International Journal for Electronic Crime Investigation

ISSN: 2522-3429 (Print)
ISSN: 2616-6003 (Online)

**DOI:** https://doi.org/10.54692/ijeci.2024.0802192

Vol. 8 issue 2 Apr-Jun 2024

# Enhanced Malware Detection Using Deep Learning: A Comprehensive Framework for Feature Extraction and Classification

**Zohaib Ahmad**
Faculty of Electronics and Information Engineering, Beijing University of Technology, Beijing, China.
Correspondence Author: ahmedzohaib03@gmail.com

**Received:** March 21, 2024; **Accepted:** April 27, 2024; **Published:** June 14, 2024

## ABSTRACT

The exponential growth and sophistication of malware necessitate new detection strategies. The rapid evolution of malware makes traditional manual heuristic practices ineffective in perceiving new malware variants. Machine learning systems have proven essential for automating the dynamic and static analysis, as they cluster similar malware behaviors and classify new infections based on their similarity to approved malware families. This research validates that deep learning networks can accomplish higher accuracy than customary machine learning approaches. Deep learning has multiple neural network layers, which allows it to better automatically ascertain and classify malware variants. It offers a framework for removing multiple signature sets, including parts, opcode, bytecode, and system calls, from malware files. Experimental consequences indicate that the most accurate feature vector is the feature vector generated through system calls. This study concludes that deep learning approaches outperform traditional shallow machine learning systems in terms of malware recognition and classification precision.

**Keywords:** Malware analysis, deep learning, feature extraction, neural networks

## 1. INTRODUCTION

Malware refers to malicious software designed to compromise the reliability, security, and operation of a computer structure without the user's permission. This type of software can attain the attacker's malicious objectives, such as hijacking the computer or stealing data. Modern antivirus programs rely on signature-based revealing techniques that generate various signatures for identified malware and store them in a database for later identification. However, signature-based procedures are limited in effectiveness since they cannot identify malware whose signatures are not yet recorded in the database. Attackers often use procedures such as polymorphism, encryption and obfuscation to evade detection, making it problematic for signature-based systems to keep up with the ever-changing threat landscape [1] [2].

To address this encounter, static and dynamic analysis techniques are used to find variants of known threats. Signatures generated by these techniques can be used to classify unknown samples into existing families and group malware. Static analysis examines code without executing it to find configurations and extract data i.e. text, byte sequence, call graphs and opcodes, and. Memory dump tools retrieve and examine protected code from system memory, while disassembler tools reverse engineer executables to generate assembly instructions. However, static analysis is difficult because obfuscation, encryption, polymorphism, and transformations hinder decompilation [3] [4].

Malicious code is dynamically analyzed using a virtual or controlled environment and tools such as Process Explorer, Regshot, and Process Monitor. This approach looks closely at instruction traces, function parameters, function calls, etc. [5][6] Dynamic analysis does not destroy executables, but records malware behavior into a feature vector space, which is time-consuming and resource-intensive. Also, some viruses may behave differently in virtual environments or be triggered in certain circumstances, making them difficult to identify [7] [8].

Machine learning systems can already automate malware analysis and classification, reducing the need for thorough manual investigation, even if static and dynamic procedures are still necessary. Unknown malware is categorized into different families using machine learning procedures i.e. clustering and classification, which can look for patterns in static and dynamic examination [9][10].

Determining if a file contains malware-filled is a classification issue. To do this, we have used various machine learning procedures including Support Vector Machine (SVM), and Decision Tree. Typically, a dataset covers files that are to be classified as benign or malicious. A model has been trained using the training samples of these datasets, which are alienated into a test set and a training set. Cross-validation procedures enhance the evaluation of the model. The quantity of properly classified documents determines the quality of the model, and the model can predict the labels of the test set after training. However, common machine learning practices are limited in precision by their shallow learning architecture, while deep learning procedures with more automated feature learning capabilities can advance accuracy [11][12].

Deep learning procedures incorporate many layers to extract information from the lowest to the highest layer. After each layer detects a particular feature and forward it to the subsequent layer, the subsequent layer merges the features of lower-level into the features of the higher-level. This aggregation process continues until the final layer determines whether the file is benign or malicious. Deep learning models can extract features on their own, whereas traditional machine learning necessitates features to be explicitly served into the network. In this training, we use the dataset to test machine and deep learning performances and compare the outcomes.

Given the preceding discussion, the following key point's summaries the main contributions and findings of this research paper.

- Draw attention to how deep learning networks can outclass machine learning performances in terms of accuracy.
- Provide a comprehensive approach to remove signature sets such as opcodes, bytecode, and system calls from malware files.
- Explain how to find variants of known malware threats using static and dynamic study performances.
- Classify malware using various machine-learning systems
- Explain how to train and evaluate these algorithms using labelled datasets.
- Explain how deep learning models can self-extract features compared to traditional machine learning procedures that require manual input of features.
- Demonstrate that deep learning techniques can significantly advance detection and classification precision.
- Provide experimental evidence that deep learning performances offer high accuracy and performance compared to traditional shallow machine learning practices.

The remainder of the document is organized as: Related work is offered in Section 2, our deep-learning approach for malware revealing is familiarized in Section 3, and its effectiveness is assessed in Section 4 when compared to other malware detection alternatives. Next comes Section 5, which brings the paper to a conclusion.

## 2. RELATED STUDIES

Machine learning techniques have been widely used in various fields such as recommender systems [18], PVC pipe break detection [15][16][17], sentiment analysis [14], and wildfire detection [13]. Applying these methods to the field of malware detection is the main goal of this research. Traditional antivirus software often fails to identify malware variants due to missing signatures in their databases. Machine learning systems overcome this limitation by identifying small patterns of behavior and classifying these variants into known families.

The studies listed below demonstrate the development of machine learning based malware classification.

- In [19], the authors used static analysis to obtain text, byte sequence, and system resource information. For classification, they used algorithms such as RIPPER, Naive Bayes, and multi-classification systems.
- In [20], the authors used bytecode n-grams as features to evaluate techniques including Naive Bayes, SVM and decision trees. The decision tree algorithm performed better than other algorithms.
- To compare and classify malware programs into families based on the similarity of their call graphs, the authors in [21] proposed a system that utilizes call graphs as features and distance measures.
- The authors in [22] use K-nearest neighbors to classify malware and visualize it using image processing procedures.
- The authors in [23] showed that function length and frequency are important in identifying malware families by classifying Trojans based on these characteristics.
- The authors in [24] studied the use of labeled and unlabeled examples for malware classification in semi-supervised learning systems.
- To significantly reduce the runtime overhead, the authors in [25] projected an incremental method for behavior type analysis that combines the clustering and classification structure.

- The authors in [26] used decision tree method and the random forest systems to categorize worms and identify them using variable length instruction sequences.
- Using pcap files, the developers [27] focused on the malware's network activity. Using the J48 decision tree classifier, they were able to achieve good classification results by extracting flow information to create a behavior graph and using features that reflect the network behavior.
- The authors of [28] use a graph structure consisting of malware behavior to analyze dynamic analysis data. They then classify the data using an SVM trained on a similarity matrix. These papers show the progression of malware detection strategies from static and dynamic analysis to more complex machine learning procedures, which improve classification accuracy and flexibility in identifying new malware variants.

## 3. MODEL DEVELOPMENT METHODOLOGY

The process for constructing the malware detection model consists of many main steps:

### *3.1. Data Collection*

The data used to train and evaluate the model came from Microsoft's malware dataset, which is accessible on Kaggle. This collection contains 10,868 malware files from distinct families, including Ramnit, Kelihos_ver3, Simda, Vundo Tracur, Obfuscator.ACY, Kelihos_ver1, and Gatak. Each file in the collection has a unique identity and a class label that indicates its family.

### *3.2. Data Preprocessing*

The raw data must be converted into a feature vector space before analysis can begin. The dataset is used to create four different kinds of feature sets:

- 00 to FF are the hexadecimal codes that designate the frequency of bytecodes.
- Opcode frequency: The frequency of machine language commands like CMP, ADD and SUB.
- The frequency of sections in an object file, such as.init,.text, and.bss, are relocatable sections.
- System calls with one-hot encoding: API calls made by user applications to get kernel services

Following a min-max scaler approach standardization, these characteristics are randomized and divided into training (80%) and testing (20%) samples. Cross-validation, which divides the training samples into several lesser samples for training and the samples for validation, are used to prevent overfitting.

### *3.3 Feature Extraction and Analysis*

To produce an extensive input for the model, the processed feature sets are examined. For classification, a variety of machine-learning systems are used, such as SVM approach, Decision Trees and Naïve Bayes. Cross-validation is utilized to separate the dataset into the training samples and testing samples to advance model assessment.

### *3.4. Model Training and Validation*

The dataset is allocated into training and testing subsets to solve the classification issue. The system is trained on the sample of training, and its precision is calculated on the samples of testing. Additionally, deep learning techniques are used, extracting characteristics from the level of lowest to the highest across numerous layers. The file is categorized as either benign or malicious by the last layer.

## 4. PERFORMANCE MEASURE

The confusion matrix was utilized to construct the following measures, which was utilized to weigh the efficacy of deep neural networks (DNN) and machine learning processes for malware classification:

- TP (True Positive): The quantity of files that were precisely categorized as harmful.
- The quantity of files correctly categorized as benign (TN, True Negative).
- FP (False Positive): The files those were incorrectly identified as harmful.
- FN (False Negative): The files those were incorrectly identified as benign.

  Key performance metrics include:
- Percentage of real harmful files that are correctly categorized (TP / (TP + FN)) is identified as the True Positive Rate (TPR) or recall.
- The ratio of benign files mistakenly labelled as malicious (FP / (FP + TN)) is known as the False Positive Rate (FPR).
- F1-Score: ((2 * accuracy * Recall) / (Precision + Recall)) is the harmonic mean of accuracy and recall.
- Precision: Total proportion of accurately categorized files ((TP + TN) / (TP + TN + FP + FN)).
- Precision: The proportion of harmful files accurately recognized among all files designated as malicious (TP / (TP + FP)).

## 5. EXPERIMENTAL RESULTS

In the field of malware analysis, traditional research has focused on binary classification, classifying data into two categories: harmful and benign. Our research, on the other hand, seeks to predict specific malware file family types by solving the more difficult multi-class classification task.

Our strategy involves applying machine learning practices to create nine binary classifiers, each specialized for predicting a single malware family, to solve the multi-class classification challenge. The class that receives the highest score from each classifier is assigned as the expected class during classification. Interestingly, the random forest classifier functions differently as it does not require the creation of a binary classifier beforehand; instead, it immediately classifies instances into several classes.

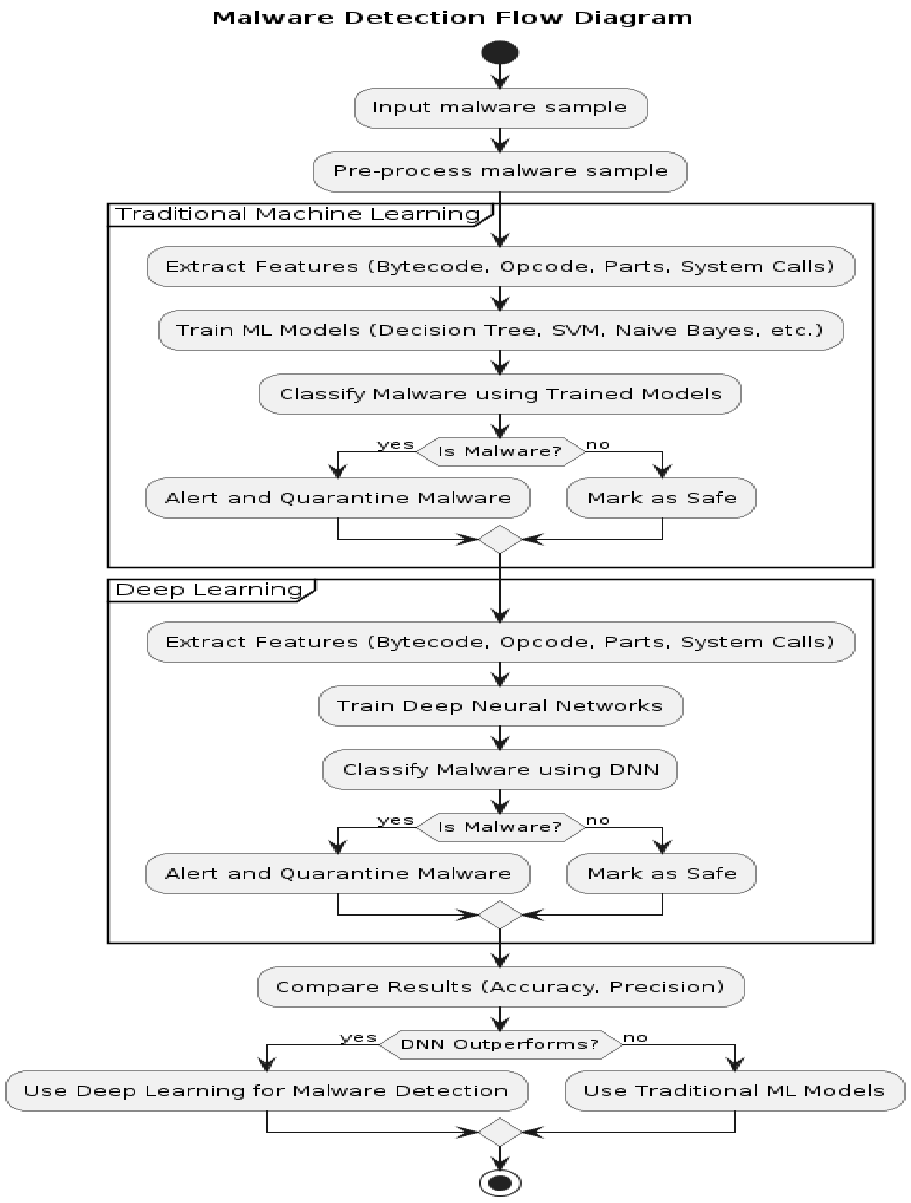
Malware Detection Flow Diagram
Input malware sample
Pre-process malware sample
Traditional Machine Learning
Extract Features (Bytecode, Opcode, Parts, System Calls)
Train ML Models (Decision Tree, SVM, Naive Bayes, etc.)
Classify Malware using Trained Models
yes
Is Malware?
no
Alert and Quarantine Malware
Mark as Safe
Deep Learning
Extract Features (Bytecode, Opcode, Parts, System Calls)
Train Deep Neural Networks
Classify Malware using DNN
yes
Is Malware?
no
Alert and Quarantine Malware
Mark as Safe
Compare Results (Accuracy, Precision)
yes
DNN Outperforms?
no
Use Deep Learning for Malware Detection
Use Traditional ML Models

**Figure 1: Flow diagram for malware detection**

### *5.1 Evaluation Method*

We used a rigorous 3-fold cross-validation method for assessment. This technique splits the dataset into three subsets: one-third is used to assess the model's performance, while the other two-thirds are used to train the algorithms.

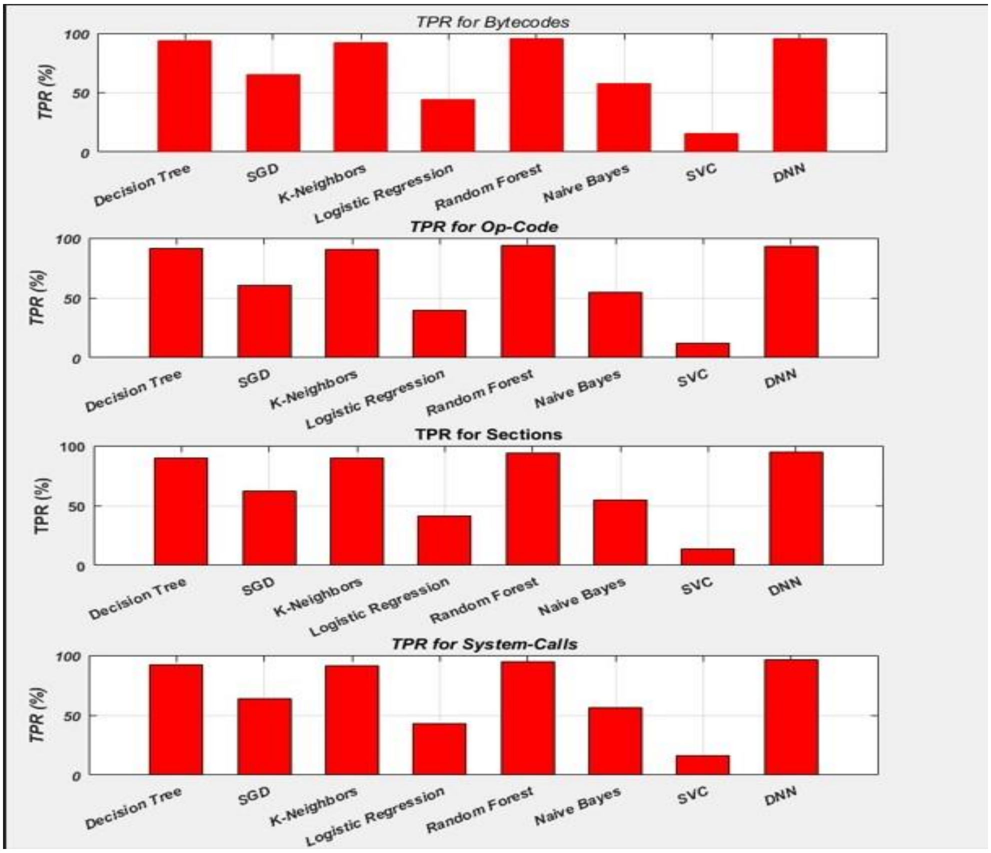

**Figure 2 (a): Comparison of TPR for Different Algorithms (Bytecodes, Op-Code, Sections, and System-Calls)**

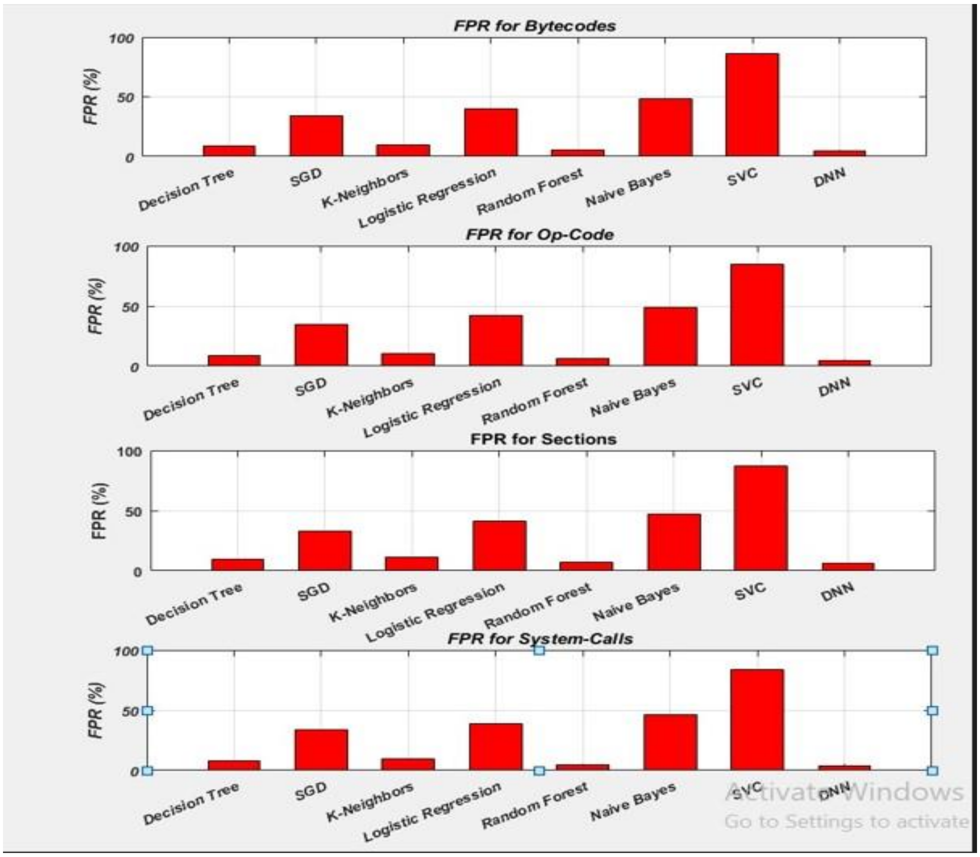

**Figure 2 (b): Comparison of FPR for Different Algorithms (Bytecodes, Op-Code, Sections, and System-Calls)**

### *5.2. Performance Metrics Comparison*

We contrasted the accuracy, True Positive Rate (TPR), and False Positive Rate (FPR) of deep learning strategies with machine learning performances based on shallow learning.

Figure 2, Byte-Code, Sections, Op-Code, and System-Calls: illustrate the TPR and FPR for different feature representations used in the classification process. Across all representations, deep-learning models exhibit superior TPR and lower FPR compared to shallow-learning methods.

The success of suggested deep learning malware classification system is attributed to the effective use of the gradient descent and back-propagation performances. These mechanisms facilitate iterative adjustment of model weights to minimize loss, thereby enhancing overall accuracy, and TPR, and reducing FPR.

## 6. CONCLUSION

This study offers a thorough framework for deep learning-based improved malware detection. In comparison to conventional machine learning techniques, the suggested method provides improved accuracy by using feature samples include operational codes, system calls, byte codes and sections. According to the experimental results, deep learning models greatly enhance malware variant identification and classification, offering a reliable countermeasure to the constantly changing threat landscape. Subsequent research endeavors will centre on refining the deep learning models and investigating supplementary feature sets to augment detection proficiency.

## REFERENCES

[1] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs." *In Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, IEEE, pp. 49-63, 2002.

[2] M. Christodorescu and S. Jha, "Testing malware detectors." *ACM SIGSOFT Software Engineering Notes,* vol. 29, no. 4, pp.34-44, 2004.

[3] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering." In *NDSS*, vol. 9, pp. 8-11, 2009.

[4] Z. Li, A. Goyal, Y. Chen, and V. Paxson, "Automating analysis of large-scale botnet probing events." *In Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pp. 11-22, 2009.

[5] Y. Ye, D. Wang, T. Li, and D. Yet, "An intelligent PE-malware detection system based on association mining." *Journal in computer virology,* vol. 4, pp.323-334, 2008.

[6] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models." *arXiv preprint arXiv: 1804.04637* ,2018.

[7] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and J. Blasco, "Dendroid: A text mining approach to analyzing and classifying code structures in android malware families." *Expert Systems with Applications*, vol. 41, no. 4, pp. 1104-1117, 2014.

[8] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification." In Proceedings of the 8th international symposium on visualization for cyber security, pp. 1-7. 2011.

[9] S. Poudyal, "Multi-level analysis of Malware using Machine Learning". *The University of Memphis*, 2021.

[10] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection." In 2010 IEEE symposium on security and privacy, IEEE, pp. 305-316. 2010.

[11] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning." *Journal of computer security*, vol.19, no. 4, pp. 639-668, 2011.

[12] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine

learning aided Android malware classification." *Computers & Electrical Engineering*, vol.61, pp.266-274, 2017.

[13] W. Lidong, Z. Huixi, Z. Yin, H. Keyong and A. Kang, "A deep learning-based experiment on forest wildfire detection in machine vision course." *IEEE Access*, vol.11, pp. 32671-32681, 2023.

[14] C. Anmol, M. Pulkit and G. Mohit, "Sentiment analysis of text using deep convolution neural networks." *In 2017 Tenth international conference on contemporary computing (IC3),* IEEE, pp. 1-6, 2017.

[15] Y. Xiangyang, Z. Jian'e, L. Ruohan ,W. Yajiao, G. Lili, Y. Zhonghan ,and H. Xiaoqing, "Diagnosis of sewer pipe defects on image recognition of multi-features and support vector machine in a southern Chinese city." *Frontiers of Environmental Science & Engineering*, vol.13, pp. 1-13, 2019.

[16] M. Adrien, K. Nikos, R. Christian C,and M. Dirk, "Machine learning classifiers for surface crack detection in fracture experiments." *International Journal of Mechanical Sciences*, vol. 209, pp. 106698, 2021.

[17] K. M Safeer, Z. Kaiman, W. Nansong, and U. Ishaq, "Robotics and Deep Learning Framework for Structural Health Monitoring of Utility Pipes." *In 2019 IEEE International Symposium on Measurement and Control in Robotics (ISMCR),* IEEE, pp. B3-3. 2019.

[18] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new persp ectives." *ACM computing surveys (CSUR)*, vol. 52, no. 1, pp. 1-38, 2019.

[19] D. Schultz, E. Eskin, F. Zadok, and S. Stolfo, "Data mining methods for detection of new malicious executables." *In Proceedings 2001 IEEE Symposium on Security and Privacy. S&*P 2001, IEEE, pp. 38-49, 2000.

[20] N. Lakshmanan, J. Gregoire and M, BS "Detecting packed executables based on raw binary data." *Technical report*, 2010.

[21] W. Hu, W. Hu, and S. Maybank, "Adaboost-based algorithm for network intrusion detection." *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 38, no. 2, pp.577-583, 2008.

[22] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild." *Journal of Machine Learning Research,* vol.7, no. 12, 2006.

[23] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD: A multiple classifier system for accurate payload-based anomaly detection." *Computer networks*, vol.53, no. 6, pp. 864-881, 2009.

[24] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification." *In Proceedings of the sixth ACM conference on data and application security and privacy,* pp. 183-194. 2016

[25] L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security

intrusion detection." *IEEE Communications surveys & tutorials,* vol. 18, no. 2, pp. 1153-1176, 2015.

[26] S. A. Yeswanth, N.K. Reddy, B and V. R. Gupta, "Malware Detection Using Machine Learning Techniques." *In Smart Data Intelligence: Proceedings of ICSMDI 2022, Springer Nature Singapore*, pp. 95-107, 2022

[27] W. Mayuri, D.T, Fabio and S. Mark, W. Mayuri , D. T, Fabio and Stamp, Mark "Detecting malware evolution using support vector machines." *Expert Systems with Applications*, vol.143, pp. 113022, 2020.

[28] K. Xu, H. Shen, and H. Chen, "Trafficav: An effective and explainable detection of mobile malware behavior using network traffic." *In 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS*), IEEE, pp. 1-6. 2016.