Iraqi Journal of Information and Communications Technology(**IJICT**)
Vol. 7, Issue 3, December 2024
www.ijict.edu.iq
ISSN:2222-758X
e-ISSN: 2789-7362

EFFICIENT IOT MALWARE DETECTION TECHNIQUE USING RECURRENT NEURAL NETWORK

Marwa A. Abd Al Abbas[1], **Ban M. Khammas**[2]
[1,2] Department of Networks Engineering, College of Information Engineering, Al-Nahrain University, Jadriya,Baghdad, Iraq
Lamarmarwa94@gmail.com[1], bankhammas@coie-nahrain.edu.iq[2]
Corresponding Author: **Ban M. Khammas**

Received:14/05/2023; Revised:04/07/2023; Accepted:15/08/2023
DOI:**10.31987/ijict.7.3.249**

***Abstract*- Because of the Internet of Things (IoT) growing impact on many different uses and their expanding computational and analytical capacities, they are a potential threat victim for malware intended to hack particular IoT gadgets. Therefore, in this paper, a successful Recurrent Neural Network (RNN) is proposed for malware detection. Multiple trials with varied hyper parameter values in this research are trained. When employing RNN for malware categorization, thorough trials demonstrated that embedding size is more important than the input size. RNN performance with two different feature vectors was assessed using hyper parameters to validate RNN as an efficient solution for malware detection. Natural Language Processing (NLP) and feature selection are the two feature vectors. A paired t-test was also employed in this paper to see if the findings were meaningful to one another. Compared to the chi2 feature selection, RNN with NLP attained the maximum AUC value and a reasonable variance. Based on the actual results. The proposed effective malware detection approach proves 99% detection accuracy with NLP techniques and 89% with feature selection techniques, hitherto results with the RNN classifier.**

keywords: **Malware detection , Recurrent Neural Network (RNN), Deep learning, Opcode, Feature selection, Feature reduction.**

I. Introduction

In today's world, IoT smart devices permeate every aspect of our society, including companies, medicine, housing, automobiles, games, mentoring networks, entertainment, and more. This interaction, however, causes significant worries because a large amount of network traffic and a wide range of traffic classes flow over IoT networks, such as those produced by manufacturing machines, self-driving cars, care sensors, home automation, and other critical devices [1]. From an attacker's perspective, IoT devices are appealing targets for malware attacks because, unlike PCs, they are always online, have no antivirus installed, and flawed user account passcodes give hackers simple access to powerful shells (such as BusyBox). Malware is computer software intended to harm the Operating System (OS). Based on its intended use and behaviour, malware is classified as adware, spyware, virus, worm, trojan, rootkit, backdoor, ransomware, and Command and Control (CC) bot. As researchers create novel methods, the malware creators enhance their capacity to elude discovery. Based on these developments, IoT devices represent a significant new area of security study. It is critical to know which strategies are appropriate for protecting IoT [2].

Machine Learning (ML) and Deep Learning (DL) are powerful data discovery methods for learning regarding 'normal' and 'abnormal' actions in how IoT devices and their parts communicate. IoT systems must advance from just to be efficient and safe to enable safe interaction with security-based intelligence supported by DL/ML approaches among devices [3]. DL is the bedrock of modern artificial intelligence, it is frequently employed in computer vision, speech recognition, robotics, and a variety of additional fields [4]. DL is a bigger subject of ML in which supervised, unsupervised, and semi-supervised learning are performed using a bigger deep neural network [5]. It offers the advantage of identifying an unstructured data model, and most individuals are familiar with the media contained in such data, such as photos, sound, video, and text. DL has produced very promising results for a wide range of natural language processing problems. Specifically, topic classification, attitude analysis, question answering, and language translation [6].

Many algorithms in the field of Natural Language Processing (NLP) have evolved; Recurrent Neural Networks (RNN) are examples of DL (for sequence modelling). RNNs are a type of return network because they aim to take advantage of serial or sequence data. As an outcome, provided an order (or series) of inputs, RNN can accurately identify objects. For this case, preceding calculations decide what will occur [7]. Because RNN is flexible and powerful, using it is a feasible option in malware hunting. During the training step, RNN effectively considers and learns the opcode series with varying length sequences [8].

In this work, two methods are proposed for malware classification based on RNN. The first method involves RNN with feature selection (chi2), while the second involves RNN by applying Natural Language Processing techniques. The remaining parts of the work are as follows: Section II includes the subsequent malware detection, and Section III discusses the suggested approach. Section IV discusses the experiment's findings, and Section V brings the paper to a conclusion.

II. RELATED WORK

Deep learning-based malware detection has been shown to be particularly effective in malware detection. Many researchers' works in the field of malware detection will be presented in this section.

An N-gram technique was used Bojan Kolosnjaji et al.,2017 [9], as well as a combination of convolutional neural network (CNN) and RNN to carry out hierarchical feature extraction, to select ideal Opcodes for malware detection. The success rate for detection was 89%, with a median precision of 85.6% and recall of 89.4%, stated to the researchers.

Matilda Rhode et al ., 2018 [10], the authors investigated the possibility of predicting whether an executable is malicious or benign based on a short snapshot of behavioral data. They propose a method that uses an ensemble of recurrent neural networks and evaluate their approach on a dataset containing 1,126 malicious and 1,126 benign executables. The findings of their experiments suggest that their strategy achieves 94% accuracy during the first 5 seconds of operation.

Zhongru Ren et al., 2020 [11], the authors proposed two end-to-end methods for malware detection without manual feature engineering. They used a sampling method to preprocess the dex files in Android APKs and trained two DL models, namely DexCNN and DexCRNN, on the preprocessed sequences. The evaluation was conducted on a dataset consisting of 8000 benign APKs and 8000 malicious APKs, and the results showed that DexCNN achieved a detection accuracy of 93.4%, while DexCRNN achieved a detection accuracy of 95.8%.

Ruitao Feng et al., 2020 [12], the authors examined the efficacy of a sequence-based learning strategy combined with performance optimization techniques employing RNN models for spotting malicious apps on the device end. To do this, they provide a feature selection approach that uses a repeating components removal method to filter out unnecessary elements, shortening the length of the feature sequence for chosen sequence-based features. An appropriate number sequence for input into a neural network is created out of the condensed feature sequence. The evaluation's findings demonstrate that

their method achieves a high level of accuracy (97.85%).
Sanket Shukla et al., 2019 [13], used RNNs to extract and process localized features for sequence classification in malware detection. This approach achieved an average accuracy of 90% for detecting stealthy malware and 94% for detecting traditional malware applications.
Hamed HaddadPajouh et al., 2018 [14], they analyzed the Opcode sequences using RNNs to find IoT malware. With a dataset containing 281 harmful ARM-based IoT samples and 270 ARM-based IoT benign samples, their approach obtained a high accuracy rate of 98.18 percent.
Hamid Darabian., 2018 [15], used a sequential pattern mining technique to identify the most prevalent Opcode sequences used by malicious IoT applications. Maximum Frequent Patterns (MFP) of Opcode sequences can be used to distinguish between malicious and benign IoT programs. After that, the classification feature is assessed for RNN and Convolution Neuron Network (SVM). In particular, the accuracy reached up to 99% in detecting unknown IoT malware.
According to Radhakrishnan et al., 2021 [16], DL techniques identify IoT malware well. In particular, their proposed approach makes used RNN to investigate the execution procedure codes of IoT frameworks. They employed an IoT malware sample dataset of 271 benign and 282 dangerous apps to test their methods. The 4 trained techniques were then evaluated on the 104 untrained samples. The proposed model's second setup has a greater accuracy of 99.08%.

III. **METHODOLOGY**

This study proposes a malware detection approach based on static analysis and an RNN model. Natural Language Processing (NLP) techniques such as word embedding, padding, and masking are employed for effective malware detection. The feature selection technique is used in another scenario. The proposed strategy is then tested with various parameters on RNN networks, such as embedding size and input size. The flow diagram of the proposed technique is shown in Fig. 1. It begins with a dataset of executable files, both malicious and benign, which is divided into training and testing sets. Preprocessing involves three steps: feature extraction, labelling, and tokenization. Two cases are then applied: either NLP or feature selection. Next, the model is generated, and the dataset is classified.

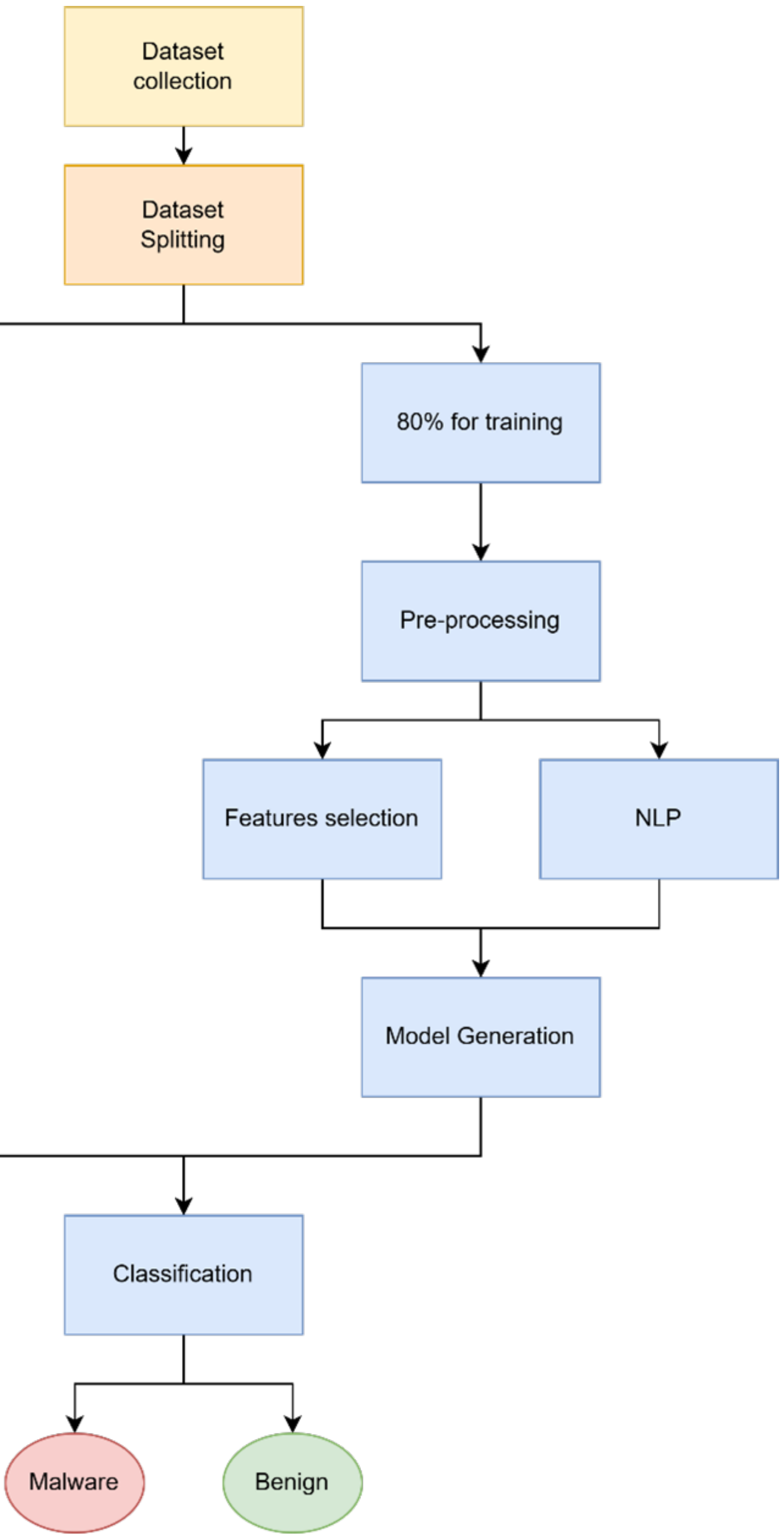

Figure 1: Flowchart of the proposed system.

The main steps of the proposed systems can be identified as follows:

1) **The Dataset:** The dataset used in this study consists of 3971 executable files, comprising 3247 malware files from distinct families and 724 benign files. To ensure efficient processing, each file is limited to a maximum size of 1MB, and any files larger than this are excluded from the dataset.
2) **Splitting the dataset:** To ensure the reliability of our prediction model, we randomly split the dataset into two parts using an 80:20 ratio. Specifically, 80% (3176) of the executable files were used for training the DL algorithm to develop the prediction model. The remaining 20% (795) of the files were reserved for testing the effectiveness of the model.
3) **Preprocessing:** There are three implicit phases in this step:
   - **Feature extraction:** in this step use Bag of Words (BoW) and filtering techniques in this phase as follows. (BoW) entails removing unwanted characters such as "#$%& " deleting English stop words and changing each word to lower case such as "an", "the", and "is". Filtering technique is a quick and efficient method for extracting text features on a large scale. In this work, high word frequency filtering is used to remove words with a high frequency.
   - **Label encoding:** It is the process of converting the categorical labels in the two types of data(benign and malicious) into numerical values so that the model can use them in this project.
   - **Tokenization** It refers to the method of breaking down a text files into smaller parts known as tokens. Following the tokenization step, this work proposes the employment of two models. The first model is RNN classifier with feature selection. The second model is RNN classifier with NLP technique. The details of each model are presented, respectively, in the following subsection III-A and subsection III-B.

A. RNN classifier with feature selection

In this model, we will utilize an RNN classifier based on chi-square feature selection, which will select features depending on the relevance of each word retrieved during the vectorization process. In the case of feature selection, converting text to numerical vectors is useful. As a type of vectorization, we will employ Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF is a well-known vectorization method it emphasizes the significance for each word in the document into account. TF-IDF is used just in model one. It computes a numerical value for each word in a document, which represents its importance in the document relative to other words. It calculates two important measures: Term Frequency (TF) and Inverse Document Frequency (IDF). The TF component counts the number of occurrences of an item in the document (malware and benign file), whereas the IDF component determines the frequency a word is throughout all documents in the corpus. By multiplying these two measures, TF-IDF vectorizer gives higher weightage to words that are more important in the document and less common across all the documents in the corpus. The TF-IDF vectorizer generates a matrix that is sparse with each row representing a document and each column representing a distinct word in the corpus. The value in each cell of the matrix represents the TF-IDF score for the related word in the relevant document.
Eqs. (1),(2), and (3) represents how to compute the TF and IDF formulas as follows [17]:

$$\text{TF}_{i,j} = \frac{n_{i,j}}{\sum n_{k,j}} \tag{1}$$

Here, $n_{i,j}$ is the count of occurrences of the word t_i in file d_j, and $n_{k,j}$ is the total of all occurrences in file d_j.

$$\text{IDF}_{t,D} = \log \frac{|D|}{|n_t|} \tag{2}$$

Here, $|D|$ is the total number of documents in the corpus, and $|n_t|$ is the number of documents containing the phrase t_i. If the word is not found in the corpus, the dividend is 0 (in general, use $1 + |n_t|$).

$$W_{i,j} = \frac{\text{TF}_{i,j} \times \log |D|}{|n_t|} \tag{3}$$

Here, $W_{i,j}$ is the weight of the word t_i. It has been demonstrated that a high word frequency in one file and a low word frequency across the entire file set can result in a high-weight TF-IDF.

After compute the TF-IDF of each feature the chi-square feature selection is used to choose the most important features. The chi-square method involves selecting the most significant features of a file and feeding these features into a DL system for classification. For filtering the TF-IDF features, a function called SelectKBest was employed in our work. This function accepts two arguments, the selected score function (chi2), and k, the number of features with the best chi2 that will be checked [18]. By utilizing this strategy, the advantage lies in the fact that it requires only a small set of features (k) to be analyzed, instead of scrutinizing all the available features, leading to faster and more efficient processing. The selected features select from above method (chi2) feature selection are fed to the RNN model to classify the dataset files. The RNN architecture used is comprised of 7 layers. We have trained different architectures, and the highest performance was achieved with the architecture shown in Table I.

TABLE I
RNN network architecture based on chi-square features selection

Layer type	**Output shape**	**Param**
Input layer	(None, 100)	0
Embedding	(None, 100, 128)	17408
LSTM	(None, 100, 256)	394240
LSTM	(None, 100, 128)	197120
LSTM	(None, 100, 64)	49408
LSTM	(None, 100, 32)	12416
Dropout	(None, 100, 16)	3136
Dense	(None, 100, 1)	17
Total params: 673,745 Trainable params: 673,745 Non-trainable params: 0		

B. RNN classifier with NLP technique

The second proposed model employs RNN classifiers based on natural language processing techniques. This work uses word embedding and (padding& masking) techniques of NLP. These steps in the current work include feature extraction, label encoding, tokenization mentioned above, and word embedding and padding/masking. These techniques are used to optimize the model's performance in handling and processing the textual data present in the dataset. Using these NLP techniques improves the model's ability to understand the text's context and meaning, thus leading to better predictions and accuracy. In the second approach, we used two NLP techniques: word embedding and padding/masking. The details of these two methods are described as follows.

1) Word embedding: In this study, we will use the Embedding layer method as a type of word embedding. Each word in the dataset files is converted generating a dense vector that captures word semantic associations, allowing the neural network to better understand the meaning of the text. This word embedding layer can improve classification model accuracy by allowing the neural network to learn more meaningful representations of the words [19].

2) Padding and masking: Because the Opcode files' size varies and DL models often require fixed-size inputs, these inputs must be transformed. Padding can be used to join together short sequences of variable durations. However, long ones are typically unified by truncating some sequence. The maximum sequence value in this paper is 23000 words; this value was determined based on a boxplot; the boxplot approach, as shown in Fig. 2, was used to summarize and compare data sets visually. Boxplots, also known as box and whisker plots, are excellent charts for displaying the distribution of data points across a specific parameter. These graphs show variations inside parameters determined, such as outliers, the median, the mean, and where a vast amount of data points fall inside the "box". In this work, the box is drawn according to the number and size of the words, and it became clear that 75% of the files contain 23,000 words (this value is adopted, in this work, in padding and masking techniques.

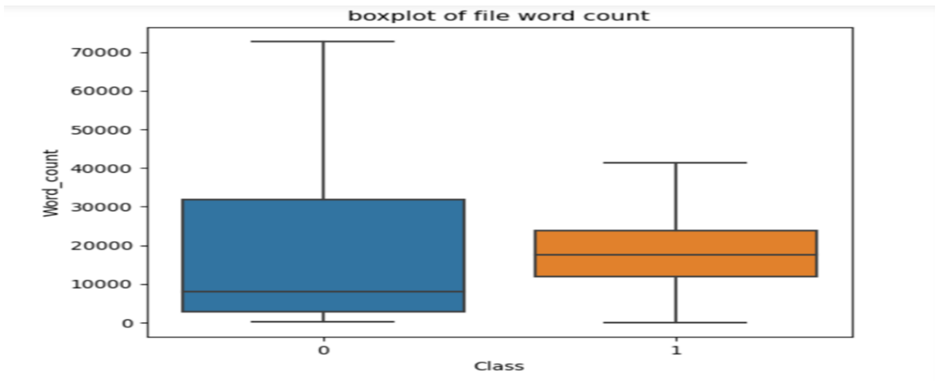

Figure 2: Illustration of the boxplot.

After splitting and pre-processing, the dataset was subjected to word embedding with padding and masking. Subsequently, the processed dataset was fed to the RNN. Applied RNN with an embedding layer and achieved the best accuracy using the architecture outlined in Table II.

TABLE II
The RNN with NLP model architecture

Layer type	**Output shape**	**Param**
Embedding	(None,5000, 75)	75075
Simple RNN	(None, 5000, 150)	33900
Simple RNN	(None, 150)	45150
Dense	(None, 100, 1)	151
Total params: 154,276 Trainable params: 154,276 Non-trainable params: 0		

IV. EXPERIMENTAL RESULTS

The proposed approach distinguishes between harmful and benign software. Several experiments were conducted to compare the effectiveness of RNN with NLP techniques and RNN with the Chi-squared feature selection method in terms of various DL performance evaluation metrics, such as Classification Accuracy (ACC), False Positive Rate (FPR), Precision, True Negative Rate (TNR), Recall, False Negative Rate (FNR), F-measure, and True Positive Rate (TPR). These metrics were calculated for each RNN classifier [19-22]. True Positive (TP) describes the number of malware instances correctly identified as malware, True Negative (TN) refers to the number of benign files properly categorized as typical, and False Positive (FP) relates to the number of normal files incorrectly classified as malware. False Negative (FN) refers to the number of malware wrongly classified as normal [23, 24]. Below are the equations used to compute each of the evaluation metrics:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{5}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{6}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} \tag{7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8}$$

$$\text{F1-Score} = \frac{2 \times \text{TP}}{2 \times (\text{TP} + \text{FP} + \text{FN})} \tag{9}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{10}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FP} + \text{FN}} \tag{11}$$

The RNN used different input and embedding sizes, as shown in Table III, and the highest results were achieved with an embedding size of 150. While input size does not significantly affect performance, it can be handled with padding and masking techniques, ensuring compatibility within the model. However, the embedding process directly impacts how the model represents and understands the input features. Changes in embedding can affect the model's performance by altering correlations between features.

TABLE III
The impact of input and embedding size on the proposed model performance

Method	**Input size**	**Embedding size**	**Accuracy (%)**
RNN with NLP	5000	25	96.3
	10000	25	96.01
	15000	25	96.02
	23000	25	96.0
	5000	50	98.0
	5000	75	98.9
	5000	100	99.0
	5000	150	99.3
RNN with CHI2 feature selection	5000	25	86.5
	10000	25	86.01
	15000	25	85.6
	23000	25	85.0
	5000	50	87.0
	5000	75	87.9
	5000	100	88.6
	5000	150	89.4

Table IV displays the performance of both RNN models, one with Chi-square and the other with NLP, with respect to various performance metrics. Overall, the second approach with NLP showed higher performance across all metrics compared to the first approach with Chi-square.

TABLE IV
The outcomes of classification accuracy and DL performance tests

Method	TP	TN	FP	FN	ACC (%)	Precision (%)	Recall (%)	F-measure (%)
RNN with CHI2 feature selection	79	632	30	54	89.4	72	59	65
RNN with NLP	161	622	1	11	99.3	99	93	96

The two RNN classifiers showed a significant improvement in classification accuracy. The accuracy increased from 89.4 to 99.3 when using feature selection and NLP techniques, respectively, as demonstrated in Fig. 3.

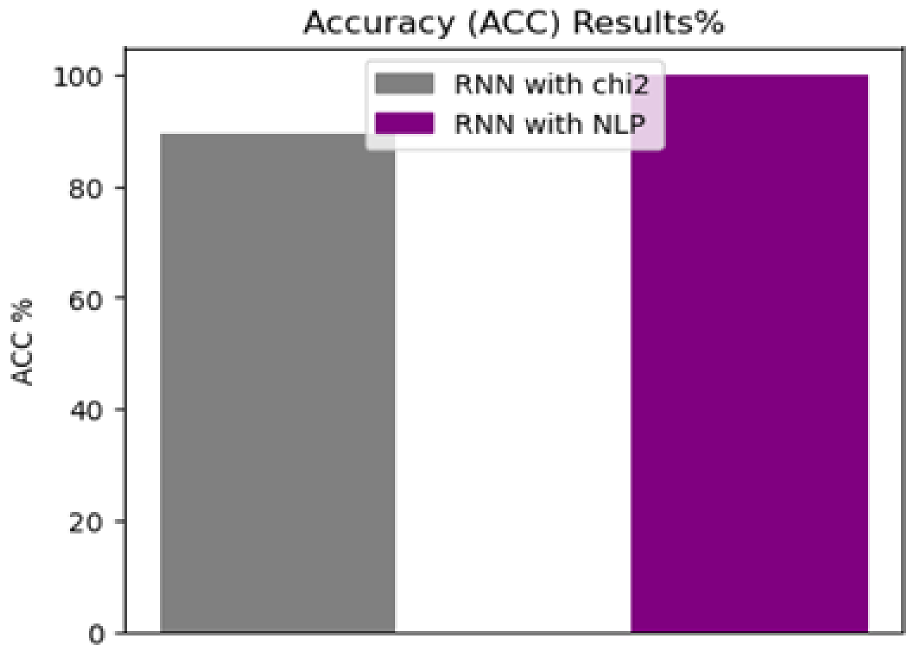

Figure 3: The accuracy of categorization was assessed.

TPR, FPR, Recall, TNR, FNR, and other metrics have also improved in value using NLP techniques, as shown in Table III. Fig. 4 depicts the improvement in TPR, FPR, TNR, and FNR results with NLP.

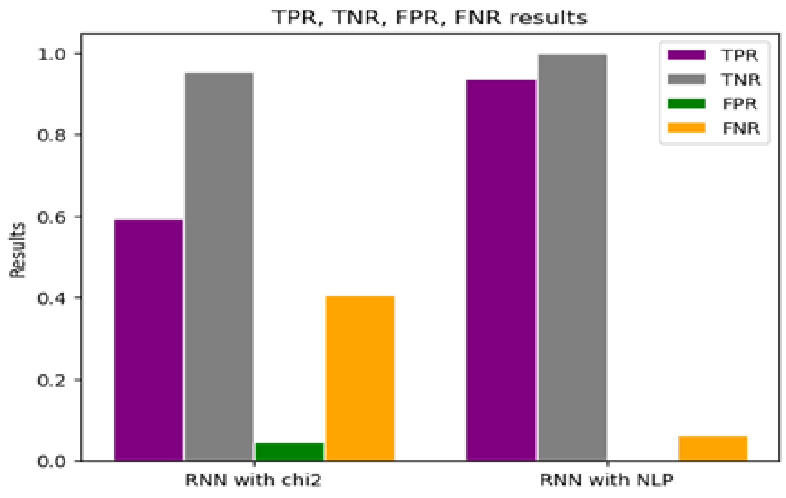

Figure 4: TPR, FPR, TNR, and FNR experimental results.

Fig. 5, Fig. 6, and Fig. 7 show the experimental outcomes of various DL evaluation metrics as recall, precision, f-measure with NLP, and feature selection.

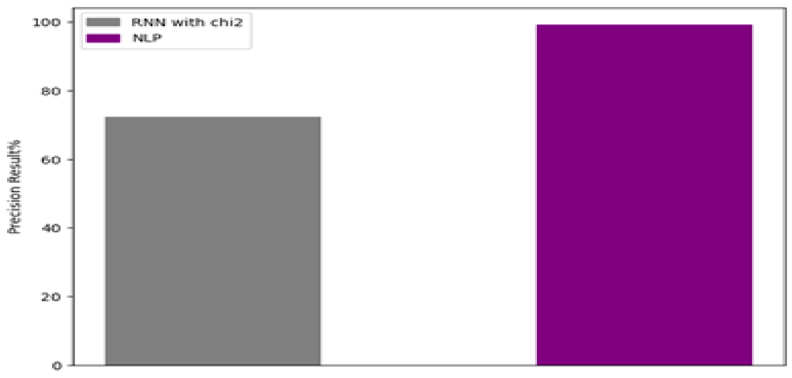

Figure 5: Precision results from NLP and feature selection.

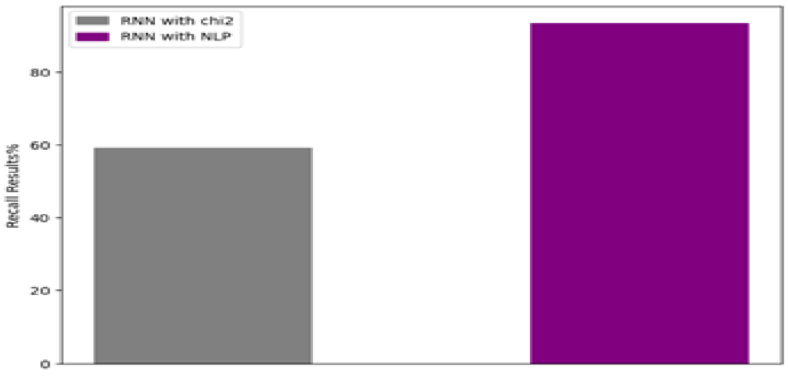

Figure 6: Recall results using NLP and feature selection.

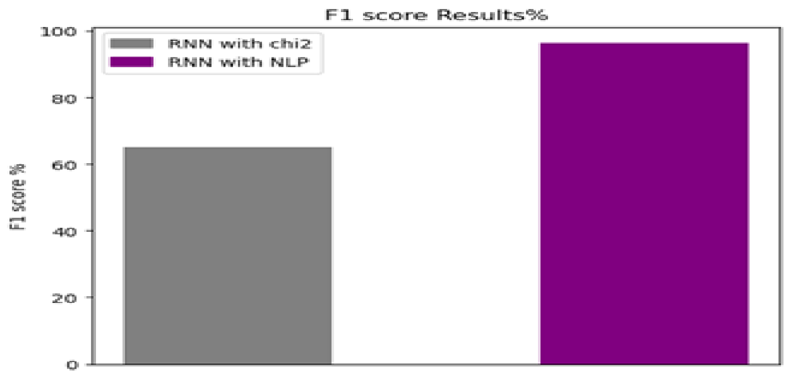

Figure 7: F-measure results using NLP and feature selection.

This work outperforms the previous study by [25] which used NLP (One Hot Encoding) based on static analysis with Opcode features and RNN. While Hamed HaddadPajouh [14] and G. Radhakrishnan [16] , these two works didn't base their work on the NLP, while in our work, we used the NLP as a basis for our work. This paper is based on static analysis and Opcode but with different techniques of NLP that have been shown to perform better [25]. As shown in Table V, the previous work achieved an accuracy of 97.8%, 98.1% and 99.08%, while this proposed system employed RNN with word embedding and achieved an accuracy of 99.3%.

V. DISCUSSION AND CONCLUSION

In conclusion, this work proposes two DL models for malware classification - an RNN model based on chi-square feature selection and an RNN based on NLP. The former relies on selecting the most relevant features and training the model based on those selected features. At the same time, the latter employs modern techniques such as word embedding and achieves high performance in all performance metrics evaluated. For instance, the difference in the accuracies of the two approaches

TABLE V
Comparison of methods

Methods	**Method of Analysis**	**Feature Type**	**Used Methods**	**Feature Extraction**	**Hyperparameters Tested**	**Accuracy (%)**
Proposed	Static	Opcode	Word embedding	Filtering	Input size (n), Step size (k), and Recurrent Neuron (r)	99.3
Sudan Jha [25]	Static	Opcode	One-hot encoding	BoW & Filtering	Input size (n), Embedding size (k), and Recurrent Neuron (r)	97.8
Hamed Had-dadPajouh [14]	Static	Opcode	Used Information Gain (IG) feature selection			98.1
G. Radhakr-ishnan [16]	Static	Opcode	Used Information Gain (IG) feature selection	Filtering		99.08

is that when not using chi-square, the system learned from all features, enabling a better understanding of the file structures and the relationships between features (words). On the other hand, when applying feature selection like chi-square, the models were limited to a subset of high-ranked features, resulting in a less comprehensive understanding of the file contents. Compared to the current study, classification DL based on NLP proved to have higher and better performance. These findings suggest that NLP-based DL approaches are a promising avenue for improving malware classification accuracy.

Funding

None

ACKNOWLEDGEMENT

The author would like to thank the reviewers for their valuable contribution in the publication of this paper.

CONFLICTS OF INTEREST

The author declares no conflict of interest

REFERENCES

[1] R. Pecori, A. Tayebi, A. Vannucci, and L. Veltri, "IoT Attack Detection with Deep Learning Analysis," in 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, United Kingdom: IEEE, Jul. 2020, pp. 1–8. doi: 10.1109/IJCNN48605.2020.9207171.
[2] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: Analysing the Rise of IoT Compromises"2015.
[3] N. A. Khalil and B. M. Khammas, "AN EFFECTIVE AND EFFICIENT FEATURES VECTORS FOR RANSOMWARE DETECTION VIA MACHINE LEARNING TECHNIQUE," Iraqi J. Inf. Commun. Technol., vol. 5, no. 3, pp. 23–33, Dec. 2022, doi: 10.31987/ijict.5.3.205.
[4] Y. Yue, S. Li, P. Legg, and F. Li, "Deep Learning-Based Security Behaviour Analysis in IoT Environments: A Survey," Secur. Commun. Netw., vol. 2021, pp. 1–13, Jan. 2021, doi: 10.1155/2021/8873195.
[5] B. I. Farhan and A. D. Jasim, "IMPROVING DETECTION FOR INTRUSION USING DEEP LSTM WITH HYBRID FEATURE SELECTION METHOD," Iraqi J. Inf. Commun. Technol., vol. 6, no. 1, pp. 40–50, Apr. 2023, doi: 10.31987/ijict.6.1.213.
[6] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du, and G.-J. Ahn, "Uncovering the Face of Android Ransomware: Characterization and Real-Time Detection," IEEE Trans. Inf. Forensics Secur., vol. 13, no. 5, pp. 1286–1300, May 2018, doi: 10.1109/TIFS.2017.2787905.
[7] T. Iqbal and S. Qureshi, "The survey: Text generation models in deep learning," J. King Saud Univ. Comput. Inf. Sci., vol. 34, no. 6, pp. 2515–2528, Jun. 2022, doi: 10.1016/j.jksuci.2020.04.001.
[8] A. Yazdinejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, and M. Y. Chen, "Cryptocurrency malware hunting: A deep Recurrent Neural Network approach," Appl. Soft Comput., vol. 96, p. 106630, Nov. 2020, doi: 10.1016/j.asoc.2020.106630.

[9] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep Learning for Classification of Malware System Call Sequences," in AI 2016: Advances in Artificial Intelligence, B. H. Kang and Q. Bai, Eds., in Lecture Notes in Computer Science, vol. 9992. Cham: Springer International Publishing, 2016, pp. 137–149. doi: 10.1007/978–3–319–50127–7_11.
[10] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," Comput. Secur., vol. 77, pp. 578–594, Aug. 2018, doi: 10.1016/j.cose.2018.05.010.
[11] R. Feng, J. Q. Lim, S. Chen, S.-W. Lin, and Y. Liu, "SeqMobile: An Efficient Sequence-Based Malware Detection System Using RNN on Mobile Devices," in 2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS), Singapore: IEEE, Oct. 2020, pp. 63–72. doi: 10.1109/ICECCS51672.2020.00015.
[12] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for android IoT devices using deep learning," Ad Hoc Netw., vol. 101, p. 102098, Apr. 2020, doi: 10.1016/j.adhoc.2020.102098.
[13] S. Shukla, G. Kolhe, S. M. Pd, and S. Rafatirad, "RNN-Based Classifier to Detect Stealthy Malware using Localized Features and Complex Symbolic Sequence," in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA: IEEE, Dec. 2019, pp. 406–409. doi: 10.1109/ICMLA.2019.00076.
[14] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting," Future Gener. Comput. Syst., vol. 85, pp. 88–96, Aug. 2018, doi: 10.1016/j.future.2018.03.007.
[15] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K. R. Choo, "An opcodeâbased technique for polymorphic Internet of Things malware detection," Concurr. Comput. Pract. Exp., vol. 32, no. 6, Mar. 2020, doi: 10.1002/cpe.5173.
[16] G. Radhakrishnan, K. Srinivasan, S. Maheswaran, K. Mohanasundaram, D. Palanikkumar, and A. Vidyarthi, "WITHDRAWN: A deep-RNN and meta-heuristic feature selection approach for IoT malware detection," Mater. Today Proc., p. S2214785321002960, Feb. 2021, doi: 10.1016/j.matpr.2021.01.207.
[17] C. Liu, Y. Sheng, Z. Wei, and Y. Q. Yang, "Research of Text Classification Based on Improved TF-IDF Algorithm," in 2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE), Lanzhou: IEEE, Aug. 2018, pp. 218–222. doi: 10.1109/IRCE.2018.8492945.
[18] E. Dias Canedo and B. Cordeiro Mendes, "Software Requirements Classification Using Machine Learning Algorithms," Entropy, vol. 22, no. 9, p. 1057, Sep. 2020, doi: 10.3390/e22091057.
[19] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated Deep Learning for Cyber Security in the Internet of Things: Concepts, Applications, and Experimental Analysis," IEEE Access, vol. 9, pp. 138509–138542, 2021, doi: 10.1109/ACCESS.2021.3118642.
[20] Y. K. Saheed and M. O. Arowolo, "Efficient Cyber Attack Detection on the Internet of Medical Things-Smart Environment Based on Deep Recurrent Neural Network and Machine Learning Algorithms," IEEE Access, vol. 9, pp. 161546–161554, 2021, doi: 10.1109/ACCESS.2021.3128837.
[21] B. A. N.G. and S. S., "Deep Radial Intelligence with Cumulative Incarnation approach for detecting Denial of Service attacks," Neurocomputing, vol. 340, pp. 294–308, May 2019, doi: 10.1016/j.neucom.2019.02.047.
[22] R. V. Mendonca et al., "Intrusion Detection System Based on Fast Hierarchical Deep Convolutional Neural Network," IEEE Access, vol. 9, pp. 61024–61034, 2021, doi: 10.1109/ACCESS.2021.3074664.
[23] B. M. Khammas, "The Performance of IoT Malware Detection Technique Using Feature Selection and Feature Reduction in Fog Layer," IOP Conf. Ser. Mater. Sci. Eng., vol. 928, no. 2, p. 022047, Nov. 2020, doi: 10.1088/1757-899X/928/2/022047.
[24] B. M. Khammas, I. Ismail, and M. N. Marsono, "Pre-filters in-transit malware packets detection in the network," TELKOMNIKA Telecommun. Comput. Electron. Control, vol. 17, no. 4, p. 1706, Aug. 2019, doi: 10.12928/telkomnika.v17i4.12065.
[25] S. Jha, D. Prashar, H. V. Long, and D. Taniar, "Recurrent neural network for detecting malware," Comput. Secur., vol. 99, p. 102037, Dec. 2020, doi: 10.1016/j.cose.2020.102037.