Computer Science and Engineering 2024, 14(3): 56-66
DOI: 10.5923/j.computer.20241403.02

Leveraging Deep Learning Techniques for Enhanced Intrusion and Malware Detection Performance

Asmaa Ourdighi*, Sarah Yamina Messaoudi, Ikram Belabed

Department of Computer Science, University of Sciences and Technology of Oran –Mohamed Boudiaf, El M'Naouer Oran, Algeria

Abstract As computer networks become increasingly complex; the network security sector faces evolving cyber threats, highlighting the critical role of Intrusion Detection Systems (IDS) in identifying attacks. Currently, Deep Learning (DL) is gaining momentum as a preferred technique due to its ability to generalize in classification tasks. This study evaluates DL techniques for IDS and Malware Detection Systems (MDS) by comparing their performance under identical conditions. The choice of DL methods challenges conventional notions about designing effective neural architectures and input data types, including tabular data. Hence, we assess a basic ANN, more suitable for our case, alongside Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) model combined with Long Short-Term Memory (LSTM) tailored for sequential or temporal data. DL networks undergo testing on the NLS-KDD and Malware datasets, achieving an accuracy of 99.99% for IDS and 99.97% for MDS, with RNN-LSTM emerging as the top performer in both cases.

Keywords Artificial Intelligence, Deep learning, Intrusion Detection Systems, Malware Detection Systems

1. Motivation and Related Works

Nowadays, computer networks are faced with a significant influx of data from sources such as the Internet of Things (IoT), cybersecurity, mobile devices, businesses, social networks, healthcare, etc. To effectively protect these networks, intelligent analysis and automated solutions are essential. Artificial intelligence (AI), particularly machine learning and deep learning techniques, offer a powerful solution [1]. By leveraging the capabilities of AI, we can develop intelligent applications capable of analyzing large volumes of data, detecting threats in real-time, and automating security responses [2].

The DL introduction into cybersecurity has brought significant advancements in cyber-attacks detection and prevention [3]. Deep learning has significantly evolved within the cybersecurity industry, demonstrating its crucial role in enhancing threat detection and bolstering system resilience. Companies such as Darktrace utilize artificial intelligence to detect suspicious behaviors across networks. Cylance employs sophisticated deep learning algorithms to proactively identify and mitigate malware attacks by analyzing intricate behavioral patterns within files. Vectra Networks utilizes advanced deep learning techniques to monitor and promptly detect internal threats, offering automated threat responses. PatternEx enhances accuracy by reducing false positives through machine learning, while Sophos AI effectively identifies ransomware and targeted attacks using robust deep learning methodologies. These advancements underscore how deep learning plays a pivotal role in safeguarding systems against cyber threats, swiftly identifying malicious activities and bolstering overall network and computer security. In this section, we showcase a selection of research studies employing DL for IDS.

In Alom et al. [4], they employ a Deep Belief Network (DBN). A DBN is a deep generative model comprised of a visible layer and multiple hidden layers of latent variables. While connections exist between the layers, there are no connections between units within each layer [5]. The proposed system is capable of detecting attacks, and the accuracy of network activity is also identified and classified into five groups based on factors such as limited, incomplete, and nonlinear data sources. Compared to the existing system, the detection accuracy reaches 97.5% after 50 iterations. However, the DBN requires initial unsupervised pre-training and careful adjustment of hyperparameters to achieve good performance, which can involve numerous trials and errors.

Next, Tang et al. [6] implemented a Deep Neural Network (DNN) for IDS in a Software Defined Network (SDN) controller to monitor all flows of OpenFlow switches. They trained the model on the NSL-KDD dataset for binary classification (normal/anomaly) using only 6 basic features out of the 41 available. The model was optimized by varying the learning rate from 0.1 to 0.0001. Their model achieved an accuracy of 75.75%. Other works, such as those by [7], [8], [9], [10], pursued similar approaches.

* Corresponding author:
ouras2003@hotmail.com (Asmaa Ourdighi)
Received: Jun. 6, 2024; Accepted: Jun. 24, 2024; Published: Jul. 6, 2024
Published online at http://journal.sapub.org/computer

DL algorithms, especially Convolutional Neural Networks (CNNs), have demonstrated remarkable capabilities in automatically extracting intricate patterns and features from complex data, such as network traffic [11]. For instance, in the work of [12], a CNN was implemented to model network traffic events as time-series of TCP/IP packets within predefined time periods. Drawing inspiration from natural language processing techniques, the authors utilized a 1D Convolution layer [13]. This approach enabled modeling network traffic events as chronological data series, where instead of using 2D image data as input, the CNN processed a series of data in 1D format organized over time intervals. Different architectures were proposed, each containing an input layer, hidden layers with one or more CNN layers, and output layers such as FFN or RNN/LSTM/GRU to determine the optimal architecture. All experiments were conducted over 1000 epochs, and the CNN-LSTM achieved a high accuracy of 99% on the KDDCup 99 dataset.

Wu et al. [14] proposed an IDS model using Convolutional Neural Networks (CNNs) to automatically select traffic features from raw datasets, improving class accuracy and reducing the false alarm rate (FAR). Similarly, Xiao et al. [15] proposed an efficient IDS based on CNN, initially performing feature extraction using techniques like principal component analysis (PCA) and autoencoder (AE). They transformed the one-dimensional vector (feature set) into a two-dimensional matrix before inputting it into the convolutional neural network. Experimental results on the KDD Cup'99 dataset demonstrated efficiency in terms of learning and testing phase times, though with lower detection rates for U2R and R2L classes compared to other attack classes. Studies by [16], [17] recommend analysing network traffic using DL models, following a similar approach. However, CNNs may face challenges in capturing temporal dependencies within data sequences critical for detecting specific intrusion patterns, and their limited interpretability complicates understanding detected attack patterns.

In reference [18], IDS based on RNN using GRU as the main memory with a multilayer perceptron and softmax classifier was proposed. Testing on the KDD Cup'99 and NSL-KDD datasets showed good detection rates compared to other methodologies, with lower detection rates observed for minority attack classes like U2R and R2L.

Although our work focuses on deep learning, other machine learning studies have also achieved equally remarkable performances. In the work by Ferrag et al. [41], decision tree-based algorithms were utilized to assess performance on the CICIDS and BOT-IoT datasets, achieving respective accuracies of 96.665% and 96.995%. Similarly, Kunhare et al. [42] employed a random forest algorithm to select relevant features for reducing irrelevant attributes in intrusion detection. They conducted a comparative study using various classifiers including k-nearest neighbors (k-NN), support vector machine (SVM), logistic regression (LR), decision tree (DT), and naive Bayes (NB) to evaluate different metrics of intrusion detection systems (IDS). The particle swarm optimization (PSO) algorithm was applied to optimize the selected features on the NSL-KDD dataset, resulting in an accuracy of approximately 99.26%.

In conclusion, these examples underscore that the utilization of deep learning in Intrusion Detection Systems (IDS) and Malware Detection Systems (MDS) remains a pertinent research topic. DL enables exploration of various ANN approaches, continuously improving system performance [19]. Factors such as structure, data flow, neuron density, layer number, and deep activation filters contribute to expanding the perspectives of these approaches. However, variations in training and testing conditions, datasets used, and output classes considered may lead to comparative survey challenges in objectivity and effectiveness. Our work offers a concise comparative analysis aiming to improve IDS and MDS performances by presenting referenced DL methods. Based on related works, RNN and CNN approaches show efficiency in major performances. Additionally, the characteristics of input data play a crucial role in designing artificial neural network models. Different types of data, such as time series, sequential inputs, spatial data, or tabular datasets, often demand specific architectures tailored to handle their respective input data propagation. As a result, our study investigates three approaches (ANN, RNN-LSTM, and CNN) to observe and evaluate IDS and MDS performances based on input data nature. Following experimentation, we integrated LSTM into CNN architecture for further improvement.

2. Theoretical Background

The utilization of deep learning techniques offers numerous advantages, particularly in its capacity to extract intricate patterns and generalize to novel data. Our study aims to assess the performance of existing models. Among the various architectures examined, we have selected the most commonly used static and dynamic models, along with hybrid versions: Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), and Convolutional Neural Networks (CNNs) paired with Long Short-Term Memory (LSTM). In this section, we briefly outline the theoretical background of these approaches.

2.1. ANN Approach

A simple Artificial Neural Network (ANN) consists of multiple layers of neurons: an input layer, one or more hidden layers, and an output layer, as illustrated in Figure 1. Each neuron in a layer receives weighted inputs, sums them, and passes them through an activation function before transmitting them to the next layer. This process is repeated until the data reaches the output layer, where the final output is generated.

Learning in a simple ANN typically occurs through gradient descent. During the training phase, the network iteratively adjusts the weights and biases of the connections between neurons to minimize a loss function, often employing optimization techniques.

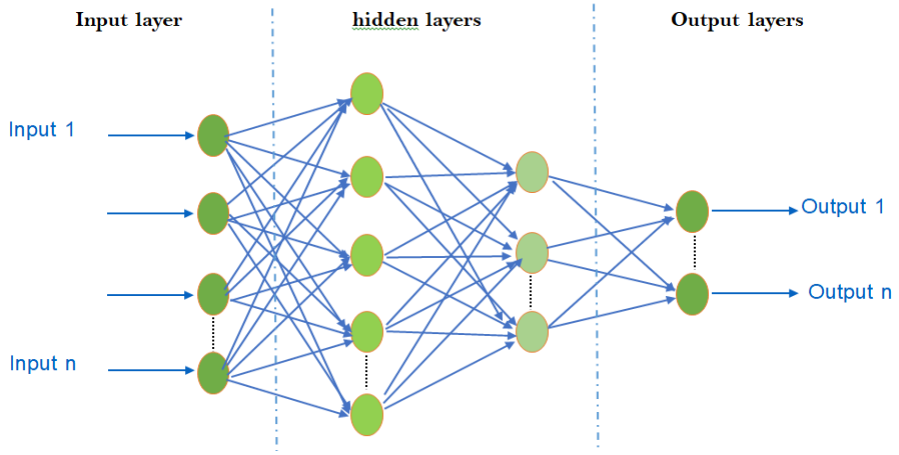


Figure 1. ANN topology

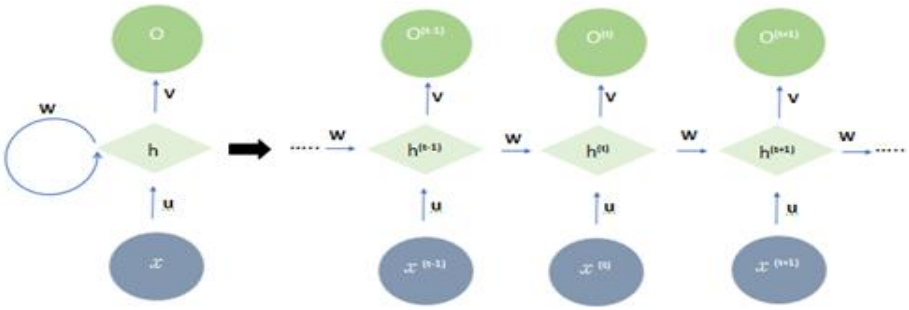


Figure 2. RNN topology

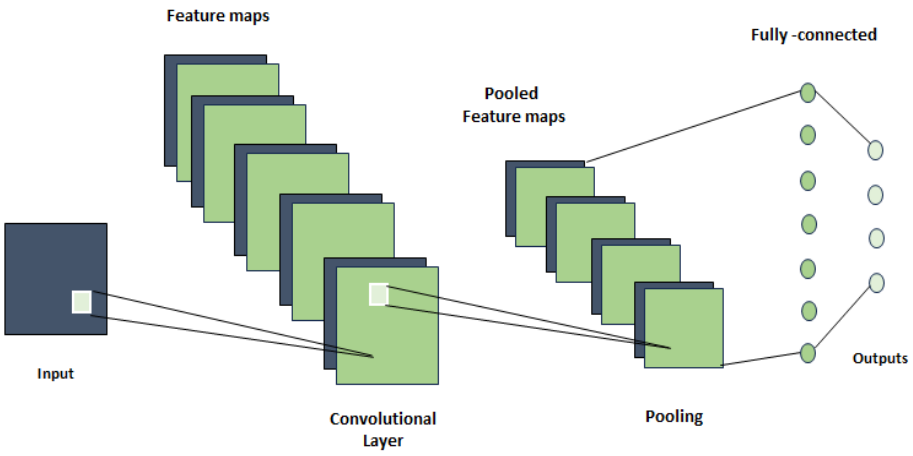


Figure 3. CNN topology

While simple ANNs are effective for many tasks, they may encounter overfitting issues with large datasets and deep architectures. Additionally, they may not always effectively capture temporal or spatial dependencies in the data.

2.2. RNN Approach

The main characteristic of an RNN is its utilization of recurrent loops, enabling the network to transfer information across different time steps. At each step of the sequence, the RNN takes into account the current input along with the internal state, or memory, computed from preceding steps. It then generates an output and updates its internal state for use in the subsequent step [20]. This recurrent mechanism empowers the RNN to capture long-term dependencies within the sequence.

The Figure 2 illustrates how the hidden state of an RNN at time step t is determined based on the hidden state at the previous time step, the input at time step t, and the associated weights and biases. The RNN consists of input layer x, hidden layer h, and output layer o. When unfolding the loop,

the standard RNN repeats this structure multiple times, with the state h of each iteration serving as input to the next. Denoting the input, hidden, and output layers at time t as x(t), h(t), and o(t) respectively, the output o(t) is calculated as follows:

$$\begin{aligned} a^{(t)} &= b_1 + W h^{(t-1)} + U x^{(t)} \\ h^{(t)} &= \sigma(a^{(t)}) \\ o^{(t)} &= b_2 + V h^{(t)} \end{aligned} \quad (1)$$

In (1), b1 and b2 are bias vectors, U, V, and W are the weighting matrices of the input-to-hidden connection, hidden-to-output connection, and hidden-to-hidden connection, respectively, and σ is an activation function, for example, sigmoid, tanh, ReLU.

Traditional RNNs may face challenges in capturing long-term dependencies or processing very long sequences due to a problem known as vanishing or exploding gradients.

2.3. CNN Approach

Proposed in the work of LeCun et al. [21], CNNs are feed-forward artificial neural networks capable of recognizing simple objects with high shape variability [22]. CNNs are a specific type of artificial neural network designed for supervised learning, particularly for processing data with a grid-like structure such as images or temporal sequences [13].

CNNs are typically structured as a sequence of layers, alternating between convolutional layers, activation layers (such as ReLU), and pooling layers. Additionally, fully connected layers may be appended at the end of the network for classification purposes, as described in Figure 3.

First, the input layer progressively extracts increasingly abstract features from the input data, enabling the model to better understand and make decisions about the data.

The convolutional layers apply filters to the input to extract important features. Each filter is a weight matrix that is learned during the network training, as shown in (2).

$$Output\ [i,j]=\sum(W * Input\ [\ i:i+K, j:j+K])++B \quad (2)$$

Where Output[i, j] represents the output value, W is the weight matrix, Input [i:i+K, j:j+K] is the input region covered by the filter, K is the filter size. B is the bias term.

Following each convolution operation, an activation function is applied to introduce non-linearity into the model. The ReLU function is commonly utilized for this purpose.

Pooling layers decrease the spatial dimensions of the extracted features by retaining the most significant values. Max pooling stands out as the prevalent pooling technique, where the feature map is partitioned into non-overlapping regions and the maximum value within each region is selected.

Finally, fully connected layers are utilized for the final classification or prediction task. In these layers, every neuron is connected to all neurons in the preceding layer. Each neuron within a fully connected layer calculates a weighted sum of the outputs from the previous layer, subsequently applying an activation function. Equation (3) illustrates the computation of the fully connected layer.

$$Output = f(\sum(W * Input) + B) \quad (3)$$

Where Output is the neuron's output, W is the weight associated with each connection, Input is the output of the previous layer, B is the neuron's bias, and f is the activation function applied to the weighted sum.

However, CNNs are less effective at capturing sequential dependencies in data, making them less suitable for tasks involving temporal or textual sequences.

In the context of cybersecurity, CNNs are indispensable for intrusion detection due to their ability to process structured data such as images and videos. CNNs excel in extracting hierarchical features from visual data, enabling precise identification of anomalies such as unauthorized objects or suspicious behaviors in security systems. Their capability to reduce false positives and integrate seamlessly into automated surveillance systems enhances real-time threat detection and response, significantly bolstering the security of networks and infrastructures against malicious attacks.

2.4. LSTM Approach

No Introduced by Hochreiter and Schmidhuber [23], the Long Short-Term Memory (LSTM) networks represent a refined iteration of Recurrent Neural Networks (RNNs), adept at overcoming the challenge of vanishing or exploding gradients through the integration of a complex structure and long-term memory mechanisms. Their primary function lies in capturing prolonged dependencies within sequential data. LSTMs utilize specialized LSTM units, specifically engineered to manage sequences while accounting for temporal dependencies [24].

The Long Short-Term Memory (LSTM) layer, a fundamental component of recurrent neural networks, is intricately structured with multiple gates and a memory cell. These elements work collaboratively to meticulously regulate the flow of information within the network. Below, we present several gates, including the forget gate, the input gate, and the output gate, which regulate the flow of information through the memory cell. The forget gate, in (4), controls the amount of past information to forget or retain in long-term memory. The input gate determines which new information should be stored in long-term memory, as shown in (5). Lastly in (6), the output gate controls the amount of information to be transmitted to the output based on the current state of the memory cell.

$$f_t=\sigma(W_f\cdot[h_{t-1},x_t]+ b_f) \quad (4)$$

$$i_t=\sigma(W_i\cdot[h_{t-1},x_t]+ b_i) \quad (5)$$

$$o_t=\sigma(Wo\cdot[h_{t-1},x_t]+ b_o) \quad (6)$$

In addition, Equation (7) and Equation (8) describes Update Gate and Update Cell. Equation (9) illustrates Output of LSTM.

$$\boldsymbol{C}_t=tanh(Wc\cdot[\ h_{t-1},x_t]+ b_c) \quad (7)$$

$$C_t=.f_t\cdot C_{t-1}+ i_t\cdot \boldsymbol{C}_t \quad (8)$$

$$h_t = o_t \cdot tanh(C_t) \quad (9)$$

Where $x_t$is the input at time t, h_{t-1} is the output from the previous time step, f_t, i_t and o_t are the values of the forget, input, and output gates respectively at time step t, W_f, W_i, W_c and W_o are the weights associated with each gate, b_f, b_i, b_c and b_o are the biases associated with each gate and σ is the sigmoid function.

These equations describe how information is filtered, updated, and propagated through the LSTM cell at each time step in a recurrent network.

After selecting methods, the subsequent step in designing the network topology is to define the input and output layers. This decision is influenced by the specific features of the dataset used. In the following section, we delineate our choices for each IDS and MDS framework.

In summary, while LSTM offers substantial advantages in hybrid models, it's crucial to acknowledge and address their limitations during both design and implementation phases. Thoughtful compromises are necessary to maximize LSTM's benefits while mitigating its drawbacks within hybrid architectures. LSTM plays a crucial role in intrusion detection by effectively processing temporal data sequences, such as network activity logs. By identifying anomalies based on suspicious activity patterns over extended periods, LSTM enhances security systems' ability to promptly detect and respond to potential threats. This contribution is indispensable for fortifying the security of networks and computer systems against malicious attacks.

3. Datasets Used

For this study, we utilized two datasets commonly used: the NSL-KDD dataset for IDS and the Malware dataset for MDS. The NSL-KDD dataset is a commonly referenced dataset for intrusion detection in computer networks. It was developed to improve upon the original KDD Cup 1999 dataset by addressing certain limitations and rendering it more realistic [25] [26]. The NSL-KDD dataset contains four categories of network attacks: "DoS" (Denial of Service) attacks, "Probe" attacks, "R2L" (Unauthorized Remote Access) attacks, and "U2R" (Privilege Escalation) attacks. Each entry in the NSL-KDD dataset is tagged with an attack class, signaling whether it represents an attack or a non-malicious activity. In this dataset, there are 125,973 instances with 41 attributes or features. The features are divided into three main type as follows: features extracted from the TCP/IP connection, features to access TCP packet payload and time-based traffic features and host-based traffic features [27].

For malware case, based on the characteristics of the observations, the Malware dataset was created in a Unix/ Linux-based virtual machine for classification purposes, containing benign and malware software for Android devices. The dataset comprises 100,000 observation data points and 35 features.

4. Methodology

For this study, each employed approach exhibits distinct characteristics, all of which play a vital role in effectively detecting intrusions and malware. The detection methodology hinges on binary classification, as illustrated in Figure 4. All approaches adhere to the same block diagram design. In this section, we will provide comprehensive descriptions of the design steps.

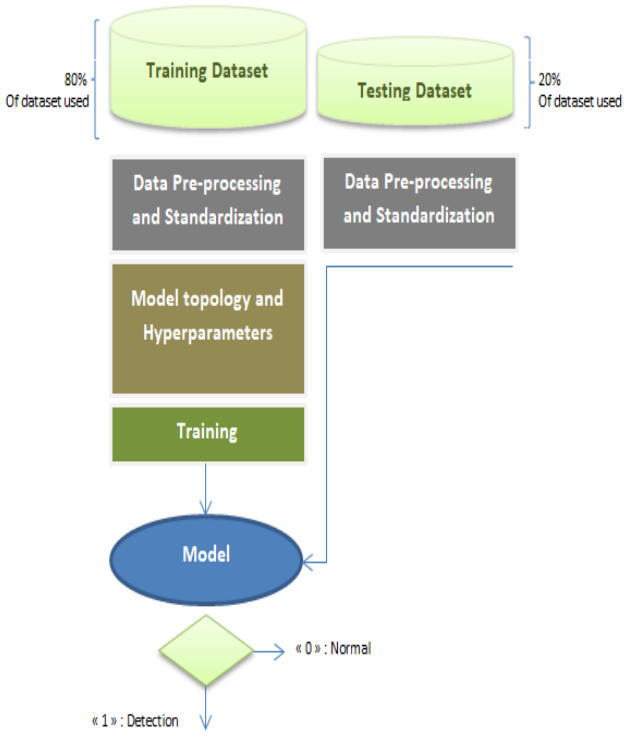


Figure 4. Block diagram of the proposed architectures for IDS and MDS

4.1. ANN Model

As Artificial Neural Networks (ANNs) can be utilized for intrusion or malware detection by analyzing network flow behavior. Prior to input into the ANN, collected information requires preprocessing, which could entail normalization, discretization, or feature transformation to suit the ANN input. Consequently, the remainder of the ANN architecture, including the number of hidden layers, neurons per dense layer, activation functions, etc., remains to be defined randomly [28].

Then, the simple ANN is trained using a labeled dataset. These labeled data are typically divided into two categories: normal or intrusion data for IDS, and benign or malware data for MDS. The ANN learns to recognize characteristic patterns of intrusions from these training data. Once the ANN is trained, it is evaluated on an independent test dataset.

For the IDS, the ANN was constructed using the TensorFlow library. The model comprises five dense layers with 64 neurons each and utilizes Rectified Linear Unit (ReLU) activation functions. The output layer is configured with a single unit and uses a sigmoid activation function.

For the MDS, a sequential architecture in Keras was employed. The model starts with an input layer with 27 neurons, corresponding to the number of attributes in a data instance. Subsequently, 50 hidden layers were added with ReLU activation functions. The output layer consists of two neurons representing the classification classes, utilizing a softmax activation function which generates probabilities for each output class.

4.2. RNN-LSTM Model

To detect intrusions using an LSTM, the network is trained on a pre-processed dataset transformed into sequences of vectors, where each vector represents a measurement or feature at a given time. During training, the LSTM learns to predict the class of each input sequence, i.e., whether it is an intrusion or not. It also learns to update its internal state (cell state) based on the input sequence, allowing it to consider long-term contextual information. Once the LSTM is trained, it can be used to detect intrusions by providing input data sequences and observing the predictions made by the model. If the LSTM predicts a high probability of intrusion for a given sequence, this may indicate the presence of malicious activity.

Meanwhile, the MDS model is sequentially constructed. Initially, the input dimensions determine the size of the third dimension, and the output size is set to two neurons. Subsequently, the model consists of two LSTM layers, each employing the ReLU activation function. Additionally, two dense layers are included, with the final layer outputting a single value with 2 features, totalling 258 trainable parameters.

4.3. CNN-LSTM

In this final part, we developed a CNN and Long Short-Term Memory (LSTM) model for our IDS. The model begins with a 1D Convolution Layer consisting of 32 filters, a kernel size of 9, padding to ensure the output size matches the input size, and a ReLU activation function. It outputs a one-dimensional array of 41 points with 32 channels, totaling 320 trainable parameters. The second layer is a MaxPooling1D Layer with a pooling window size of 2. The third layer is an LSTM Layer with 16 units and a dropout of 0.2. Finally, the last layer is a Dense Layer with a single output unit.

For malware detection, the CNN-LSTM model includes multiple layers. It starts with an Embedding Layer that converts inputs into dense vectors of size 8. Following this, a Batch Normalization Layer is applied to normalize activations from the previous layer, aiding in training stability. The model also incorporates a 1D Convolution Layer with 32 filters of size 9. Subsequently, a MaxPooling1D Layer reduces spatial dimensions by extracting maximum values within windows of size 2. An LSTM Layer with 512 units follows to capture sequential dependencies in the data. Lastly, a Dense Layer with a single unit and sigmoid activation produces the final output of the model, representing a probability between 0 and 1.

5. Resultats and Discussions

We implemented three Deep Learning models: ANN, CNN, and RNN, each incorporating LSTM. These models were trained and tested on two subsets of data: the NSL-KDD dataset for intrusion detection and the Malware dataset for malware detection, as shown in Table 1 and Table 2, respectively.

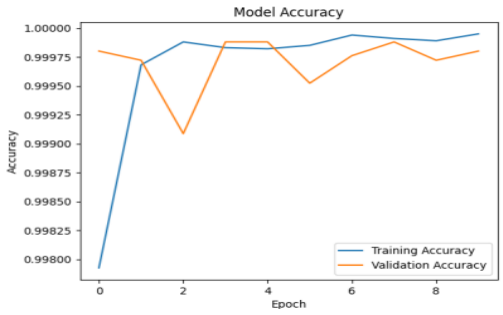
Table 1. Distribution of binary classification data in the NSL-KDD dataset

Subsets	Total	Normal	Attack
KDDTrain	125973	67343	58630
KDDTest	22544	12833	9711

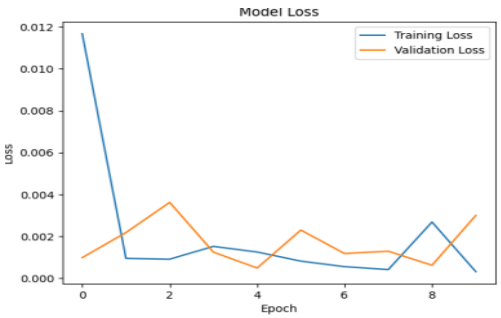
Table 2. Distribution of binary classification data in the NSL-KDD dataset

Subsets	Total	Normal	Malware
MalwareTrain	80000	40000	40000
MalwareTest	20000	10000	10000

Several tests were conducted to obtain the best hyperparameters for each model. These parameters, although they cannot be adjusted during the learning phase, have a significant impact on the models' performance during training. They include variables that determine the network structure (number of neurons, number of layers, activation function, etc.), batch size, and the number of iterations. The objective was to achieve a high-performing model with minimal error rate and maximum accuracy.

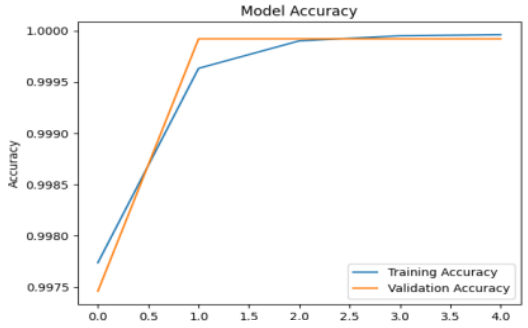


(a)

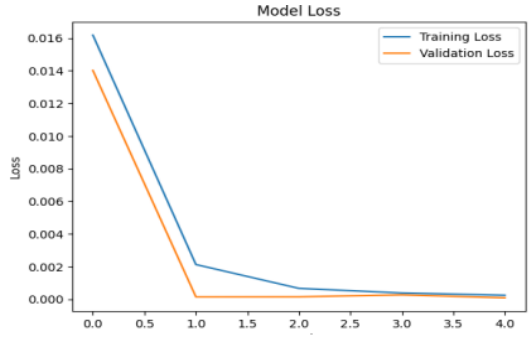


(b)

Figure 5. Evolution of training and validation accuracy (a) and loss (b) of the ANN model over epochs for intrusion detection in the NSL-KDD dataset

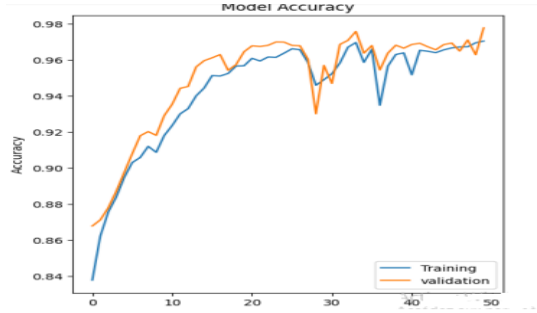


(a)



(b)

Figure 6. Evolution of training and validation accuracy (a) and loss (b) of the RNN-LSTM model over epochs for intrusion detection in the NSL-KDD dataset



(a)

(b)

Figure 7. Evolution of training and validation accuracy (a) and loss (b) of the CNN-LSTM model over epochs for intrusion detection in the NSL-KDD dataset

For the IDS, the models were trained for approximately 5 to 50 epochs. Figures 5(a), 6(a), and 7(a) show that both training and validation accuracy consistently increased from the beginning to the end, reaching maximum values approaching 1 for all approaches. Additionally, it was noted that the loss value decreased significantly during training, as depicted in Figures 5(b), 6(b), and 7(b). This indicates that the ANN, RNN-LSTM, and CNN-LSTM models learned better and made improved predictions after each optimization epoch.

During the testing phase, we achieved remarkably similar accuracy rates between the ANN and RNN-LSTM models, reaching 99.98% and 99.99%, respectively, while the CNN-LSTM model achieved 97.72%, as shown in Figure 8.

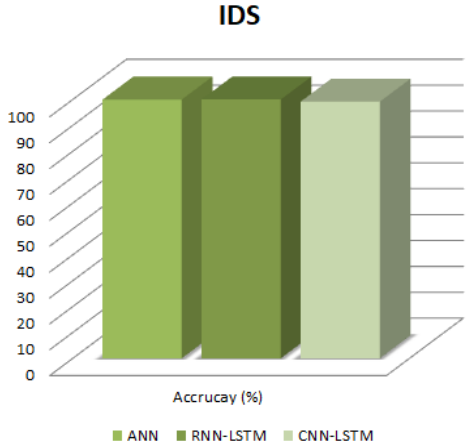


Figure 8. Optimal accuracy results for the proposed DL methods for intrusion detection

Additional metrics such as precision, recall, and F1-score are detailed in Table 3, highlighting the performance of these methods and showcasing the shortcomings of RNN-LSTM in our scenario.

Table 3. Results of precision, recall, and F1-score for the proposed DL methods for intrusion detection

Model	precision	recall	f1-score
ANN	0.99	0.99	0.99
RNN-LSTM	0.99	1	0.99
CNN-LSTM	0.97	0.99	0.98

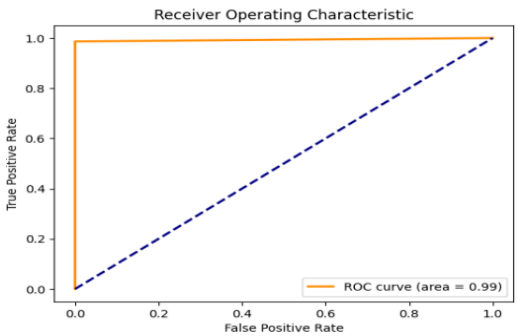
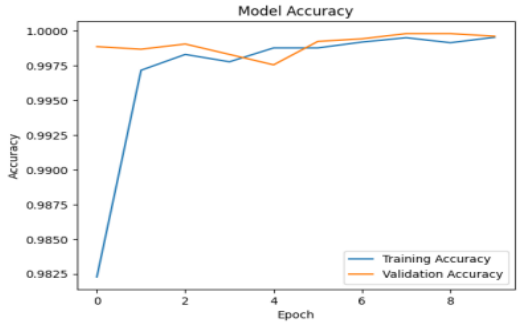


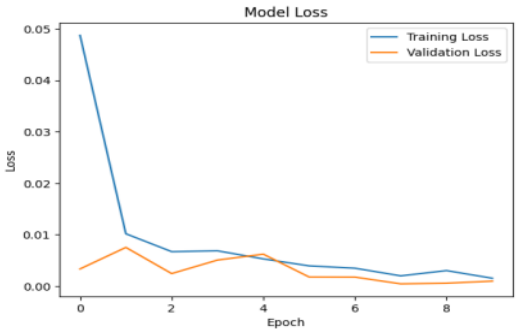
Figure 9. ROC Curve of RNN-LSTM for intrusion detection

In summary, the RNN-LSTM model exhibits the best observed results, demonstrating discrimination performance as depicted in the ROC curve illustrated in Figure 9. For the RNN-LSTM model, the AUC (Area Under the Curve) is 0.99,

indicating very high performance. This means the model has nearly perfect ability to correctly classify positive and negative samples. It effectively discriminates between the two classes, showing high precision and reliability in prediction.

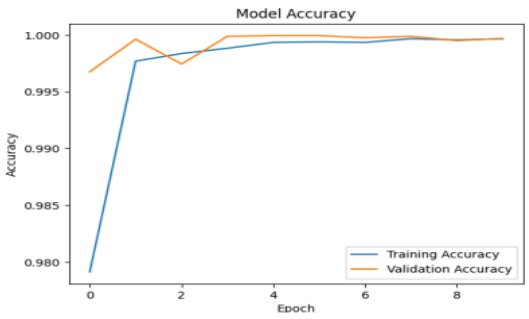


(a)

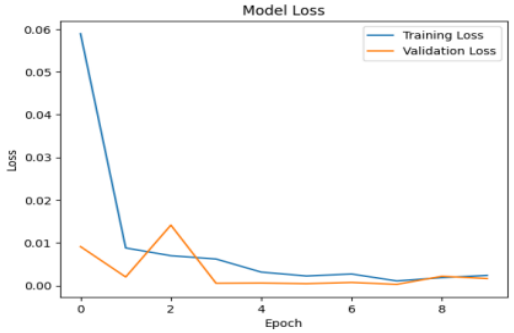


(b)

Figure 10. Evolution of training and validation accuracy (a) and loss (b) of the ANN model over epochs for intrusion detection in the Malware dataset

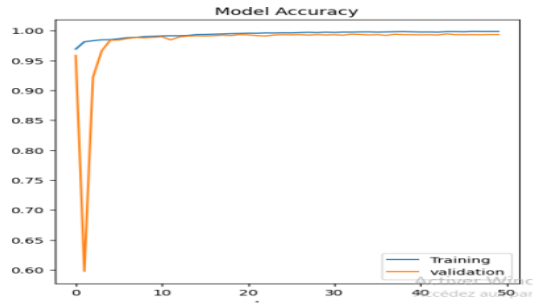


(a)

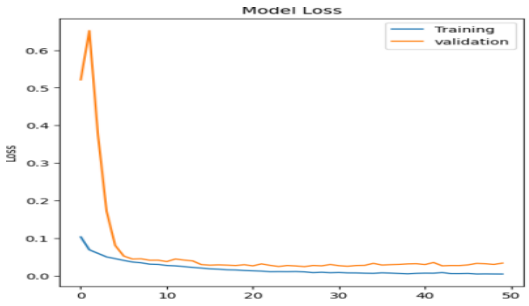


(b)

Figure 11. Evolution of training and validation accuracy (a) and loss (b) of the RNN-LSTM model over epochs for intrusion detection in the Malware dataset



(a)



(b)

Figure 12. Evolution of training and validation accuracy (a) and loss (b) of the CNN-LSTM model over epochs for intrusion detection in the Malware datase

For the MDS, all approaches were trained for 10 to 50 epochs. Figures 10(a), 11(a), and 12(a) illustrate the evolution of accuracy achieved by all methods, demonstrating stability and consistency in performance. Notably, the models converged to minimal loss values, with training and validation losses approaching parity as shown in Figures 10(b), 11(b), and 12(b).

Subsequently, Figure 13 indicates the test accuracy for each approach. These models were tested on the test subset, yielding good accuracies: 99.95% for ANN, 99.97% for LSTM-based RNN, and 99.14% for CNN-LSTM.

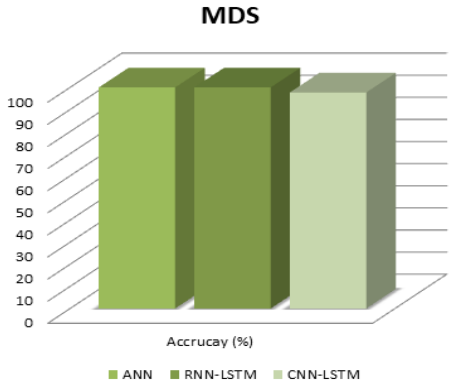


Figure 13. Optimal accuracy results for the proposed DL methods for Malware detection.

Table 4. Results of precision, recall, and F1-score for the proposed DL methods for Malware detection

Model	precision	recall	f1-score
ANN	0.99	0.99	0.99
RNN-LSTM	0.99	0.99	1
CNN-LSTM	0.91	0.75	0.82

Table 5. Related works in IDS

Authors & Years [ref]	Model	Dataset	Performance
Tavallaee et al, 2009 [29]	Random Forest	NSL-KDD	ACC=82.02%
Bhupendra Ingre et al, 2015 [30]	ANN	NSL-KDD	ACC=81.2%
Vinayakumar, et al, 2017 [12]	CNN	KDD'99	ACC =99.99%
Hsu et al, 2021 [22]	RNN; LSTM CNN-LSTTM	NSL-KDD	ACC=83.28%; ACC=89.23%; ACC=94.12%
Sandee et al, 2018 [31]	Deep auto-encoder	NSL-KDD	ACC =99.99%
Chia-Ming Hsu et al, 2019 [32]	CNN-LSTM	NSL-KDD	ACC=94.12%
Pramita Sree Muhuri et al, 2020 [33]	RNN-LSTM	NSL-KDD	ACC=99.83%

Table 6. Related works in MDS

Authors & Years [ref]	Model	Dataset	Performance
Hwang et al, 2020 [34]	DNN	malware Dataset	ACC = 94%
Yazdinejad et al, 2020 [35]	LSTM	malware Dataset	ACC = 98%
Dang et al, 2021 [36]	CNN	Malware dataset	ACC=94.3%
Satheesh kumar Sasidharan et al, 2021 [37]	LSTM	Malware dataset	ACC=99.23%
Ban et al, 2022 [38]	CNN	malware dataset	ACC = 98%
Akhtar et al, 2022 [39]	CNN-LSTM	Malware dataset	Acc = 99%
Gyamfi et al, 2022 [40]	CNN	Malware dataset	ACC=96%

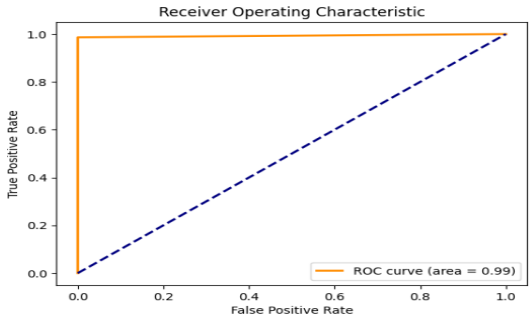


Figure 14. ROC Curve of RNN-LSTM for Malware detection

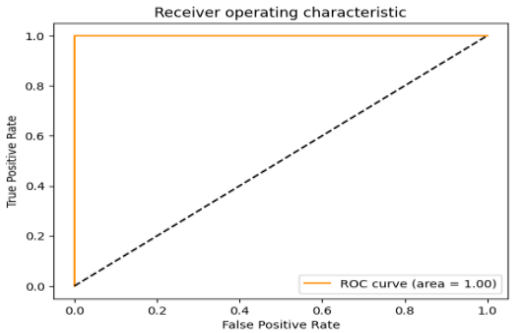


Figure 15. ROC Curve of a simple RNN for Malware detection

In Table 4, we documented the performance of all approaches, recording parameters such as precision, f1-score, and recall. Our study emphasizes the success of the RNN-LSTM model, achieving a high AUC (Area Under the Curve) score of 0.99. This indicates strong performance in binary classification tasks, supported by Figure 14, which demonstrates the model's effectiveness in accurately classifying positive and negative samples. These results underscore the RNN-LSTM model's excellent discriminatory ability and precision in classification, reaffirming its effectiveness in providing accurate predictions.

However, the focus of this study was to improve the performance of MDS. Despite the high performance achieved by RNN-LSTM, we further explored by attempting to train RNN without LSTM. As illustrated in Figure 15, the highest accuracy reached was 99.99% using a basic RNN, with precision scoring 1, recall and F1-score both scoring 0.99, and achieving an AUC of 1. These results reflect ideal discriminatory ability and maximum precision in classification.

Table 5 and Table 6 showcase various findings from studies conducted within the same framework. We have successfully achieved our objective, as our models have outperformed the performance metrics of the comparative models.

6. Conclusions

In our study aimed at enhancing intrusion detection and preventing cyber-attacks, we integrated deep learning into various methodologies. Inspired by prior research, we implemented different models: Artificial Neural Networks (ANNs), Recurrent Neural Networks with Long Short-Term Memory (RNN-LSTM), and hybrid Convolutional Neural Networks with LSTM (CNN-LSTM). Using the NSL-KDD and Malware Dataset, our objective was to achieve accurate detection and promptly uncover attacks using a DL-based system. Remarkably, we found that the RNN-LSTM system exhibited superior performance, achieving an accuracy of 99.99% for IDS. For malware detection, all DL approaches surpassed 99% accuracy, with ANN showing particularly notable improvement.

REFERENCES

[1] H.S. Iqbal, Machine Learning: Algorithms, Real-World Applications and Research. SN Computer Science, Vol. 2, No. 60, pp. 1-2, 2021.

[2] S. Mahdavifar and A. Ghorban, Application of deep learning to cybersecurity: A survey, Neurocomputing, pp. 149-176, 2019.

[3] D. Sumeet and D. Xian, Data mining and machine learning in cybersecurity. Auerbach Publications, 2016.

[4] M.Z. Alom, V. Bontupalli and T.M Taha, Intrusion detection using deep belief networks, National Aerospace and Electronics Conference (NAECON), pp.339–344, 2015.

[5] M. Macas and C. Wu, Review: Deep Learning Methods for Cybersecurity and Intrusion Detection Systems, IEEE LATINCOM, 2020.

[6] T. A Tang, L. Mhamdi, D. McLernon, A. Zaidi, S., Ghogho and A. Mounir, 2016, "Deep learning approach for network intrusion detection in software defined networking", International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258–263. IEEE.

[7] K. Grosse., N. Papernot, P. Manoharan, M. Backes and P. McDaniel, Adversarial examples for malware detection, S.N. Foley, D. Gollmann, E. Snekkenes (Eds.), Computer Security – ESORICS 2017, Springer, Springer International Publishing, Cham, pp. 62-79, 2017.

[8] A.E. Cil, K. Yildiz and A. Buldu, Detection of ddos attacks with feed forw ard based deep neural network model, Expert Syst. Appl., Vol.169, pp. 114520, 2021. https://doi.org/10. 1016/j.eswa.2020.114520.

[9] V. Hussain and J. Hnamte, Deep learning based intrusion detection system: software defined network, 2021 Asian Conference on Innovation in Technology (ASIANCON), IEEE, pp. 1-6, 2021.

[10] C.S Wu and S. Chen, A heuristic intrusion detection approach using deep learning model, International Conference on Information Networking (ICOIN), pp. 438-442, 2023.

[11] V. Hnamte and J. Hussain, Dependable intrusion detection system using deep convolutional neural network: A Novel framework and performance evaluation approach, Telematics and Informatics Reports, Vol.11, pp.100077, 2023. https://doi. org/10.1016/j.teler.2023.100077.

[12] R. Vinayakumar, K.P. Soman and P Poornachandran, 2017, "Applying convolutional neural network for network intrusion detection", International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13-16 September 2017, DOI: 10.1109/ICACCI.2017.8126009.

[13] Kim Y., 2014, "Convolutional neural networks for sentence classification", Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, pp.1746-1751, arXiv: 1408.5882, 2014. https://doi.org/ 10.3115/v1/D14-1181.

[14] K. Wu, Z. Chen and A.W .Li, A novel intrusion detection model for a massive network using convolutional neural networks". IEEE Access, Vol. 6, pp.50850 – 50859, 2018.

[15] Y. Xiao, C. Xing, T. Zhang and Z. Zhao, An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks, IEEE Access, Vol.7, pp. 42210 – 42219, 2019. DOI: 10.1109/ACCESS.2019.2904620.

[16] M. Zhu, K. Ye and C.Z. Xu, Network anomaly detection and identification based on deep learning methods, CLOUD 2018: 11th International Conference, Held as Part of the Services Conference Federation, pp. 219–234, USA, 2018. https://doi. org/10.1007/978-3-319-94295-7_15.

[17] M.S. ElSayed, N.A .Le Khac, M.A. Albahar and A. Jurcut, A novel hybrid model for intrusion detection systems in sdns based on cnn and a new regularization technique, J. Netw. Comput. Appl., Vol.191, pp. 103160, 2021.

[18] C. Xu, J. Shen, X. Du and F. Zhang, An Intrusion Detection System Using a D eep Neural Network With Gated Recurrent Units. IEEE Access, Vol.6, pp.48697-48707, 2018.

[19] S. Forrquue Ahmed, M. S. Bin Alam, M. Hassan, M.R. Rozbu, T. Ishtiak, N. Rafa, M. Mofijur, A.B.M Shawkat Ali and A.H. Gandomi, Deep learning modelling techniques: current progress, applications, advantages, and challenge, Artificial Intelligence Review, Vol.56, pages 13521–13617, 2023.

[20] K. Danqing, Y., Lv and C. Yuan-yuan, Short-term traffic flow prediction with LSTM recurrent neural network, IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 1-6, 2017.

[21] Y. LeCun, P. Haffner, L. Bottou and Y. Bengio, Object recognition with Gradient-Based learning, Lecture Notes in Computer Science, (1681), pp 319–345, 1999.

[22] C.M. Hsu, M.Z. Azhari, H.Y. Hsieh, S.W. Prakosa and J.. S. Leu, Robust Network Intrusion Detection Scheme Using Long-Short Term Memory Based Convolutional Neural Networks, Mobile Networks and Applications, (26), pp. 1137–1144, 2021. https://doi.org/10.1007/s11036-020-01623-2.

[23] S, Hochreiter and J. Schmidhuber, Long short-term memory. Neural Comput., Vol.9, No.8, pp.1735–80, 1997.

[24] F. Weijiang and G. Naiyang, 2017, "Audio visual speech recognition with multimodal recurrent neural networks". International Joint Conference on Neural Networks (IJCNN), Changsha, Hunan, P.R. China, pp. 4-8.

[25] S. Revathi and A. Malathi, A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection International. Journal of Engineering Research & Technology (IJERT), Vol.2, No.12, pp. 1848-1852, 2013.

[26] R. Bala and R. Nagpal, A review on KDD CUP99 AND NSL-KDD dataset , International Journal of Advanced Research in Computer Science, Udaipur, Vol.10, No.2, pp. 64-67, 2019. DOI:10.26483/ijarcs.v10i2.6395.

[27] R.A.R. Mahmood, A.,H. Abdi and M. Hussin, Performance Evaluation of Intrusion Detection System using Selected Features and Machine Learning Classifiers, Baghdad Science Journal, Vol.18, No.2, P-ISSN: 2078-8665, 2021. DOI: http://dx.doi.org/10.21123/bsj.2021.18.2(Suppl.).0884.

[28] A Javaid, Q., Niyaz, W. Sun and M. Alam, A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), pp. 21-26, 2016.

[29] M. Tavallaee, E. Bagheri, W. Lu and A.A. Ghorbani, 2009, A "Detailed Analysis of the KDD CUP 99 DataSet". Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009, pp. 1-6, Ottawa, 8-10 July.

[30] I. Bhupendra and A. Yadav, Performance analysis of NSL-KDD dataset using ANN, Communication. Engneering. System IEEE pp. 92–96, 2015.

[31] G. Sandeep, K. G. Mirnal and S. Aroj., Deep learning approach onnetwork intrusion detection system using nsl-kdd dataset, International Journal of Computer Network and Information Security (IJCNIS), Vol.11, No.3, pp. 8–14, 2018.

[32] C.M. Hsu, Y. Hsieh and S.W. Parakosa, Using Long-Short-Term Memory Based Convolutional Neural Networks for Network Intrusion Detection: 11th EAI International Conference, WiCON 2018, Taipei, Taiwan, 2019.

[33] P.S. Muhuri, P. Chatterjee, X. Yuan, K. Roy and A. Esterline, Using a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) to Classify Network Attacks, Information, Vol.11, No.5, pp. 243, 2020.

[34] C. Hwang, J.. Hwang, J. Kwak and T. Lee, Platform-independent malware analysis applicable to Windows and Linux environments. Electronics 9, 5, 793, 2020.

[35] A. Yazdinejad, H. Haddad Pajouh, A. Dehghantanha, R.M Parizi, G. Srivastava and M.Y Chen., Cryptocurrency malware hunting: A deep Recurrent Neural Network approach, Applied Soft Computing, 96, 2020.

[36] D. Dang, F. Di Troia and M. Stamp, 2021, "Malware classification using long short-term memory models", ICISSP 2021 - Proceedings of the 7th International Conference on Information Systems Security and Privacy, pp.743-752.

[37] K.S. Satheesh and T. Ciza, 2021, "MemDroid - LSTM Based Malware Detection Framework for Android Devices", 2021 IEEE Pune Section International Conference (PuneCon), Pune, India, 2021.

[38] Y. Ban, S. Lee, D. Song, H. Cho and .L.H. Yi, "FAM: Featuring Android Malware for Deep Learning-Based Familial Analysis", IEEE Access, 10, pp. 20008- 20018, 2022.

[39] M.S. Akhtar and T. Feng, Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time, Symmetry, Vol.14, No.11, pp. 2308, 2022.

[40] N.K. Gyamfi, N. Goranin, D. Ceponis and H.A. Cenys., Malware Detection Using Convolutional Neural Network, A Deep Learning Framework: Comparative Analysis, Journal of internet services and information security.. Innovative Information Science & Technology Research Group (ISYOU), Vol.12, No.4, pp. 102-115, 2022.

[41] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, H. Janicke, RDTIDS: Rules and Decision Tree-Based Intrusion Detection System for Internet-of-Things Networks, Future Internet, Vol. 12, No. 44, pp. 1- 14, 2020.

[42] N. Kunhare, R. Tiwari and J. Dhar, Particle swarm optimization and feature selection for intrusion detection system, Sadhana, Vol. 45, No. 109, pp. 1-14, 2020.