INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 Issue: IV Month of publication: April 2024

DOI: https://doi.org/10.22214/ijraset.2024.59911

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue IV Apr 2024- Available at www.ijraset.com

Robust Intelligent Malware Detection Using Deep Learning

Dr. P.Sruthi [1], Dr.Y.Ambica[2], Thumula Pranay Krishna Kumar[3], Thota Ajitha[4], Nilagiri Venkat Prasad[5]
[1]*Associate Professor, Department of CSE (AI&ML), CMR College of Engineering & Technology, Hyderabad,Telangana State, India*
[2]*Assistant Professor, Department of CSE (AI&ML), CMR College of Engineering & Technology, Hyderabad, Telangana State, India*

Abstract: In the modern era of technology, malicious software, or malware, holds a serious security hazard as computer users, businesses, and governments see an uptick in malware attacks. In attempts to identify unknown malware, current malware detection solutions use dynamic as well as static examination of malware signatures and behavior patterns, which takes time and is unsuccessful. Modern malware employs evasive strategies such as metamorphosis and polymorphism to rapidly alter its actions and produce a multitude of variants. Machine learning algorithms (MLAs) are being used more and more to do an efficient malware analysis because new malware is primarily versions of current malware. Extensive feature engineering, feature learning, and feature representation are needed for this. It is likely to fully eliminate the feature engineering stage by utilizing sophisticated MLAs like deep learning. Even though there have been a few fresh investigations in the field, the algorithms' performance is skewed by the training set. It is a prerequisite to reduce bias and figure out these techniques holistically in order to develop new, improved techniques for successful zero-day malware detection. This paper fills a vacuum in the literature by comparing and contrasting deep learning architectures with standard MLAs for malware detection, classification, and categorization using public and private datasets. The public and private dataset's train and test splits, which were gathered during distinctly different periods, are not connected to one another in the experimental study. Furthermore, we provide a new method of image processing with ideal parameters for deep learning architectures and MLAs. In response to a thorough scientific assessment of these methodologies, deep learning architectures perform more efficiently than traditional MLAs. All in all, our work suggests a scalable and multimodal deep learning system for real-time malware detection through visual means. An improved technique for successful zero-day malware detection is the visualization and deep learning architectures for static, dynamic, and image processing based blended methods in a big data environment.

Keywords: Malicious software, Deep learning, Machine learning, Data security, CNN, LSTM

I. INTRODUCTION

The swift evolution of technology has impacted both personal and a commercial everyday operation in the contemporary era of Industry 4.0. The modern notion of the information society has emerged as a result of the Internet of Things (IoT) and its numerous applications. However, privacy concerns provide a significant obstacle to reaping the rewards of this industrial revolution, since cybercriminals pursue specific PCs and networks in an attempt to steal personally identifiable data for financial gain and cripple systems. These attackers utilize malware or malicious software to put systems at considerable risk and highlight vulnerabilities [1]. A computer program intended to harm the operating system (OS) is known as malware.

According to its activities and goal, malware goes by a variety of labels. Among them are adware, spyware, viruses, worms, trojans, rootkits, backdoors, ransomware, and command and control (C&C) bots. Malware detection and mitigation is an ongoing concern in the world of cyber security. Malware programmers get better at avoiding detection as researchers build new methods.

A. Malware

Malware, an acronym for "malicious software," refers to any software that is intentionally created with the intention of causing harm to a computer, server, client, or computer network. Malware is an advanced threat that can be designed to do a variety of illegal activities.

The following are some common malware categories:

1) *Viruses:* Computer programs that multiply and contaminate other files on the system are called viruses. They usually require human interaction to spread, such as when an infected executable file is opened.
2) *Worms:* Worms, as opposed to viruses, spread automatically when users interact with them. By exploiting flaws in computer networks, they proliferate and spread from one system to another.

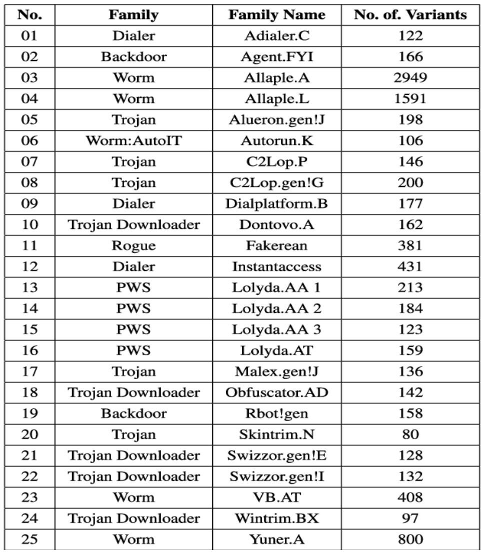
3) *Trojan:* Trojans are malicious programs that pretend to be reliable software. They commonly trick users into downloading and opening them, which gives hackers unauthorized access to compromised systems. Malware must be avoided by putting effective cybersecurity measures into place. Using firewalls, installing and updating antivirus software, applying security patches to software, avoiding shady email attachments and links, and practicing safe browsing are a few of these precautions. The impact of ransomware attacks can also be mitigated by routine data backups.
4) *Dialer Adialer.C:* "Dialer Adialer.C" is the name of a particular kind of malware called a trojan dialer. This spyware calls premium-rate phone numbers on your computer, which might lead to unforeseen and expensive phone bills. The majority of PCs with a modem attached to a phone line are impacted.

B. Data Security

The process of safeguarding digital information against theft, tampering, or illegal access is known as data security. The complete gamut of information security is covered by this idea. It encompasses administrative and access controls in addition to the physical security of hardware and storage devices. It also addresses organizational policies and procedures as well as the logical security of software programs.

C. DataSet

Utilizing the "MALIMG" binary malware dataset. This dataset comprises 25 malware families[1]. To create and test models for machine learning methods, the application will transform the binary information into grayscale images. These techniques, which are known as MalConv CNN and MalConv LSTM and another method known as EMBER, transform binary input to pictures before creating a model.



No.	Family	Family Name	No. of. Variants
01	Dialer	Adialer.C	122
02	Backdoor	Agent.FYI	166
03	Worm	Allaple.A	2949
04	Worm	Allaple.L	1591
05	Trojan	Alueron.gen!J	198
06	Worm:AutoIT	Autorun.K	106
07	Trojan	C2Lop.P	146
08	Trojan	C2Lop.gen!G	200
09	Dialer	Dialplatform.B	177
10	Trojan Downloader	Dontovo.A	162
11	Rogue	Fakerean	381
12	Dialer	Instantaccess	431
13	PWS	Lolyda.AA 1	213
14	PWS	Lolyda.AA 2	184
15	PWS	Lolyda.AA 3	123
16	PWS	Lolyda.AT	159
17	Trojan	Malex.gen!J	136
18	Trojan Downloader	Obfuscator.AD	142
19	Backdoor	Rbot!gen	158
20	Trojan	Skintrim.N	80
21	Trojan Downloader	Swizzor.gen!E	128
22	Trojan Downloader	Swizzor.gen!I	132
23	Worm	VB.AT	408
24	Trojan Downloader	Wintrim.BX	97
25	Worm	Yuner.A	800

Table – 1: Malware Families

II. RELATED WORK

A. Recognition of fraudulent Singleton Files on a huge-scale

94% of the billions of program binary files that showed up on 100 million computers over a 12-month period are found to be on a single machine, according to our analysis of a dataset. Given that the proportion of benign to harmful singleton files is 80:1, polymorphism in malware is one reason for the high number of singleton files; however, polymorphism in malware is also influenced by other factors. It is difficult to accurately detect the tiny percentage of adverse singletons due to the large quantity of benign singletons. We give an extensive analysis of the traits, features, and distribution of malicious and benign singleton files. By drawing on the knowledge gained from this investigation, we construct a classifier that only uses static features, allowing us to spot 92% of the remaining perilous singletons at a 1.4% false positive rate, even though the majority of harmful singleton files heavily utilize packing and camouflage techniques, which we don't try to de-obfuscate. In verdict, we exhibit the resilience of our classifier against significant categories of automated avoidance attempts.

B. *Comprehending Malware Activity by the Extraction of API Calls*

Using fillers or software tools that cause opaque code to avoid recognition by safeguarded scanners is one of the latest tactics used by malware writers. By using escape methods like metamorphism and polymorphism, malware can elude the detection strategies used today. Thus, removing payloads concealed within densely packed executables is a monumental effort for security researchers and the anti-virus business. It is standard procedure to employ software tools for static or manual unzipping, and to examine application programming interface (API) commands in order to locate malware. But acquiring these elements for inverse concealment from the packaged executables is a tedious task that necessitates a thorough understanding of low-level programming, which includes kernel and assembly code. With the aim to clarify how API call features might be used intentionally this work recommends a robotic means of collecting those aspects and analysing them. There is a dearth of literature on features in Malodes, in spite a few investigations being done on utilizing API call characteristics and similar techniques to arrive to file birthmarks. To attempt to close this gap, we make an effort to autonomously analyze and categorize API function request behavior in keeping with any malicious intent concealed in a packed application. This study develops a fully automated approach in four steps, identifying six primary categories of suspicious API call feature behavior.

C. *Detection of zero-day Malware with Supervised learning Techniques using API Call Signatures*

Code obfuscation techniques are used in the generation Of zero-day or unknown malware. These techniques permit the parent code to be modified to produce offspring copies with identical functionality but distinct signatures. The existing methods described in research are not able to identify zero-day malware with the necessary efficiency and accuracy. In the presented study, we have suggested and assessed a unique approach that uses many data mining approaches to efficiently and accurately identify zero-day malware based on the frequency of Windows API calls.

III. OBJECTIVE

At the moment, we use both static and dynamic analysis of request data to detect threats. Static analysis uses signatures to find out if a packet is normal or carries an offensive signature. We do this by juxtaposing the contents in a new request packet with the attack signature still in place. Although dynamic analysis costs a long time, it uses dynamic program execution to find malware or attacks. The contributor relies on machine learning algorithms to assess the prediction performance of different machine learning algorithms, including Support Vector Machine, Random Forest, Decision Tree, Naïve Bayes, Logistic Regression, KNearest Neighbors, and Deep Learning Algorithms like Convolution Neural Networks (CNN) and LSTM (Long Short-term Memory), in an effort to overcome this obstacle and increase detection accuracy with both old and new malware attacks.

IV. SYSTEM REQUIREMENTS

A. *Hardware Requirements*

1) Processor- Pentium –IV
2) Speed- 1.1 Ghz
3) RAM-256 MB(min)
4) Hard Disk- 20 GB
5) Key Board- Standard Windows Keyboard
6) Mouse- Two or Three Button Mouse
7) Monitor- SVGA

B. *Software Requirements*

1) Operating System- Windows Family
2) Programming Language- Python (python 3.7.0)

V. METHODOLOGY

This section evaluates the suggested approach's accuracy, precision, recall, and F-measure to show how reliable it is at identifying malware. Furthermore, we show that our suggested defence against junk code attacks is sustainable.

A. *Static and Dynamic Analysis*

In contemporary times, transaction data is analyzed both statically and dynamically to identify malicious attacks. By correlating a packet's signature to pre-existing attack signatures, static research demonstrates whether the packet is legitimate or has a vulnerability that an attacker could exploit.

The perpetual execution flow was used in dynamic examination to evaluate malware and exploits. But the simulation program gets utilized the night before bed. With the aim to increase the ability to spot with old as well as novel malware and viruses (Long Short-Term Memory), the author is using machine learning techniques for this challenge, such as SVM Algorithm, Random Forest, Decision Tree, Naive Bayes and Logistic Regression, KNearest Neighbors, and Deep Learning such as Convolution Neural Networks (CNN) and LSTM. CNN and LSTM perform more effectively than any other algorithm.

B. CNN

A kind of deep learning model called Convolutional Neural Networks (CNNs) is mainly employed for image recognition applications. They are perfect for detecting malware based on images since they are very good at finding patterns in huge visual information databases. A CNN's fundamental design is made up of multiple layers, each serving a distinct function. In the process of classifying malware, CNNs are trained to identify patterns in byte-level picture data that correspond to certain malware families. The input photos are represented as byte-level images. an exclusive CNN architecture that has shown to have superior performance and computational efficiency, for this classification issue. Multiple convolution 1D layers, pooling 1D layers, and fully connected layers can exist in a CNN network. The filters in the convolutional 1D layer skate across the 1D sequence data to gather the most relevant features. A novel feature set known as a feature map is created by clustering the features that originate from each filter. The length and number of filters are picked using a hyperparameter tuning strategy. On every aspect, this in turn utilizes the non-linear activation function, or ReLU.

C. LSTM

Long Short-Term Memory, or LSTM, is a form of RNN that is frequently implemented for time series analysis and natural language processing. However, it may also be utilized for picture-based classification tasks, such as object detection and image captioning. New investigations have shown that malware detection is another application for LSTMs.Classified IoT malware families and examined the pixel value sequence in malware sample photos using a multilevel deep learning architecture with LSTM. Classified obfuscated binaries from imagery using LSTM in conjunction with a CNN, and they applied transfer learning to increase classification accuracy. Overall, current studies has demonstrated good outcomes using LSTMs for malware detection.

D. Data Analysis

The malware binary files are converted into two-dimensional malware images, with the sizes of these images differing throughout the twenty-five tested families. To ensure that the malware imagery gathered in the first phase fit the input size of the CNN model being used, the malware images must be scaled as part of the pre-processing stage for malware information. Reducing the size of the input photos is the main advantage of this scaling strategy; this assists in accelerating up training and alleviate the computational strain on the CNN model that is being used. Furthermore, the key tactile characteristics of the malicious images are preserved throughout the re-dimensionalization process. By proactively removing pertinent features from data sequences, LSTM networks can eliminate the requirement for human feature engineering. This is particularly helpful for finding malware variants that have never been seen before or zero-day threats.

E. Measures Metrics

This is an appraisal of the several metrics for deep learning models used in image-based malware classification. Metrics are F1-Score, Precision, Accuracy, Recall.

Block Diagram

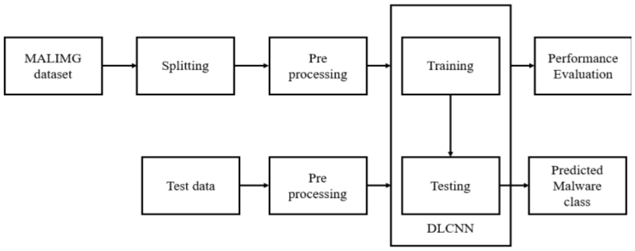


Fig.2 – Block Diagram

Flow Chart

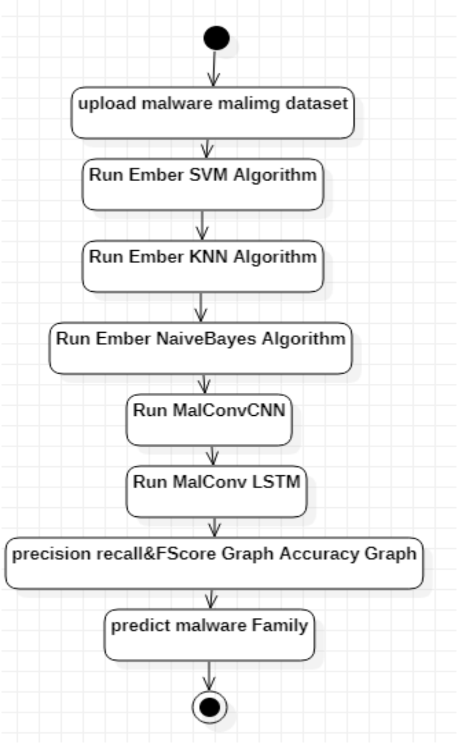
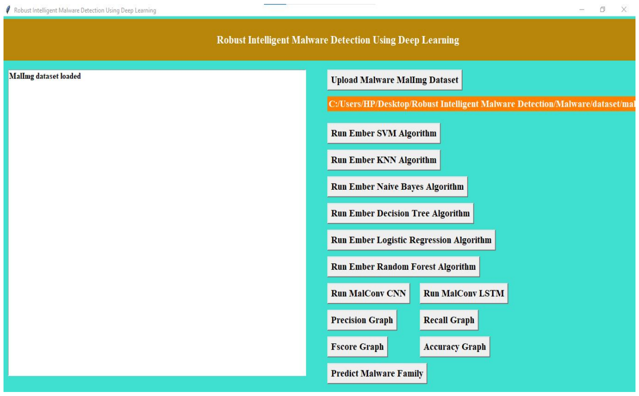

upload malware malimg dataset
Run Ember SVM Algorithm
Run Ember KNN Algorithm
Run Ember NaiveBayes Algorithm
Run MalConvCNN
Run MalConv LSTM
precision recall&FScore Graph Accuracy Graph
predict malware Family

Fig.3– Work Flow

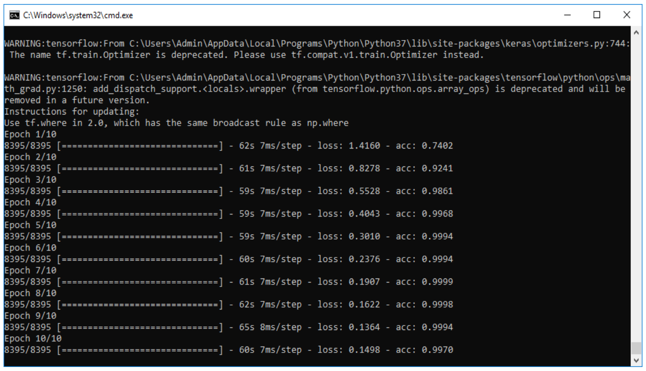
VI. RESULTS


Robust Intelligent Malware Detection Using Deep Learning
Robust Intelligent Malware Detection Using Deep Learning
Upload Malware MalImg Dataset
Run Ember SVM Algorithm
Run Ember KNN Algorithm
Run Ember Naive Bayes Algorithm
Run Ember Decision Tree Algorithm
Run Ember Logistic Regression Algorithm
Run Ember Random Forest Algorithm
Run MalConv CNN
Run MalConv LSTM
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family

MalImg dataset loaded

C:/Users/HP/Desktop/Robust Intelligent Malware Detection/Malware/dataset/mal

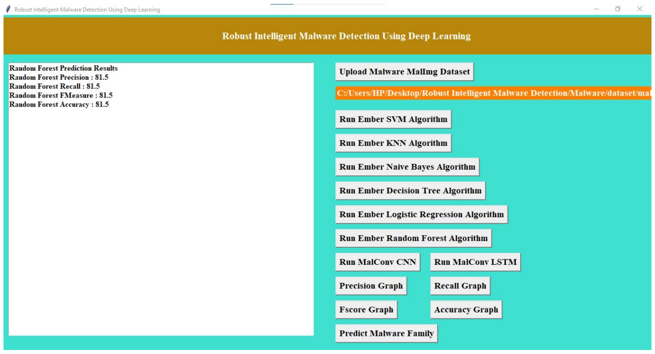
Fig.4 Interface



```
WARNING:tensorflow:From C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\keras\optimizers.py:744:
 The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorflow\python\ops\ma
th_grad.py:1250: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be
removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/10
8395/8395 [==============================] - 62s 7ms/step - loss: 1.4160 - acc: 0.7402
Epoch 2/10
8395/8395 [==============================] - 61s 7ms/step - loss: 0.8278 - acc: 0.9241
Epoch 3/10
8395/8395 [==============================] - 59s 7ms/step - loss: 0.5528 - acc: 0.9861
Epoch 4/10
8395/8395 [==============================] - 59s 7ms/step - loss: 0.4043 - acc: 0.9968
Epoch 5/10
8395/8395 [==============================] - 59s 7ms/step - loss: 0.3010 - acc: 0.9994
Epoch 6/10
8395/8395 [==============================] - 60s 7ms/step - loss: 0.2376 - acc: 0.9994
Epoch 7/10
8395/8395 [==============================] - 61s 7ms/step - loss: 0.1907 - acc: 0.9999
Epoch 8/10
8395/8395 [==============================] - 62s 7ms/step - loss: 0.1622 - acc: 0.9998
Epoch 9/10
8395/8395 [==============================] - 65s 8ms/step - loss: 0.1364 - acc: 0.9994
Epoch 10/10
8395/8395 [==============================] - 60s 7ms/step - loss: 0.1498 - acc: 0.9970
```

Fig.5 Loading/Training Model


Robust Intelligent Malware Detection Using Deep Learning
Robust Intelligent Malware Detection Using Deep Learning
Upload Malware MalImg Dataset
C:/Users/HP/Desktop/Robust Intelligent Malware Detection/Malware/dataset/mal
Run Ember SVM Algorithm
Run Ember KNN Algorithm
Run Ember Naive Bayes Algorithm
Run Ember Decision Tree Algorithm
Run Ember Logistic Regression Algorithm
Run Ember Random Forest Algorithm
Run MalConv CNN
Run MalConv LSTM
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family

```
Random Forest Prediction Results
Random Forest Precision : 81.5
Random Forest Recall : 81.5
Random Forest FMeasure : 81.5
Random Forest Accuracy : 81.5
```

Fig.6 Measures Metrics

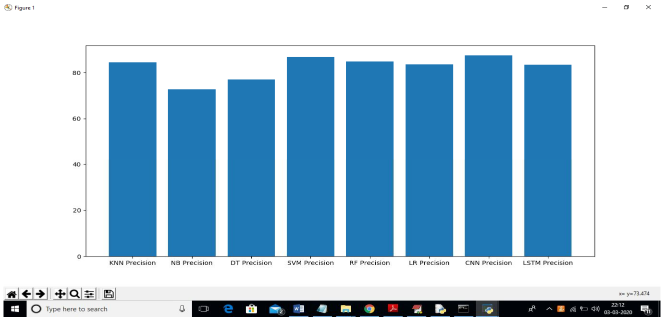
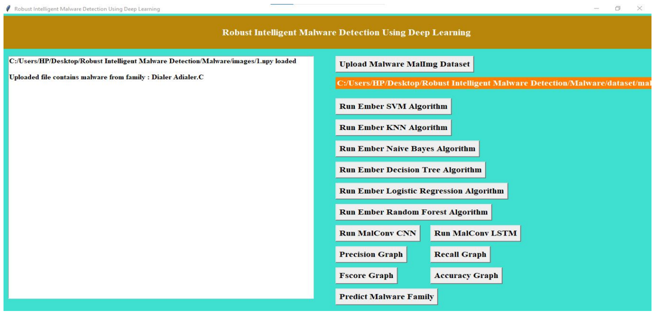


Fig.7 Measures Graphs


Robust Intelligent Malware Detection Using Deep Learning
Robust Intelligent Malware Detection Using Deep Learning
Upload Malware MalImg Dataset
C:/Users/HP/Desktop/Robust Intelligent Malware Detection/Malware/dataset/mal
Run Ember SVM Algorithm
Run Ember KNN Algorithm
Run Ember Naive Bayes Algorithm
Run Ember Decision Tree Algorithm
Run Ember Logistic Regression Algorithm
Run Ember Random Forest Algorithm
Run MalConv CNN
Run MalConv LSTM
Precision Graph
Recall Graph
Fscore Graph
Accuracy Graph
Predict Malware Family

```
C:/Users/HP/Desktop/Robust Intelligent Malware Detection/Malware/images/1.npy loaded

Uploaded file contains malware from family : Dialer Adialer.C
```

Fig.8 Predict Malware Family

VII. CONCLUSION

To investigate ways to detect malware, this paper evaluated deep learning architectures and classical machine learning algorithms (MLAs). It likewise created a highly scalable framework called ScaleMalNet that can detect, classify, and categorize zero-day malware. The architecture is based on static and dynamic analysis as well as image processing techniques. This structure uses a two-stage procedure for malware analysis and applies deep learning to the malwares that originate from end user hosts. For malware categorization in the first phase, a hybrid of static and dynamic analysis was utilized. Using image processing strategies, malwares were separated into related malware categories in the second stage. Deep learning-based approaches beat classical MLAs, according to a variety of experimental analyses carried out by applying modifications in the models on both the publicly available benchmark datasets and privately obtained datasets in this study. By extending a few extra ones to the current architectures, the newly designed framework can be scaled out to analyze a bigger variety of malwares in real-time. It can now analyze a huge number of malwares in real-time. Further research should investigate these variations using current attributes that could be included in the current data. The main conclusions, shortcomings, and potential applications of this work can be summed up as follows: • A scalable malware detection framework with two stages is suggested. • The performances obtained by deep learning architectures outperformed classical MLAs in static, dynamic, and image processing-based malware detection and categorization. • The proposed framework uses state-of-the-art method, deep learning, which detects the malware in first level and in second level the malware is categorized into the corresponding categories.

The Malimg dataset has a very unbalanced collection of malware families. A cost-sensitive tactics can be used to address the imbalanced problem of multiclass malware families. This makes it easier to incorporate the cost elements into deep learning architectures' backpropagation learning approach. The cost item mostly reflects the relevance of the classification, giving a greater value for classes with fewer samples and a lower value for those with more samples. In an antagonistic information, deep learning architectures are susceptible to attack. Deep learning architectures can be easily tricked by the generative adversarial network approach of generating samples during testing or deployment. The deep learning architectures' robustness isn't dealt with in the specified work. offered the severity of malware defection in circumstances where safety is a concern, this is one of the most essential directions for future research. A single misclassification has the potential to harm the organization in multiple ways.

REFERENCES

[1] Anderson, R., Barton, C., Böhme, R., Clayton, R., Van Eeten, M. J., Levi, M., ... & Savage, S. (2013). Measuring the cost of cybercrime. In The economics of information security and privacy (pp. 265-300). Springer, Berlin, Heidelberg.

[2] Li, B., Roundy, K., Gates, C., & Vorobeychik, Y. (2017, March). LargeScale Identification of Malicious Singleton Files. In Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (pp. 227-238). ACM.

[3] Alazab, M., Venkataraman, S., & Watters, P. (2010, July). Towards understanding malware behaviour by the extraction of API calls. In 2010 Second Cybercrime and Trustworthy Computing Workshop (pp. 52-59). IEEE.

[4] Tang, M., Alazab, M., & Luo, Y. (2017). Big data for cybersecurity: vulnerability disclosure trends and dependencies. IEEE Transactions on Big Data.

[5] Alazab, M., Venkatraman, S., Watters, P., & Alazab, M. (2011, December). Zero-day malware detection based on supervised learning algorithms of API call signatures. In Proceedings of the Ninth Australasian Data Mining Conference-Volume 121 (pp. 171-182). Australian Computer Society, Inc.

[6] Alazab, M., Venkatraman, S., Watters, P., Alazab, M., & Alazab, A. (2011, January). Cybercrime: the case of obfuscated malware. In 7th ICGS3/4th e-Democracy Joint Conferences 2011: Proceedings of the International Conference in Global Security, Safety and Sustainability/International Conference on e-Democracy (pp. 1-8). [Springer].

[7] Alazab, M. (2015). Profiling and classifying the behavior of malicious codes. Journal of Systems and Software, 100, 91-102.

[8] Huda, S., Abawajy, J., Alazab, M., Abdollalihian, M., Islam, R., & Yearwood, J. (2016). Hybrids of support vector machine wrapper and filter-based framework for malware detection. Future Generation Computer Systems, 55, 376-390.

[9] Raff, E., Sylvester, J., & Nicholas, C. (2017, November). Learning the PE Header, Malware Detection with Minimal Domain Knowledge. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (pp. 121-132). ACM.

[10] Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., ... & Van Steen, M. (2012, May). Prudent practices for designing malware experiments: Status quo and outlook. In Security and Privacy (SP), 2012 IEEE Symposium on (pp. 65-79). IEEE.