AN UNSUPERVISED MALWARE DETECTION SYSTEM FOR WINDOWS BASED SYSTEM CALL SEQUENCES

Ragaventhiran J[1], Vigneshwaran P[2*], Mallikarjun M Kodabagi[3], Syed Thouheed Ahmed[4], Prabu Ramadoss[5], Prisma Megantoro[6]

[1]School of Engineering, Presidency University, Bengaluru, Karnataka, India

[2]Department of Networking and Communications, SRM Institute of Science and Technology, Kattankulathur, Chennai, India

[3,4]School of Computation and Information Technology, REVA University, Bengaluru, Karnataka, India

[5]Senior Solution Architect, Dassault Systemes Austrila Pty Ltd, West Perth, WA, Austrila

[6]Faculty of Advanced Technology and Multidiscipline, Universitas Airlangga, Indonesia

Email: vigneshwaranpandi1981@gmail.com[2*] (corresponding author)

DOI: https://doi.org/10.22452/mjcs.sp2022no2.7

ABSTRACT

Malware attacks have grown in prominence in recent years, posing severe security risks and resulting in significant financial losses. The ability to rapidly and reliably classify malware is vital to cybersecurity due to the exponential growth of malware variants. The role of artificial intelligence plays a significant role in cybersecurity industry. Recently, in the field of malware detection deep learning technique seeks more attention than the machine learning techniques due to the complexity of its behavior. Because the deep learning technique performs well than the machine learning techniques in terms of accuracy and it is well suited for large amount of data. The input attribute for the proposed model is windows-based system call sequence which is collected from NT mal detect project. In this work, the unsupervised deep learning technique used for text classification namely LSTM autoencoder and the performance of proposed model compares with existing DL methods such as CNN, RNN and LSTM with the performance parameters of accuracy, precision, recall and F1-measure.

Keywords: LSTM Autoencoder, LSTM, RNN, CNN, Malware Detection, PE files.

1.0 INTRODUCTION

In the era of digital world, storing and manipulating all the records are converting in the form of digitized documents. The rise of digitized document facilitates the backup process and it reduces the workload of humans. The advantage of digitized document has equivalent form of disadvantages too in form of malicious attack established for the system. The establishment of malicious attack can be done for many reasons which includes information theft, identity theft, demanding money, to take control of remote system etc., This scenario shows that always a system getting expose to influence its information which is reside on it. This phenomenon shows that there is a high alert security system has to be provided. The term malware can be defined as any malicious piece of code or software which causes harm to the system. There are numerous techniques has been in industry to protecting the malware and it can also broadly classify into two categories they are static and dynamic malware analysis [1] [2]. Both of the detecting techniques relies on the process of significant feature from the malware sample which has the capability to expose the behavior of malware. In static malware detection approach, the features were taken without executing the malware sample for example string pattern, opcodes and bytes sequences [3]. Dynamic malware detection, in contrast to static malware detection, extracts features by executing malware samples in a virtual environment, such as registry changes, system calls, API calls, network traffic, and so on. [4]. While comparing these two detection techniques, it shows that static method fails to detect new variants of malwares [5] [6] so it can conclude that dynamic based approaches show better performance in form of detection rate than the static based approach. They used machine learning approach such as SVM, Nave Bayes, Decision Tree, and K-nearest neighbour to learn and classify the retrieved features of both normal and malicious files for the above-mentioned two approaches [7] [8] [9]. But in some cases, in it is proven that machine learning algorithms have been shown to be capable of producing higher accuracy and

larger amounts of data since it does not extract required meaningful pattern from the extracted features to learn the behavior of the malicious file [10]. In this regard, the malware detection technique was replaced by deep learning technique in place of machine learning techniques to improve high detection rate for enormous amount of data.

As a result, the windows-malware detection framework used in this study is based on deep learning. The system calls were obtained as an input parameter from both benign and malware files, and these calls were fed into the deep learning model to improve the efficiency of the created model. This scenario helps to detect even unknown pattern of new incoming malware files.

The summary of the contribution in this work has been enumerated below:

- The data preprocessing has been carried out using TFID (Term Frequency Inverse Document) vectorizer to extract significant features from the system call sequence
- Visualization of windows-based system calls for both benign and malicious samples was done in form of word cloud representation
- The process of classification was done by implementing the deep learning algorithm using LSTM Autoencoder
- Finally, the proposed deep learning solution were compared and analyzed against well known deep technique used malware detection system such as CNN, RNN and LSTM-RNN

2.0 RELATED WORKS

Amer et al. [11] suggested a malware detector that extracts API programs simultaneously sequences from Windows PE files. Author has utilized Markov chain model to predict and classify the benign and malicious call sequences. Xie et al. [12] suggested a malware detection algorithm for cell phone based malicious files. Here hidden Markov model is deployed to discriminate benign system calls from malicious calls. Ravi et al. [13] presented a model which uses windows API call sequences as an input attributes in order to distinguish between malicious samples from the normal files. To study the behavior pattern and to do prediction of system call sequences of malicious files association-based rule mining has been used here. This approach makes to detect the malware during run time also. Ki et al. [14] presented a model to detect the malicious API call sequence using DNA sequence alignment algorithms. This algorithm capable of detecting new unknown windows based malicious samples. By collecting API call sequences, Tang et al. [15] established a malware detection tool for Windows systems. Then these sequences were transformed into color feature map and it is turned to be color images. Furthermore, these images are then classified into benign and malicious using Convolutional neural network. Xiaofeng et al. [16] anticipated a combined model for malware detection framework which is built upon a machine learning as well as deep learning technique. The first component of the model captures the association between a large number of extracted system calls, while the second half performs the categorization. Wang et al. [17] suggested a malware detection method based on API calls, with RNN autoencoder used to identify these extracted API calls. These autoencoders can also be constructed to detect various variations of malicious samples by combining multiple decoders.

By using windows-based API call sequences, Asha et al. [18] established a paradigm for multiclassifying malware families. Here author has utilized Rete algorithm to generate rule-based pattern matching process. Apart from this, multidimensional random forest has been used for the process of classification. At the edge of discussion part of the existing solution, it given a decision that only few of the work are relies on deep learning technique and none of the work are depends on unsupervised technique. Since the non-availability of labelled malwares in the formation of malware dataset, here unsupervised technique plays a huge role to tackle this situation. Here System calls were considered as input instead of API calls, and to study its behavior pattern LSTM autoencoder has been incorporated in an unsupervised way. In recent days LSTM autoencoder seeks more attention towards for text classification. System calls sequence is in the textual form, hence LSTM autoencoder has chosen here for malware detection process. Majchrzycka et al. [19] developed a Secure Development Strategy model to overcome problems such as data leaks and to safe guard from outsider attacks in mobile devices. Also, they propose an iSec security tool which was designed to implement security in mobile devices. The Table 1 shows the comparison of existing systems with the methodologies used and accuracy. Anshumaan et al. [22] proposed a Email spam classification method using LSTM and Bi-LSTM model. The Bi-LSTM model achieved a higher performance. Islabudeen et al. [23] proposed a smart approach for intrusion detection and prevention system to ease attacks using machine learning approaches.

Table 1: A comparison of existing systems

Authors	Dataset and samples used	Methodology	Performance
Cesare, S et al., 2014 [1]	mwcollect alliance network with 17,430 real malware from honeypots.	The feature vectors of string-based signatures are used to create a distance measure. Static analysis	Q-Grams - 0.62 Q-Grams + Optima Distance - 0.43
Galal, H.S., et al., 2016 [2]	9993 samples from VirusSign	Behaviour based features model using DT, RF and SVM	Decision Tree - 97.19%, Random Forest - 96.84%, SVM - 93.98%
Burnap, P et al., 2018 [6]	VirusTotal API. The PE contains 594 harmful files.	A Logistic Regression model based on an ensemble classifier.	SVM - 68.08%, MLP - 79.40%, BayesNet - 77.70%, RF - 86.52%, MOPR - 93.76%
Fan et al., 2018 [7]	Genome Project dataset, Drebin Dataset, FallDroid - I, FallDroid - II	FalDroid	FalDroid - 94.2%
Lin et al., 2018 [8]	4-gram API fragment sequence	privacy-preserving Naive Bayes classifier (PP-NBC).	94.93%
Xiao F et al., 2019 [10]	malware samples from VX Heaven.	Stacked AutoEncoders and the Behavior-Based Deep Learning Framework (BDLF)	98.60%
Amer E et al., 2020 [11]	Intelligence and Security Informatics Data Sets brazilian-malware- dataset	Using Markov chain sequences to depict the link between API functions to represent malware and goodware	Prediction - 0.997 FPR of 0.000 FNR of 0.007.
Xie L et al., 2010 [12]	Data from monitoring of I/O events examine correlations of these events.	Behavior based malware detection system named pBMDS	Detection rate - 92.1% False Positive - 6.3% False Negative - 1.6%
Ravi C et al., 2012 [13]	VXHeavens and benign executable samples collected from a freshly installed copy of Windows XP.	3rd order Markov chain (4-grams)	99.38%
Ki Y et al., 2015 [14]	Malicia-project and VirusTotal samples totaling 23,080	Dynamic analysis. DNA sequence alignment algorithms	99.80%
Tang M et al., 2019 [15]	The Virus Share community has 9 virus families, each with 1000 variants.	Visualization and deep learning techniques are used.	True Positive Ratio, precision, recall and F1 > 99%, FPR < 0.1%
Xiaofeng, L et al., 2019 [16]	Malicious samples from Virus Share and VirusTotal, as well as samples from Windows 7 and Windows XP system exe files	dynamic behavior	AUC 99.3%
Wang X et al., 2016 [17]	Public API call sequence dataset (Kim 2016). For coarse grained evaluation 7430 samples and fine grained evaluation 4932 samples.	dynamic behavior	99.90%

3.0 PROPOSED MODEL

This section gives the detailed view of proposed solution and the working of the inner components. In figure 1 the techniques involved in each modules of the proposed methodology can be visualized clearly. The modules present in the proposed solution consists of data collection stage, data preprocessing stage, Dimensionality reduction stage, classification stage and finally evaluation.

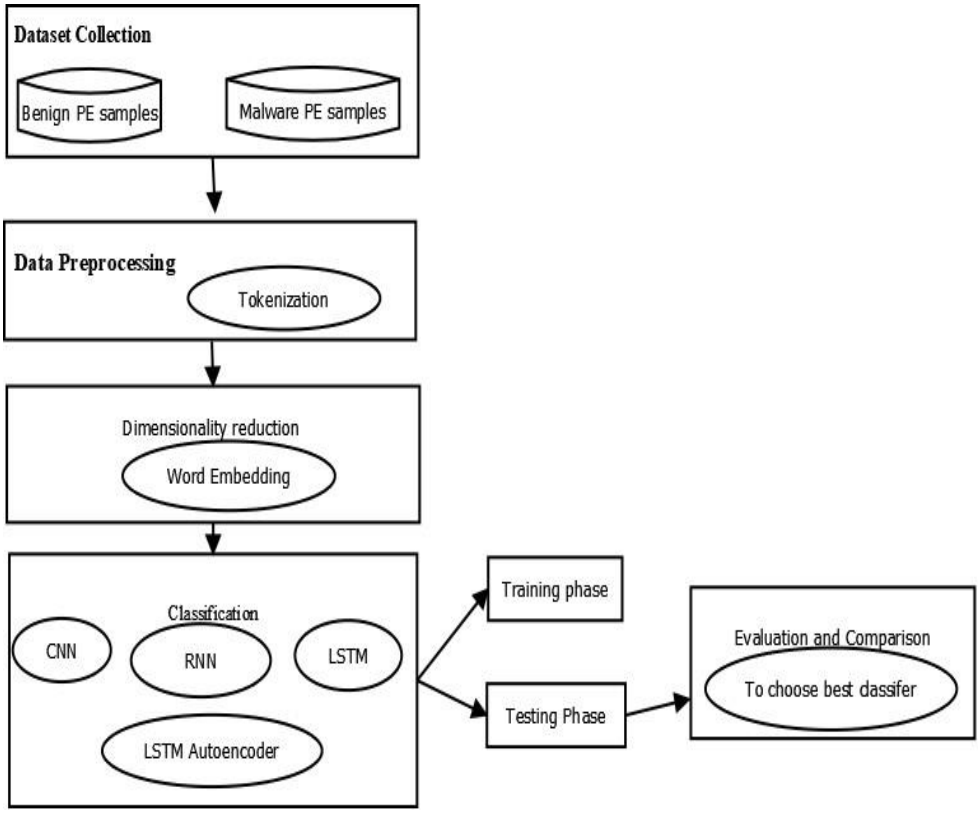

Fig. 1: The Malware Detection System's Proposed Architecture

3.1 Dataset Collection

The dataset used for this work is taken from the project called Ntmaldetect [20]. The dataset comprises of API system calls which are extracted from the windows-based PE format files. This database includes both normal and malicious samples placed in separate folders. This dataset has 73 benign files and 152 malware files. All the system calls were placed in form of text sequence in the .txt file. Figures 2 and 3 are examples of the word cloud representation for benign file as well as malware file correspondingly. 'NtQueryVirtualMemory' is the maximum occurrence system call in the benign file Whereas 'NtProtectVirtualMemory' is the maximum occurrence system call in the malicious file. In these two-word cloud, the two 100 maximum occurred system calls has been given. Based on the text size, its frequency can be determined. The larger size of text shows the high rate of recurrence and its text size decreased gradually based on its frequency.

Fig. 3: Word Cloud for normal file

3.2 Data preprocessing

The collected data is in the form of text, so here tokenization process has been carried to make the textual data suitable for deep learning architecture. Tokenization is the process of breaking the entire sentences into individual words or tokens. The process of breaking can be depending upon the language gets preferred. Suppose for instance if the system call sequence has 40 system calls means it will be break into 40 tokens

3.3 Dimsenonality Reduction

In this stage, the required quantity of significant features will be extracted from the tokens. For this task to accomplish, the word embedding technique has been employed using embedding layer given in Keras package of Python. This technique is used to assign meaningful dense vectors for each tokens or words. It is also can be defined as the updated model of Bag of words (BAG) model. The dense vectors extracted from the words can also be used for another work. Finally, the designed embedding layer can be used as the first layer for the deep learning model or classifier.

3.4 Classification

In this stage, the classification process are carried out using four CNN (Convolutional neural network), RNN (Recurrent neural network), Long short-term memory (LSTM), and LSTM autoencoder are examples of deep learning approaches. This classification layer accustomed to distinguish between malicious and lawful API call sequences. In table 1 the tuned hyperparameters used for the four deep learning technique has been illustrated. The techniques used for API calls are listed and described below:

3.5 Convolutional Neural Network

It's a form of neural network technology that's been built specifically for image processing. This type of neural network requires only less pre-processing techniques. This architecture consists of filtering, maxpooling, flatten and convolution layers. The connectivity of the neurons present in this architecture resembles the working of human brain. It has recently attracted interest in the field of natural language processing techniques.

3.6 Recurrent Neural Network

It's a type of artificial neural network in which it has internal memory to process the long sequences for prediction. The neurons of the layers in the architecture of RNN is connected in the form of directed graph along temporal space. Since the RNN has the ability to remember the information about the previous task, so its current state of decision is partially depending on past experience. In this way, it is well suited for prediction like task.

3.7 LSTM

It is just an updated neural network model of RNN with same functionality. The two technical problem exhibits by the RNN are vanishing gradients and exploding gradients were resolved by the LSTM. The internal structure of LSTM is built by LSTM layers which composed of multiple recurrent connected blocks in other words it can say that memory blocks. Each block comprises of multiple memory cells along with three multiplicative units called input gate, output gate and forget gates. This kind of architecture design provide continuous manipulation operation of memory cells.

In a typical LSTM unit in figure 4, A cell, an input gate, an output gate, and a forget gate are all present. Three gates control the flow of data into and out of the cell, allowing it to retain information for an indefinite period of time. Ideally, it makes LSTM because crucial events can have unpredictable lags, for analysing time series data and making predictions the vanishing gradient problem might occur when training typical RNNs. LSTMs were created to solve this problem. This means that when there are time gaps of more than between relevant input events and target signals, there are 5 to 10 discrete time steps ordinary RNNs cannot learn. The vanishing error problem calls into question RNNs' practical advantages over time window-based feedforward networks. LSTM, a more contemporary model, is not affected by this issue. To bridge time gaps greater than 1000 discrete time steps, LSTM can learn to enforce "constant error carrousels" (CECs) within specific units, known as cells, to ensure that errors flow in the same direction.

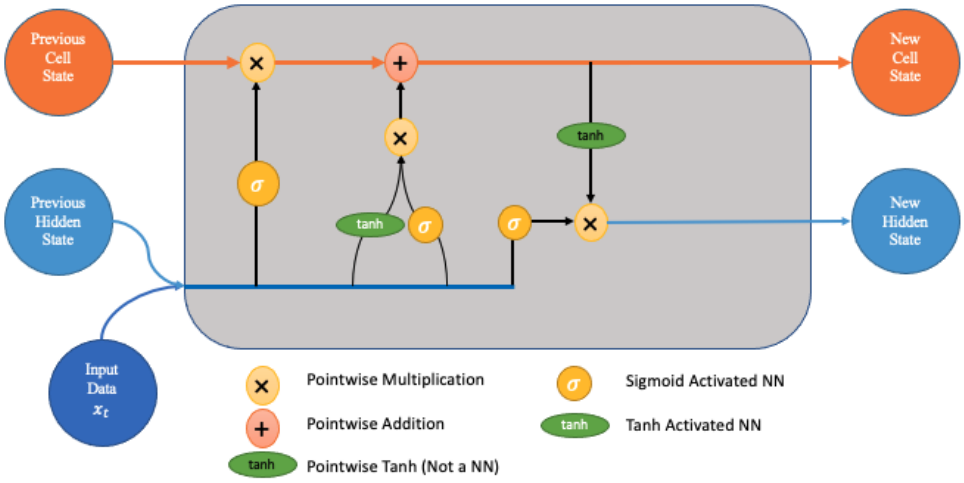

Fig. 4: Architecture of LSTM

3.8 LSTM Autoencoder

LSTM autoencoder is a special variant implementation of LSTM which possess two blocks they are encoder and decoder.

The encoder part is used to compress the input data sequence and the decoder part is used to recreate the compressed the sequence data. This architecture has dropout layer, time distributed layer, LSTM layer, repeat

vector layer, and LSTM layer. The intention of dropout layer is to eliminate overfitting problem, the repeat vector is to repeat the input for 'n' number of times. Finally, the time distributed layer is to develop a vector of output layer size by retrieving the information from the previous layer. The number of neurons employed in the architecture, as well as the number of layers of LSTM autoencoder for this work has been visualized in the figure 5&6.

3.9 Evaluation

Finally, the evaluation stage is completed to determine the proposed model's compliance with several quality measures such as accuracy, precision, recall, and F1-measure, as shown in Table 2. The test dataset is utilized to conduct various test for evaluation purposes. Using these metrics, a comparison of three more deep learning algorithms, including CNN, RNN, and LSTM, with the proposed LSTM autoencoder was performed. It is demonstrated towards the conclusion of the evaluation that the proposed method outperforms the remaining deep learning strategies.

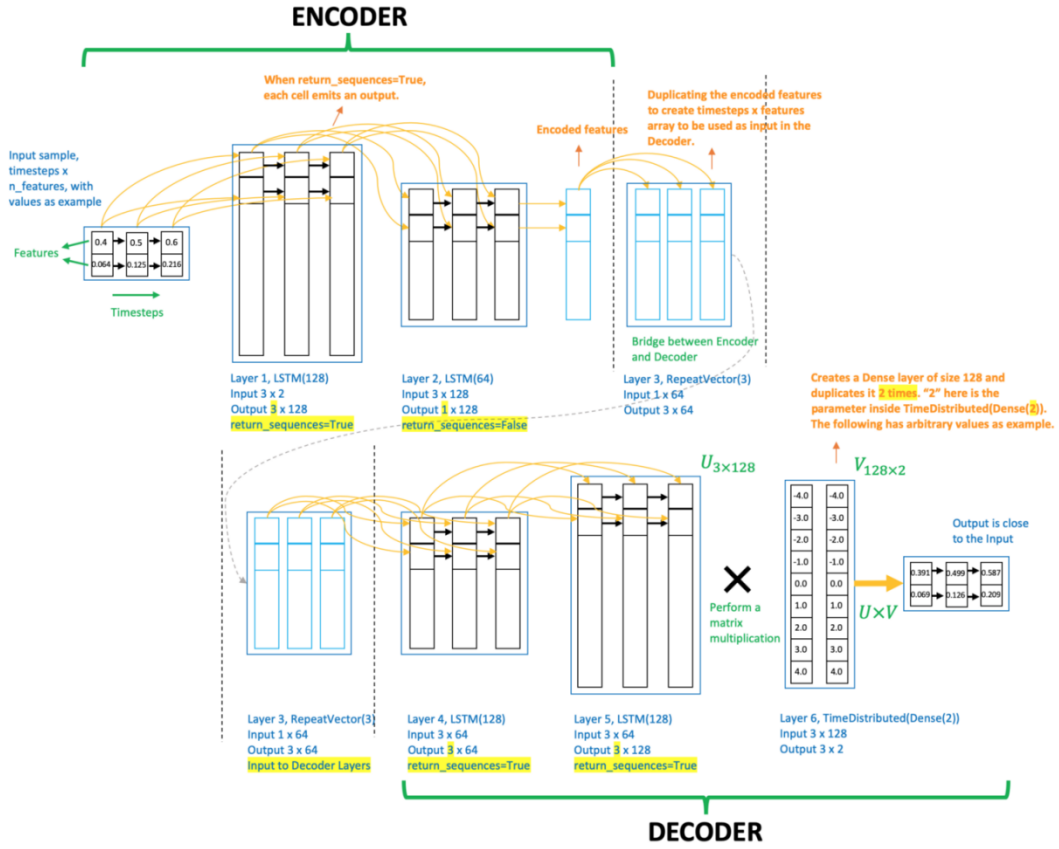

Fig. 5: LSTM Autoencoder flow diagram

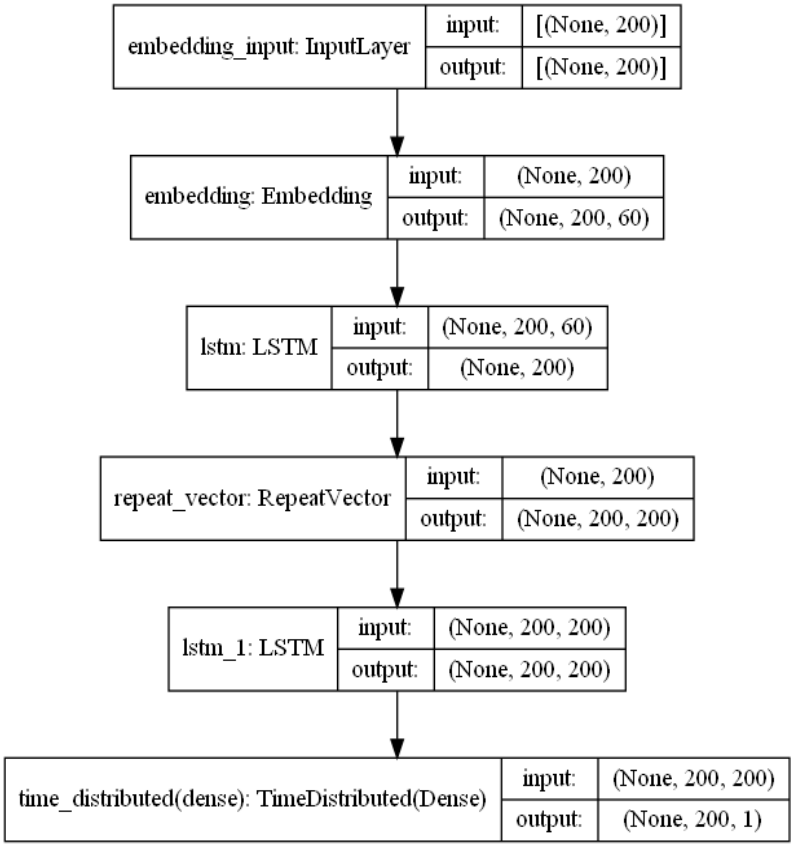

Fig. 6: Architecture of LSTM Autoencoder

4.0 RESULTS AND DISCUSSION

The experimental setup for this proposed malware detection framework has been done using python version 3.7 along with tensorflow as backend process. The model has tested and evaluated in the windows 10 operating system with 4GB RAM at 1.23Ghz frequency of CPU speed. Accuracy, precision, recall, and F1-measure were used to evaluate the suggested model's performance. The description of each performance metrics with its respective equation has been given below.

Table 2: Values for Deep Learning Model Hyperparameters

HyperParameters	**Values**			
	CNN	**RNN**	**LSTM**	**LSTM Autoencoder**
Epochs	10			
Hidden layers	2		1	3
Activation function	output layer - sigmoid function Hidden layer - Relu function			output layer - time distributed function Hidden layer - Relu function
Neurons	output layer - 01 hidden Layer - 10	output layer - 01 hidden Layer - 04	output layer - 01 hidden Layer - 64	output layer - 01 hidden Layer - 200
Optimizer	Adam			
Metrics	Accuracy, Loss (MSE)			
Loss Function	Binary cross entropy			

4.1 Accuracy

It is determined using equation (1) and is defined as the ratio of the number of successfully predicted sequences to the total number of samples in a dataset.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

4.2 Precision

It is described as a classifier's capacity to correctly identify malware samples as attacks. The precision of the classifier is calculated using Equation (2).

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

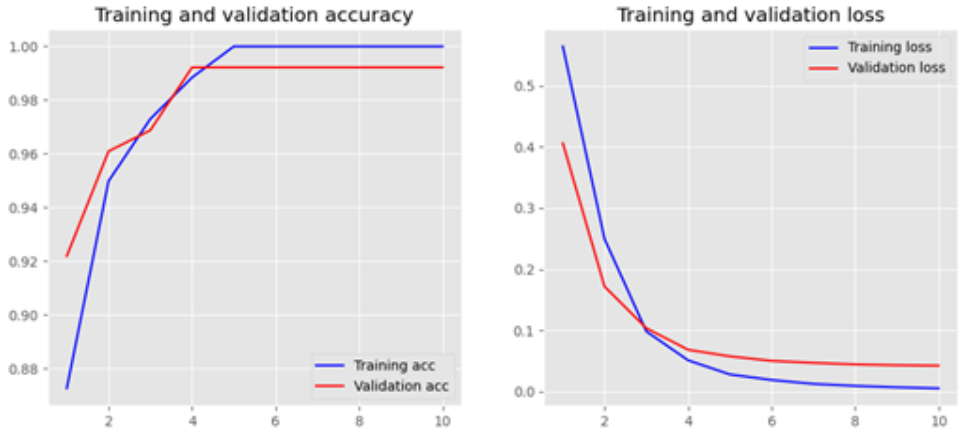

Fig. 7: Performance of CNN

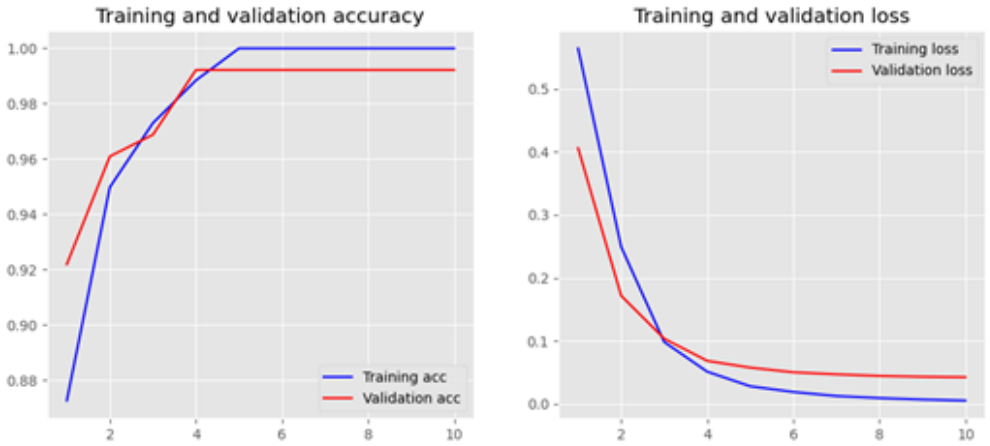

Fig. 8: Performance of RNN

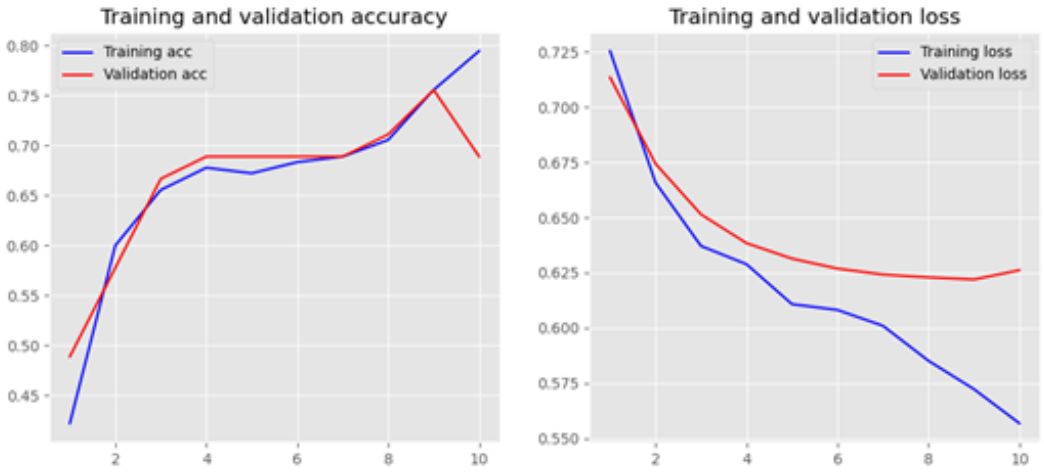

Fig. 9: Performance of LSTM

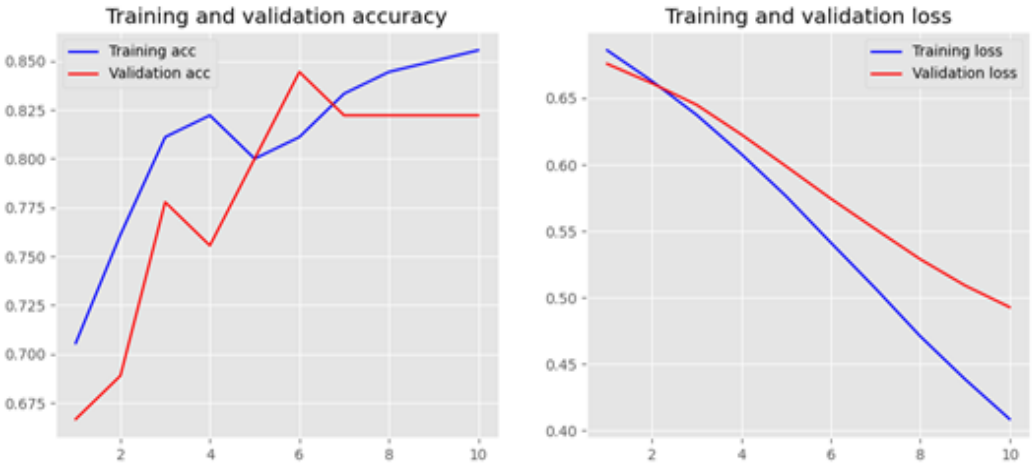

Fig. 10: Performance of LSTM Autoencoder

4.3 Recall or detection rate

It's defined as the number of malicious samples that have been appropriately classified as malware. Equation (3) is used to calculate the recall of the classifier

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

4.4 F1 Score

The weighted melodic mean of precision and recall value is what it's called. For the classifier equation (4) is used to estimate the F-measure.

$$F1 - measure = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{4}$$

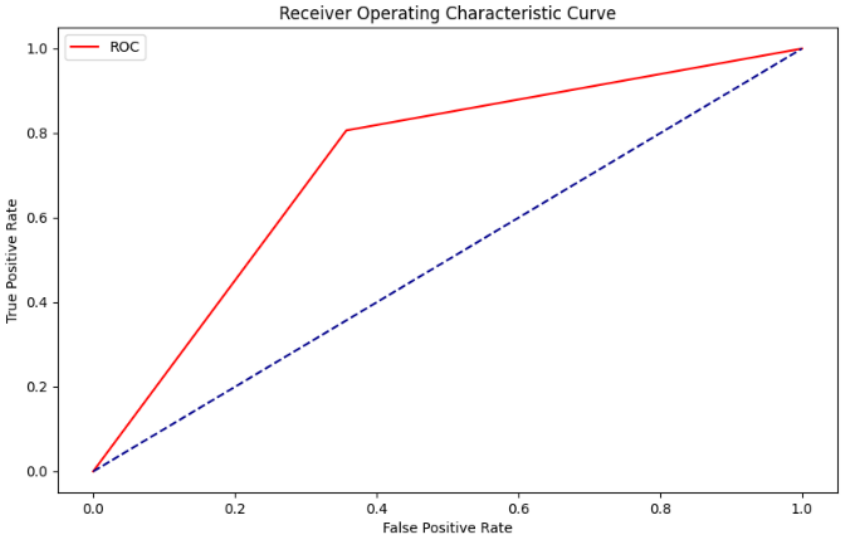

Fig. 11: RoC – CNN

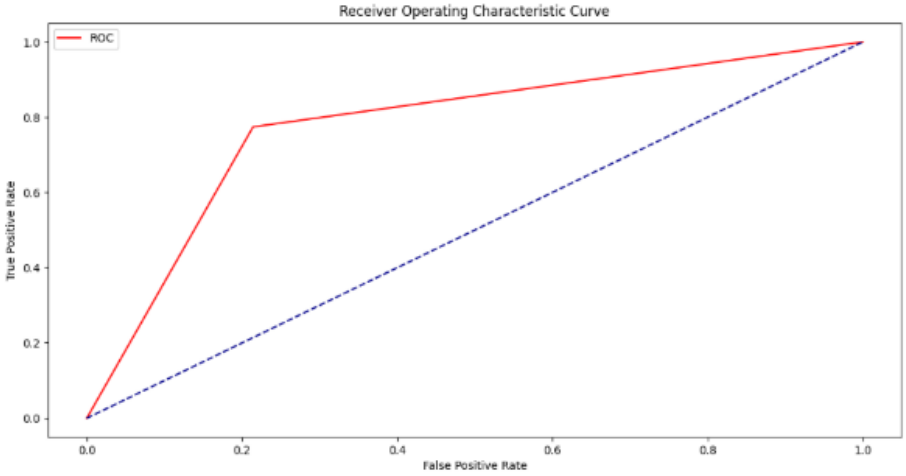

Fig. 12: RoC – RNN

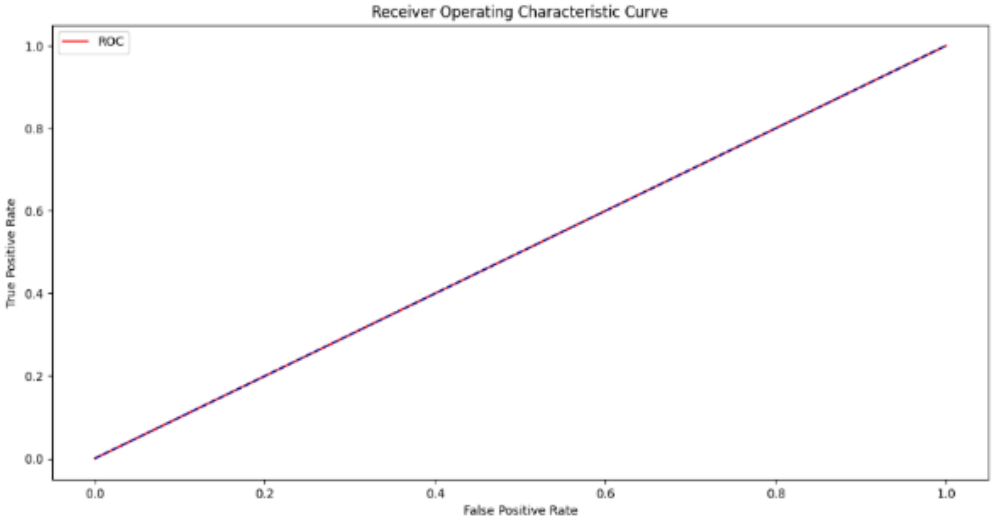

Fig. 13: RoC – LSTM

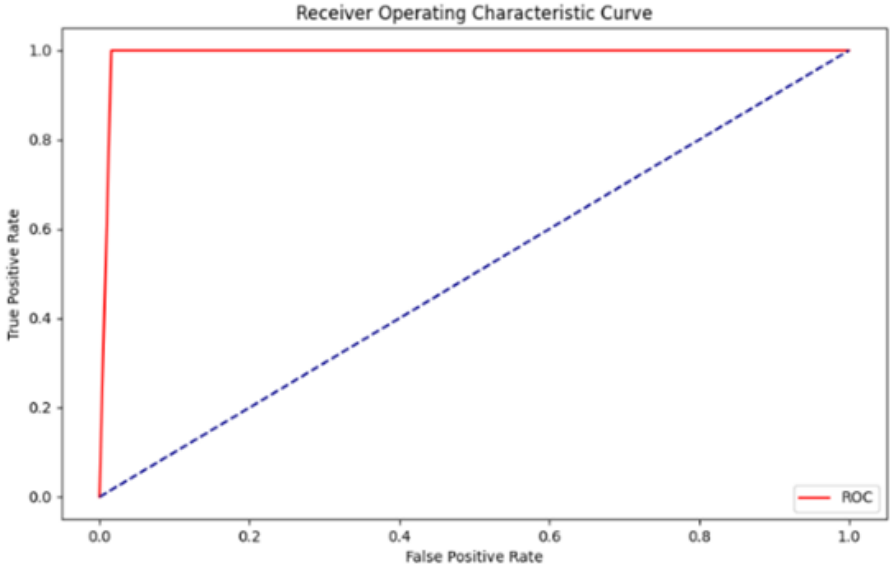

Fig. 14: RoC – LSTM Autoencoder

True positive (TP) are the number correctly labelled malware from malware samples, True Negative (TN) are the number of correctly labelled normal from the normal samples, False Positive (FP) are the number of wrongly labelled malware from normal samples and FN (False Negative) are the number wrongly labelled normal from malware samples. Table 2, the performance metrics for four deep learning techniques has been demonstrated. By analyzing these values, it gives the detailed insights of each technique. Based on these values, it can say that the LSTM autoencoder performs better than the remaining three algorithms. To the next level of LSTM autoencoder, CNN performs better than the LSTM and RNN. Both LSTM and RNN performs in the same range. RoC (Receiver operating Characteristics curve) is the plot between FPR and TPR rate of the binary classifier against different threshold. The figure 11,12,13 and 14 shows the RoC curve for CNN, LSTM, RNN and LSTM autoencoder correspondingly. The RoC curve for LSTM overlaps with the blue dotted line, it shows the poor performance of it. For CNN and RNN, the gap between those lines are moderate but for LSTM autoencoder the gap between those lines are high. The figure 7, 8,9 and 10 shows the accuracy and loss plots between training and testing time for each epochs. Plotted the results of all approaches, including our suggested way, and compared them to other methods, including the LSTM Autoencoder. The method that has been proposed outperforms well with the others. The results are shown in Table 3. As per the result and comparsion of previous methods the Deep Learning model is more systmeatic than the tradional machine learning models.

Table 3: Comparative Analysis of Various Deep Learning Techniques With Proposed Model

Algorithm Used	**Accuracy**	**Precision**	**Recall**	**F1-measure**	**Training Time (sec)**	**Prediction Time (sec)**
CNN	82.22	0.81	0.967	0.882	9.137	0.213
SVM	84.50	0.74	0.95	0.80	12.15	1.5
LSTM-RNN	68.89	0.688	1	0.815	6.761	0.503
RNN	68.89	0.904	0.612	0.73	3.667	0.291
LSTM Autoencoder	**99.22**	**0.984**	**1**	**0.992**	**2.921**	**0.27**

Overall, It is observed from the comparative analysis, the LSTM Deep Learning model gives accuracy of 99.22% among the other existing methods. Meanwhile, the variation in accuracy is not significantly different from the LSTM [22]. LSTM Autoencoder algorithms continue to outperform CNN and SVM methods. In our proposed smethod we have used the LSTM autoencoder with 200 hidden layers and also the words are categorised by using the Bag of Word method. During the preprocessing the tockenizing method is used to make the suitable textual data for deep learning models.

5.0 CONCLUSION

This paper explores about the four deep learning techniques applied for the design of malware detection framework. This model has been evaluated against very small amount of malware dataset. So, the most well performed algorithm like LSTM and RNN yield least detection rate when comparing to the remaining deep learning techniques. In this regard, the unsupervised classifier namely LSTM autoencoder stand in a best position in terms of detection rate. It can capable of providing better performance even for smaller dataset. Here most significant input attribute such as API based system calls has been considered for the designing of malware detection framework. This work can be further improved by incorporating the different set of dynamic based features extracted from the malware samples. Moreover, this model can also be enhanced by performing multilabel classification. The investigating of malware family's classification gives more insights about the new variants of malwares for further analysis process.

REFERENCES

[1] S. Cesare, Y. Xiang, and W. Zhou, "Control flow-based malware variant detection", *IEEE Transactions on Dependable and Secure Computing*, Vol. 11, No. 4, July – Aug 2014, pp. 307-317, doi: 10.1109/TDSC.2013.40.

[2] H. S., Galal, Y. B. Mahdy, and M. A. Atiea,"Behavior-based features model for malware detection", *Journal of Computer Virology and Hacking Techniques*, Vol. 12, No. 2, June 2015, pp. 59–67, doi: 10.1007/s11416-015-0244-0.

[3] E. Gandotra, D. Bansal and S. Sofat, "Malware Analysis and Classification: A Survey", *Journal of Information Security*, Vol. 5, No.2, April 2014, pp. 56-64, doi: 10.4236/jis.2014.52006.

[4] Y. Qiao, Y. Yang , J. He, C. Tang, and Z. Liu, "CBM: Free, Automatic Malware Analysis Framework Using API Call Sequences", In: Sun F., Li T., Li H. (eds) *Knowledge Engineering and Management. Advances in Intelligent Systems and Computing, vol 214. Springer, Berlin, Heidelberg*, 2014, doi: 10.1007/978-3-642-37832-4_21.

[5] D. Ucci., L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis", *Computers & Security*, Vol. 81, March 2019, pp. 123-147, doi: 10.1016/j.cose.2018.11.001.

[6] P. Burnap, R. French, F. Turner, and K. Jones, "Malware classification using self organising feature maps and machine activity data", *computers & security*, Vol. 73, March 2018, pp. 399-410. doi: 10.1016/j.cose.2017.11.016.

[7] M. Fan, J. Liu, X. Luo, K. Chen, Z. Tian, Q. Zheng, and T. Liu, "Android malware familial classification and representative sample selection via frequent subgraph analysis" *IEEE Transactions on Information Forensics and Security*, Vol. 13, No. 8, August 2018, pp.1890-1905, doi: 10.1109/TIFS.2018.2806891.

[8] Z. Lin, X. Fei, S. Yi, M. Yan, X. Cong-Cong and H. Jun, "A secure encryption-based malware detection system." *KSII Transactions on Internet and Information Systems (TIIS)*, Vol. 12, No. 4, April 2018, pp.1799-1818. doi: 10.3837/tiis.2018.04.022.

[9] S. T. Ahmed and S. Sankar, "Investigative protocol design of layer optimized image compression in telemedicine environment". Procedia Computer Science, Vol. 167, April 2020, pp. 2617-2622, doi: 10.1016/j.procs.2020.03.323.

[10] F. Xiao, Z. Lin, Y. Sun and Y. Ma, "Malware detection based on deep learning of behavior graphs", *Mathematical Problems in Engineering*, Vol. 2019, February 2019, pp. 1-10, doi: 10.1155/2019/8195395.

[11] E. Amer and I. Zelinka, "A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence", *Computers & Security*, vol. 92, February 2020, pp. 1-15, doi: 10.1016/j.cose.2020.101760.

[12] L. Xie and X. Zhang., "pBMDS: a behavior-based malware detection system for cellphone devices" *In Proceedings of the third ACM conference on Wireless network security, Hoboken, New Jersey*, ACM, 2010, pp. 37-48.

[13] C. Ravi and R. Manoharan, "Malware detection using windows API sequence and machine learning", *International Journal of Computer Applications*, Vol. 43, No. 17, April 2012, pp. 12-16, doi: 10.5120/6194-8715.

[14] Y. Ki, E. Kim and H. K. Kim, "A novel approach to detect malware based on API call sequence analysis", *International Journal of Distributed Sensor Networks*, Vol. 11, No. 6, June 2015, pp.1-9, doi: 10.1155/2015/659.

[15] S. T. Ahmed and K. K. Patil, "An investigative study on motifs extracted features on real time big-data signals", in *Proceedings of the 2016 International Conference on Emerging Technological Trends (ICETT), Kollam, India*, IEEE, 2016, pp. 1-4. doi: 10.1109/ICETT.2016.7873721.

[16] L. Xiaofeng, J. Fangshuo, Z. Xiao, Y. Shengwei, S. Jing and P. Lio "ASSCA: API sequence and statistics features combined architecture for malware detection", *Computer Networks*, Vol. 157, July 2019, pp. 99-111, doi: 10.1016/j.comnet.2019.04.007.

[17] X. Wang and S. M. Yiu, "A multi-task learning model for malware classification with useful file access pattern from API call sequence*", arXiv preprint arXiv:1610.05945*, October 2016, doi: 10.48550/arXiv.1610.05945.

[18] M. A. Jerlin and K. Marimuthu, "A New Malware Detection System Using Machine Learning Techniques for API Call Sequences", *Journal of Applied Security Research*, Vol. 13, No. 1, December 2017, pp. 45-62, doi: 10.1080/19361610.2018.1387734.

[19] A. Majchrzycka and A. Poniszewska-Marańda, "Secure development model for mobile applications", *Bulletin of the Polish Academy of Sciences Technical Sciences*, Vol. 64, No. 3, September 2016, pp. 495-503, doi: 10.1515/bpasts-2016-0055.

[20] C. W. Kim, "Ntmaldetect: A machine learning approach to malware detection using native api system calls" *arXiv preprint arXiv:1802.05412*, February 2018, pp. 1-8, doi: 10.48550/arXiv.1802.05412.

[21] M. Mohammadi, M. Sarmad, N. R. Arghami. "An Extension of the Outlier Map for Visualizing the Classification Results of the Multi-Class Support Vector Machine" *Malaysian Journal of Computer Science*, Vol. 34, No. 3, July 2021, pp. 308-323, doi: 10.22452/mjcs.vol34no3.5.

[22] Anshumaanmishra, and V. Pandi, "Classifications of E-MAIL SPAM using Deep Learning Approaches", *Advances in Parallel Computing Algorithms, Tools and Paradigms*, Vol. 41, November 2022, pp. 414-421, doi: 10.3233/APC220058.

[23] M. Islabudeen and M. K. K.Devi, "A Smart Approach for Intrusion Detection and Prevention System in Mobile Ad Hoc Networks against Security Attacks", *Wireless Personal Communication*, Vol. 112, No. 1, January 2020, pp. 193–224, doi: 10.1007/s11277-019-07022-5.