Journal of Theoretical and Applied Information Technology
15th October 2024. Vol.102. No. 19
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COMPARATIVE ANALYSIS AND HOW EFFICIENT DEEP LEARNING METHODS OF MALWARE DETECTION

FIRAS SHIHAB AHMED [1], NORWATI MUSTAPHA [2], NOR FAZLIDA MOHD SANI HEAD[3], RAIHANI MOHAMED[4]

[1]Faculty of Computer Science and Information Technology, University Putra Malaysia, Selangor, Malaysia

[2]Associate Professor, Department of Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia, Malaysia

[3]Associate Professor, Department of Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia, Malaysia

[4]Senior Lecturer, Department of Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia, Malaysia

E-mail: [1]firasshahab48@gmail.com , [2]norwati@upm.edu.my, [3] fazlida@upm.edu.my, [4] raihanimohamed@upm.edu.my

ID 55351 Submission	Editorial Screening	Conditional Acceptance	Final Revision Acceptance
16-08-2024	17-08-2024	11-09-2024	30-09-2024

ABSTRACT

Due to the massive interconnectivity among Internet devices in the Internet of Things (IoT), this led to security challenges in confronting attacks by malware. Detecting malware attacks in the IoT environment is considered a crucial matter that constitutes a challenge for researchers to contribute an accurate method to build a protection system capable of providing security for existing applications in the IoT environment. Today, most of the current research explores deep-learning methods for malware detection. This paper presents an approach that includes analysis to compare the performance of deep learning methods based on opcode in detecting malware in IoT. Four deep learning methods which include Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and Gated Recurrent Unit (GRU) are evaluated and compared for accuracy, precision, recall, and F-measure. The idea of this study is based on pre-processing and feature selection by identifying outlier values inside opcodes using the Interquartile range (IQR) technique. Then, the Recursive Feature Elimination (RFE) method has been applied to determine the important features and the suitable hyperparameters to reduce memory space. There are two data sets used in this study to evaluate the performance of the deep learning methods. The first dataset is generated by an IoT-based application with two classes which is considered smaller size than the second dataset which comprises nine different classes. The experimental results showed that the performance of the LSTM method outperformed compared to the other methods which were based on methods for measuring performance and reliability such as accuracy, precision, recall, and F-measure for both data sets. Moreover, used result of receiver operating characteristic (ROC) curves and precision-recall (PR) curves confirm that LSTM is the best method to detect malware. These results will be used as reference results to address the weaknesses of each deep learning method.

Keywords: *Malware Detection, Deep Learning, , AI Methods, Efficiency*

Journal of Theoretical and Applied Information Technology
15th October 2024. Vol.102. No. 19
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1. INTRODUCTION

The Internet of Things (IoT) is defined as a large network of things associated with each other through different sensing devices in a vast range of applications [1, 2] Due to its continuous services over the Internet, IoT devices generate a tremendous amount of different sensory data over time, this led to security challenges in confronting attacks by malware [3, 4]. Therefore malware detection is a fundamental matter in IoT-based applications. The main issue in detecting malware is ineffective when using the signature method to identify code that is suspected of security changes [5]. For this reason, many researchers have explored and proposed many techniques and methods to detect malware in different attack vectors [6].

In the process of detecting malware, many relied on traditional machine learning methods to find appropriate solutions, such as [7-9], this research focused on discovering this malware that is based on portable executable files (PE) that run on operating systems as the Windows operating system was widely used. Machine learning offers various methods and models that aid in providing adequate protection systems among IoT-based Moreover, machine learning methods require additional time and effort to extract the basic features for detecting malware.

Currently, deep learning methods have been relied upon to solve many problems such as extracting important features and selecting features with minimal feature extraction efforts, in addition to their ability to deal with the high dimensions of the dataset [10]. However, the problem lies in working with high-dimensional or very large data, because the data has not been processed sufficiently for deep learning methods to be able to detect malware more efficiently [11, 12].

2. PROBLEM STATEMENT

One of the most important problems and challenges facing researchers in building a malware detection model is the suitability of the detection model designed using one of the deep learning methods and its suitability to the size and type of data used, as there are no fixed criteria for choosing one of the deep learning methods to detect malware except through trial and error to determine the suitability of the detection model. By identifying the weaknesses of each method and showing the extent of its impact and improvement in performance after data processing, this study clarified the extent of the response of each method through the performance measurement tools that were explained in Tables (2-7). Therefore, this study came to determine some criteria through an analytical study of these methods after using two different groups in terms of size and work environment and the extent to which the accuracy rate of these methods is affected by the stages of data processing before using them. Therefore, this study is considered a contributing factor in determining the suitability of the deep learning methods model to detect malware and what should be taken into account in determining the appropriate method.

Because there is a gap in providing adequate protection for IoT applications, this research provides a comparative analysis of deep learning methods for detecting malicious software based on opcode sequences in the IoT environment-based applications and in the Windows environment. The analysis will compare four deep learning methods, which are RNN, LSTM, CNN, and GRU. The analysis is imperative to serve as benchmark performance for all malware detection methods focusing on opcodes. The rest of this paper is organized as follows. Section 2 reviews the related works on malware detection based on deep learning methods. Section 3 presents the supervised learning methodology used to perform malware detection along with the dataset evaluation metrics. Section 4 discusses the experimental results and finally, Section 5 concludes the paper.

3. RELATED WORKS

The diversity of malware has made the process of detecting and controlling it extremely difficult. The term malware represents many types of malicious software, for example, ransomware, viruses, spyware, and many other types. The literature has shown an increase of deep learning methods being explored in malware detection.

Research by [13] provides a general and comprehensive overview of malware that runs on the Android system and gives an idea of the most common types of malware. The LSTM model was used by examining the structure of system actions, calls, or other active communication (API), which were created from Android applications, the LSTM algorithm is well suited for modelling sequential information, making it a potential tool for identifying malware that reveals subtle communication patterns over time. The performance results achieved were 96.65% accuracy, 93.04% precision, 96.53% recall, and 94.07% F1 score.

The LSTM algorithms were also proposed by [14] together with the Markov model to detect malware in the IoT by identifying anomalies in the network which is not practically possible for simple edge-based computing devices. The idea of removing outlier values before inputting them into the proposed model results in 92.48% accuracy.

[15] Address of outliers and noise in the data set for malware in the Internet of Things environment (IoT). A new approach was proposed that combines the cornetropy model and the deep learning model CNN to deal with a complex data set due to outliers and noise of the features used in the detection model since CNN is skilled in learning hierarchical features from big data.

Similar work by [16] proposed a multi-channel CNN algorithm to detect malware in IoT with the other two deep-learning algorithms LSTM, and RNN to discriminate against extracted system calls and opcode sequences for dwarf files. The work of the CNN algorithm is based on two channels connected in parallel, whereby each channel takes a sequence of the opcodes as input while the other channel works with system calls. The main motivation of the work is to deal with a larger number of features at the same time using a single model. To achieve this type of model only by implementing a multi-channel architecture based on deep learning algorithms. The results of this research have shown that CNN outperforms the remaining considered techniques by achieving a high accuracy of 99.8%.

Another study to detect new and unknown malware through detecting network anomalies is [17]. It is not enough to compare current anomalies with the expected normal range because most current methods have low rates of detecting new or unknown types of attacks. Therefore, this study proposed a model that predicts the parameter values of network sensors and control units in systems by integrating a one-dimensional convolutional neural network (1D_CNN) and GRU to identify the temporal and spatial correlation for the sensors, and control units. The method is based on calculating deviation and statistical analysis to achieve anomaly detection in control systems. The results have shown precision and recall of this method are 99% and 85% respectively with an average F1 score is 91%.

Botnet attacks in the IoT environment have raised major security concerns. Therefore, [18] presents an approach to detect malware of botnet attacks by selecting features within edge environments. Harnessing Chi-square analyses, and Redundant Feature Elimination (RFE) are some techniques used strategically to find meaningful subsets of features. GRU and machine learning models were used to evaluate 19 classifiers. Preliminary results confirmed the potential of the Gated recurrent unit (GRU) model, especially when coupled with intrinsic feature selection based on Lasso method.

[19] Presents a new approach to detecting malware in the Internet of Things (IoT) environment by using deep-world methods such as LSTM and CNN. The proposed approach consists of three steps. In the first step, the data set is pre-processed using scaling, normalization, and noise removal. In the second step, features are identified using a single fast encoder followed by an ensemble classifier based on LSTM and CNN to detect malware and finally evaluate the results. The results showed that the proposed methods outperformed the benchmark methods on standard data sets with an average accuracy of 99.5%.

4. MATERIALS AND METHOD

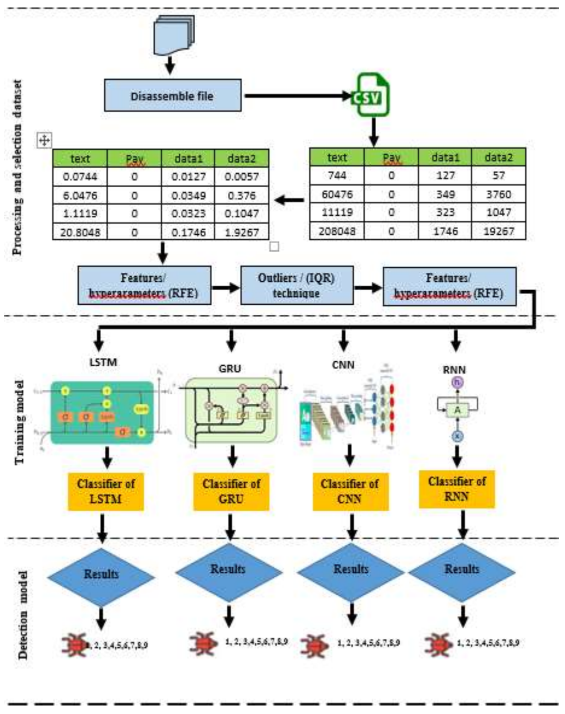
The malware classification process based on sequence opcode features with deep learning as shown in Figure 1. The process of classifying malware will be carried out on two different opcode datasets whereby the first dataset will be classified into two class labels and the second dataset will be classified into nine different types of malwares there are sub-sections will be detail out the datasets, pre-processing and feature selection, model validation, algorithms, and evaluation metrics.

4.1 Dataset

This study focuses on deep learning methods to detect malware in the Internet of Things (IoT) environment and in the Windows system environment. For the purpose, there are two datasets used in experiments. The first dataset is generated by IoT-based application that comprises 552 malware samples with two different classes. With the Raspberry Pie II, it is worth noting that AMD processors have been widely used in cloud-edge devices, thus qualifying the Raspberry Pi II as a IoT cloud-edge device. The dataset used in this research was obtained from the Debian Linux package repositories from https://pkgs.org/. The second dataset is provided by Microsoft in a Windows environment which was obtained from https://www.kaggle.com/competitions/malware-Classification/data?select=trainLabels.csv. It is the malicious code families which divided in nine categories with 10,869 malware samples.

4.2 Pre-Processing and Feature Selection

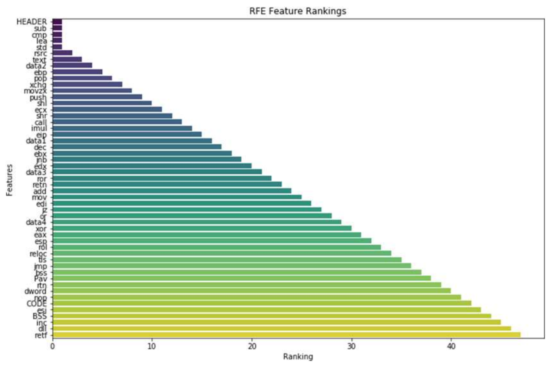
In creating a dataset that can be used as input to deep learning algorithms, the Object-Dumb tool was used to decompile all samples to extract Opcode sequences in each sample. After that, the Opcode sequences will be processed with various pre-processing steps that include normalizing, centering, and scaling.



text	Pay	data1	data2
744	0	127	57
60476	0	349	3760
11119	0	323	1047
208048	0	1746	19267

text	Pay	data1	data2
0.0744	0	0.0127	0.0057
6.0476	0	0.0349	0.376
1.1119	0	0.0323	0.1047
20.8048	0	0.1746	1.9267

Figure 1: Malware Classification Process



(a) First dataset

The Python code was used to convert this sequence file into an Excel file before splitting it into a training and testing set. In addition, other two techniques have been used which are Recursive Feature Elimination (RFE) and Interquartile range (IQR) techniques. The RFE as a feature selection method is used to create a subgroup of the dataset that contains themost important features to prevent an overfitting issue during processing and build a model capable of classification with high accuracy. It is based on determining the weights of the features in the data set, which reflect the importance of each feature in the group. Figure 2 represents the process of RFE in this study that deals with a dataset in the Internet of Things environment and the Windows system, which contains large and high-dimensional and numerous outliers that affect the performance of the model for malware classification. Figure 3 shows the ranking features of the first dataset and the second set.

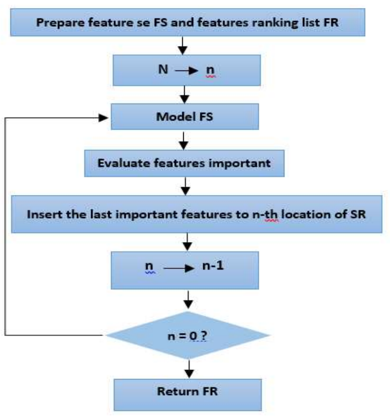
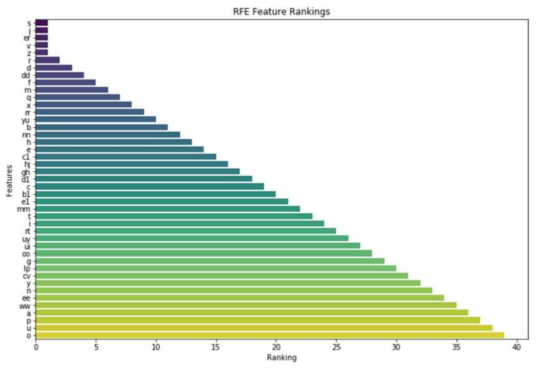

Prepare feature se FS and features ranking list FR
N → n
Model FS
Evaluate features important
Insert the last important features to n-th location of SR
n → n-1
n = 0 ?
Return FR

Figure 2: The process of Recursive Feature Elimination (RFE)



(b) Second dataset

Figure 3: Ranking features of (a) first dataset (b) second dataset

QR has been applied to identify outlier values inside features (opcode). It measures the statistical dispersion of the data values as a measure of overall distribution. IQR is equivalent to the difference between the first quartile (Q1) i.e. 25% and the third quartile (Q3) i.e. 75% respectively. Outliers' data are measured by measuring the lower and upper limits, meaning any data point that is below the lower limit or outside the upper limit is considered extreme as shown in Figure 4.

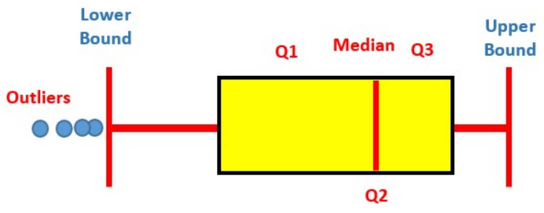
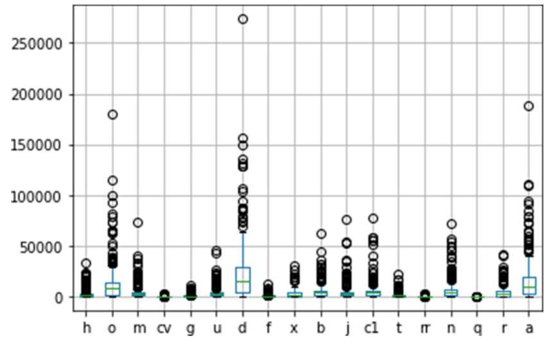
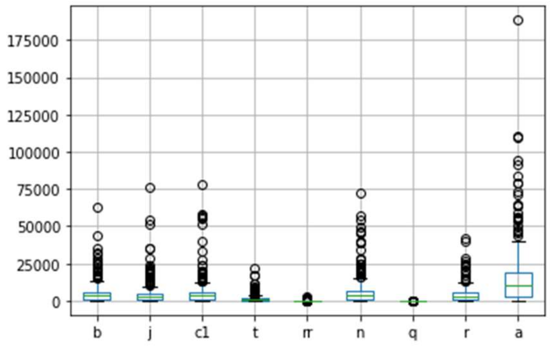


Figure 4: The Interquartile Range (IQR)

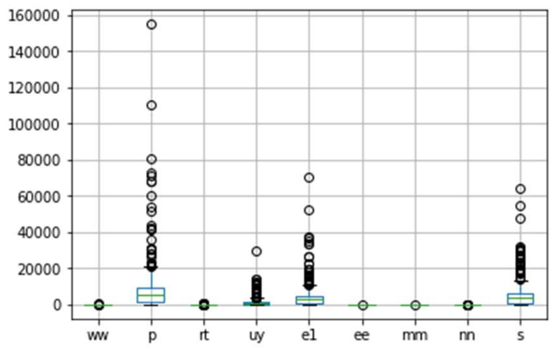
Outliers present in both datasets can be visualized using Boxplots as shown in Figure 5 for the first and second datasets.



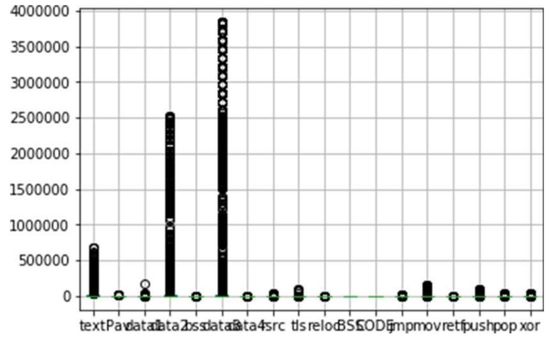
(a) First dataset



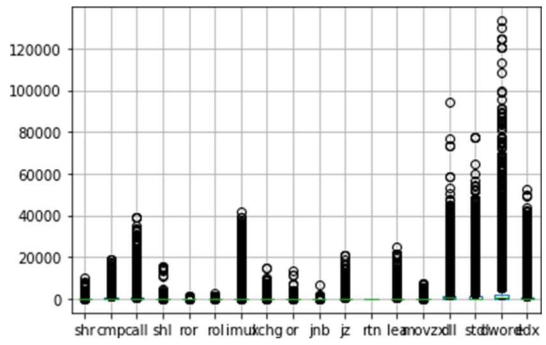
(b) First dataset (cont'd)



(c) Frist dataset (cont'd)



(d) Second dataset



(e) Second dataset (cont'd)

Figure 5: Boxplot of outliers in the first dataset (a-c) and second dataset (d-e)

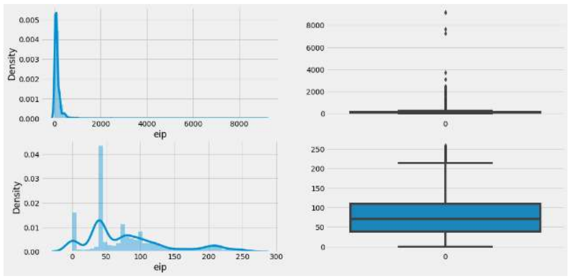
Figure 6 shows an example of removing the outliers from one of the feature ***eip*** by setting the threshold to detect outliers using the filtering conditions above the limits.

4.3 Model Validation

The cross-validation k-fold is a method for performance validating of the deep learning methods, where the dataset is split into k-fold each iteration using one fold as testing data and the remaining folds as training data [20] As a result, the procedure is repeated until every dataset has been assessed. The results are typically repeated along

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

with the values' mean scores. In this study, malware classification was performed based on the cross-validation (k-fold) method for training and testing as shown this Figure 7, where a 10-fold cross-validation was set up. Here, using one fold for testing, and the rest of the fold nine of data are used for training, and the process is repeated until it reaches 10 fold.



```
count    10868.000000
mean       121.681450
std        235.653595
min          0.000000
25%         39.000000
50%         76.000000
75%        127.000000
max       9139.000000
Name: eip, type: float64
```

The placement marks column:
75th quartile: 127.0
25th quartile: 39.0
IQR of (eip) = 88.0

Figure 6: Example of removing outliers from feature ***eip***

As mentioned above, the features that have high degrees of importance in the datasets were selected by applying the RFE technique and then determining outliers and removing them by IQR technique. This process was performed on the datasets before applying deep learning algorithms to detect malware.

4.4 Algorithms

Four deep learning algorithms are used in the comparative experiments, which are Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Convolutional Neural Network (CNN). All algorithms were implemented using the Anaconda Navigator, TensorFlow, Scikit-learn: Machine learning, Jupyter Notebook, and tools in Python.

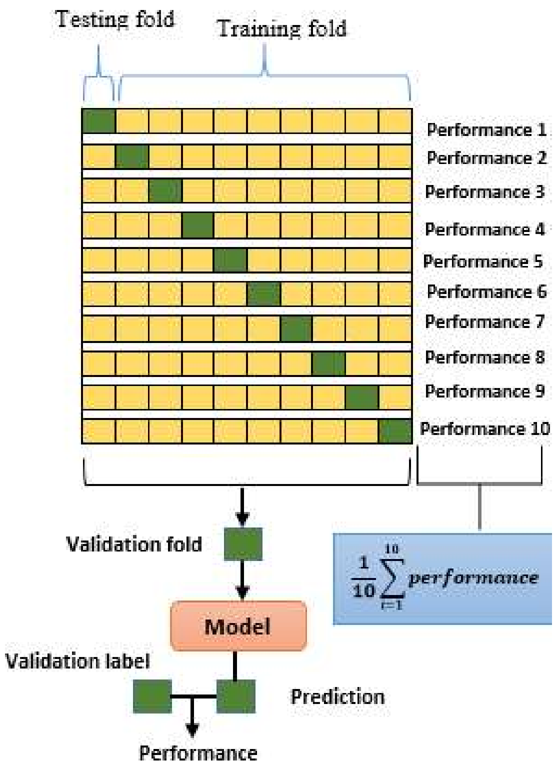


Figure 7: Validation methodology

4.4.1 Recurrent Neural Networks (RNN)

Recurrent Neural Networks RNN is an extension of a neural network with feed-forward and is called recurrent because it performs the same task with each element of the sequence while relying on the output of previous calculations. There is also another way to Recurrent Neural Networks RNN it has a memory that obtains information about what has been calculated so far. The algorithm is powerful in modelling sequences through the presence of periodic connections [21].

It use *X = (x_1,x_(2……….,),x_T)* to represent the, input vector sequence. The hidden vector sequence *H = (h_1,h_(2……….,),h_T)* and output vector sequence *Y= (y_1,y_(2……….,),y_T)* are calculated with t belongs to *[1,T]* as shown in Equations 1 and 2:

$$h_{t} = \sigma(W_{xh}\, X_t + W_{hh}\, h_{t-1} + b_h) \quad (1)$$

$$y_{t} = W_{hk}\, h_t + b_y \quad (2)$$

Where W and b is the weight matrix and bias term, X_t is the input vector at time t, h_{t-1} is the state at time t-1, is a nonlinearity activation function as shown in Figure 8.

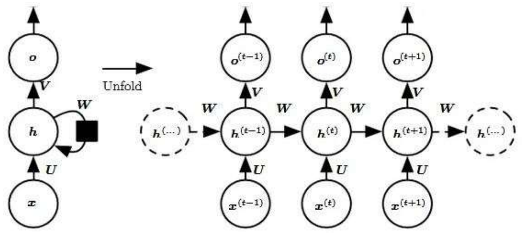


Figure 8: Recurrent Neural Networks RNN [21]

4.4.2 Long Short-Term Memory (LSTM)

The goal for discovering the LSTM algorithm in 1997 is to solve the vanishing gradient problem in RNN algorithm. So the structure of the LSTM algorithm was built from three gates to address this problem: the input gate, the forgetting gate, and the output gate.

The value cell task is to remember random periods and also organize the three gates for the flow of information into and out of the cell and trace the relationships between elements of the input sequence as a new value flows into the cell. As for the forget gate, it controls the extent to which values are kept in the cell. The process of storing information takes place through cells, and memory is processed through gates as shown in Figure 9 [21, 22].

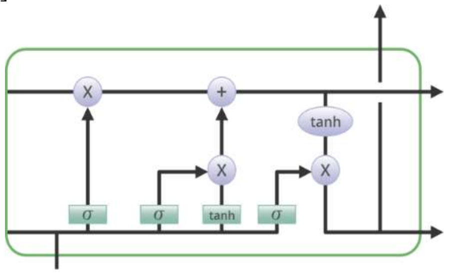


Figure 9: Long Short-Term Memory (LSTM)

Forget Gate

The task of the forgetting gate is to identify information that is no longer useful in the process of training the model. It is done by feeding the gate with inputs x_t (input at a given time) and h_{t-1} (output of the previous cell) by multiplying them using the weight matrices, and then the bias b_f is added. Through the activation function σ, the result is passed to give a binary output. If the cell state is 0, the information will be forgotten, but if the value is 1, the information will be kept for use in the future, the forget equation as shown in Equation 3:

$$f_{t=}\ \sigma(\,W_f\,[h_{t-1,}x_t] +\ b_f) \qquad (3)$$

Where:

- W_f represents the weight matrix associated with the forget gate.
- $W_f\,[h_{t-1,}x_t]$ denotes the concatenation of the current input and the previous hidden state.
- b_f is the bias with the forget gate.
- σ is the sigmoid activation function.

Input Gate

The information is organized and the values that will be remembered are filtered similarly to the forget gate using the inputs h_{t-1} and x_t. After that, a vector will be created using the tanh function, which gives outputs from -1 to +1, containing all possible values h_{t-1} and x_t. Finally, the vector and regularized values are multiplied to obtain useful information. The input gate equation is as shown in Equations 4 and 5:

$$i_t\ =\ \sigma\,(\,W_i\,[\,h_{t-1}\,, x_t\,] +\ b_i) \qquad (4)$$
$$\hat{C}_t\ =\ tanh\,(\,W_c\,[\,h_{t-1}\,, x_t\,] +\ b_c) \qquad (5)$$

The previous state is multiplied by $f_t\ \hat{C}_{t-1}$ ignoring the previously selected information and direction, and then inserted $i_t\,\hat{C}_t$ where this represents the updated candidate values adjusted for the amount we chose to update each state value as shown in Equation 6:

$$\hat{C}_t\ =\ \ f_t\ C_{t-1}\ +\ i_t\ \hat{C}_t \qquad (6)$$

Output Gate

The task of the output gate is to extract information from the state of the cell that is useful in the training process to present it as output to this gate. The vector is created by applying the tanh function to the cell. After that, the sigmoid function organizes the information and filters it for the values that should be used (h_{t-1} and x_t). In the final stage, the vector values and the structured values are multiplied and then they are sent as outputs and inputs to the cell that will be the next. The equation of this gate will be as shown in Equation 7:

$$O_t\ =\ \sigma(W_O\,.\,[h_{t-1}\,, x_t] +\ b_O) \qquad (7)$$

4.4.3 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit GRU algorithm is an updated version of the LSTM algorithm and is a bit more dramatic. In this algorithm, the forget gate and the input gate are combined to become one gate called the update gate. Along with making various modifications, it mixes the hidden state and the cell state. The resulting model, which is clearer than traditional LSTM models, The equations of GRU is shown in Equation 8 to Equation 11:

Update Gate: $z_t = \sigma(W_z * [h_{t-1}, x_t])$ (8)
Reset Gate: $r_t = \sigma(W_r * [h_{t-1}, x_t])$ (9)
Hidden State: $\tilde{h}_t = tanh(W * [r_t \odot h_{t-1}, x_t])$ (10)
Final Hidden State: $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$ (11)

Here, σ represents the sigmoid function, tanh is the hyperbolic tangent function, W_z, W_r, and W are parameter matrices, h_{t-1} is the previous hidden state, x_t is the current input, ⊙ represents element-wise multiplication, and h_t is the current hidden state as shown in Figure 10.

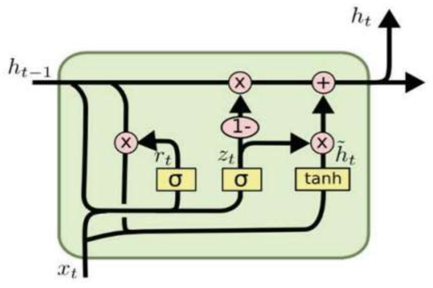


Figure 10: Gated Recurrent Unit (GRU)

4.4.4 The Convolutional Neural Network (CNN)

CNN algorithm is considered a deep learning method that is widely used in analyzing visual images, where the (CNN) structure uses a special technique called Convolution instead of relying only on matrix multiplications like traditional neural networks. This algorithm uses the process of convolution, where it performs a process, which combines two functions to show how one changes the shape of the other. The input (x) for each layer in the CNN model has three dimensions, height, width, and depth, where the height (m) is equal to the width. In addition, the depth is indicated by the channel number. The convolutional layer calculates a dot product between its inputs and the weights, as in the Equation 12, which is similar to NLP, but the inputs are small-scale regions of the initial image volume. The nonlinearity or activation function is then applied to the output of the convolutional layer as shown in Figure 11 [23, 24].

$$h^k = f(W^k * \chi + b^k) \quad (12)$$

Where: W^k=weight, b^k= bias, h^k =maps, k = generating feature

4.5 Evaluation Metrics

The evaluation and percentage metrics are based on the confusion matrix as shown in Table 1, where the rows in the matrix represent instances of the actual class and each column represents instances of the expected class. It is created through the results extracted from the malware classification. The correct predictions are the number of values distributed for each category, taking into account the total expected results after classification. Through this table, TP means that the instance was correctly predicted to be benign and that it was predicted to be benign. TN means that the instance was correctly predicted to be malware and that it was predicted to be malware. FP means that the instance predicted as malware was incorrectly predicted as benign FN means the instance predicted as benign was incorrectly predicted as malware. so through the confusion matrix, we can extract evaluation metrics to give insight into the performance of the detection model, where these metrics determine the accuracy and validity of the training model to detect malware by the following functions[5].

Table 1: Confusion matrix

	Predicted	
Actual	Positive class (Benign)	Negative class (Malware)
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

• ***Accuracy:*** Accuracy is the number of correct predictions from all predictions made. It can be calculated using Equation 13.

$$Accuracy = \frac{TP+}{TP+TN+FP+FN} \quad (13)$$

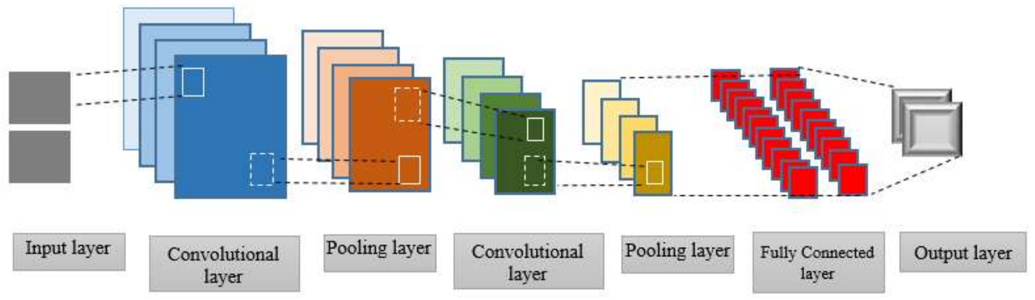


Figure 11: Convolutional Neural Network (CNN)

• ***Precision:*** Precision measures the percentage of expected malware that is correctly classified as malware and the formula is shown in Equation 14.

$$Precision = \frac{TP}{TP+FP} \qquad (14)$$

• ***Recall:*** recall or detection rates is the percentage of malware that was correctly predicted and it can be calculated using Equation 15.

$$Recall = \frac{TP}{TP+FN} \qquad (15)$$

• ***F-Measure:*** is the harmonic mean of precision and recall, and it is considered a very important measure for the success of the detection model when the layers are unbalanced in the information retrieval process, as precision measures the importance of the result, while recall measures the number of truly relevant results that were returned.
The formula as given in Equation 16.

$$F - measure = \frac{2 \times TP}{2 \times TP+FP+FN} \qquad (16)$$

Measuring Precision, recall, and F-measure is necessary because accuracy alone is insufficient and it can be misleading. The confusion matrix is a means of describing the breakdown of errors in the predictions for an unseen dataset. Precision and recall will give the exactness and completeness of the model respectively while the F-measure or F1-score gives the balance [25].

5. RESULTS AND DISCUSSION

The experiments aimed to evaluate and compare the performance of four deep learning algorithms in detecting malware based on opcode features. Those algorithms are Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Convolutional Neural Networks, (CNN), and Recurrent Neural Networks (GRU). The idea of this research is based on pre-processing by removing the outliers with IQR and reducing the memory size of the features from 64-bit to 8-bit by applying the Pandas equations.

In addition, the RFE method is used as feature selection to improve the result significantly. Comparison experiments of different deep learning algorithms against two different datasets with and without pre-processing and feature selection were implemented.

Tables 2 and 3 contain the full results of accuracy, precision, recall, and F-measure for each algorithm under study for the first dataset before and after applying the preprocessing and feature selection step. As noticed from Table 2, LSTM gives better performance compared to other algorithms with 94% accuracy, 0.94 precision, 0.94 recall, and 0.94 F-measure. The modest performance achieved by

GRU, RNN followed by CNN showed the lowest performance. In Table 3, it is noticed that there is an improvement in the performance after applying IQR and RFE methods in all algorithms and LSTM remains the best in terms of accuracy, precision-recall, and F-measure with 97.06%, 0.97, 0.97 and 0.97 respectively. Similarly, concerning the performance of the algorithms against the second dataset, it can be seen there is improvement in the

Observed from the results in the tables, the low performance of the algorithms due to the large size of the training set without selection of important features, along with the presence of outliers led to poor performance of the Algorithms. At the same time, the very poor result shown by RNN is due to the inability of this algorithm to deal with large dataset that have long sequences

Performance of all the methods before and after applying IQR and RFE methods and again LSTM still gives an excellent result as shown in Table 4 and Table 5.

The results in Table 3 and Table 5 are analyzed based on Receiver Operating Characteristic (ROC) curve and Precision-Recall (PR) curve. They played a role in understanding the method of various systems in the presence of uncertainty. The Area under Curve (AUC) will be used as a summary of the model skill. The model skill will be compared with a no-skill classifier. A model with no skill is represented at the point (0.5, 0.5) for first dataset and different values of no skill for the second dataset. Table 6 and Table 7 show the results of AUC for ROC and PR of four deep learning methods across the first and second datasets respectively.

Table 2: Results on first dataset (without IQR and RFE)

Algorithm	Epoch	Batch size	Accuracy	Precision	Recall	F- Measure
RNN	100	65	78.38%	0.80	0.78	0.78
GRU	100	65	88.29%	0.90	0.90	0.88
CNN	100	65	34.34%	0.92	0.34	0.43
LSTM	**100**	**65**	**94%**	**0.94**	**0.94**	**0.94**

Table 3: Results on first dataset (with IQR and RFE)

Algorithm	Epoch	Batch size	Accuracy	Precision	Recall	F- Measure
RNN	100	65	92.16%	0.92	0.92	0.92
GRU	100	65	97.05%	0.98	0.97	0.98
CNN	100	65	36.75%	0.75	0.37	0.49
LSTM	**100**	**65**	**97.06 %**	**0.97**	**0.97**	**0.97**

Table 4: Results on second dataset (without IQR and RFE)

Algorithm	Epoch	Batch size	Accuracy	Precision	Recall	F- Measure
RNN	100	65	13.52%	0.02	0.14	0.03
GRU	100	65	67.57%	0.76	0.68	0.69
CNN	100	65	45.84%	0.53	0.46	0.45
LSTM	**100**	**65**	**90.02%**	**0.92**	**0.83**	**0.85**

Table 5: Results on second dataset (with IQR and RFE)

Algorithm	Epoch	Batch size	Accuracy	Precision	Recall	F- Measure
RNN	100	65	8.02%	0.01	0.09	0.01
GRU	100	65	96.03%	0.96	0.96	0.96
CNN	100	65	90.58%	0.92	0.91	0.91
LSTM	**100**	**65**	**98.26%**	**0.98**	**0.98**	**0.98**

5.1 Receiver Operating Characteristic (ROC) Curve

The Receiver Operating Characteristic (ROC) curve summarizes the trade-off between TP rate and the FP rate for a predictive model that uses different probability thresholds. It has two dimensions whereby the x-axis indicates the False Positive FP rate and the y-axis indicates the True Positive TP rate [26]

Table 7: Area under Curve (AUC) of ROC and PR for second dataset

Algorithms	ROC AUC	PR AUC
Recurrent Neural Networks (RNN)	0.97	0.96
Gated Recurrent Unit (GRU)	0.99	0.99
Convolutional Neural Network (CNN)	0.54	0.52
Long Short-Term Memory (LSTM)	1.00	0.98

Table 6: Area under Curve (AUC) of ROC and PR for first dataset

Algorithms	ROC AUC	PR AUC
Recurrent Neural Networks (RNN)	0.952	0.84
Gated Recurrent Unit (GRU)	0.987	0.99
Convolutional Neural Network (CNN)	0.992	0.58
Long Short-Term Memory (LSTM)	0.997	0.99

5.1.1 Receiver Operating Characteristic (ROC) Curve (first dataset)

Figure 12 shows the ROC curves for RNN, CNN, GRU and LSTM for the first dataset with 0.97, 0.54, 0.99 and 1.00 respectively.

5.1.2 Receiver Operating Characteristic (ROC) Curve (second dataset)

Figure 13 shows the ROC curves of second dataset for RNN is 0.952 with no skill at 0.06, CNN is 0.992 with no skill at 0.50, GRU is 0.987 with no skill at 0.99 and LSTM is 0.997 at no skill 0.99.

5.2 Precision-Recall (PR) Curve

There are many ways to evaluate the classifier, and among these methods are the Precision-Recall curve (PR) and the receiver operating characteristic (ROC) curve. PR It is a graph that expresses the Precision values on the y-axis and the recall values on the x-axis. In other words, the PR curve contains the y-axis and the x-axis.

5.2.1 Precision-Recall (PR) Curve (first dataset)

Figure 14 shows the PR curve for RNN, CNN, GRU and LSTM for first dataset with 0.52, 0.96, 0.99, and 0.98 respectively.

5.2.2 Precision-Recall (PR) Curve (second dataset)

Figure 15 shows the PR curve for RNN, CNN, GRU and LSTM for first dataset with 0.84, 0.58, 0.99, and 0.99 respectively.

5.3 DISCUSSION

Nowadays, there are many methods for malware detection, most notably deep learning methods, but many do not distinguish which is the best and most appropriate for detecting malware. The importance of this research is to identify the weaknesses, strengths, and efficiency of deep learning methods in detecting malware based on the opcode, this research used a set of data of different sizes and working environments.

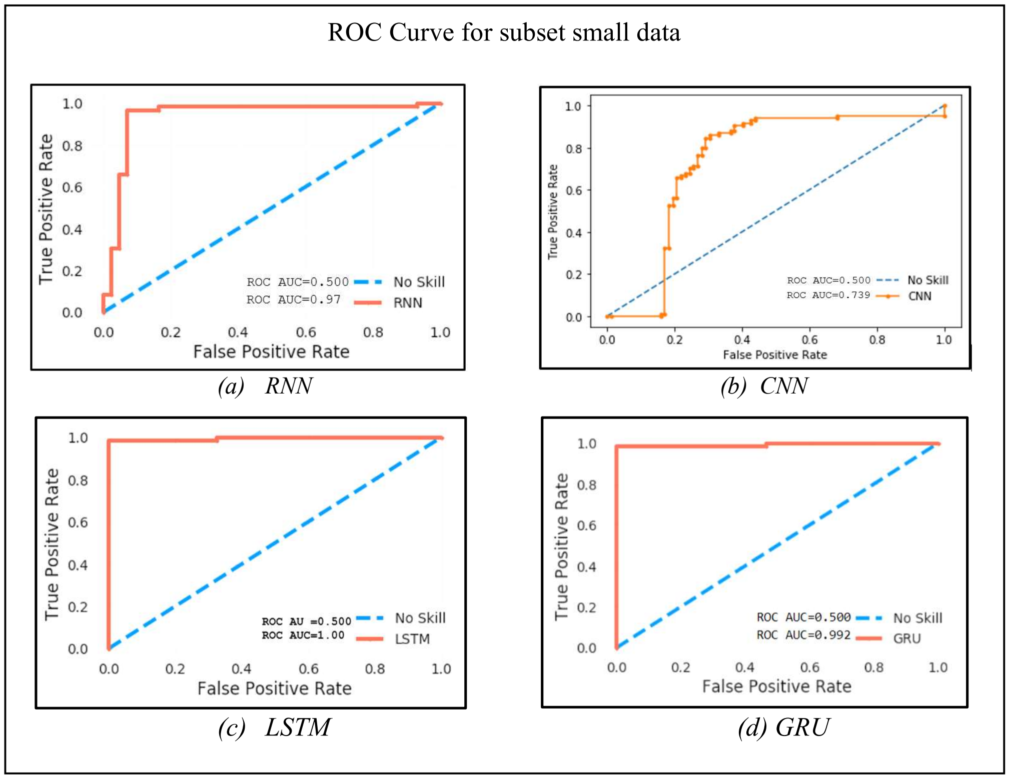


Figure 12: ROC curves for RNN, CNN, GRU, and LSTM (first dataset)

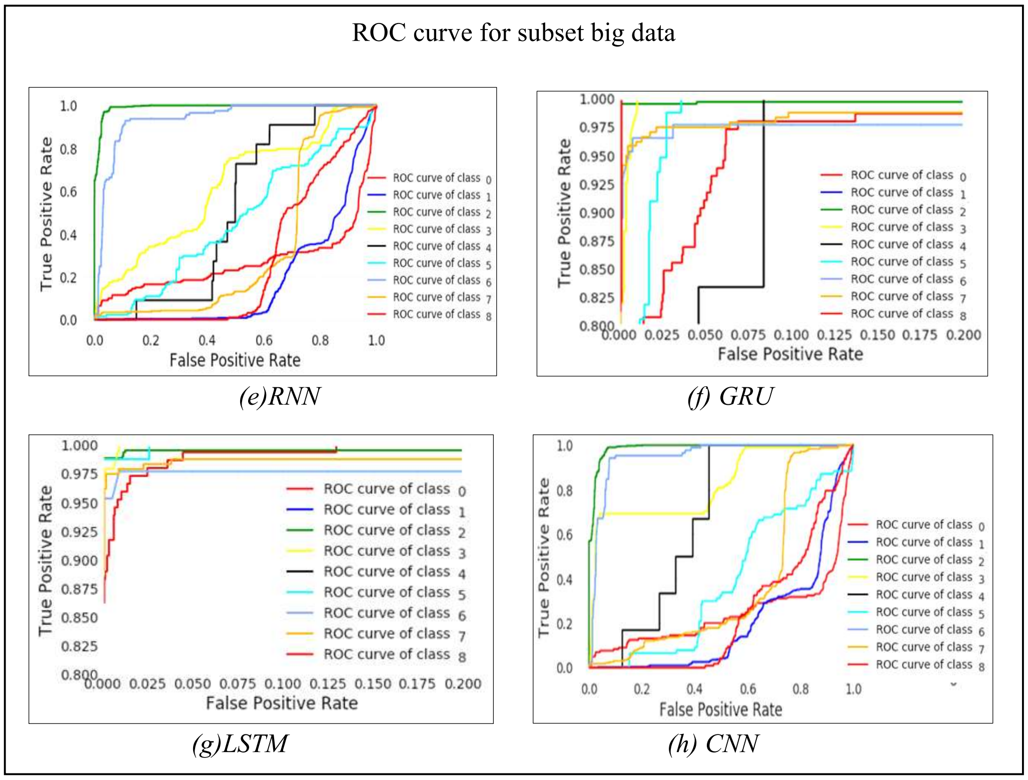
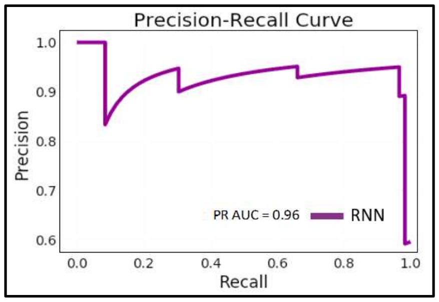
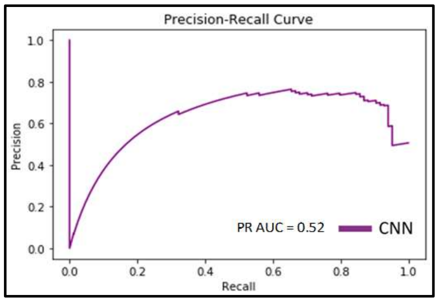


Figure 13: ROC curves for RNN, CNN, GRU, and LSTM (second dataset)

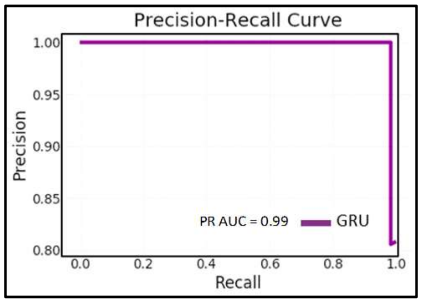
PR curve for subset first dataset



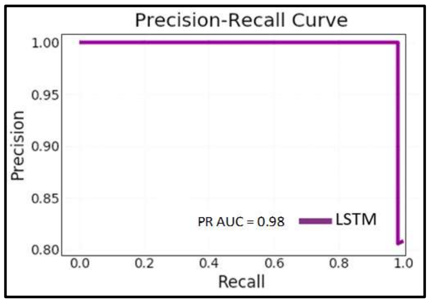
(i) RNN



(j) CNN



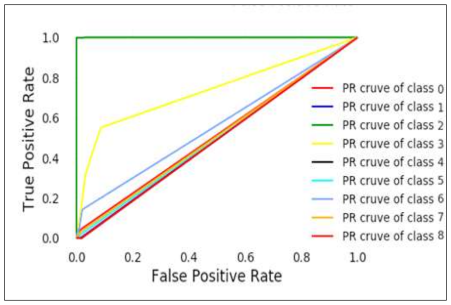
(k) GRU



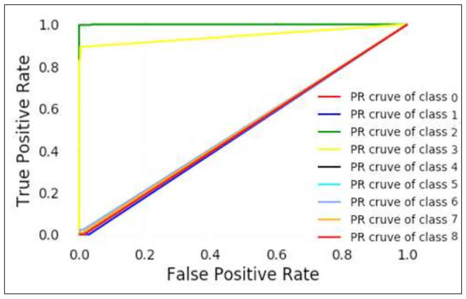
(l) LSTM

Figure 14: PR curves for RNN, CNN, GRU, and LSTM (first dataset)

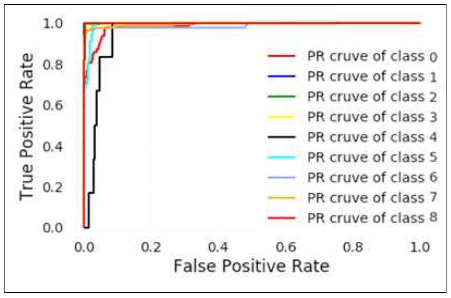
PR curve for sub-second dataset



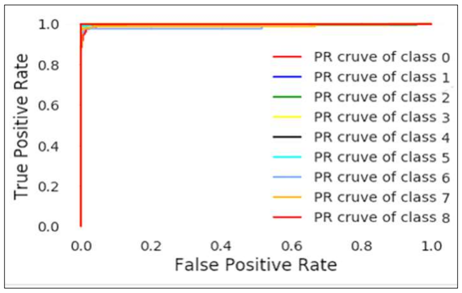
(m) RNN



(n) CNN



(p) GRU



(q) LSTM

Figure 15: PR curves for RNN, CNN, GRU, and LSTM (second dataset)

This study provided an evaluation and analysis of the performance of deep learning methods by using performance measurement tools for each method through using of a small data set and a large data set. The extent to which these algorithms are affected by the size and type of the data set has been proven. It is better to use the (LSTM) algorithm when the data set is large or unbalanced because it is capable of processing large data without falling into overfitting or vanishing gradients, where It relies on three gates, which reflect the high ability to process large data efficiently, through the accuracy and evaluation results shown in Tables (2-7), it becomes clear to us that the (GRU) algorithm is the closest in performance to the (LSTM) algorithm, but it suffers from the inability to unbounded counting of data sets, when the data set is large, as it just uses two, gets to process the data. As for the (RNN) algorithm, its performance was not good, as it originally suffered from the problem of vanishing gradient, especially with large data, and this is what previous research confirmed. As for the (CNN) algorithm, it was the worst among all deep learning methods because it specializes in processing image data more, and according to the results of this study, it is not able to process the data well except after converting the data set into an image format.

As mentioned above, the AUC curve of two curves, the Receiver Operating Characteristic (ROC) curve, and the Precision-Recall (PR) curve have been used in this comparative analysis study. The important point of using the ROC and PR curves together is finding close or shared points to give the best evaluation of the models used in this study as shown in Tables 6 and 7. In addition, AUC and ROC curves will help us to know the feasibility of the classifier's performance of deep learning methods and to evaluate the efficiency of detecting malware. Therefore, ROC measures the performance of the classification model at all classification thresholds and AUC measures the ability of the binary classifier to differentiate between classes and it is used as a summary of the ROC curve. Knowing how the AUC curve works, the higher the AUC, the better the model's performance at distinguishing between positive and negative classes. AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0 or otherwise AUC of 1.0 if predictions are 100% correct.

The performance of the ideal model in detecting malware depends on choosing the appropriate model for the dataset. As the performance measures that we mentioned above, it is possible to determine the appropriate model for the dataset used in this study. Observed from the results, ROC and PR AUC for RNN, GRU, LSTM, and CNN models on the first dataset is about (ROC = 0.952, PR = 0.84), (ROC=0.987, PR=0.99), (ROC=0.997, PR=0.99), and (ROC=0.992, PR=0.58) respectively. Meanwhile, the ROC AUC for the RNN, GRU, LSTM, and CNN models on the second dataset is about (ROC=0.54, PR=52), (ROC=0.99, PR=0.99), (ROC=1.0, PR=0.98), and (ROC=0.97, PR=96) respectively. Therefore, this indicates that the LSTM model is the best model among other deep learning models for malware detection using opcode datasets in the IoT environment (first dataset) as well as for categorizing malware in the Windows environment (second dataset). This is due to the strong structure of this method, which involves the input gate, forget gate, and output gate [27] in solving the problem of the vanishing gradient. LSTM has outperformed the GRU model which suffers from the problem of unbounded counting [28] due to the merging of the forget gate and the input gate into the update gate. As for the CNN model, it is an efficient method for dealing with a set of image data, so its performance was unsatisfactory in processing the opcode sequences dataset.

5.3.1 Justify the criteria using which a conclusion was reached regarding the research problem

The most important criteria of this study, which are related to the statement of the problem, are the suitability of the detection model, the limitations of each of the deep learning methods, and the extent of the response of each method in improving performance after processing the data before using it and the flexibility of each technique by moving from a small data set to a large data set and using the same parameters in terms of repetition, number of layers, and number of nodes in each method.

5.3.2 The major findings when research contribution and comparison to other such solutions presented

This study came to evaluate deep learning methods accurately and individually without working on merging between the methods to detect malware because we noticed that most previous studies were working on merging between deep learning methods to improve the performance of each method, as each method has weaknesses and strengths in dealing with different data in terms of size and type. This study came to analyze the weaknesses and strengths of each method how to use the data before and after processing it and the extent of its impact on the ability to detect malware. Therefore, the goal of this study is to understand the behaviors of deep learning methods how to overcome their weaknesses, and the extent of their suitability for the type and size of data.

6. CONCLUSION

While there are numerous approaches to malware detection—deep learning techniques being one of the most popular—there is no way to differentiate between the most effective and suitable approaches. This study presented an analysis of the performance of deep learning methods to determine the suitability of the malware detection model for the dataset, which employs a set of data with varying sizes and working circumstances to determine the efficacy, drawbacks, and efficiency of deep learning techniques for malware detection based on opcodes. It has been demonstrated how much the size and nature of the data collection influence these algorithms. This paper presented a comparative analysis of four deep learning methods which are LSTM, GRU, CNN, and RNN to detect malware in the Internet of Things environment and the Windows system. Two sets of data were used based on sequences of operating code (opcode). The experimental results showed that the performance of the LSTM method outperformed compared to the other methods under study in terms of accuracy, precision, recall, and F-measure for both datasets. In addition, these results are supported by the analysis of Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves to confirm that LSTM is the best method to detect malware. These results and evaluations will be used as reference results to address the weaknesses of each deep learning method with the same datasets. They can also be used to explore deep learning methods with other data sets from different environments to evaluate deep learning methods.

- The limitations and future work

Not counting the training and testing time of the methods used, where time is an important element in evaluating the efficiency of each method. So The future work of this study is to measure the performance of deep science methods not only through accuracy, but the time factor must be introduced because it is considered very important in the efficiency of the detection model, especially in the large data set, addition to and the number of features used, and to improve the method of selecting features by assigning weights of the importance of features and excluding unimportant and duplicate features.

REFERENCES

[1] M. H. Alsharif *et al.*, "A comprehensive survey of energy-efficient computing to enable sustainable massive IoT networks," *Alexandria Engineering Journal,* vol. 91, pp. 12-29, 2024.

[2] C. Singh and A. K. Jain, "A Comprehensive Survey on DDoS Attacks Detection & Mitigation in SDN-IoT Network," *e-Prime-Advances in Electrical Engineering, Electronics and Energy,* p. 100543, 2024.

[3] Y. Bobde, G. Narayanan, M. Jati, R. S. P. Raj, I. Cvitić, and D. Peraković, "Enhancing Industrial IoT Network Security through Blockchain Integration," *Electronics,* vol. 13, no. 4, p. 687, 2024.

[4] Q. Chen, D. Li, and L. Wang, "Network Security in the Internet of Things (IoT) Era," *Journal of Industrial Engineering and Applied Science,* vol. 2, no. 4, pp. 36-41, 2024.

[5] B. Nawaal, U. Haider, I. U. Khan, and M. Fayaz, "Signature-Based Intrusion Detection System for IoT," in *Cyber Security for Next-Generation Computing Technologies*: CRC Press, 2024, pp. 141-158.

[6] T. Shi, R. A. McCann, Y. Huang, W. Wang, and J. Kong, "Malware Detection for Internet of Things Using One-Class Classification," *Sensors,* vol. 24, no. 13, p. 4122, 2024.

[7] C. Ni and S. C. Li, "Machine learning enabled industrial iot security: Challenges, trends and solutions," *Journal of Industrial Information Integration,* p. 100549, 2024.

[8] P. Jayaraman, K. K. Nagarajan, P. Partheeban, and V. Krishnamurthy, "Critical review on water quality analysis using IoT and machine learning models," *International journal of information management data insights,* vol. 4, no. 1, p. 100210, 2024.

[9] E. Altulaihan, M. A. Almaiah, and A. Aljughaiman, "Anomaly Detection IDS for Detecting DoS Attacks in IoT Networks Based on Machine Learning Algorithms," *Sensors,* vol. 24, no. 2, p. 713, 2024.

[10] Chaganti R, Ravi V, Pham TD. Deep learning based cross architecture internet of things malware detection and classification. Computers & Security. 2022;120:102779.

[11] Nguyen KDT, Tuan TM, Le SH, Viet AP, Ogawa M, Le Minh N, editors. Comparison of three deep learning-based approaches for IoT malware detection. 2018 10th international conference on Knowledge and Systems Engineering (KSE); 2018: IEEE.

[12] Alomari ES, Nuiaa RR, Alyasseri ZAA, Mohammed HJ, Sani NS, Esa MI, et al. Malware detection using deep learning and correlation-based feature selection. Symmetry. 2023;15(1):123.

[13] Kumar M, Singh S, Pilania U, Arora G, Jain M. LSTM-based Approach for Android Malware Detection. Procedia Computer Science. 2023;230:679-87.

[14] Shanmuganathan V.Suresh A. LSTM-Markov based efficient anomaly detection algorithm for IoT environment. Appl Soft Comput. 2023;136(C):11.

[15] Luo X, Li J, Wang W, Gao Y, Zhao W. Towards improving detection performance for malware with a correntropy-based deep learning method. Digital Communications and Networks. 2021;7(4):570-9.

[16] Manoharan S, Sugumaran P, Kumar K, Engineer A. Multichannel based IoT malware detection system using system calls and opcode sequences. Int Arab J Inf Technol. 2022;19(2):261-71.

[17] Xie X, Wang B, Wan T, Tang W. Multivariate abnormal detection for industrial control systems using 1D CNN and GRU. Ieee Access. 2020;8:88348-59.

1] Ahmadi M, Ulyanov D, Semenov S, Trofimov M, Giacinto G. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy; New Orleans, Louisiana, USA: Association for Computing Machinery; 2016. pp. 83–94.

[2] Azmoodeh A, Dehghantanha A, Choo K-KR. Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning. IEEE transactions on sustainable computing. 2018;4(1):88-95.

[3] D'Orazio CJ, Choo K-KR, Yang LT. Data exfiltration from Internet of Things devices: iOS devices as case studies. IEEE Internet of Things Journal. 2016;4(2):524-35.

[4] Watson S, Dehghantanha A. Digital forensics: the missing piece of the internet of things promise. Computer Fraud & Security. 2016;2016(6):5-8.

[5] Ahmed FS, Mustapha N, Mustapha A, Kakavand M, Foozy CFM. Preliminary Analysis of Malware Detection in Opcode Sequences within IoT Environment. Journal of Computer Science. 2020.

[6] Burguera I, Zurutuza U, Nadjm-Tehrani S, editors. Crowdroid: behavior-based malware detection system for android. Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices; 2011.

[7] Rathore MM, Paul A, Hong W-H, Seo H, Awan I, Saeed S. Exploiting IoT and big data analytics: Defining smart digital city using real-time urban data. Sustainable cities and society. 2018;40:600-10.

[8] Vigneswaran RK, Vinayakumar R, Soman K, Poornachandran P, editors. Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. 2018 9th International conference on computing, communication and networking technologies (ICCCNT); 2018: IEEE.

[9] Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Al-Nemrat A, Venkatraman S. Deep learning approach for intelligent intrusion detection system. Ieee Access. 2019;7:41525-50.

[10] Chaganti R, Ravi V, Pham TD. Deep learning based cross architecture internet of things malware detection and classification. Computers & Security. 2022;120:102779.

[11] Nguyen KDT, Tuan TM, Le SH, Viet AP, Ogawa M, Le Minh N, editors. Comparison of three deep learning-based approaches for IoT malware detection. 2018 10th international conference on Knowledge and Systems Engineering (KSE); 2018: IEEE.

[12] Alomari ES, Nuiaa RR, Alyasseri ZAA, Mohammed HJ, Sani NS, Esa MI, et al. Malware detection using deep learning and correlation-based feature selection. Symmetry. 2023;15(1):123.

[13] Kumar M, Singh S, Pilania U, Arora G, Jain M. LSTM-based Approach for Android Malware Detection. Procedia Computer Science. 2023;230:679-87.

[14] Shanmuganathan V.Suresh A. LSTM-Markov based efficient anomaly detection algorithm for IoT environment. Appl Soft Comput. 2023;136(C):11.

[15] Luo X, Li J, Wang W, Gao Y, Zhao W. Towards improving detection performance for malware with a correntropy-based deep learning method. Digital Communications and Networks. 2021;7(4):570-9.

[16] Manoharan S, Sugumaran P, Kumar K, Engineer A. Multichannel based IoT malware detection system using system calls and opcode sequences. Int Arab J Inf Technol. 2022;19(2):261-71.

[17] Xie X, Wang B, Wan T, Tang W. Multivariate abnormal detection for industrial control systems using 1D CNN and GRU. Ieee Access. 2020;8:88348-59.

[18] A., Jyotsna; E. A., Mary Anita. A Novel Paradigm for IoT Security: ResNet-GRU Model Revolutionizes Botnet Attack Detection. International Journal of Advanced Computer Science & Applications. 2023;14(12).

[19] Riaz S, Latif S, Usman SM, Ullah SS, Algarni AD, Yasin A. Malware detection in internet of things (IoT) devices using deep learning. Sensors. 2022;22(23):9305.

[20] Varoquaux G. Cross-validation failure: Small sample sizes lead to large error bars. Neuroimage. 2018;180(Pt A):68-77.

[21] Meng F, Fu Y, Lou F, Chen Z, editors. An effective network attack detection method based on kernel PCA and LSTM-RNN. 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC); 2017: IEEE.

[22] Prakash S, Jalal AS, Pathak P, editors. Forecasting covid-19 pandemic using prophet, lstm, hybrid gru-lstm, cnn-lstm, bi-lstm and stacked-lstm for india. 2023 6th International Conference on Information Systems and Computer Networks (ISCON); 2023: IEEE.

[23] Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of big Data. 2021;8:1-74.

[24] Mayya A, Alkayem NF, Shen L, Zhang X, Fu R, Wang Q, et al., editors. Efficient hybrid ensembles of CNNs and transfer learning models for bridge deck image-based crack detection. Structures; 2024: Elsevier.

[25] Sabharwal A, Sedghi H, editors. How Good Are My Predictions? Efficiently Approximating Precision-Recall Curves for Massive Datasets. UAI; 2017.

[26] Nahm FS. Receiver operating characteristic curve: overview and practical use for clinicians. Korean journal of anesthesiology. 2022;75(1):25-36.

[27] Staudemeyer RC, Morris ER. Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:190909586. 2019.

[28] Jordan ID, Sokół PA, Park IM. Gated recurrent units viewed through the lens of continuous time dynamical systems. Frontiers in computational neuroscience. 2021;15:678158.