JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 08, ISSUE 05, 2021

Deep Learning CNN based an Artificial Intelligence Approach for Malware Detection

Umarani Koppula, Nagaraju Medasani

Department of Computer Science and Engineering

Sree Dattha Group of Institutions, Hyderabad, Telangana, India.

Abstract

Malicious software or malware continues to pose a major security concern in this digital age as computer users, corporations, and governments witness an exponential growth in malware attacks. Current malware detection solutions adopt Static and Dynamic analysis of malware signatures and behaviour patterns that are time consuming and ineffective in identifying unknown malwares. Recent malwares use polymorphic, metamorphic, and other evasive techniques to change the malware behaviours quickly and to generate large number of malwares. Since new malwares are predominantly variants of existing malwares, machine learning algorithms (MLAs) are being employed recently to conduct an effective malware analysis. This requires extensive feature engineering, feature learning and feature representation. By using the advanced MLAs such as deep learning, the feature engineering phase can be completely avoided. Though some recent research studies exist in this direction, the performance of the algorithms is biased with the training data. There is a need to mitigate bias and evaluate these methods independently in order to arrive at new enhanced methods for effective zero-day malware detection. To fill the gap in literature, this work evaluates classical MLAs and deep learning architectures for malware detection, classification, and categorization with both public and private datasets. The train and test splits of public and private datasets used in the experimental analysis are disjoint to each other's and collected in different timescales. In addition, we propose a novel image processing technique with optimal parameters for deep learning convolutional neural networks (DLCNN) architectures. Overall, this work proposes an effective visual detection of malware using a scalable and hybrid deep learning framework for real-time deployments. The visualization and deep learning architectures for static, dynamic and image processing-based hybrid approach in a big data environment is a new enhanced method for effective zero-day malware detection. Finally, the simulations revealed that the proposed DLCNN resulted in superior performance as compared to existing models.

Keywords: Deep learning, robust intelligent malware detection, Machine learning algorithms (MLAs).

1. INTRODUCTION

In this digital world of Industry 4.0, the rapid advancement of technologies has affected the daily activities in businesses as well as in personal lives. Internet of Things (IoT) and applications have led to the development of the modern concept of the information society. However, security concerns pose a major challenge in realising the benefits of this industrial revolution as cyber criminal's attack individual PC's and networks for stealing confidential data for financial gains and causing denial of service to systems. Such attackers make use of malicious software or malware to cause serious threats and vulnerability of systems [1]. A malware is a computer program with the purpose of causing harm to the operating system (OS). A malware gets different names such as adware, spyware, virus, worm, trojan, rootkit, backdoor, ransomware and command and control (C&C) bot, based on its purpose and behaviour. Detection and mitigation of malware is an evolving problem in the cyber security field. As researchers develop new techniques, malware authors improve their ability to evade detection.

When Morris worm made its appearance as the first ever computer virus in 1988-89, antivirus software programs were designed to detect the existence of such a malware by finding a match with the virus definition database updated from time to time. This is called signature-based malware detection, which can also perform a heuristic search to identify the behavior of malware. However, the major challenge in such classical approaches is that new variants of malware use antivirus evasion techniques such as code obfuscation and hence such signature-based approaches are unable to detect zero-day malwares [2]. Signature-based malware detection system requires extensive domain level knowledge to reverse engineer the malware using Static and dynamic analysis and to assign a signature for that. Moreover, signature-based system requires larger time to reverse engineer the malware and during that time an attacker would encroach into the system. In addition, signature-based system fails to detect new types of malwares. Security researchers have identified that hackers predominantly use polymorphism and metamorphism as obfuscation techniques against signature-based detection. In order to address this problem, software tools are used to manually unpack the codes and analyse the application programming interface (API) calls.

Since this process is a resource intensive task, [3] presented an automated system to extract API calls and analyse the malicious characteristics using a four-step methodology. In step 1, the malware is unpacked. In step 2, the binary executable is disassembled. Step 3 involves API call extraction. Step 4 involves API call mapping and statistical feature analysis. This was enhanced in [4] using a 5- step methodology incorporating machine learning algorithm (MLA) such as SVM with n-gram features extracted from large samples of both the benign and malicious executables with 10-fold cross validations. Later, in [5] a comparative study of various classical machine learning classifiers for malware detection was performed, and a framework for zero-day malware detection was proposed. To handle malicious code variants, the sequence of API calls and their frequency of appearance of API calls passed into similarity-based mining and machine learning methods [7]. The detailed experimental analysis was done on very large data set and to extract the features from malware binaries a unified framework proposed. In [8], API calls features and a hybrid of support vector machine (SVM) and Maximum-Relevance Minimum Redundancy Filter (MRMRF) heuristics were employed to present novel feature selection approaches for enhanced malware detection. Recently, with the increase in unknown malware attacks, the detailed information on obfuscated malware is discussed by [6] and many researchers are improving the MLAs for malware detection [9]. This forms the motivation of this research work.

2. RELATED WORK

Machine learning algorithms (MLAs) rely on the feature engineering, feature selection and feature representation methods. The set of features with a corresponding class is used to train a model in order to create a separating plane between the benign and malwares. This separating plane helps to detect a malware and categorize it into its corresponding malware family. Both feature engineering and feature selection methods require domain level knowledge. The various features can be obtained through Static and Dynamic analysis. Static analysis is a method that captures the information from the binary program without executing. Dynamic analysis is the process of monitoring malware behavior at run time in an isolated environment. The complexities and various issues of Dynamic analysis are discussed in detail by [10]. Dynamic analysis can be an efficient long-term solution for malware detection system. The Dynamic analysis cannot be deployed in end-point real time malware detection due to the reason that it takes much time to analyze its behaviour, during which malicious payload can get delivered. Malware detection methods based on Dynamic analysis are more robust to obfuscation methods when compared to statically collected data. Most commonly, the commercial anti-malware solutions use a hybrid of Static and Dynamic analysis approaches. The major issue with the classical machine learning based malware detection system is that they rely on the feature engineering, feature learning and feature representation techniques that require an extensive domain level knowledge [11], [12], [13]. Moreover, once an attacker comes to know the features, the malware detector can be evaded easily [14]. To be successful, MLAs require data with a variety of patterns of malware. The publicly available benchmark data for malware analysis research is very less due to the security and privacy concerns. Though few datasets exist, each of them has their own harsh criticisms as most of them are outdated. Many of the published results of machine learning based malware analysis have used their own datasets. Even though publicly available sources exist to crawl the malware datasets, preparing a proper dataset for research is a daunting task. These issues are the main drawbacks behind developing generic machine learning based malware analysis system that can be deployed in real time. More importantly, the compelling issues in applying data science techniques were discussed in detail by [15].

In recent days, deep learning, which is an improved model of neural networks has outperformed the classical MLAs in many of the tasks which exist in the field of natural language processing (NLP), computer vision, speech processing and many others [16]. During the training process, it tries to capture higher level representation of features in deep hidden layers with the ability to learn from mistakes. MLAs experience diminishing outputs as they see more and more data whereas deep learning captures new patterns and establishes associations with the already captured pattern to enhance the performance of tasks. There exist few research studies towards the application of deep learning architectures for malware analysis to improve cyber security [13], [11], [12], [17], [18], [18]–[24]. However, with Industry 4.0, the number of malwares is rapidly increasing in recent times. Since the continuous collection of malwares in real time results in Big Data, the existing approaches are not scalable with very high requirements for storage and time in making efficient decisions.

3. PROPOSED METHODOLOGY

Deep learning or deep neural networks (DNNs) takes inspiration from how the brain works and forms a sub module of artificial intelligence. The main strength of deep learning architectures is the capability to understand the meaning of data when it is in large amounts and to automatically tune the derived meaning with new data without the need for a domain expert knowledge. Convolutional neural networks (CNNs) and Recurrent neural networks (RNNs) are two types of deep learning architectures predominantly applied in real-life scenarios. Generally, CNN architectures are used for spatial data and RNN architectures are used for temporal data. The combination of CNN and LSTM is used for spatial and temporal data analysis.

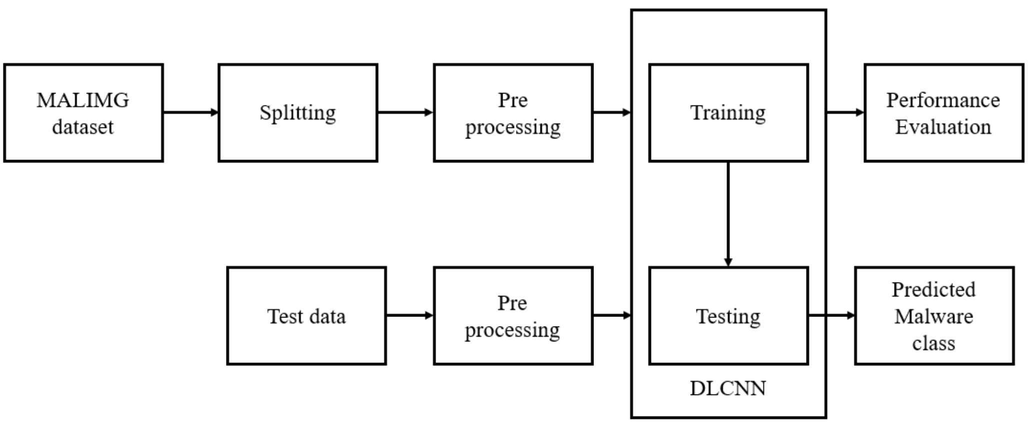

Fig. 1. Proposed block diagram.

Fig. 1 shows the block diagram of proposed method. Initially, MALIMG dataset is spitted into 80% for training and 20% for testing. Then, dataset pre-processing operation is performed to normalize the entire dataset. Further, DLCNN classifier is used for prediction of malware attack from test sample. The performance evaluation is carried out to show supremacy of proposed method.

3.1 MALIMG dataset

CICDDoS2019 contains benign and the most up-to-date common DDoS attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter-V3 with labelled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols, and attack (CSV files). Generating realistic background traffic was our top priority in building this dataset. We have used our proposed B-Profile system to profile the abstract behaviour of human interactions and generates naturalistic benign background traffic in the proposed testbed. For this dataset, we built the abstract behaviour of 25 users based on the HTTP, HTTPS, FTP, SSH and email protocols.

3.2 Pre-processing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

3.3 Splitting the Dataset

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance

3.4 DLCNN

A feed forward neural network (FFN) creates a directed graph in which a graph is composed of nodes and edges. FFN passes information along edges from one node to another without formation of a cycle. Multi-layer perceptron (MLP) is a type of FFN that contains 3 or more layers, specifically one input layer, one or more hidden layer and an output layer in which each layer has many neurons, called as units in mathematical notation. The number of hidden layers is selected by following a hyper parameter tuning approach. The information is transformed from one layer to another layer in forward direction without considering the past values. Moreover, neurons in each layer are fully connected.

Convolutional network or convolutional neural network or CNN is supplement to the classical feed forward network (FFN), primarily used in the field of data processing. It is shown in Fig. 2, where all connections and hidden layers and its units are not shown. Here, m denotes number of filters, ln denotes number of input features and p denotes reduced feature dimension, it depends on pooling length. In this work, CNN network composed of convolution 1D layer, pooling 1D layer and fully connected layer. A CNN network can have more than one convolution 1D layer, pooling 1D layer and fully connected layer. In convolutional 1D layer, the filters slide over the 1D sequence data and extracts optimal features. The features that are extracted from each filter are grouped into a new feature set called as feature map. The number of filters and the length are chosen by following a hyper parameter tuning method. This in turn uses non-linear activation function, ReLU on each element.

JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 08, ISSUE 05, 2021

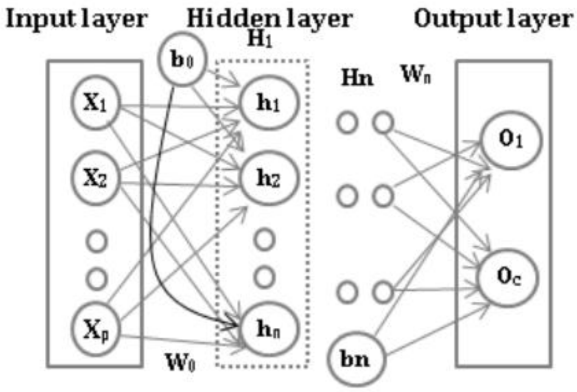

Fig. 2. DNN architecture.

The dimensions of the optimal features are reduced using pooling 1D layer using either max pooling, min pooling or average pooling. Since the maximum output within a selected region is selected in max pooling, we adopt max pooling in this work. Finally, the DLCNN network contains fully connected layer for classification. In fully connected layer, each neuron contains a connection to every other neuron. Instead of passing the pooling 1D layer features into fully connected layer, it can also be given to recurrent layer, LSTM to capture the sequence related information. Finally, the LSTM features are passed into fully connected layer for classification.

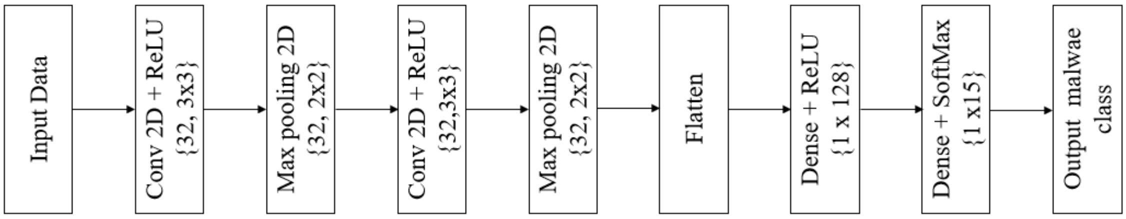

Fig. 3. Architecture of DLCNN for malware detection.

Table 1. Layers description.

Layer Names	No. of filters	Kernel size	Feature size
Conv 2D +ReLU	32	3 x 3	62x62x32
Max pooling 2D	-	3 x 3	31x31x32
Conv 2D+ReLU	32	3 x 3	29x29x32
Max pooling 2D	-	3 x 3	14x14x32
Flatten	-	1x6272	1x6272
Dense +ReLU		1 x 128	1 x 256

Dense + SoftMax		1 x 15	1 x 15

Convolutional neural networks are generally composed of three parts. Convolution layer for feature extraction. The convergence layer, also known as the pooling layer, is mainly used for feature selection. The number of parameters is reduced by reducing the number of features. The full connection layer carries out the summary and output of the characteristics. A convolution layer is consisting of a convolution process and a nonlinear activation function ReLU. A typical architecture of CNN model for malware class recognition is shown in Fig. 3.

The leftmost data is the input layer, which the computer understands as the input of several matrices. Next is the convolution layer, the activation function of which uses ReLU. The pooling layer has no activation function. The combination of convolution and pooling layers can be constructed many times. The combination of convolution layer and convolution layer or convolution layer and pool layer can be very flexibly, which is not limited when constructing the model. But the most common CNN is a combination of several convolution layers and pooling layers. Finally, there is a full connection layer, which acts as a classifier and maps the learned feature representation to the sample label space.

It is assumed that the size of the input picture is $50 * 50 * 3$. If placed in a fully connected feedforward network, there are 7500 mutually independent links to the hidden layer. And each link also corresponds to its unique weight parameter. With the increase of the number of layers, the size of the parameters also increases significantly. On the one hand, it will easily lead to the occurrence of over-fitting phenomenon. On the other hand, the neural network is too complex, which will seriously affect the training efficiency. In convolutional neural networks, the parameter sharing mechanism makes the same parameters used in multiple functions of a model, and each element of the convolutional kernel will act on a specific position of each local input. The neural network only needs to learn a set of parameters and does not need to optimize learning for each parameter of each position.

Data stability is the local invariant feature, which means that the natural data will not be affected by the scaling, translation, and rotation of the data size. Because in deep learning, data enhancement is generally needed to improve performance, and fully connected feedforward neural is difficult to ensure the local invariance of the data. This problem can be solved by convolution operation in convolutional neural network.

DLCNN Layers: According to the facts, training, and testing of DLCNN involves in allowing every source data via a succession of convolution layers by a kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer and utilize SoftMax layer with classification layer to categorize the objects with probabilistic values ranging from $[0,1]$.

Convolution layer as depicted in Fig. 4 is the primary layer to extract the features from a source data and maintains the relationship between pixels by learning the features of data by employing tiny blocks of source data. It's a mathematical function which considers two inputs like source data $I(x, y, d)$ where x and y denotes the spatial coordinates i.e., number of rows and columns. d is denoted as dimension of a data (here $d = 3$, since the source data is RGB) and a filter or kernel with similar size of input data and can be denoted as $F(k_x, k_y, d)$.

ISSN- 2394-5125 VOL 08, ISSUE 05, 2021

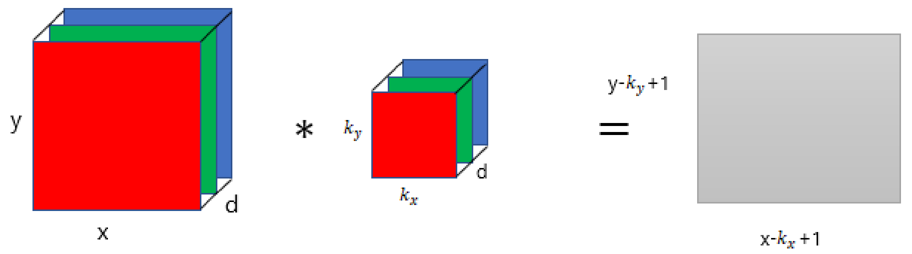

Fig . 4. Representation of convolution layer process.

The output obtained from convolution process of input data and filter has a size of $C\left((x-k_x+1),(y-k_y+1),1\right)$, which is referred as feature map. An example of convolution procedure is demonstrated in Fig. 5 (a). Let us assume an input data with a size of 5×5 and the filter having the size of 3×3. The feature map of input data is obtained by multiplying the input data values with the filter values as given in Fig. 5(b).

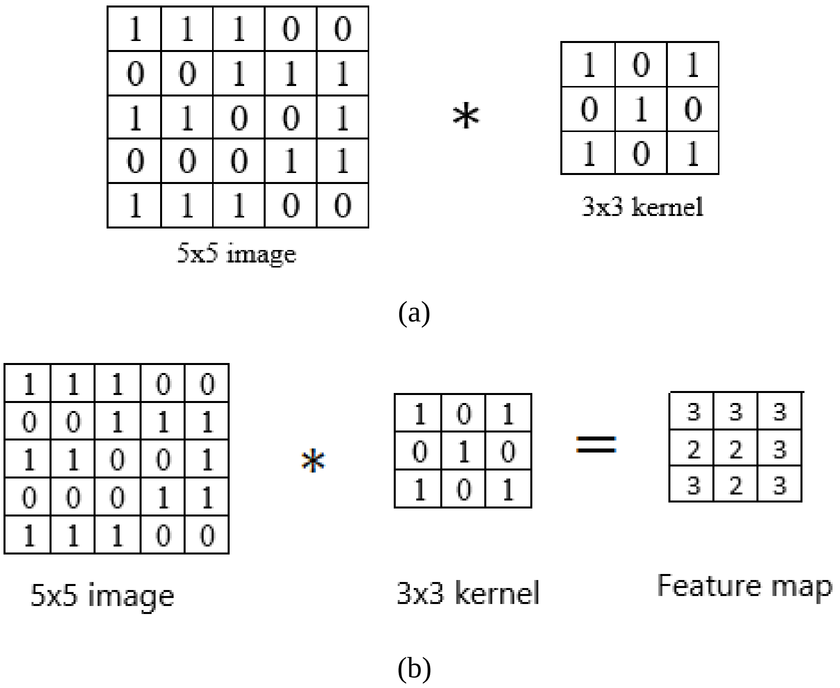

Fig. 5. Example of convolution layer process (a) a data with size $\mathbf{5 \times 5}$ is convolving with $\mathbf{3 \times 3}$ kernel (b) Convolved feature map.

Generally, as seen in the above picture SoftMax function is added at the end of the output since it is the place where the nodes are meet finally and thus, they can be classified. Here, X is the input of all the models and the layers between X and Y are the hidden layers and the data is passed from X to all the layers and Received by Y. Suppose, we have 10 classes, and we predict for which class the given input belongs to. So, for this what we do is allot each class with a particular predicted output. Which means that we have 10 outputs corresponding to 10 different class and predict the class by the highest probability it has. In Fig. 6, and we must predict what is the object that is present in the picture. In the normal case, we predict whether the malware is A. But in this case, we must predict what is the object that is present in the picture. This is the place where softmax comes in handy. As the model is already trained on some data. So, as soon as the picture is given, the model processes the pictures, send it to the hidden layers and then finally send to softmax for classifying the picture. The softmax uses a One-Hot encoding Technique to calculate the cross-entropy loss and get the max. One-Hot Encoding is the technique that is used to categorize the data. In the previous example, if softmax predicts that the object is class A then the One-Hot Encoding for:

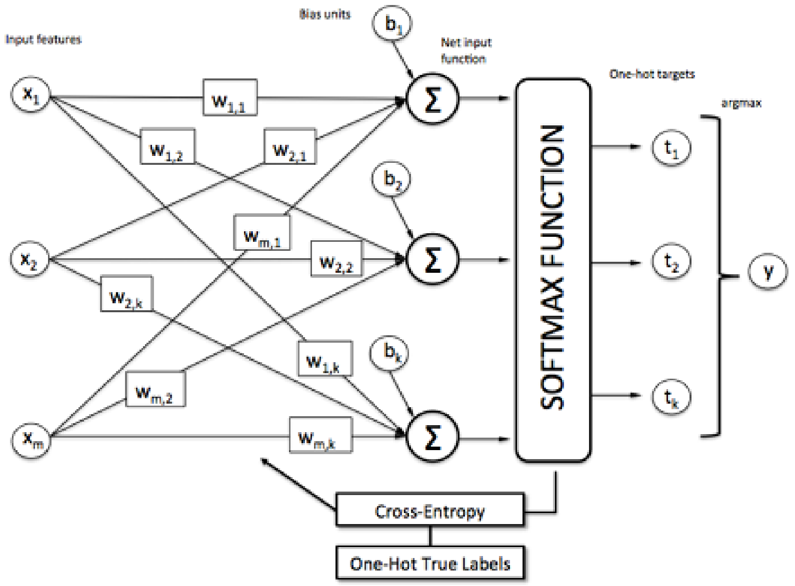

Fig.6. Malware class prediction using SoftMax classifier.

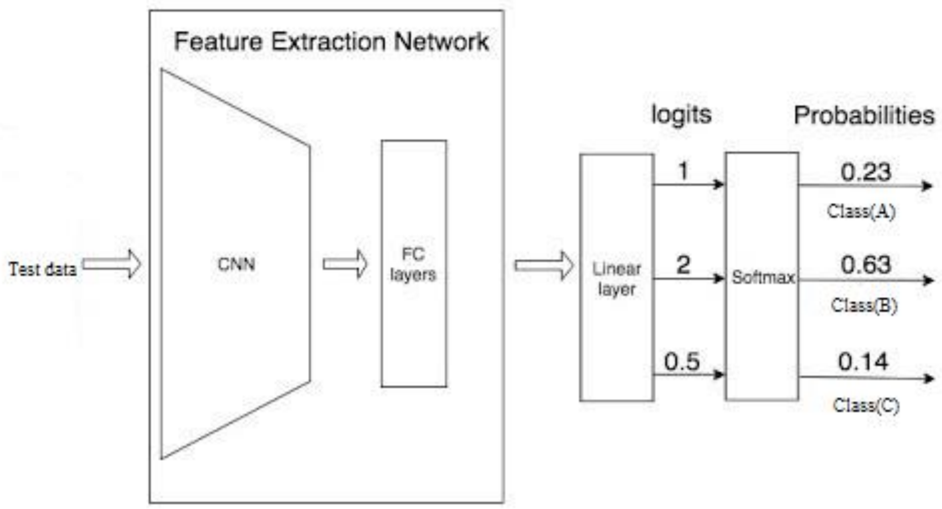

Fig.7. Example of SoftMax classifier.

Class A will be [1 0 0]

Class B will be [0 1 0]

Class C will be [0 0 1]

From the Fig. 7, we see that the predictions are occurred. But generally, we don't know the predictions. But the machine must choose the correct predicted object. So, for machine to identify an object correctly, it uses a function called cross-entropy function. So, we choose more similar value by using the below cross-entropy formula.

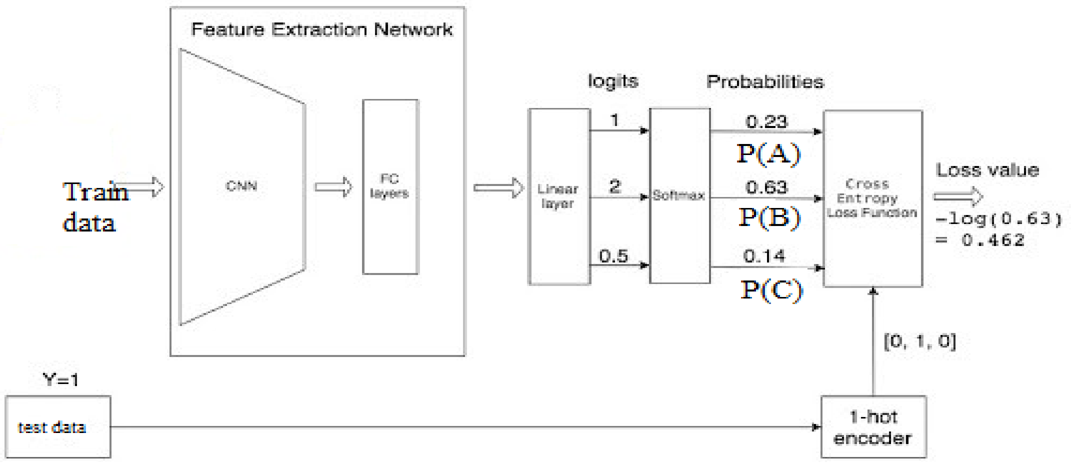

Fig.8. Example of SoftMax classifier with test data.

In the above Fig. 8, we see that 0.462 is the loss of the function for class specific classifier. In the same way, we find loss for remaining classifiers. The lowest the loss function, the better the prediction is. The mathematical representation for loss function can be represented as: -

$$LOSS = np.sum(-Y \ * \ np.log(Y\_pred))$$

4. RESULTS

The dataset was formed by transforming malware binaries into a matrix. This matrix has 8-bit unsigned integer. This matrix can be visualized as a grayscale image which contains values in the range of [0, 255], 0 represents black and 255 represents white. We converted the 2D matrix into 1D vector form, resulting in a 1x1024 size array. L_2 normalization is employed for newly formed data. Next, the dataset was randomly divided into 70% training and 30% testing dataset with both these datasets containing samples for each malware family.

We have prepared the datasets for conducting the experimental analysis using the following pre-processing stages:

1) Ember: Using domain level knowledge, various features from parsed PE file as well as format-agnostic features such as raw byte histogram, byte entropy histogram are taken from [27], and strings are extracted and passed into the LightGBM model. Since the performance of LightGBM model is good as compared to MalConv model, they use gradient boosted decision tree (GBDT) in LightGBM with default parameters consisting of 100 trees and 31 leaves per tree. Following, in this work we evaluate the performance of classical MLAs and DNNs for malware classification using the Ember dataset.

2) MalConv: MalConv is an architecture proposed in [11] for malware detection which composed of 3 different sections are undergone, namely (1) pre-processing (2) convolution and (3) fully connected. In the pre-processing section, the raw byte sequences from the binary files are passed into embedding layer. The embedding layer contains 257 as the size of the dictionary of embeddings and 8 as the embedding dimension. Embedding layer maps bytes into fixed length feature vector representation. In convolution section, MalConv contains two convolution 1D layers. Each convolution 1D layer contains 512 (kernel size 4, 128 filters) units and 500 strides. These convolution layers follow the gated convolution approach. Convolution layer follows a temporal maxpooling which uses 4000 as pooling length to reduce the dimension and to handle the information sparsity issue. Fully connected section is composed of 2 fully connected layers: the first fully connected layer contains 128 units, and

the second fully connected layer contains 1 unit with sigmoid non-linear activation function. SVM is used at the last layer for classification with LSTM.

3) Variants of MalConv: The slight variation to the strides, SELU nonlinear activation function of the MalConv model and removed the DeConv regularization by [12]. The convolution section contains two convolution layers, a maxpooling followed by another two convolution layers. The first two convolution layers contain 32 units with strides 4 and the next two convolution layers contain 16 units with strides of 8. The last two layers follow the global average pooling with 4 fully connected layers.

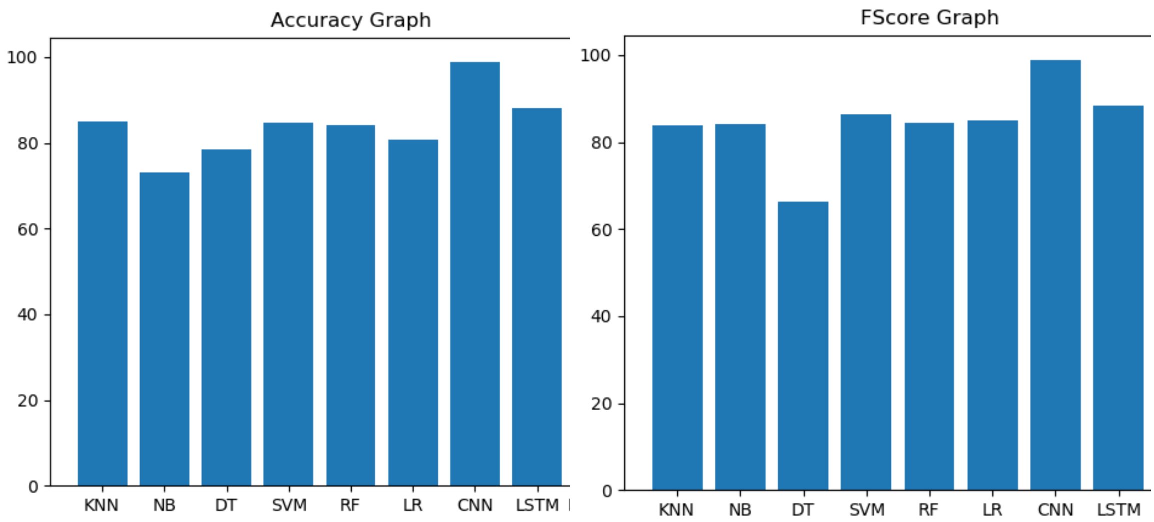

Fig. 9. Performance comparison of accuracy and F-score obtained using existing and proposed models.

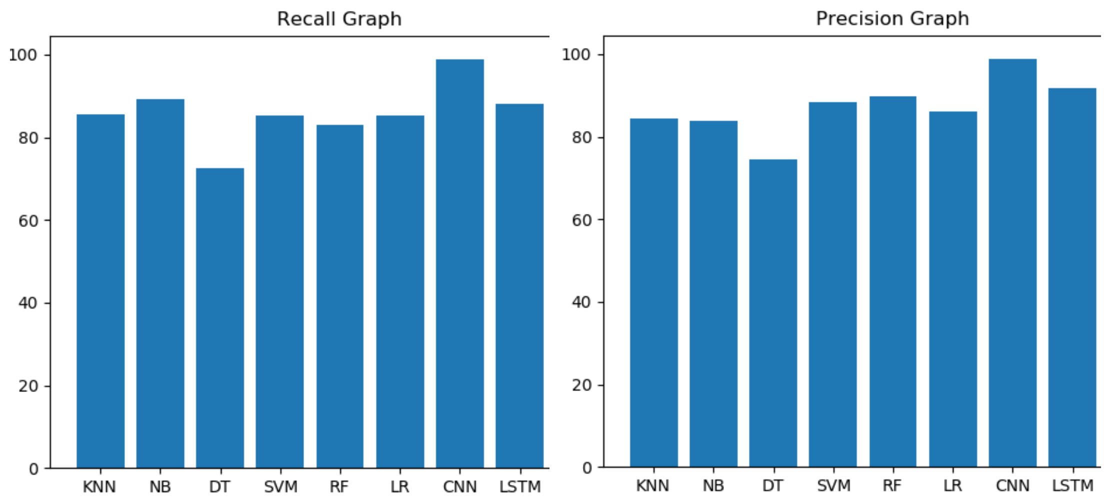

Fig. 10. Performance comparison of recall and precision obtained using existing and proposed models.

5. CONCLUSION

This paper proposed an efficient malware detection and designed a highly scalable framework to detect, classify and categorize zero-day malwares. This framework applies DLCNN on the collected malwares from end user hosts and follows a two-stage process for malware analysis. In the first stage, a hybrid of static and dynamic analysis was applied for malware classification. In the second stage, malwares were grouped into corresponding malware categories using image processing approaches. Various experimental analysis conducted by applying variations in the models on publicly available benchmark dataset and indicated the proposed model outperformed classical MLAs. The developed framework is capable of analyzing large number of malwares in real-time and scaled out to analyse even larger number of malwares by stacking a few more layers to the existing architectures. Future research entails exploration of these variations with new features that could be added to the existing data.

REFERENCES

[1] R. Anderson et al., ‘‘Measuring the cost of cybercrime,’’ in The Economics of Information Security and Privacy. Berlin, Germany: Springer, 2013, pp. 265–300.

[2] B. Li, K. Roundy, C. Gates, and Y. Vorobeychik, ‘‘Large-scale identification of malicious singleton files,’’ in Proc. 7th ACM Conf. Data Appl. Secur. Privacy. New York, NY, USA: ACM, Mar. 2017, pp. 227–238.

[3] M. Alazab, S. Venkataraman, and P. Watters, ‘‘Towards understanding malware behaviour by the extraction of API calls,’’ in Proc. 2nd Cybercrime Trustworthy Comput. Workshop, Jul. 2010, pp. 52–59.

[4] M. Tang, M. Alazab, and Y. Luo, ‘‘Big data for cybersecurity: Vulnerability disclosure trends and dependencies,’’ IEEE Trans. Big Data, to be published.

[5] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, ‘‘Zero-day malware detection based on supervised learning algorithms of API call signatures,’’ in Proc. 9th Australas. Data Mining Conf., vol. 121. Ballarat, Australia: Australian Computer Society, Dec. 2011, pp. 171–182.

[6] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, and A. Alazab, ‘‘Cybercrime: The case of obfuscated malware,’’ in Global Security, Safety and Sustainability & e-Democracy (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 99, C. K. Georgiadis, H. Jahankhani, E. Pimenidis, R. Bashroush, and A. Al-Nemrat, Eds. Berlin, Germany: Springer, 2012.

[7] M. Alazab, ‘‘Profiling and classifying the behavior of malicious codes,’’ J. Syst. Softw., vol. 100, pp. 91–102, Feb. 2015.

[8] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, ‘‘Hybrids of support vector machine wrapper and filter-based framework for malware detection,’’ Future Gener. Comput. Syst., vol. 55, pp. 376–390, Feb. 2016.

[9] E. Raff, J. Sylvester, and C. Nicholas, ‘‘Learning the PE header, malware detection with minimal domain knowledge,’’ in Proc. 10th ACM Workshop Artif. Intell. Secur. New York, NY, USA: ACM, Nov. 2017, pp. 121–132.

[10] C. Rossow, et al., ‘‘Prudent practices for designing malware experiments: Status quo and outlook,’’ in Proc. IEEE Symp. Secur. Privacy (SP), Mar. 2012, pp. 65–79.

[11] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas. (2017). ‘‘Malware detection by eating a whole exe.’’ [Online]. Available: https://arxiv.org/abs/1710.09435

[12] M. Krcál, O. Švec, M. Bálek, and O. Jašek. (2018). Deep Convolutional Malware Classifiers Can Learn from Raw Executables and Labels Only. [Online]. Available: https://openreview.net/forum?id=HkHrmM1PM

[13] M. Rhode, P. Burnap, and K. Jones, ‘‘Early-stage malware prediction using recurrent neural networks,’’ Comput. Secur., vol. 77, pp. 578–594, Aug. 2018.

[14] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, Evading Machine Learning Malware Detection. New York, NY, USA: Black Hat, 2017.

[15] R. Verma, ‘‘Security analytics: Adapting data science for security challenges,’’ in Proc. 4th ACM Int. Workshop Secur. Privacy Anal. New York, NY, USA: ACM, Mar. 2018, pp. 40–41.

[16] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[17] A. F. Agarap and F. J. H. Pepito. (2017). ‘‘Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification.’’ [Online]. Available: https://arxiv.org/abs/1801.00318

[18] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, and P. de Geus, ‘‘Malicious software classification using VGG16 deep neural network’s bottleneck features,’’ in Information Technology-New Generations. Cham, Switzerland: Springer, 2018, pp. 51–59.

[19] J. Saxe and K. Berlin, ‘‘Deep neural network-based malware detection using two dimensional binary program features,’’ in Proc. 10th Int. Conf. Malicious Unwanted Softw. (Malware), Oct. 2015, pp. 11–20.

[20] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi, ‘‘Malware detection with deep neural network using process behavior,’’ in Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC), vol. 2, Jun. 2016, pp. 577–582.

[21] W. Huang, J. W. Stokes, ‘‘Mtnet: A multi-task neural network for dynamic malware classification,’’ in Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment, Cham, Switzerland: Springer, Jul. 2016, pp. 399–418.

[22] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, ‘‘Malware classification with recurrent networks,’’ in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Apr. 2015, pp. 1916–1920.

[23] T. Shibahara, T. Yagi, M. Akiyama, D. Chiba, and T. Yada, ‘‘Efficient dynamic malware analysis based on network behavior using deep learning,’’ in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2016, pp. 1–7.

[24] S. H. Ebenuwa, M. S. Sharif, M. Alazab, and A. Al-Nemrat, ‘‘Variance ranking attributes selection techniques for binary classification problem in imbalance data,’’ IEEE Access, vol. 7, pp. 24649–24666, 2019.

[25] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, ‘‘Malware images: Visualization and automatic classification,’’ in Proc. 8th Int. Symp. Vis. Cyber Secur. New York, NY, USA: ACM, Jul. 2011, p. 4.

[26] F. C. C. Garcia, and F. P. Muga, II, (2016). ‘‘Random forest for malware classification.’’ [Online]. Available: https://arxiv.org/abs/arXiv: 1609.07770

[27] H. S. Anderson and P. Roth. (2018). ‘‘EMBER: An open dataset for training static PE malware machine learning models.’’ https://arxiv.org/ abs/1804.04637