Iraqi Journal of Information and Communications Technology(IJICT)
Vol.6, Issue 3, December 2023

www.ijict.edu.iq

ISSN:2222-758X
e-ISSN: 2789-7362

PERFORMANCE EVALUATION OF DEEP LEARNING TECHNIQUES IN THE DETECTION OF IOT MALWARE

Ayat T. Salim [1], **Ban Mohammed Khammas** [1]
[1] Department of Computer Networks Engineering, College of Information Engineering, AL-Nahrain University, Iraq
ayat.tariq@coie-nahrain.edu.iq , bankhammas@coie-nahrain.edu.iq
Corresponding Author: Ban Mohammed Khammas
Received:21/11/2022; Revised: 06/01/2023; Accepted:17/01/2023
DOI:10.31987/ijict.6.3.233

***Abstract*- Internet of Things (IoT) equipment is rapidly being used in a variety of businesses and for a variety of reasons (for example, sensing and collecting data from the environment in both public and military settings). Because of their expanding involvement in a wide range of applications and their rising computational and processing capabilities, they are a viable attack target for malware tailored to infect specific IoT devices. This study investigates the potential of detecting IoT malware using different deep learning techniques: the classic feedforward neural network (FNN), convolutional neural networks (CNN), long short-term memory (LSTM), and recurrent neural networks (RNN). The proposed method analyses the execution operation codes of IOT app sequences using modern NLP (natural language processing) methods. The current work utilized an IoT application dataset with 500 malware (collected from the IOTPOT dataset) and 500 goodware samples to train the proposed algorithms. The trained model is tested against 2971 fresh IoT malware and goodware samples. The samples were input into deep learning models, and performance metrics were obtained. The results demonstrate that the RNN model had the best accuracy (99.19%) in detecting fresh malware samples. On the other hand, the results were compared by the time required for training; the CNN model shows that it could achieve high accuracy (98.05%) with less training time. A comparison with various deep learning classifiers demonstrates that the RNN and CNN techniques produce the best results.**

keywords: **IoT Security, Deep Learning, Intrusion Detection System, Cyber-Attack**

I. INTRODUCTION

The Internet of Things (IoT) is, in fact, a networked collection of devices that can detect, process, and transmit data [1]-[4]. There are several uses for the Internet of Things, including medical, mobility, intelligent buildings, urban governance, and agriculture [5]-[9]. By 2025, it is anticipated that more than 63 million Internet of Things devices will indeed be on the marketplace [10]. According to Ericsson's projections, roughly 3.5 billion cellular Internet of Things connections will be established by 2023 [11]. Because of the extensive usage and vital role of the Internet of Things networks, cybercriminals have devised harmful and complex cyberattacks targeting IoT end nodes to exploit IoT nodes and infrastructural facilities [12]- [14]. Mirai was among the first malware strains to universally use the Internet of Things. Mirai orchestrated a botnet of compromised Internet of Things devices to launch a Distributed Denial of Service (DDoS) cyberattack [15]. The release of Mirai's script demonstrated that it is effortless to develop harmful IoT-based malware cyberattacks.

For example, the "BrickerBot" malware grows by using the Mirai source code, joins the infected system to a botnet, erases the firmware, and completely reboots the device after infection. Different and intelligent techniques have been proposed to detect cyberattacks; one of the smart techniques is the deep learning technique. Deep Learning (DL) represents a technique that has been widely used to improve the accuracy and solidity of cybersecurity detection and defense systems [16, 17]. DL's capacity to understand complex patterns inside complex intrusions and its resistance against unanticipated hostile attempts make it a potential solution for identifying and safeguarding IoT networks from cyberattacks. DL approaches are

ISSN:2222-758X
e-ISSN: 2789-7362

frequently used in the fields of security, privacy, and forensics [14], [18]-[22]. The current work aims to investigate and report on the results of applying various deep learning models to a gathered dataset of IoT software products through extracted Operational Code (OpCode) from malicious and benign recordings using the proposed technique. The rest of this paper is arranged as follows: Section II evaluates pertinent literature, Section III details the technique adopted, Section IV discusses the methodology used, Section V analyzes the experimental data, Section VI examines the findings, and Section VII wraps up this article.

II. RELATED WORKS

The methods that are now in use to particularly identify malware that uses OpCode sequences are described in this section.

Azmoodeh et al. and Naveen et al. [23, 24] suggested converting OpCodes into vector spaces and using a deep Eigenspace technique to distinguish between malicious and benign files. Azmoodeh et al. suggest feeding the vector space values directly to the Eigenspace model without change; on the other hand, Naveen et al. created N-gram opcode sequences and fed them to the model. Azmoodeh et al. and Naveen et al. use accuracy matrices to evaluate their model and achieve an accuracy of 99.68% and 98.37%, respectively.

Hamad S. et al. [25] present a simple cross-architecture antimalware solution for IoT devices. The proposed approach examines the sequence representations of the executable file's operation codes (OpCodes) by utilizing Deep Bidirectional Representations from Transformers (BERT) and embedding an advanced natural language processing (NLP) method. The retrieved sentence embedding from BERT is input into a hybrid CNN, Bi-LSTM, and LocAtt model that has been customized to capture contextual information and long-term relationships between OpCode sequences. The proposed deep learning (DL) model incorporates the convolutional neural networks (CNN), local attention mechanisms (LocAtt), and bidirectional long short-term memory (Bi-LSTM) in one model to detect malware accurately. The accuracy acquired by BERTDeep-Ware reached 99.93%.

Jeon J. et al. [26] proposed a hybrid malware detection model called "HyMalD." This model conducts dynamic and static analyses concurrently to discover disguised malware, which static analysis alone cannot detect. First, it extracts static aspects of the opcode sequence using a predefined dataset and then dynamically collects the API call sequence. The retrieved features are trained using the Bi-LSTM and SPP-Net models. This model reached an accuracy of 92.5%.

Jahromi et al. [27] constructed a revised "Two-hidden-layered Extreme Learning Machine (TELM)." This relies on malware sequence components, such as OpCode sequences, in malware detection. This model proposes using a method that benefits from avoiding backpropagation when training neuron networks by using partly linked networks between the input and the first hidden layer. In the second layer, these are subsequently aggregated into a fully linked network. Finally, they employ an ensemble to raise the system's reliability and accuracy for detecting malware threats. Compared to stacked LSTM and CNN, the suggested technique expedites the learning and detection phases of malware detection and achieves an accuracy of 99.65%.

Radhakrishnan et al. [28] proposed a technique involving using a recurrent neural network (RNN) model to analyze ARM-base OpCode sequences. The authors used a small dataset composed of 271 files of benign IoT OpCodes and 282

ISSN:2222-758X
e-ISSN: 2789-7362

files of malicious IoT OpCodes to train their model; they then used 104 unknown files of OpCodes to test their model, which achieved an accuracy of 99.08%.

Vasan et al. [29] discuss a robust cross-architecture IoT malware threat detection system utilizing advanced ensemble learning and propose a malware hunting methodology based on advanced ensemble learning (MTHAEL). Their proprietary MTHAEL model, which improves existing methods for detecting IoT malware, uses a stacking ensemble of heterogeneous pattern selection techniques to see it. The model involves using an ensemble of RNN and CNN deep learning algorithms and training the model on a dataset composed of 15,482 files of malware OpCodes and 5,655 files of benign OpCodes. The proposed model achieves an accuracy of 99.98%.

All of the proposed approaches still need to be improved to detect malware with very high detection accuracy while keeping model complexity to a minimum. According to the previously shown relevant research, a lightweight solution is highly required in an IoT environment, which can be achieved by fine-tuning the hyperparameter of the deep learning algorithms. This study aimed to create a deep learning model that efficiently detects IoT malware with as little complexity as possible.

III. Deep Learning Algorithms

Deep learning algorithms have gained popularity in malware detection in recent years. In this section, some of the widely used algorithms are introduced.

A. *Convolutional Neural Networks (CNN)*

One deep learning method well-known for handling image analytics is CNN. CNN has lately been widely employed for textual analysis and sentiment analysis. Additionally, it is recognized as the regularized kind of multilayer perceptron, which resists overfitting and mimics biological neurons in the human brain.

Like other deep learning approaches, CNN comprises one output, an input layer, and numerous hidden layers. There are mainly two types of hidden layers in a convolutional neuron network, which are: a) the convolution layer, which is responsible for applying filters to the data and learning the features from the filter output. b) the max-pooling layer. Here is where the pooling action takes place, choosing the most significant number of elements from the feature map region the filter has covered. The most prominent aspects of the prior feature map would therefore be included in the output after the max-pooling layer, which would be a feature map [30].

B. *Simple Recurrent neural network (RNN)*

As the name suggests, an RNN is a neural network that repeatedly uses the prior output as an input. RNN's nature makes it suited for prediction problems.

The most notable characteristic of an RNN is the hidden state, which aids in storing details for each sequence. The settings for its parameters that forecast the series are the same for all hidden layers doing the same task [31].

ISSN:2222-758X
e-ISSN: 2789-7362

C. *Long Short-Term Memory (LSTM)*

LSTM is a unique form of RNN designed to overcome the difficulty of long-term dependencies. It has been focusing on text processing for better results [31].

D. *Feedforward Neural Network (FNN)*

Artificial neural networks, called feedforward neural networks, do not have cyclical connections between the units. RNNs are more complex than feedforward neural networks, the first artificial neural network type to be created. They are known as "feedforward" networks because information only moves forward (there are no loops) between the input nodes, hidden nodes, and output nodes in that order [32].

IV. METHODOLOGY

This section will discuss the methodology used to evaluate the suggested deep learning models. As shown in Fig. 1, the process consists of two stages (training and testing), each consisting of several sub-phases âmainly data gathering, OpCode extraction, pre-processing, and training and testing the deep learning models. The only difference between the two stages is the feature selection phase in the training stage, which requires selecting features that contribute the most to classifying the files into benign and malware. Following is a simple description of the operation done in each phase:

- Data gathering and OpCode extraction: this phase involves collecting benign and malware IoT executables and using reverse engineering tools; the operational codes are extracted and saved in a text file for the training and testing datasets.
- Feature selection: in the training stage, the features that contribute the most to accurately classifying the executable files as benign and malicious were selected using the information gain technique.
- In pre-processing to convert the training and testing datasets to a format understood by the deep learning models, text filtering and tokenizing techniques are used.
- Training and testing: these phases involve feeding the data to the deep learning models and evaluating their performance in classifying the files as benign or malware using evaluation metrics.

www.ijict.edu.iq
ISSN:2222-758X
e-ISSN: 2789-7362

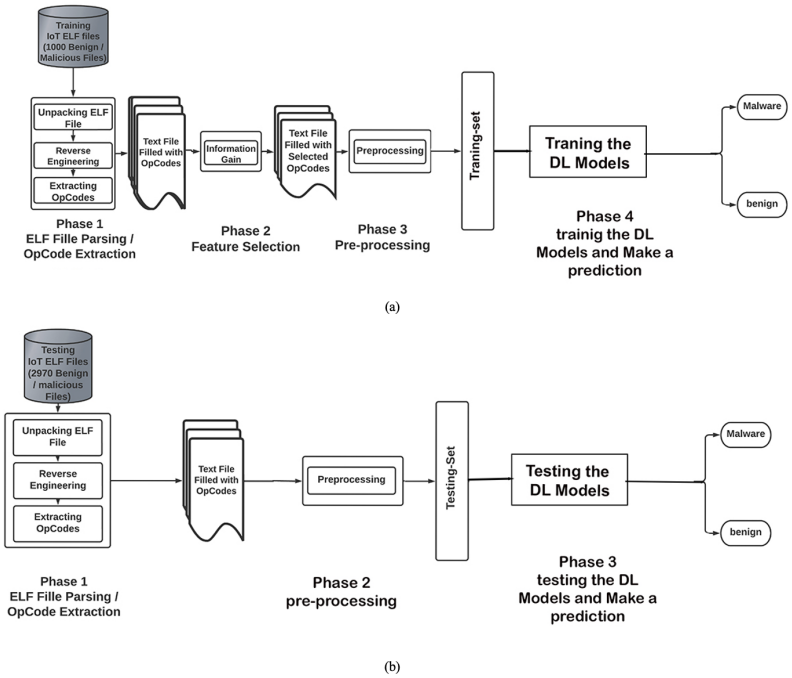

Figure 1: (a) shows steps conducted when training the models, (b) shows steps conducted when testing the models

A. *Dataset Description*

The latest DL approaches for identifying IoT malware were evaluated using an IoT dataset consisting of 3971 samples in text file format. This dataset has 724 samples of goodware and 3247 samples of malware. The malware files were obtained from the IoTPoT dataset, which contains malware executable files used to extract the malicious OpCode. This dataset was provided by a cybersecurity research team led by Prof. Katsunari Yoshioka at Yokohama National University. The benign opcode was extracted from the Ubuntu Linux system's system files residing in the paths /usr/bin, /sbin, and /bin. The most recent version of the IoTPoT dataset [33, 34] can be found on this site: https://sec.ynu.codes/iot.

B. *Dataset Preparation*

A text corpus was constructed using simple Python code to showcase various feature engineering and representation approaches, resulting in a data frame with two columns, one of which involves a succession of operational codes such as ADD, MOV, SUB, and PUSH. The other column specifies the sample category, such as "goodware" or "malware." The second step is to select the OpCodes that contribute the most to correctly classifying the file as malicious or benign. The information gain technique was used as a feature selection method for this step. Eq. (1) shows the information gain formula, where an OpCode feature called f, c is how many classes available (malicious and non-malicious); and Dv is the OpCode flow where the feature f may be found. Wi is the percentage of DV in the training dataset that belongs to class i. Sorting each IG's values into decreasing order allowed us to determine the most important characteristics necessary for defining a

ISSN:2222-758X
e-ISSN: 2789-7362

threshold ($\alpha > 0.30$). The third step involved using text filtering and the tokenizer technique, which filters the text from any unnecessary special characters and assigns each OpCode a numeric value. Then aggregate these numeric values in a list that represents the opcode, and finally use the padding technique to make these lists the same size by post-padding with zeroes. Fig. 2 shows the order of the pre-processing steps.

$$IG(D, \boldsymbol{f}) = \sum_{i=1}^{c} -p \ln p_i - \sum_{V \in \{0,1} \frac{|D_V|}{|D|} \sum_{i=1}^{c} -w_i \ln w_i \tag{1}$$

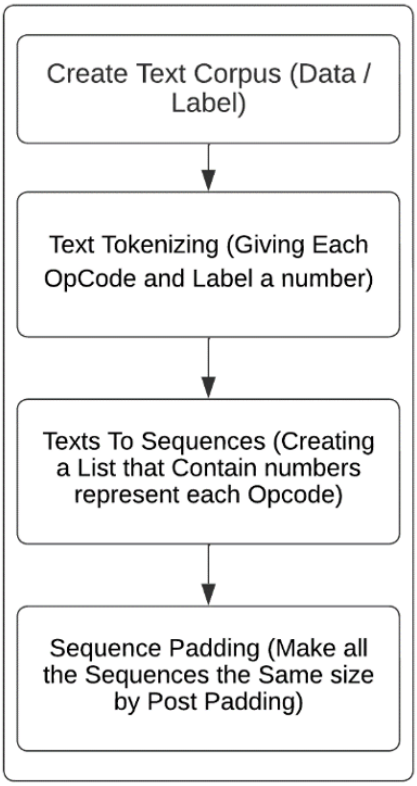

Figure 2: Pre-processing steps

After shaping the data into a format understood by the deep learning models, the data was fed to four deep learning models to train them and evaluate them individually using the training and testing datasets.

V. EXPERIMENT RESULTS

The proposed models are evaluated with different settings to achieve the best possible accuracy. The first thing the models were tested against was how accuracy and training time would affect the number of OpCodes in each text file. While fixing the batch size to 64 , randomly select an Opcode number to keep in the file, and then the number of OpCodes

ISSN:2222-758X
e-ISSN: 2789-7362

that provide the best accuracy and training time are chosen to be tested against different batch size values, as shown in Fig. 3.

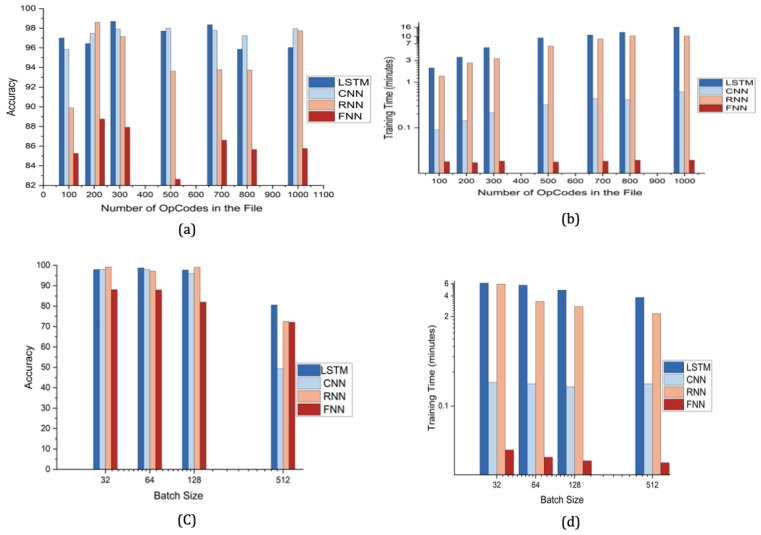

Figure 3: System performance: (a) Number OpCodes vs the Provided Accuracy, (b) Number OpCodes vs Training Time, (c) Batch Size vs the Provided Accuracy, and (d) Batch Size vs the Training Time

The figures above describe the model performance in terms of accuracy and training time; here, the model was tested using a different number of OpCodes in the provided text files (mainly 100 OpCodes, 203 OpCodes,302 OpCodes, 500 OpCodes, 682 OpCodes, 800 OpCodes and 1000 OpCodes) and the accuracy of the model was obtained, and the training time was calculated. as shown in Fig 3 (a) and Fig. 3 (b) the models perform differently in each OpCodes group, however, the models provide an acceptable accuracy compared to the consumed training time which was increased proportionally with an increment of the OpCodes number which reached a maximum value of 16.07 minutes when the opcode number was 1000 OpCodes in the LSTM model as illustrated in detail in Table I .

TABLE I
A COMPARISON BETWEEN THE OPCODE NUMBER AND THE PROVIDED ACCURACY/TRAINING-TIME

OpCodes Number	**LSTM** **Acc. (%) / time (min)**	**CNN** **Acc. (%) / time (min)**	**RNN** **Acc. (%) / time (min)**	**FNN** **Acc. (%) / time (min)**
100	97.01 / 2.046	95.85 / 0.09	89.9 / 1.363	85.26 / 0.018
203	96.43 / 3.53	97.48 / 0.144	98.59 / 2.64	88.76 / 0.0171
302	98.69 / 5.72	97.91 / 0.213	97.14 / 3.302	87.92 / 0.0185
500	97.71 / 9.33	98.01 / 0.321	93.64 / 6.161	82.63 / 0.0177
682	98.35 / 10.8	97.78 / 0.436	93.77 / 8.902	86.6 / 0.0184
800	95.86 / 12.35	97.24 / 0.411	93.74 / 10.37	85.65 / 0.0193
1000	96.03 / 16.07	97.95 / 0.607	97.74 / 10.28	85.76 / 0.0194

Based on the obtained result, An OpCode number equal to 302 is chosen , which gives acceptable accuracy and training

times. After selecting the suitable number of OpCodes, the models were evaluated against the batch size, a hyperparameter specifying how many samples must be processed before the inner model parameters are updated. Fig. 3 (c), Fig. 3 (d), and Table 2 show the accuracy and training time provided by the models. Some models' performance was excellent in one batch size group and bad in another group. Based on the result above, the 32 and 64 batch sizes are so close in their performance that the batch size is chosen to be 32 as it provided the best accuracy possible and acceptable training times.

TABLE II
A COMPARISON BETWEEN THE BATCH SIZE AND THE PROVIDED ACCURACY / TRAINING-TIME

Batch size	LSTM Accuracy (%) / training time (min)	CNN Accuracy (%) / training time (min)	RNN Accuracy (%) / training time (min)	FNN Accuracy (%) / training time (min)
32	97.91 / 6.13	98.05 / 0.22	99.19 / 5.93	88.05 / 0.023
64	98.69 / 5.72	97.91 / 0.213	97.14 / 3.302	87.92 / 0.0185
128	97.68 / 4.84	95.89 / 0.19	99.02 / 2.8	81.93 / 0.016
512	80.55 / 3.79	49.31 / 0.21	72.43 / 2.21	72.1 / 0.015

Typically, the following metrics are used to assess deep learning's efficiency in detecting attacks:

- True Positive (TP): signifies that a malicious program has been successfully detected as harmful.
- True Negative (TN): signifies that a benign program was accurately identified as non-malicious.
- False Positive (FP): implies that a benign program was incorrectly identified as harmful.
- False Negative (FN) indicates that malware was not discovered and classified as benign.

Additionally, the following measures are used to determine the performance of a deep learning model:

1) Accuracy, the proportion of samples is appropriately classified.
2) Precision is the percentage of correctly predicted malware samples.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

3) Recall The proportion of malware that was accurately classified

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

4) The F1 Rating is a summed recall and accuracy score.

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{4}$$

A. *processing environment*

All testing was conducted on a PC with 8 GB of memory and a basic Core i7 (1.8GHz) processor. Furthermore, the source code was written in Python 3.9.5, and the library for the DL work used Keras version 2.3.

B. *Deep learning models final settings*

Table III shows the final setting of the deep learning model, which involves the defied layers, the number of epochs used for training the models, the activation functions, the used optimizer, and the used batch size.

ISSN:2222-758X
e-ISSN: 2789-7362

TABLE III
FINAL SETTING FOR THE DL MODELS

DL Algorithm	**Convolutional Neural Network (CNN)**	**Simple RNN (Recurrent Neural Network)**	**Long Short-Term Memory (LSTM)**	**Feedforward neural network (FNN)**
Epochs	5	5	5	5
Neurons	302 Input Embedding Convolutional layer 1D Max pooling Flatten layer One output	302 Input Embedding 512 Simple RNN 128 Simple RNN 16 Simple RNN 1 Output	302 Input Embedding 128 LSTM 64 LSTM 16 LSTM 1 Output	302 Input 32 Dense One output
Activation function	Hidden layers: relu Output layers: sigmoid	Hidden layers: relu Output layers: sigmoid	Hidden layers: tanh Output layers: sigmoid	Hidden layers: relu Output layers: sigmoid
Optimiser	Adam	Adam	Adam	Adam
Batch size	32	32	32	32

C. Accuracy

The accuracy measure was generated using Eq. 2 to evaluate the accuracy of DL models in distinguishing malware from benign samples, and Fig. 4 shows model performance. As shown,the RNN model outperforms other models and obtains an accuracy of 99%. Next in line was the CNN model, with an accuracy of 98%, and the LSTM and FNN models gave the least accuracy of 97% and 88%, respectively.

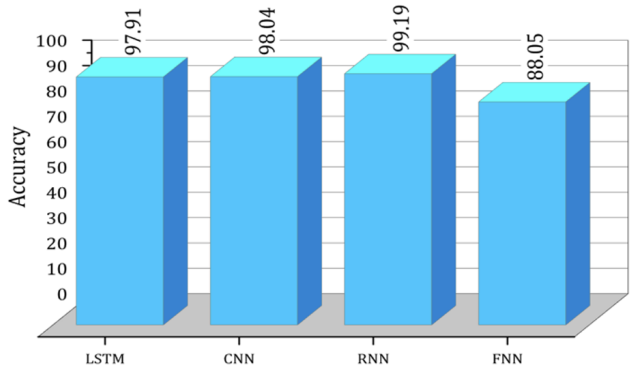

Figure 4: Accuracy measured by the Dl model

D. Precision

The precision of models was determined using the precision performance measurement metric specified in Eq. (3) for this dataset. Fig. (5) depicts the metric-based performance of DL models. As can be seen, the precision for all models is the same at 92%.

ISSN:2222-758X
e-ISSN: 2789-7362

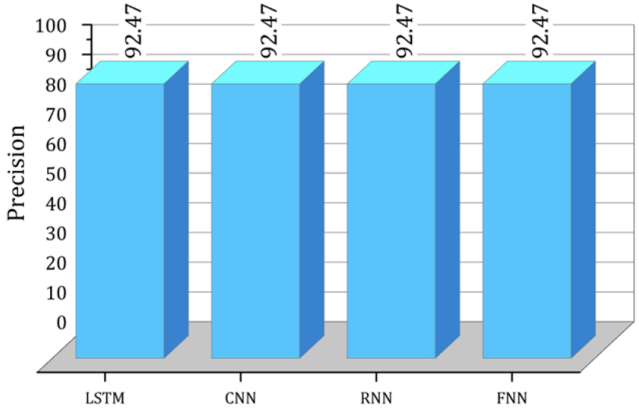

Figure 5: The precision of the DL models

E. Recall

The recall of all models was evaluated using Eq. (4), as shown in Fig. (6). The figure shows that the LSTM model and CNN model gave the same recall of 90%. And the RNN model gave us 91%, and the FNN model gave us 82%.

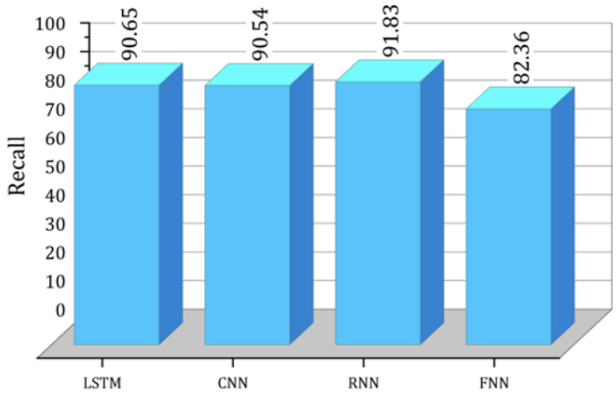

Figure 6: The recall of the DL models

F. F1 measure

Finally, the F1 measurement of DL models was obtained using Eq. (5), as shown in Fig. (7), for the used dataset: LSTM and CNN gave us a result of 91%, and the RNN model and the FNN model gave us 92% and 87%, respectively.

ISSN:2222-758X
e-ISSN: 2789-7362

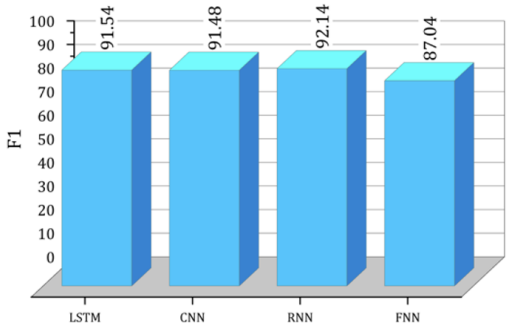

Figure 7: The F1 measure of the DL models

VI. Discussion

Fig. 4 and Table 4, show the training accuracy, training loss, testing accuracy, and training time of all trained DL models over several epochs. The number of epochs that the proposed model needs to get the best outcomes. For dataset, initially, each model is evaluated with a batch size of 64,128 , or 512 records, which provided appropriate timing but poor performance for all models, as opposed to a smaller batch size, such as 32 records, which produces a better result but takes longer to process. As shown in Fig. 4 and Table 4, the RNN model reached 99% accuracy in five epochs, and the training took only 356 seconds (about 5.93 minutes). Compared to the proposed RNN model, the proposed CNN was a bit less accurate, with an accuracy rate of 98%, but it took less training time, about 13 seconds in five epochs. For both models (RNN and CNN), the loss for training and validation is shown in Table 4, and it shows that the model can manage this type of data without encountering overfitting.

On the other hand, the LSTM and FNN models show much less accuracy than the previous models, with an accuracy of 97% and 88%, respectively. Additionally, Table II shows an enormous difference between testing and training loss for the FNN model, which indicates the data is overfitted, but this model was the fastest in training time; it takes only 2 seconds. Lastly, the LSTM model gave us an accuracy of 97%, but it took the longest training time, about 6.13 minutes, to be precise, and the model did not face any overfitting. The paper also evaluates the models based on other matrices. Figures 5-7 show the model's performance based on the precision, recall, and F1 measurement; for all models, they achieve the same precision of 92%, indicating that the models messed up in different places but have the same precision. In terms of recall, the LSTM model and the CNN model manage to have the same recall values of 90%, and for the RNN and FNN models, as before, they show results of 91% and 82%, respectively. Lastly, the F1 measurement was calculated, and the RNN model showed the highest results of 92%, followed by the CNN and LSTM models, which gave us a measure of 91%. Finally, the FNN model received 87%.

TABLE IV
TRAINING AND TESTING RESULTS FOR THE IMPLEMENTED MODELS

DL Algorithm	Convolutional neural network (CNN)	Simple Recurrent Neural Network (RNN)	Long Short-Term Memory (LSTM)	Feedforward neural network (FNN)
Epochs	5	5	5	5
Accuracy training	99.10	99.00	99.10	91.10
Loss training	0.0513	0.0404	0.0491	0.8550
Accuracy testing	98.05	99.19	97.91	88.05
Loss testing	0.0578	0.0447	0.0735	3.2395
Timing	13 s	356 s	368 s	1.40 s

VII. CONCLUSION

With the advent of the IoT and the widespread usage of IoT networks to deliver a wide range of high-quality electronic services, fraudsters are attempting to exploit IoT networks and devices to damage their performance. Furthermore, they endanger the privacy of information sensed, processed, and sent through networks, and there is an ongoing battle between virus writers and security groups. This article used four alternative models on a collection of malicious and benign IoT samples to analyze and evaluate state-of-the-art deep learning algorithms. Several tests were conducted to examine the dataset, sample distribution, and score of every attribute for classification tasks. Overall, the RNN and CNN models provided the highest accuracy, at 99.19% and 98.05%, respectively. This result can be improved by using an ensemble of the two models. At the same time, maintain a low training time by efficiently tuning the hyperparameters of the models.

Funding

None

ACKNOLEDGEMENT

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] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Future Generation Computer Systems*, vol. 78, Elsevier, pp. 544–546, 2018.
[2] J. Jeon, J. Kim, S. Jeon, S. Lee, and Y.-S. Jeong, "Static Analysis for Malware Detection with Tensorflow and GPU," in *Advances in Computer Science and Ubiquitous Computing*, Springer, 2021, pp. 537–546.
[3] S. Nakhodchi, A. Dehghantanha, and H. Karimipour, "Privacy and security in smart and precision farming: A bibliometric analysis," in *Handbook of Big Data Privacy*, Springer, 2020, pp. 305–318.
[4] S. Walker-Roberts, M. Hammoudeh, and A. Dehghantanha, "A systematic review of the availability and efficacy of countermeasures to internal threats in healthcare critical infrastructure," *IEEE Access*, vol. 6, pp. 25167–25177, 2018.
[5] A. D. Dwivedi, L. Malina, P. Dzurenda, and G. Srivastava, "Optimised blockchain model for internet of things based healthcare applications," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019, pp. 135–139.

[6] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, "A decentralised privacy-preserving healthcare blockchain for IoT," *Sensors*, vol. 19, no. 2, p. 326, 2019.
[7] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, Q. Zhang, and K.-K. R. Choo, "An energy-efficient SDN controller architecture for IoT networks with blockchain-based security," *IEEE Trans Serv Comput*, vol. 13, no. 4, pp. 625–638, 2020.
[8] A. Yazdinejad, R. M. Parizi, G. Srivastava, A. Dehghantanha, and K.-K. R. Choo, "Energy efficient decentralised authentication in internet of underwater things using blockchain," in *2019 IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1–6.
[9] M. Zhong, Y. Zhou, and G. Chen, "Sequential model-based intrusion detection system for IoT servers using deep learning methods," *Sensors*, vol. 21, no. 4, p. 1113, 2021.
[10] P. Newman, "IoT Report: How Internet of Things technology growth is reaching mainstream companies and consumers," *Business Insider*, vol. 28, 2019.
[11] W. Peters, A. Dehghantanha, R. M. Parizi, and G. Srivastava, "A comparison of state-of-the-art machine learning models for OpCode-based IoT malware detection," in *Handbook of Big Data Privacy*, Springer, 2020, pp. 109–120.
[12] Gaurav A, Gupta BB, Panigrahi PK, "A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information systems," *Enterprise Information Systems*, 2022 Jan 8:1–25.
[13] A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Big data and Internet of Things security and forensics: Challenges and opportunities," *Handbook of Big Data and IoT Security*, pp. 1–4, 2019.
[14] P. N. Bahrami, A. Dehghantanha, T. Dargahi, R. M. Parizi, K.-K. R. Choo, and H. H. S. Javadi, "Cyber kill chain-based taxonomy of advanced persistent threat actors: Analogy of tactics, techniques, and procedures," *Journal of information processing systems*, vol. 15, no. 4, pp. 865–889, 2019.
[15] H. F. Atlam and G. B. Wills, "IoT security, privacy, safety and ethics," in *Digital twin technologies and smart cities*, Springer, 2020, pp. 123–149.
[16] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences (New York)*, vol. 460, pp. 83–102, 2018.
[17] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019.
[18] K. Bolouri, A. Azmoodeh, A. Dehghantanha, and M. Firouzmand, "Internet of things camera identification algorithm based on sensor pattern noise using color filter array and wavelet transform," in *Handbook of Big Data and IoT Security*, Springer, 2019, pp. 211–223.
[19] S. Homayoun et al., "DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Generation Computer Systems*, vol. 90, pp. 94–104, 2019.
[20] T. Mackey, J. Kalyanam, J. Klugman, E. Kuzmenko, and R. Gupta, "Solution to detect, classify, and report illicit online marketing and sales of controlled substances via Twitter: using machine learning and web forensics to combat digital opioid access," *Journal of Medical Internet Research*, vol. 20, no. 4, p. e10029, 2018.
[21] B. M. Khammas, "The performance of IoT malware detection technique using feature selection and feature reduction in fog layer," in *IOP Conference Series: Materials Science and Engineering*, 2020, vol. 928, no. 2, p. 022047.
[22] B. M. Khammas, S. Hasan, N. Nateq, J. S. Bassi, I. Ismail, and M. N. Marsono, "First Line Defense Against Spreading New Malware in the Network," in *2018 10th Computer Science and Electronic Engineering (CEEC)*, 2018, pp. 113–118.
[23] A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust malware detection for the internet of (battlefield) things devices using deep eigenspace learning," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 88–95, 2018.
[24] N. Naveen, M. A. Safwan, T. G. Manoj Nayaka, and N. Nischal, "Deep Learning Based Malware Detection for IoT Devices," in *ICDSMLA 2020*, Springer, 2022, pp. 1247–1254.
[25] S. A. Hamad, Q. Z. Sheng, and W. E. Zhang, "BERTDeep-Ware: A Cross-architecture Malware Detection Solution for IoT Systems," in *2021 IEEE 20th International Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom)*, 2021, pp. 927–934.
[26] J. Jeon, B. Jeong, S. Baek, and Y.-S. Jeong, "Hybrid Malware Detection Based on Bi-LSTM and SPP-Net for Smart IoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4830–4837, 2021.
[27] A. N. Jahromi et al., "An improved two-hidden-layer extreme learning machine for malware hunting," *Computer Security*, vol. 89, p. 101655, 2020.
[28] G. Radhakrishnan, K. Srinivasan, S. Maheswaran, K. Mohanasundaram, D. Palanikkumar, and A. Vidyarthi, "A deep-RNN and meta-heuristic feature selection approach for IoT malware detection," *Materials Today Proceedings*, 2021.
[29] D. Vasan, M. Alazab, S. Venkatraman, J. Akram, and Z. Qin, "MTHAEL: Cross-architecture IoT malware detection based on neural network advanced ensemble learning," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1654–1667, 2020.
[30] A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, pp. 1–5.
[31] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D*, vol. 404, p. 132306, 2020.
[32] W. Zhang, H. Li, Y. Li, H. Liu, Y. Chen, and X. Ding, "Application of deep learning algorithms in geotechnical engineering: a short critical review," *Artificial Intelligence Review*, vol. 54, no. 8, pp. 5633–5673, 2021.
[33] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: A novel honeypot for revealing current IoT threats," *Journal of Information Processing*, vol. 24, no. 3, pp. 522–533, 2016.

