Malware Prediction Using LSTM Networks

Malware Prediction using LSTM Networks

Saba Iqbal, Abrar Ullah, Shiemaa Adlan and Ahmad Ryad Soobhany
School of Mathematical and Computer Science,
Heriot-Watt University Knowledge Park, Dubai, UAE
{si104,a.ullah,sia1,r.soobhany}@hw.ac.uk

Abstract: With a recent increase in the use of the Internet, there has been a rise in Malware attacks. Malware attacks can lead to stealing confidential data or make the target a source of further attacks. The detection of Malware has been posing a unique challenge. Malware analysis is the study of malicious code to prevent cyber attacks and vulnerability assessment. This article aims for classification of malware using a deep learning model to obtain an accurate and efficient performance. The system proposed in this study extracts a number of features and trains the Long Short-Term Memory (LSTM) model. The study utilises hyper-parameter tuning to improve the accuracy and efficiency of the LSTM model. The findings revealed 99.65% accuracy using sigmoid function that outperforms other activation function. This work can be helpful in malware detection to improve security posture.

keywords
Malware, Security, Neural Networks, Deep learning, LSTM

1 Introduction

There has been a continuous growth in the use and transmission of data over the Internet worldwide. The are rising concerns for the security of data at transfer and rest from various threats.To handle these threats businesses are adopting novel technologies to detect and mitigate threats that can be compromise their security. Malware has been an ongoing security for many years. The name is derived from malicious code and it comes in various forms such as viruses, worms, trojans, ransomware, rootkits etc. having different functionalities. Malware differ in their behaviour and once the system is compromised, the attacker gets unauthorised access without user's knowledge. Therefore,the requirement of conducting Malware Analysis is needed. It deals with malware functionalities along with their repercussions as discussed by [1]. There is a continous growth in the use of machine learning and deep learning approaches for malware detection. Machine learning is a sub-descipline of Artificial Intelligence where Deep Learning (DL) is a sub-category of machine learning techniques [2]. Deep Learning as also referred as Deep Neural Networks, that consists of multi feed forward layers in order to train the input data based on the type of the dataset.

This paper uses static malware analysis along with deep learning techniques used to reduce the impact of malware attacks or nullify it. [3] emphasized the evolution of malware started since the beginning of the internet world. Advanced malware leads to the inflexibility and inability of the system to detect these newly developed malwares. Even the anti-malware software are unable to detect these advanced

malwares. Hence, an advanced model should be developed which can adapt constantly to these changing malware functionalities and detecting them effectively. The DL-based approaches help systems become more intelligent and adaptive to these advanced malwares, thereby detecting them based on their behaviour.

The contribution of this paper includes prediction of various types of Malwares. The study undertakes the following:

1 Limited Malware Classification analysis systems have been developed to date.
2 The PE dataset of static malware analysis has been used in LSTM model for classification.
3 Comparative Analysis of the LSTM performances using different Hyperparameters.

2 Literature Review

This section explains the evolution of the Malware, followed by Obfuscation techniques with brief instances. The section also discusses related work in the field of Malware analysis using various Machine and Deep learning algorithms.

2.1 Camouflage evolution in Malware

The malware designers use various techniques to circumvent security and penetrate through systems undetected. Therefore, malware analysis is essential to conduct. The various types of camouflage techniques are discussed below by [4].

Encryption: The earliest camouflage method and the first encrypted virus was 'Cascade' introduced in 1987.The antivirus scanner cannot detect the encrypted virus immediately because of its encrypted main body that has to be decrypted first to access the whole malicious code. However, the virus can be detected indirectly using the string signatures of the decrypted part of the virus.

Oligiomorphism: Advanced form of encryption, also known as semi-polymorphic in the concealment of malware comprising of a collection of decryptors randomly chosen for a new target. The antivirus consumes a longer time in detecting this type of malware because they have to decrypt all the decryptor first to access the main code for detection. The first of this type was 'Whale' a Dos virus introduced in 1990.

Polymorphism: It is a complex form of both encryption and oligiomorphism. It encrypts the main code in the same way but the only difference between Polymorphic and oligiomorphic/encryption viruses is that it creates an unlimited number of unique decryptors. Polymorphism works on a principle of modifying the appearance of the code while infecting the system thereby evading the detection by not leaving behind the permanent string signatures of its variants .

Metamorphism: Composed of mutation engine that mutates the whole code unlike polymorphism. The term was coined by Igor Muttik as 'Metamorphics are body-polymorphics'. Each variant as it duplicates differs in structure, code sequence, size and syntactic properties but have the same functionality.

2.2 Obfuscation Techniques

Obfuscation Technique is a process to render unreadable source code or binary sequence of a malware while maintaining its functionality. The various obfuscation techniques are discussed below.

Junk/Dead Code Insertion: The binary sequence is modified using junk or dead instructions without any effect on the functionality or behaviour of the code. The instructions used in junk insertion does not change the value of CPU registers or the memory known to be No-operation (NOP).

Variable/Register substitution: In Variable/Register substitution, the value of registers or memory variables are changed to different values in various versions of virus evading from signature-based detection. The first virus used this obfuscation method, that modifies the binary sequence of the malicious code,was W95. Regswap in 1998. This can be detected using wild card scanning. **Instruction replacement**: It is used to replace instructions with other instructions having the same meaning/ functionality. The instructions, as illustrated below, have the same functionality of setting the register eax value to zero which makes it difficult to detect.

Subroutine reordering This method obfuscates a malicious code by randomly reordering its subroutine as shown below. It generates n! of malware variants, n denotes to the number of subroutines. **Code transposition** In code transposition, reordering sequence of instructions of the original malicious code can be done in two ways. The first method changes the order of the instructions which is then changed back to the original order using jumps or unconditional branches just before execution. The malware using this obfuscation method can easily be detected by removing the jumps and unconditional branches to get back the original malicious code. On contrary, the second method generates new malware variants by reordering independent instructions that do not have any effect on the other instructions. This method is difficult to implement causing a high detection cost.

Code integration:In code integration technique, malware first disassembles the target program, integrates itself to it thereby reassembling the new code into a new variant or generation. The first-ever malware to use this most sophisticated obfuscation technique was Zmist for Windows 95. The detection and recovery of the original malicious code are considerably difficult.

2.3 Deep learning Techniques

Machine learning (ML) used in extracting features, pattern recognition and making predictions based on the input dataset provided. ML divided into Supervised and Unsupervised learning. Deep learning frequently referred to Deep Neural Networks (DNN). The Traditional neural network only have 2-3 hidden layers whereas in DNN it can go up to 150 hidden layers refer to the hidden multilayer as ‘Deep’. These DNN models are trained using large labelled dataset that learns by itself from the input data without human intervention. It works well with a huge amount of data. DNN algorithm learns higher-level features by combining lower level features making the

model learn features at multiple levels of abstraction. It learns complex functions mapping input to output and feature extraction without any human intervention [5].

Artificial Neural Networks (ANN) [6], are inspired by human brains. ANN similar to human brain has neurons or perceptrons, which are the most critical part of ANN. ANN architecture consists of an input layer, output layer and numerous hidden layers. It is also known as Multi-layered perceptrons (MLP) where perceptrons are sigmoid neurons. Each neuron comprises of binary input weights, activation function and output. The predictions made by the neural network performed by the activation function, learning rate and multi-layered structure.The mechanism of neural network is that, each neuron is fed using inputs by the input layer in order to generate different outputs for every single neuron based on weights and bias. The outputs of each neuron from this input layer are weighted and then fed to hidden layers consisting of multiple neurons which perform complex functions. The process of feeding output of one layer as an input for another layer is known to be a feed-forward neural network. The major drawback of this feed-forward neural network is that it does not hold memory for predictions to be made on sequences or time-series. Alternatively, to overcome this drawback, another neural network developed known to be Recurrent Neural Network (RNN) having feedback loops that can store data in memory.Despite the advantage of using the RNN techniques, it has some disadvantages such as, gradient vanishing and exploding problems, and fails to process very long sequences when tanh or relu activation functions are used. RNN models are unable to store information for longer duration and incapable of handling long-term dependencies. Because of these drawbacks, it is difficult to train RNN models. These limitations can be addressed by using Long short-term memory (LSTM) that has considerable functionality of storing information for a longer period of time.

LSTM is a type of RNN, consisting of additional functionality making it superior in predicting time series data. [7] stated that LSTM was introduced to overcome the drawback of short-term memory faced by standard RNN.

The deep learning models in the field of security is already progressing incredibly where malware analysis remains one of the major area providing better and efficient performances as compared to traditional methods. This section discusses the past researches that have been carried out for analysis of the malwares using deep and machine learning.

An LSTM framework consists of three gates shown in Figure 1 that are input gate x_t/x_1 , the output gate O_t, the forget gate f_t, immediate cell state C'_t, cell stateC_t/C_1 and output state h_t/S_1 with their respective weights of unfolded single timestep LSTM architecture illustrated mathematically in below Equation 1& 2 &
3 & 4

$$f_t = \sigma(W_t W_s S_{t-1} + W_f W_x x_t) \quad (1)$$

$$i_t = \sigma(W_i W_s S_{t-1} + W_i W_x x_t) \quad (2)$$

$$O_t = \sigma(W_o W_s S_{t-1} + W_o W_x x_t) \quad (3)$$

$$C'_t = \tanh(W_c W_s S_{t-1} + W_c W_x x_t) \quad (4)$$

Figure 1 concludes that forget gate f_t can be calculated when the input X_1 having some weight and some previous output state S_0 is passed through the sigmoid function. Similarly, the input gate i_t and output gate O_t is calculated. Also, the intermediate cell state $\acute{C}_t$ is calculated similarly.

The information being discarded depends on the forget gate value,which is combined with the previous cell state value. The result is obtained by multiplying the input gate and the intermediate cell state decides about the information to be stored in the cell state. Finally, the new cell state for current input is calculated as below Equation 5.

$$C_t = (i_t * C'_t) + (f_t * C_{t-1}) \tag{5}$$

Hence, a new output stateh_t is evaluated as from Equation 6

$$h_T = O_t * \tanh(C_t) \tag{6}$$

3 Related Work

D'Angelo, G. et al. [9] have proposed their work of malware detection in a mobile Android environment using autoencoders based artificial neural network. They provided the neural network with sparse matrices having two-dimensional representation of images named as API images which represent the behaviour of the mobile application extracted by using dynamic malware analysis. The encoders-based ANN is very effective when it comes to detecting malwares as it continuously draws the most useful and best features from the two-dimensional images provided. [10] proposed the study of malware variants using image-based convolution neural network. Here they converted the raw malware binaries into coloured images using finely tuned convolution neural network for the identification of the malwares. They trained the model using ImageNet dataset and tuned it finely to cover the imbalance in the dataset through the data augmentation technique to improve the performance. They concluded that the image-based CNN has better accuracy of 98.82% which is more compared to other algorithms and the computational cost is lower than grey and coloured images analysis. [11] proposed two ends to end malware detection methods that are DexCNN and DexCRNN. The deep learning models are trained by the resampling the raw bytecodes from the classes.dex of Android applications. They used two resample methods on the classes.dex files to get it into fixed-size sequences required for the deep learning models as inputs. The accuracy for dexCNN achieved was 93.4% and accuracy for DexCRNN came out to be 95.8%. The pre-processed dataset was divided into three sets that were 80% training set, 10% validation set and 10% test set.

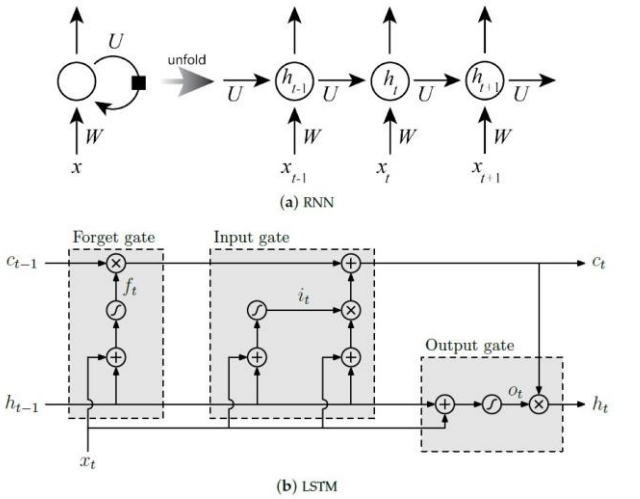

Figure 1 (a) RNN Structure and (b) LSTM Structure Source[8]

Yakura, H. et al. [12] wrote in his research about the importance of studying the malware in byte sequence which characterises its functionalities. He proposed deep learning model where the binary data of the malicious code is converted into image with an attention mechanism-based CNN. The main shortcomings of the proposed method were that the byte-sequence study becomes difficult if the packers use cryptographic methods like AES or RSA encryption. [13] proposed large datasets for the classifications of malware which were previously limited to smaller datasets and were not made public. They trained their model based on Long short-term memory neural network. [14] proposed a deep learning method using word2vec-based LSTM neural network for classification of malware and comparing it with one-hot encoding method. The malware is classified in a disassembled assembly source by extracting both the opcodes and API function thereby vectorizing it using word2vec into fewer dimensions that reduces learning rate and increased classification rate. The vectorised results were then fed to LSTM to evaluate the classification results. Traditional use of either extracting opcodes or API function in malware classification had some limitations.

Sung, Y. et al. [15] also proposed a method of classifying malwares using an advanced version of word2vec model known as fast-Text model based on bidirectional LSTM (BiLSTM). The fast-Text model-based Bi-LSTM is used for classifying malwares using opcodes and API function for more accuracy and also this model includes words in sentences by N-gram algorithms for classification. This paper focuses on extracting

Table 1 Summary of related work using ML/DL techniques

Author	Focussed security domain	DL Techniquese	Year	Accuracy	Conclusion
[9]	Mobile Se-curity	AE	2019		The resulting framework can outperform more complex and sophisticated machine learning approaches in malware classification
[10]	IOTAndroid mobile security	CNN	2020	Maling dataset-98.82% IoT-Android dataset-97.35%	More research is required for the improvement of the efficiency
[16]	Windows	TELM(Twohidden layer extreme learning)	2019	99.65%	Global dependencies of input samples needed to be extracted for both malware and botnet detection. Future work for Improvement of ELM model is required
[17]	-	Multi-level different deep learning system using tree search	2018	-	Training with a larger dataset can improve malware detection accuracy. For diverse data distributions, the traditional method cannot be fitted with deep learning model because of the complex malware data. Further research on using algorithms strategically to decrease computational should be made
[18]	IoT-Environments	GLCM and Machine Learning Techniques	2019	RF-95% Na¨ıve Bayes-89% K-NN -80%	This model can be adopted in real-life applications for detecting known IoT malware using the learned pre-processed image features
[11]		Dex-CNN & Dex-Colorinspired Neural Network	2020	DexCNN-93.4% DexCRNN95.8%	The patterns learned by the proposed model for categorising benign or malicious area index files remain unclear that requires future research
[13]	Multi-class dataset	LSTM based RNN	2019	90.63%	The multiclass dataset used was quite small for Deep learning and the resources required was high with an excellent result. Furthermore, research is required using fewer resources resulting in a good performance
[12]	Public Malware dataset	Attention mechanism with CNN	2019	-	The proposed method has a higher classification accuracy than conventional methods. It reduces the workload of manually analysing the malware by extracting sequences of the malware
[19]	Android	Monte-Carlo based Reinforc ement Learning	2020	-	The framework requires further research of using static analysis with dynamic analysis together as it limits meaningful features individually. A neural network can be used for the production of execution traces that have been trained with previously analysed data
[14]	Microsoft Malware Classification challenge dataset	Word2vec model-based LSTM	2019	97.59%	A requirement of high computing resources needed for using all the opcodes and API function. Word2vec model has better accuracy than one-hot encoding with lesser dimension
[15]	Drones & GCS (Microsoft dataset)	Fast-Text model-based Bi-LSTM	2020	96.76%	This model requires further research on the size of input vector Bi-LSTM as well as classification of malware using a random dataset

features of the malwares and then pre-processing it. The fast-Text model embeds the word2vec and one-hot encoding files with the input files which was used to feed it to Bi-LSTM for the classification of files based on families.

In the proposed method trained model of Bi-LSTM is verified using opcodes and API function where only test dataset is used. It was concluded that API function names with opcodes used for malware classification had greater accuracy than using the opcodes or APIs alone. On comparing with word2vec and one-hot encoding method, the accuracy of the proposed method came to be 1.87% and 0.39% higher than one-hot encoding and word2vec simultaneously. The accuracy achieved by the proposed method came to be 96.76%.

This method requires to determine the size of Bi-LSTM and input vector along with the ability to detect malicious files where the input dataset includes both normal and

malicious files. Future work for classifying malicious or normal file from the randomly fed input dataset will be carried out using the proposed model.

Table 1 critically analyses past research works based on Malware Analysis with Machine learning and deep learning algorithms which also includes accuracy achieved by each model used in respective security domain. We have analysed that the accuracy achieved using both static and dynamic malware techniques performed better in malware predictions than using static or dynamic techniques individually. Further, study stated that using a larger relevant dataset with API functions produces better results. It can be seen from the table that LSTM-based RNN model are frequently used because of its better performance than most of the Deep learning and ML-based models. However, their performances can be improved using ensemble models where different algorithms are combined to build an effective ensemble model. Going further there has been survey research carried out by [20] that focuses on usage of DL in IDSs giving a detailed review, its complete analysis and taxonomy of main DL architectures used in IDSs illustrated in Table 2 They also made a comparison of previous surveys based on cybersecurity DL to identify their drawbacks. They studied the DL solutions in detail that included input data, detection, deployment and evaluation strategies. After the survey, it was noticed that the earliest architectures were AE, DBN and RNN that were studied whereas CNN is newly investigated in 2017. It mentions that the ensemble architectures need to be explored more. The survey further includes proposed IDSs based on deep learning discussing the datasets used and their performances achieved.

The main drawbacks discovered in the survey was that the proposed DL based IDSs lacked the use of proper datasets. Since the proposed models used KD99 or NSL-KDD datasets containing old traffics without any novel attacks traffics without having any real-time properties. Coming onto CICIDS2017 IDS/IPS dataset and NBaIoT dataset containing traffic from stimulated environments overcame this issue. There need to be more researches done in the field of Security using deep learning models. Table 1 shows the comparison made by [20].

4 The proposed model

The main objective of this research is to create a relevant dataset, and then, using the dataset, a comparative study using Deep Learning Technique is done to detect and classify malware. Figure 2 illustrates the proposed model architecture.

For static analysis we have used indicators like file types, hashes, strings and file header data for determination of malicious files.Firstly, the filetype of the malware is found out helping to identify the targeted operating system, for instance PE32 file. The Portable Executable (PE) format is a file format for executable, object code, DLLs and others used in 32-bit and 64-bit versions of Windows operating systems. In addition to these, extracting strings from malicious files gives additional pieces of information and its functionalities.

We have extracted the PE headers of the benign and malware files which is then Pre-processed where features are extracted from headers fields.The tasks in preprocessing step involves normalisation, removal of redundant features and

labeling. Labeling is done either by signature-based anti-virus or cloud-based service where multiple anti-virus engine based scan could be performed in parallel. We labeled each sample in both malware and benign group by cloud based service Virustotal.

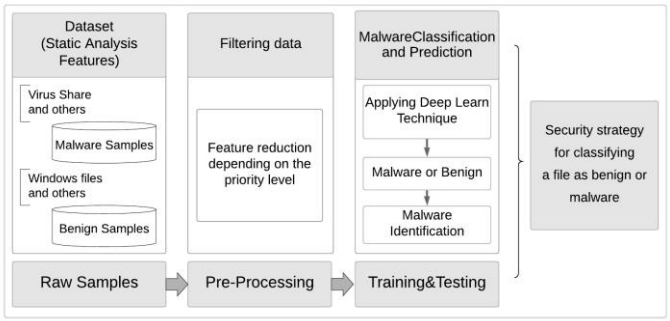

Figure 2 The Proposed Model Architecture

These samples were used in feature extraction where PE header fields are extracted to make a pre-processed dataset for malware analysis. In the second phase, Malware classification and prediction are performed, which involve applying DL techniques, Malware benign classification and Malware identification.

The dataset is then fed to the LSTM network of Deep Learning Technique having a single hidden layer and one input layer. Furthermore, the LSTM model is trained and tested using the training and testing set that has been divided in the preprocessing step in the ratio of 80:20.After implementing the aforementioned phases, the security strategy is precisely defined to classify the file as either benign or Malware.

Table 2 Survey Comparison of Deep Learning Algorithms, Source: [20]

DL Algorithm	Application	Dataset	Feature learning	Classification technique	Accuracy
2*AE	Wi-Fi intrusion (2016)	AWID	Stacked AE	Softmax	97.7%
	Wi-Fi sonation (2018)	AWID	3 SAE	K-means Clustering	94.81%
3*DBN	Network Intrusion (2014)	KDD99	LSTM	LSTM	93.94%
	Network Intrusion (2016)	10% KDD99	DBN	Soft-max regression	97.9%
	InVehicle Network (2016)	Simulation of in vehicular network	DBN	Conventional stochastic gradient descent method	97.8%
4*RNN	Network Intrusion (2015)	KDD99	LSTM	LSTM	93.82%
	Network Intrusion (2016)	KDD99	LSTM	Softmax	96.93%
	Network Intrusion (2018)	KDD99	-	Bi-directional LSTM	84.99%
	Attack detection insocial networks (2018)	NSL-KDD	LSTM	LSTM	97.5%
3*CNN	Network Intrusion (2017)	10% KDD99+NSL-KDD	CNN	SVM+1-NN	1-NN: 96.19% + 86.74% SVM: 95.27% + 77.68%
	Network Intrusion (2018)	KDD99	CNN	Softmax	97.53%

	Network Anomaly (2018)	NSL-KDD+Kyoto Honeypot + MAWILab	-	CNN	Shallow CNN outperforms moderate and deep CNN
2*Ensemble	Network Intrusion (2016)	10% KDD99	AE	DBN and BP for fine tuning	92.1%
	Network Intrusion (2018)	KDD99	1)None 2) STL: sparse AE	1) DNN 2) LSTM	DNN: 66% LSTM: 79.2%
Hybrid	Network Intrusion (2018)	NSL-KDD	GAN	GAN	-

In this section, we describe the dataset used in the analysis. Thereafter, the preprocessing, and exploration of the data are implemented using correlation technique.

4.1 Dataset Description

The data source used is a malware dataset categorized into malicious and benign. The dataset consists of 47580 of malicious and benign files with their 1002 features or information. The dataset is exported from the 'pe imports' elements of Cuckoo Sandbox reports where malware files were downloaded from virushsare.com and goodware files from portableapps.com and Windows7 x86 directories. It contains

Table 3 Top 10 Static features for Malware families

Benign Programs	**Backdoor**	**Constructor**	**Email-worm**	**Hoax**	**Rootkit**
None	None	ExitProcess	GetProcAddress	ExitProcess	GetProcAddress
GetLastError	GetProcAddress	GetModuleFileNameA	LoadLibraryA	GetProcAddress	LoadLibraryA
GetProcAddress	LoadLibraryA	CIcos	GetModuleHandleA	LoadLibraryA	VirtualProtect
CloseHandle	GetModuleHandleA	adj fptan	ExitProcess	GetModuleFileNameA	ExitProcess
GetCurrentThreadId	VirtualAlloc	FreeLibrary	CloseHandle	RegCLoseKey	VirtualFree
GetCurrentProcess	ExitProcess	GetProcAddress	VirtualAlloc	CreateThread	VirtualAlloc
Sleep	VirtualFree	LocalAlloc	RegCLoseKey	CloseHandle	RegCloseKey
MultiByteToWideCha	RegCLoseKey	vbaFreeVar	GetLastError	GetModuleHandleA	IsEqualGUID
EnterCriticalSection	GetModuleFileNameA	GetCommandLineA	GetModuleFileNameA	GetLastError	SetTimer
LeaveCriticalSection	CloseHandle	adj fdiv m64	WriteFile	WriteFile	InternetCrackUrlA

static analysis data that are the Top-1000 imported functions. The dataset has been validated first to verify its quality and usage required for the project use case. The dataset consists of malware class represented as 0 for 'Goodware' and 1 for 'Malware'.

4.2 Data Pre-processing

The dataset used is the top 1000 features of PE imports as shared by [21]. In this research we have selected 29 features that define 6 categories comprising of benign programs and malware programs further categorised into 'Backdoor', 'Constructor', 'Email-worm', 'Hoax', 'Rootkit'. The five categories of malware have been defined based on the top 10 static API call information stated in the Table 3

After the pre-processing phase, the data has to be split into train and test set before fed into the model, in order to be trained. In the experimented model, we have split the data into 80% train and 20% test. The key principle behind splitting this data, is that the more samples in the train set, the lower the variance. The training set should be big enough to achieve low variance over the model parameters. Similarly, the test data should be small enough data to observe low variance among the performance results.The idea is to split the data to achieve low variance in both cases. If the dataset is big enough to achieve lower variance on the training parameters,the increase of the training set is required, which simultaneously, will increase the training time.

4.3 Data Exploration

Figure 3 is used for visualising the correlation between the selected 29 features with malware class. So, the every square in the heat map helps us to visualise the correlation between the variables (features) on each axis. The sidebar shows the correlation range from -1 to 1. There are shades of color from light to dark that defines the linear trend between two features or variables. The darkest shade of blue is +1 and the darkest shade of pink is -1. The correlation between two features is larger if the color is darker . The diagonals that are all dark blue have values 1 because they are correlating each feature to itself.

4.4 Malware Prediction using Deep learning

This paper proposes a method for classifying malware. The proposed model is a deep learning model where it predicts whether a file is malicious or benign (0 for benign and 1 for malware). The dataset originally having 1000 features belonging

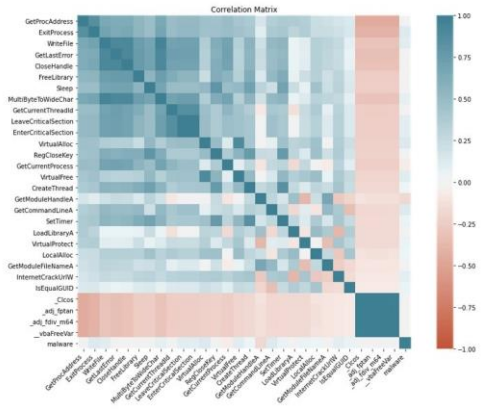

Figure 3 Heat Map of 29 features

to malicious and benign files were collected using static analysis . Out of which 29 features were selected depending on the priority level, as aforementioned. After preprocessing of the dataset, it is fed to the LSTM model, in order to be trained.The LSTM model comprising of one input layer and one hidden layer.

Hyper-parameters tuning are performed using activation functions such as Sigmoid, Relu along with different optimisers and epochs 10, 30, 50 & 100. In order to specify which works the best for the dataset. Thereafter, a comparison analysis of the experiments was conducted between 29 features and 1000 features. The output was a binary classification.

5 Results

This section presents the evaluation criteria used to assess the proposed model along with the results of the predictions.

Evaluation Criteria: The proposed Malware prediction model is evaluated using three criteria Root Mean Square Root, Mean Absolute Error and Loss Curve & Accuracy Curve.

1 RMSE- Root Mean Square Root

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(T_j - Y_j)^2} \tag{7}$$

2 MAE- Mean Absolute Error

$$MAE = MAD = \frac{1}{n}\sum_{j=1}^{n}|T_j - Y_j| \tag{8}$$

In 7 & 8 equation, T_j is the actual value while Y_j is the predicted value. '*n*' is the no of predicted observations. So, MAE can be described as the average of the absolute difference between the actual and the predicted observations. On the other hand, RMSE can be defined as the square root of MSE(Mean square error). It is the square root of the average of the squared difference between the actual and predicted observations. These metrics used for evaluation are in the range from 0-infinity and errors are within the +ve and -ve directions.

3 Loss Curve & Accuracy Curve

As explained by [22] the Loss curve is the most important to debug a model. It defines the learning rate of the neural network calculated by plotting training and validation loss. The model has a high learning rate if the loss decreases with increase in epochs and it has a low learning rate if the loss increases with epoch. On the other hand, accuracy curve generally defines the overfitting scenarios.

5.1 LSTM Model

Table 4 & Table 5 illustrates the RMSE & MAE values, that are explained for training and testing set for epoch 10, 30, 50 & 100. There is not much MAE and RMSE difference between the training and testing set for 29 features. Whereas there is a significant difference in RMSE between training and testing set for 1000 features.

Table 4 Evaluation of 29 features

	LSTM Architecture Parameters				
	Batch _size = 400 Activation function = sigmoid				
Epoch	Training MAE	Training RMSE	Testing MAE	Testing RMSE	Accuracy
10	0.03	0.17	0.03	0.17	98.97856
30	0.02	0.14	0.02	0.15	99.39680
50	0.02	0.12	0.01	0.14	99.51870
100	0.01	0.10	0.02	0.14	99.65111

Table 5 Evaluation of 1000 features

	LSTM Architecture Parameters				
	Batch size = 400 Activation function = sigmoid Features = 1000				
Epoch	Training MAE	Training RMSE	Testing MAE	Testing RMSE	Accuracy
10	0.00	0.06	0.02	0.13	99.82135
30	0.00	0.03	0.02	0.15	99.86548
50	0.00	0.03	0.03	0.17	99.89911
100	0.00	0.02	0.03	0.16	99.91172

5.2 Analysis of Accuracy & Loss Curve

Figure 4 demonstrates the accuracy curve and the loss curve for LSTM that describes the learning rate and types of overfitting cases. The graphs below are the result of training and testing of LSTM model with epochs of 10, 30, 50 and 100. The epochs were used with 29 selected features as well as the complete 1000 features to compare the performance of LSTM as neural network model works better with a larger dataset. The accuracy curve is plotted between accuracy and epoch. Now, the 'accuracy' is

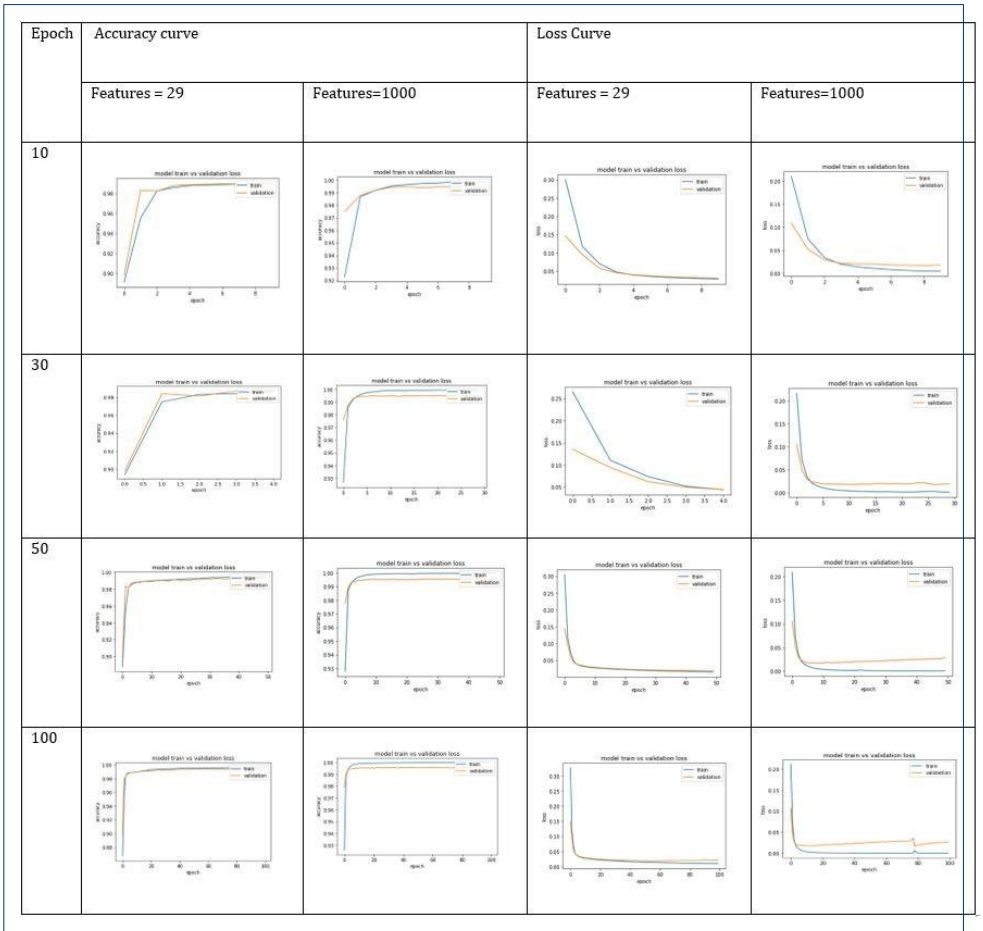

Figure 4 Analysis of LSTM Model

calculated by training the model and 'validation accuracy' is calculated by testing the model. The greater the difference between the training accuracy and the validation

accuracy, there is a strong overfitting case whereas if the difference is small little overfitting case occurs.

The loss curve is plotted against the loss(error) and epochs. Again, the 'loss' is calculated while training LSTM and also 'validation loss' is calculated while testing the model. The 'lower the loss the better is the learning rate of the model and vice versa.
The Accuracy graph curve & loss graph curve in the Figure 4 illustrates the comparison between 29 features and 1000 features and it can be concluded after looking at the graphs that our LSTM model worked best for epoch=50 & 100 with 29 features. There is no overfitting case in the accuracy curve as well as the difference in the loss is low meaning it higher learning rate.

The most important task after pre-processing is the input/output shape fed into the neural network (LSTM) model. This LSTM model works best when worked with a fewer features that consists of 29 features selected using weka but also gave better results when worked with the original dataset consisting of 1000 features. The correlation matrix graph show us the correlation between each features and how much they are dependent on each other.

After carrying out the implementation and hyper-tuning of different parameters, we found that this model worked best with Sigmoid Activation function and the best predicted model came out to be at epoch 100 where accuracy is 99.65% . In other words that the model can predict the malicious file accurately with 99.65%.

6 Conclusion

This research is about classification of a given file into malicious or benign using Deep learning technique by selecting static features for malware analysis. The model used for classification is LSTM model which is evaluated based on RMSE and MAE values.Various hyper-parameters were tuned during the experiments. Moreover, three different activation functions namely relu ,softmax and Sigmoid with different epochs were experimented. The LSTM model worked best with Sigmoid function resulting in high accuracy of 99% approximately. While, in relu activation Function the accuracy came down to 36%-49% approximately when using Optimizer as 'binary crossentropy' whereas the accuracy increased to 98% when using 'mse' optimizer. In softmax activation function the loss come out to be constant which is a hypothetical case and does not work well with the dataset.
Hence, Relu activation function worked best with 'mse' optimizer, softmax activation function is not used because it's accuracy came out to be constant with the different hyper-parameters. In Conclusion, the best activation function that suits this research is 'sigmoid function with high accuracy between 98%-99% using different parameters. The second most important factor that affected the accuracy was epoch. As the epoch increased, the accuracy slightly increased with increase in computational time. The epochs were chosen wisely for better performance of the model as computational time remains an important factor along-with the accuracy for malware detection.

This Research can be further optimised in classifying the malware and their respective families. In Addition, reducing computational time and effective analysing of these malwares can be blocked efficiently before the attacker cause a major damage to the system.

References

1. Ellen, Z.: What Is Malware Analysis? Defining and Outlining the Process of Malware Analysis. https://digitalguardian.com/blog/what-malware-analysis-defining-and-outlining-process-malware-analysis
2. in, B.: How Does Artificial Intelligence Work? https://builtin.com/artificial-intelligence
3. Sharp, R.: An Introduction to Malware. Spring (2009)
4. Rad, B.B., Masrom, M., Ibrahim, S.: Camouflage in malware: from encryption to metamorphism. International Journal of Computer Science and Network Security 12(8), 74–83 (2012)
5. Brownlee, J.: What is deep learning. Machine learning mastery 16 (2016)
6. Nielsen, M.A.: Neural Networks and Deep Learning vol. 25. Determination press San Francisco, CA, (2015)
7. Olah, C.: Understanding lstm networks (2015)
8. Nait Aicha, A., Englebienne, G., Van Schooten, K.S., Pijnappels, M., Kr¨ose, B.: Deep learning to predict falls in older adults based on daily-life trunk accelerometry. Sensors 18(5), 1654 (2018)
9. D'Angelo, G., Ficco, M., Palmieri, F.: Malware detection in mobile environments based on autoencoders and api-images. Journal of Parallel and Distributed Computing 137, 26–33 (2020)
10. Vasan, D., Alazab, M., Wassan, S., Safaei, B., Zheng, Q.: Image-based malware classification using ensemble of cnn architectures (imcec). Computers & Security 92, 101748 (2020)
11. Ren, Z., Wu, H., Ning, Q., Hussain, I., Chen, B.: End-to-end malware detection for android iot devices using deep learning. Ad Hoc Networks 101, 102098 (2020)
12. Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y., Sakuma, J.: Neural malware analysis with attention mechanism. Computers & Security 87, 101592 (2019)
13. Andrade, E.d.O., Viterbo, J., Vasconcelos, C.N., Gu´erin, J., Bernardini, F.C.: A model based on lstm neural networks to identify five different types of malware. Procedia Computer Science 159, 182–191 (2019)
14. Kang, J., Jang, S., Li, S., Jeong, Y.-S., Sung, Y.: Long short-term memory-based malware classification method for information security. Computers & Electrical Engineering 77, 366–375 (2019)
15. Sung, Y., Jang, S., Jeong, Y.-S., Hyuk, J., *et al.*: Malware classification algorithm using advanced word2vec-based bi-lstm for ground control stations. Computer Communications 153, 342–348 (2020)
16. Jahromi, A.N., Hashemi, S., Dehghantanha, A., Choo, K.-K.R., Karimipour, H., Newton, D.E., Parizi, R.M.: An improved two-hidden-layer extreme learning machine for malware hunting. Computers & Security 89, 101655 (2020)
17. Zhong, W., Gu, F.: A multi-level deep learning system for malware detection. Expert Systems with Applications 133, 151–162 (2019)
18. Karanja, E.M., Masupe, S., Jeffrey, M.G.: Analysis of internet of things malware using image texture features and machine learning techniques. Internet of Things 9, 100153 (2020)
19. Sartea, R., Farinelli, A., Murari, M.: Secur-ama: Active malware analysis based on monte carlo tree search for android systems. Engineering Applications of Artificial Intelligence 87, 103303 (2020)
20. Aldweesh, A., Derhab, A., Emam, A.Z.: Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. Knowledge-Based Systems 189, 105124 (2020)
21. Oliveira, A.: Malware Analysis Datasets: Top-1000 PE Imports. IEEE Dataport (2019)
22. Karpathy, A., et al.: Cs231n convolutional neural networks for visual recognition. Neural networks 1(1) (2016)