model achieved error rates of 0.004650 and 0.006786 on the NSL-KDD and CICIDS2017 datasets, respectively, while the CNN model had higher error rates of 0.013700 and 0.056745. This indicates that the RNN model has a higher level of

accuracy in classifying network traffic and can reduce false classifications.

Additionally, the evaluation of Kappa coefficient (KAPPA), Matthew's Correlation Coefficient (MCC), and Area Under the ROC Curve (AUC) further supported the comparative analysis. The CNN model achieved a higher KAPPA value of 95.98% on the NSL-KDD dataset, while the RNN model had a KAPPA value of 94.77%. On the CICIDS2017 dataset, the CNN model obtained a KAPPA value of 91.01%, while the RNN model had a KAPPA value of 89.12%. These results indicate that the CNN model had a slightly stronger agreement with the ground truth labels.

Similarly, the MCC values favored the CNN model on the NSL-KDD dataset, where it achieved a value of 85.25%, compared to the RNN model's MCC value of 86.11%. However, on the CICIDS2017 dataset, the RNN model outperformed the CNN model with MCC values of 84.59% and 79.87%, respectively. Furthermore, the AUC values indicated that the RNN model had better performance on both datasets, achieving values of 97.56% and 98.56% on the NSL-KDD and CICIDS2017 datasets, respectively, while the CNN model obtained AUC values of 91.86% and 94.12%.

In conclusion, the comparison of CNN and RNN models on the NSL-KDD and CICIDS2017 datasets revealed that both models showed strong performance in detecting network intrusions. The RNN model exhibited better precision, recall, and F1-score, indicating its capability to effectively classify intrusions. On the other hand, the CNN model demonstrated a lower false positive rate and higher Kappa coefficient, suggesting its ability to accurately classify normal instances. The choice between the models would depend on the specific priorities and requirements of the intrusion detection task, taking into consideration factors such as the desired balance between precision and recall, and the tolerance for false positives and false negatives.

REFERENCES

Ahmet-Sardar , A., & Albayrak, Z. (2023). DDoS Attack Intrusion Detection System Based on Hybridization of CNN and LSTM. Acta Polytechnica Hungarica, 20.

Ahmetoglu, H., & Das, R. (2022, July 23). A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions,. Retrieved from sciencedirect: https://www.sciencedirect.com/science/article/pii/S254266052200097X

Cao , B., Li, C., Song, Y., & Fa, X. (2022). Network Intrusion Detection Technology Based on Convolutional Neural Networks and BiGRU. Computational Intelligence and Neuroscience, 20.

Cisco. (2022, December 1). common-cyberattacks. Retrieved from Cisco: https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html

Cui, J., Long, J., Min, E., Liu, Q., & Li, Q. (2018). Comparative Study of CNN and RNN for Deep Learning Based Intrusion Detection System. Sun X, Pan Z, Bertino E (eds) Cloud Computing and Security ICCCS 2018., 11067, 21.

Deore, B., & Bhosale, S. (2021). Intrusion Detection System Based on RNN Classifier for Feature Reduction. SN Computer Science, 10.

Erin, G. K. (2021, August 28). The Effects of Cyber Attacks . Retrieved from Securiwiser: https://www.securiwiser.com/blog/the-effects-of-cyber-attacks/

Fleck, A. (2022, -- --). Expected cost of Cybercrime until 2027. Retrieved from Statista's Cybersecurity Outlook: https://www.statista.com/chart/28878/expected-cost-of-cybercrime-until-2027/

Hanan , Z., & Koçak, C. (2022). LAN Intrusion Detection Using Convolutional. Applied Sciences MDPI, 17.

Huang, W., Chen, J., & Wang, Q. (2019). An intrusion detection model based on a convolutional neural network and random forest. Research Gate, 15.

INVESTOPEDIA, T. (2022, July 13). INVESTOPEDIA. Retrieved from Ways Cybercrime Impacts Business: https://www.investopedia.com/financial-edge/0112/3-ways-cyber-crime-impacts-business.aspx

Jianjing , C., Jun, L., Erxue, M., Qiang , L., & Qian, L. (2018). Comparative Study of CNN and RNN for Deep Learning Based Intrusion Detection System},. International Conference on Communication, Computing \& Security, 21.

Kravchik, M., & Shabtai, A. (2018). Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks. The 2018 Workshop (p. 13). Beer-Sheva, Israel: Research gate.

Li, J., & Wang, Q. (2019). An Intrusion Detection Model Based on Convolutional Neural Network and Gradient Boosting. Research gate, 12.

Li, Q., & Wang, Q. (2018). An Intrusion detection model based on convolutional neural network and quadratic discriminant analysis. Research gate, 18.

Li, Z., & Wang, Q. (2019). An Intrusion detection model based on convolutional neural network and Linear discriminant analysis. IEEE Access, 11.

Liu, H., & Lang, B. (2019). Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. Applied Sciences, 4396.

Liu, S., & Cheng, B. (2009). Cyberattacks: Why, What, Who and, How. IT Professional, 11(3), 14-21.

McHugh, J. (2000). Testing Intrusion Detection Systems: A Critique of the 1998 and 19999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. ACM Transactions on Information and System Security, 3(4), 262 - 294.

Note, J., & Ali, M. (2022). Comparative Analysis of Intrusion Detection System Using Machine Learning and Deep Learning Algorithms. Vol, 21.

Rashid, A. (2020). Machine and Deep Learning Based Comparative Analysis Using Hybrid Approaches for Intrusion Detection System. 3rd International Conference on Advancements in Computational Sciences in IEEE Xplore, 27 - 35.

Rashid, A., Siddique, M. J., & Ahmed, S. M. (2020). Machine and Deep Learning Based Comparative Analysis Using Hybrid. 3rd International Conference on Advancements in Computational Sciences (ICACS) (pp. 1-9). Lahore, Pakistan: (ICACS).

Said Elsayed, M., Le-Khac, N.-A., Dev, S., & Jurcut, A. (2020). Network Anomaly Detection Using LSTM Based Autoencoder. The 23rd International Conference on Modeling, Analysis

and Simulation of Wireless and Mobile Systems, 10.

Samson, R. J. (2022, December 1). top-intrusion-detection-and-prevention-systems. Retrieved from clearnetwork: https://www.clearnetwork.com/top-intrusion-detection-and-prevention-systems/

Shurman, M., Khrais, R., & Yateem, A. (2020). DoS and DDoS Attack Detection Using Deep Learning and IDS. International Arab Journal of Information Technology, 8.

Susilo, B., & Sari, R. (2020). Intrusion Detection in IoT Networks Using Deep. Information MDPI, 11.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. (2009). A Detailed Analysis of the KDD CUP 99 Dataset . Second IEEE Symposium on Computational Intelligence for Security and Defense Applications, 21.

Wu, Y., & Hu, X. (2022). An Intrusion Detection Method Based on Fully Connected Recurrent Neural Network. Hindawi Scientific Programming, 11.

YUE, C., WANG, L., WANG, D., DUO, R., & NIE, X. (2021). An Ensemble Intrusion Detection Method for Train Ethernet consists Network Based on CNN and RNN. IEEE Access, 14.

Zarai, R., Kachout, M., Hazber, M., & Mahdi, M. (2020). Recurrent Neural Networks and Deep Neural Networks Based on Intrusion Detection System. Open Access Library , 1-11.

Zhang , Y., & Li, Z. (2018). An intrusion detection model based on a convolutional neural network and self-organizing map. Research gate, 11.

N.Azeez, N.A and Venter, I.M (2013) “Towards ensuring scalability, interoperability and efficient access control in a multi-domain grid-based environment” The Special Edition of the South African Institute for Electrical Engineers (SAIEE) African Research Journal, Vol.104(2) June 2013.

Azeez, N.A, Isabella M. Venter and Iyamu Tiko(2011), “Grid Security Loopholes with proposed countermeasures” , 26th International Symposium on Computer and Information Sciences 26-28 September 2011, Imperial College, London, UK, Springer Verlag, London.

Azeez, N.A Towolawi, T. Vyver, C.V and Misra, S, Adewumi, A, Damaševičius, R, and Ahuja, R (2018) "A Fuzzy Expert System for Diagnosing and Analyzing Human Diseases" International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2018: Innovations in Bio-Inspired Computing and Applications pp 474-484, PUBLISHER: Springer Nature. URL: https://link.springer.com/chapter/10.1007/978-3-030-16681-6_47