# MALWARE DETECTION USING DEEP LEARNING

**MOYYI APPALANAIDU,**

DEPARTMENT OF COMPUTER SCIENCE AND SYSTEM ENGINEERING

**ABSTRACT**:

Malware detection is a critical task in cybersecurity to protect systems and networks from malicious software threats. Traditional signature- based approaches often struggle to keep up with the rapidly evolving landscape of malware. In recent years, deep learning techniques have shown promising results in detecting malware by leveraging the sequential nature of the data.

This paper presents a novel approach that combines Long Short- Term Memory( LSTM) and Gated Recurrent Unit ( GRU) networks for effective malware detection. The LSTM and GRU architectures are bot h variants of Recurrent Neural Networks ( RNNs) known for their ability to capture long- term dependencies in sequential data. By combining these architectures, the proposed method aims to leverage the complementary strengths of LSTM and GRU models for enhanced malware detection performance.

The model's performance is rigorously evaluated using established metrics, including accuracy, precision, recall and F1- score. Comparisons are made with baseline models, including single LSTM and GRU models, as well as traditional machine learning classifiers.

The results demonstrate the effectiveness of the combined LSTM and GRU ensemble model in accurately detecting malware, particularly previously unseen variants. The ensemble approach capitalizes on the diverse capabilities of LSTM and GRU networks, resulting in improved detection rates and reduced false positives. The results obtained with model accuracy of 99% are presented.

## 1.INTRODUCTION

Malware can be defined as a set of malicious files or programs and may take many forms like Root kit, Spyware, Botnet, Trojan, Ransomware and gain unauthorized or unprivileged access to files in victim systems or servers. Malware can affect devices like desktops, laptops, mobile phones, health care devices, enterprise servers, clients, and network devices. Malware detection means finding the presence of malware in a given host or detecting the malicious behavior of a program. Email, chat clients, phone conversations, SMS messages, and even postal mail are used to communicate with other systems or software.

Malware poses significant threats to the securit y and integrity of computer systems, networks, and sensitive data. Detecting and mitigating these malicious software instances is crucial to safeguarding digital environments. Traditional approaches to malware detection often struggle to keep up with the evolving tactics employed by malware authors. In recent years, deep learning techniques have gained prominence for their ability to effectively analyze complex patterns and dependencies in sequential data.

This explores an innovative approach for malware detection that leverages the power of two popular recurrent neural network ( RNN) architectures: Long Short - Term Memory ( LSTM) and Gated Recurrent Unit ( GRU). LSTM and GRU are advanced variants of RNNs that excel at capturing long-term dependencies in sequential data, making them well- suited for analyzing the dynamic and evolving nature of malware.

By combining the strengths of LSTM and GRU models, this approach aims to enhance the accuracy and robustness of malware detection. The paper provides detailed insights into data preprocessing techniques, model configuration considerations, and t raining procedures for implementing the combined LSTM and GRU approach. It also explores strategies for customization, such as adjusting model parameters and handling imbalanced data, to optimize performance.

With the guidance offered in this paper, practitioners and researchers can develop advanced malware detection systems capable of accurately identifying and mitigating the risks posed by malware. By harnessing the capabilities of LSTM and GRU networks, organizations can strengthen their security posture, protect sensitive information, and proactively defend against evolving malware threats.

### 1.1 BACKGROUND:

The evolution of malware has mirrored the rapid advancements in technology. In the early days of computing, viruses and worms were relatively simple, often relying on basic replication and propagation mechanisms. As computing systems became more sophisticated and interconnected, malware authors adapted, developing increasingly complex and evasive strategies. Today, malware can be polymorphic, constantly changing it s code to avoid detection, or it can employ sophisticated techniques like zero - day exploits to target vulnerabilities in software and hardware.

Traditional antivirus solutions have relied on signature- based detection, where known malware patterns are matched against files and processes. While effective against known threats, these solutions struggle with zero- day attacks and polymorphic malware, as they lack predefined signatures for such variants. This limitation necessitates a shift towards more advanced detection techniques that can learn and adapt to emerging threats.

### 1.2 MOTIVATION:

The motivation for this research is twofold. Firstly, it arises from the pressing need for more effective malware detection mechanisms that can identify both known and previously unseen malware variants. Secondly, it is driven by the desire to explore the capabilities of deep learning, particularly LSTM and GRU networks, in addressing this critical cybersecurity challenge.

The arms race between cybersecurity defenders and malicious actors necessitates innovative solutions that can adapt and evolve alongside emerging threats. Traditional methods, reliant on static signatures or heuristics, often fall short in the face of polymorphic or zero -day malware. Machine learning offers a dynamic approach by learning patterns and behaviors from data. This research seeks to harness the capabilities of recurrent neural networks, which are well- suited for sequential data analysis, to enhance the accuracy and adaptability of malware detection.

### 1.3 RESEARCH OBJECTIVES:

The primary objective of this research is to develop a state- of- the- art malware detection model t hat combines LSTM and GRU networks to exploit their complementary strengths in capturing temporal dependencies. Specific research objectives include:

1. Designing a novel ensemble architecture that seamlessly integrates LSTM and GRU networks for malware detection.
2. Preprocessing and feature engineering of executable file data to prepare it for input to the model.
3. Training the ensemble model on a diverse dataset of executable files, encompassing a wide range of benign and malicious sample s.
4. Evaluating the model's performance on various metrics, including accuracy, precision, recall and F1-score.
5. Comparing the performance of the ensemble model with baseline models, including single LSTM and GRU networks, as well as traditional machine learning classifiers.
6. Discussing the implications of t he findings and the potential impact of the research on the field of cybersecurity.

### 1.4 CONTRIBUTION:

This research contributes to the fie ld of cybersecurity by proposing an innovative malware detection model t hat combines LSTM and GRU networks. The key contributions of this research include:

The development of a novel ensemble model

that harnesses the strengths of LSTM and GRU networks for improved malware detection.
Extensive experimentation and evaluation of the model's performance on a diverse dataset of executable files.
A comprehensive comparative analysis of the ensemble model with baseline models and traditional machine learning classifiers.
Insights into the adaptability and effectiveness of deep learning techniques, particularly LSTM and GRU networks, in addressing the challenge of malware detection.

## 2. RELATED WORK

Much of t he published recent research work is focused on building automatic malware detection mechanisms which use statistical methods rather than deterministic rules. Many works have also proposed Machine Learning based approaches solutions for cyber security related problems. Many of these operations may be automated to detect cyber- attacks in real time and prevent damage. It was found in the survey t hat most researchers are interested in detection of malware of different types.

A model is proposed in [ 1] using three datasets for t raining, testing, and scaling up. The paper discusses the use of cascade one sided perceptron first and cascade kernelized one sided perceptron later to identify malware files and reduce false positives. The work in [ 2] considers building a decision model with data processing, decision making and malware detection to classify and detect any suspicious malware. The malware analysis system is ML based and uses shared Nearest Neighbor technique.

In [ 3] information was collected from 2510 APK files of which 1260 were malicious apps. The paper proposed a machine learning based framework for detecting malicious apps using information from API calls and permissions. In [ 4] Microsoft office files related to malicious macros were analyzed using Machine learning methods. In [5] data is collected from packets instead of port numbers and protocols and aut o-mated malware detection was proposed using Convolution Neural Networks and other machine learning methods. In [ 6] Malware files are executed in the Cuckoo Sandbox and t raced malware process behavior to determine generated and injected processes. RNN ( Recurrent Neural Network) is used for feature extraction and later CNN is t rained to classify. In [ 7] a model based on deep auto - encoder and CNN was proposed. The data was collected from 23000 apps and was processed for use in deep learning models to identify malware in android apps. In [ 8] data was collected from VMs by running various malware like rootkits and Trojan horses. Both 2D and 3D CNN were used to impro e accuracy of detection. In [ 9] a binary file was classified to be malicious or benign. The model was tested on a dataset with 11130 binaries. In [ 10] data augmentation was used to generate variants of images obtained from malware samples. In [ 11] novel ensemble CNN based architecture was used for detection of both packed and unpacked malware. In [ 12 ] static and dynamic analysis tools were used to extract features from 7000 malware and 3000 benign files, and a classification model was built. In [ 13] a classification model is built by extracting requested permissions, vulnerable API calls, application’ s key information such as dynamic code, reflection code, native code, cryptographic code, and database from applications. In [ 14] An effective deep learning model ( MMD) was built to detect malware and 1000000 samples from EMBER dataset are considered to extract features.

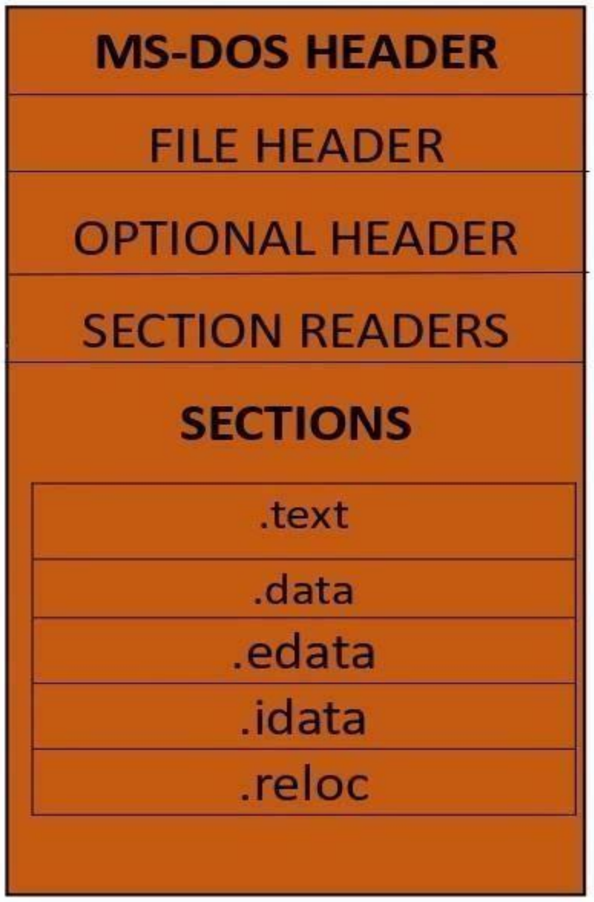

**Fig 1** PE File Format

## 3. TRAINING DATA

The selection and preparation of t raining data for malware detection are paramount to the success of the model. In our research, we adopted a rigorous approach to ensure that the t raining dataset is representative, diverse, and capable of capturing the intricacies of both benign and malicious software.

### 3.1. DATASET COMPOSITION:

Our training dataset consists of a comprehensive collection of executable files, encompassing a wide spectrum of samples. These samples are meticulously curated to strike a balance between benign and malicious instances. This balance is essential to avoid t raining bias and to enable the model to generalize effectively. The dataset includes:

#### 3.1.1. BENIGN EXECUTABLES:

A substantial portion of the dataset is comprised of benign executable files. These files are obtained from various legitimate software sources and represent the normal operating environment of computer systems. They include common applications, system utilities, and user software.

#### 3.1.2. MALICIOUS EXECUTABLES:

To challenge the model and facilitate the learning of malware patterns, we include a diverse range of known malicious executable files. These malware samples are sourced from malware repositories and security research datasets. They cover a variety of malware families, behaviors, and attack vectors.

### 3.2. DATA PREPROCESSING:

Before t raining the model, we perform extensive data preprocessing to extract relevant features from the executable files. This involves parsing file headers, extracting binary patterns, and transforming them into numerical representations suitable for input into the model. Additionally, we conduct feature engineering to enhance the discriminative power of the features, which is crucial for accurate detection.

### 3.3. DATASPLIT:

To ensure a robust evaluation of the model's performance, we split the dataset into two distinct subsets: t raining and test sets. Thet raining set accounts for approximately 80% of the dataset, while the test sets each constitute 20%. This split allows us to:

Train the model on a large and diverse t raining dataset to learn malware and benign software characteristics.
Fine- tune hyperparameters and monitor the model's performance on the validation set during training to prevent overfitting.
Evaluate the model's real- world performance on unseen data using the test set.

### 3.4. LABELLING:

Each sample in the t raining dataset is labeled as either " benign" or " malicious s" based on ground t ruth information. This labeling is critical for supervised learning, enabling the model to associate input features with the correct class labels during training.

The labels are 0 and 1 for malicious and benign with counts of 96724 and 41323 respectively.

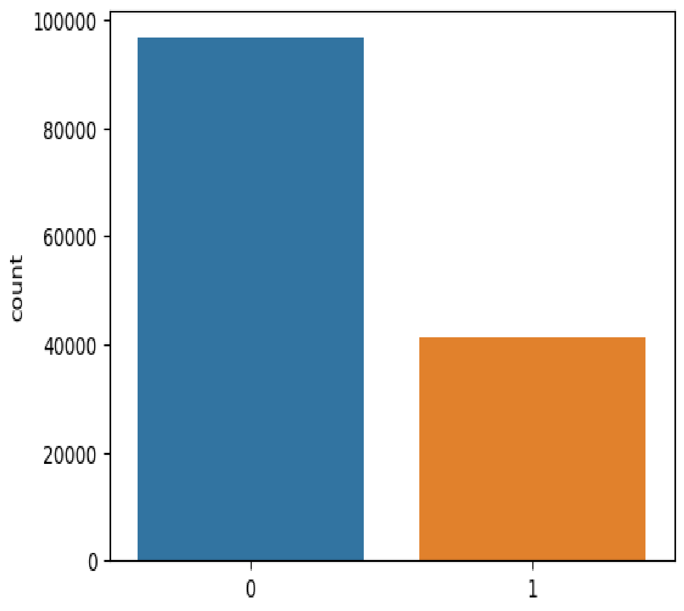

**Fig2** Label Distribution in The Dataset

## 4. MODELARCHITECTURE

The model architecture for malware detection combines the power of Long Short- Term Memory ( LSTM) and Gated Recurrent Unit ( GRU) neural networks. This architecture is designed to capture the sequential nature of malware sequences and effectively learn complex patterns and dependencies.

The input to the model consists of sequential data representing the behavior or characteristics of potential malware samples. This input is passed through both the LSTM and GRU layers in parallel. Each layer processes the input data and learns relevant representations at different levels of abstraction.

### 4.1. LSTM BRANCH:

The LSTM layer, known for its ability to capture long- term dependencies, consists of multiple LSTM units. Each unit incorporates a cell state and a set of gates, including the input gate, forget gate, and output gate. These gates control the flow of information within the unit, allowing the model to retain important information over long sequences. The LSTM layer learns to selectively update and utilize the cell state, facilitating the capture of crit ical dependencies in the malware sequences.

**4.1.1. INPUT LAYER:** The input to the LSTM branch is a sequence of feature vectors extracted from executable files. These feature vectors encapsulate essential information about the files, including characteristics, binary patterns, and other relevant data.

**4.1.2. LSTM LAYERS:** We employ one or more LSTM layers, with each layer consisting of a variable number of LSTM units. These units are responsible for processing the sequential data and capturing long- term dependencies. The output of each LSTM unit at each time step feeds into the next time step, allowing the network to remember information over extended periods.

**4.1.3. DENSE LAYER:** Following the LSTM layers, we include a densely connected layer with a RELU activation function. This layer acts as a feature extractor, transforming the LSTM's output into a more compact and representative feature space.

### 4.2. GRU BRANCH:

Similarly, the GRU layer, another variant of the RNN architecture, is also employed to capture sequential information. The GRU unit integrates an update gate and a reset gate to control the flow of information. The update gate determines the extent to which the unit updates it s hidden state, while the reset gate decides how much of the previous hidden state should be forgotten. This gating mechanism enables the GRU layer to effectively model dependencies and adapt to varying lengths of input sequences.

**4.2.1. INPUT LAYER:** Similar to the LSTM branch, the GRU branch also takes the same sequence of feature vectors as input.

**4.2.2. GRU LAYERS:** Just as in the LSTM branch, we employ one or more GRU layers. These layers consist of GRU units that process the sequential data, capturing short - term dependencies efficiently. The GRU units, like LSTM units, maintain an internal state that can capture dependencies within short time intervals.

**4.2.3. DENSE LAYER:** After the GRU layers, we introduce another densely connected layer with a RELU activation function. This layer plays a critical role in feature extraction, enabling the network to represent short-term dependencies effectively.

**4.3. MERGING LSTM and GRU BRANCHES:**
To combine the strengths of LSTM and GRU, the outputs of both layers are merged and passed through additional fully connected layers. These layers act as a classifier, mapping the learned representations to the target malware detection labels. The final layer uses an appropriate activation function, such as sigmoid or Soft Max to generate the output probabilities for each malware class.

**4.4. FINAL OUTPUT LAYER:**
The merged representation is then passed through a final densely connected layer with a sigmoid activation function. This layer serves as the classification head of the model, making binary predictions: whether the input executable file is benign or malicious. The sigmoid activation function ensures that the output is a probability score, allowing us to set a threshold for classification.

**4.5. TRAINING AND OPTIMIZATION:**
The model is t rained using labeled data, where each input sequence is associated with a binary label indicating its class ( benign or malicious). During t raining, the model adjusts its weights and biases to minimize the binary cross- entropy loss between the predicted probabilities and the actual labels. Optimization technique such as Adam is employed to expedite convergence.

**4.6. EVALUATION AND METRICS:**
To assess the model's performance, we employ a range of metrics, including accuracy, precision, recall, F1 - score. These metrics provide a comprehensive view of the model's ability to correctly classify executable files.

This model architecture facilitates the effective representation and analysis of malware sequences, enabling accurate detection. The LSTM layer captures long - term dependencies, while the GRU layer complements it by efficiently modelling sequential information. The combination of both architectures allows the model to learn intricate patterns and adapt to different characteristics of malware samples.

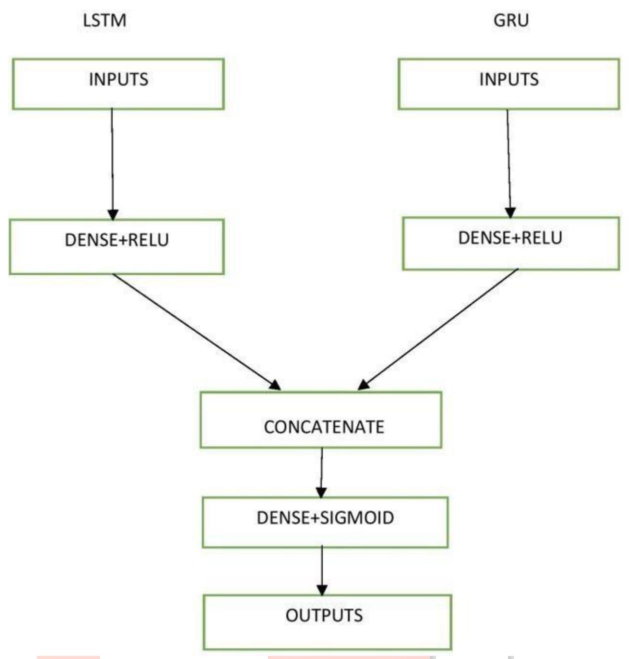

**Fig 3** System Architecture

Model: Sequential

| Layer (type) | Output Shape |
|---|---|
| LSTM input (Input Layer) | [(None, 13, 1)] |
| GRU input (Input Layer) | [(None, 13, 1)] |
| lstm (LSTM) | (None, 128) |
| gru (GRU) | (None, 128) |
| dense (Dense) | (None, 64) |
| dense_1 (Dense) | (None, 64) |
| concatenate (Concatenate) | (None, 128) |
| dense_2 (Dense) | (None, 1) |

**Fig4** Model Layers

## 5. TESTING MODEL

The testing phase of our malware detection model is a pivotal step in evaluating it s real- world efficacy. It involves subjecting the t rained model to a diverse set of executable files t hat it has never encountered before, simulating scenarios where the model needs to make predictions on previously unseen malware variants. In this comprehensive explanation, we will detail the various aspects of the testing model, including dataset preparation, metrics, and performance evaluation.

### 5.1. TESTING DATASET PREPARATION:

The testing dataset is a critical component of the testing model. It represents the real- world environment in which the model will operate. To ensure a rigorous evaluation, the testing dataset is distinct from thet raining and validation datasets, containing executable files that the model has never been exposed to during t raining. The test ing dataset is divided into two subsets:

**5.1.1. VALIDATION SET:** This subset is used during the testing phase for fine- tuning certain hyperparameters and for monitoring the model's performance without influencing the final evaluation results. It serves as a bridge between t raining and full- scale testing.

**5.1.2. TEST SET:** The primary subset used for model evaluation is the test set. It is entirely distinct from the t raining and validation sets and contains executable files that the model has never seen before. This subset is a true representation of the model's real- world performance.

### 5.2. DATA ENCODING AND PREPROCESSING:

Like the t raining phase, the test ing data undergoes encoding and preprocessing to prepare it for input into the model. This includes:

**5.2.1. EXTRACTING RELEVANT FEATURES:** Features extracted from the executable files include characteristics, binary patterns, and other relevant data.

**5.2.2. . 2 . ENCODING AND TRANSFORMATION:**
The feature vectors extracted from the testing data are encoded and t ransformed into a suitable format for input into the model, ensuring consistency with the training data preprocessing,

### 5.3. MODEL DEPLOYMENT:

During the testing phase, the t rained model is deployed to make predictions on the testing dataset. This deployment phase is crucial for evaluating the model's ability to generalize and make accurate classifications on previously unseen data.

### 5.1 METRICS FOR EVALUATION:

The performance of the model is evaluated using a comprehensive set of metrics these metrics provide insights into its abilit y to correctly classify executable files into benign or malicious categories. The key metrics used for evaluation include:

**5.1.1. ACCURACY:** Accuracy measures the overall correctness of the model's predictions. It is calculated as the rat io of correctly classified samples to the total number of samples.

**5.1.2. PRECISION:** Precision quantifies the model's ability to correctly identify malicious samples without producing false positives. It is calculated as the rat io oft rue positives to the sum of t rue positives and false positives.

**5.1.3. RECALL:** Recall, also known as sensitivity or t rue positive rate, measures the model's ability to identify all malicious samples correctly. It is calculated as the ratio of t rue positives to the sum of t rue positives and false negatives.

**5.1.4. F1-SCORE:** The F1- score is the harmonic mean of precision and recall. It provides a balance between precision and recall, considering both false positives and false negatives.

So, this section discusses the generation of customized malware and test ing combined model of Long Short -Term Memory( LSTM) and Gated Recurrent Unit ( GRU) model for classifying whether it is a malware or not. The data is pr e- processed and uses ek tree classifier to extract features and is submitted to the model for prediction.

## 6. CONCLUSION

Malware can be classified using deep learning techniques. This paper discusses a deep learning RNN model for malware detection. The paper has extensively surveyed several works published in literature. combined model of Long Short - Term Memory ( LSTM) and Gated Recurrent Unit ( GRU) was t rained and tested with malware dataset and was found to work with 99% accuracy which is quite a good improvement over [ 15]. Later the model was also tested with unknown malware generated through MSF venom, and it was correctly classified. In future the model will be fine- tuned and optimized for better accuracy. Detection in real time is another area to be explored

## 7. REFERENCE

1. Gavriluţ, D. Cimpoeşu, M., Anton, D. and Ciortuz, L., 2009, October. Mal- ware detection using machine learning. In 2009 IEEE Multiconference on Computer Science and Information Technology ( pp. 735- 741). IEE

2. Liu, L., Wang, B.S., Yu, B. and Zhong, Q. X., 2017. Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering, 18(9), pp. 1336 -1347.

3. Peiravian, N. and Zhu, X., 2013, November. Machine learning for android malware detection using permission and API calls. In 2013 IEEE 25th inter - national conference on tools with artificial intelligence (pp. 300 -305). IEEE.

4. Bearden, R. and Lo, D. C. T., 2017, December. Automated Microsoft office macro malware detection using machine learning. In 2017 IEEE International Conference on Big Data ( Big Data) ( pp. 4448- 4452). IEEE.

5. Yeo, M., Koo, Y., Yoon, Y., Hwang, T., Ryu, J., Song, J. and Park, C., 2018, January. Flow- based malware detection using convolutional neural network. In 2018 International Conf on Information Networking (pp. 910 -913).

6. Tobiyama, S., Yamaguchi, Y., Shimada, H.,Ikuse, T. and Yagi, T., 2016, June. Malware detection with deep neural network using process behavior. In 2016 IEEE 40th annual computer software and applications conference ( COMPSAC) ( Vol.2, pp. 577 -582). IEEE.

7. Wang, W., Zhao, M. and Wang, J., 2019. Effective android malware detection with a hybrid model based on deep auto encoder and convolutional neural network. Journal of Ambient Intelligence and Humanized Computing, 10( 8), pp. 3035 -3043.

8. Abdelsalam, M., Krishnan, R., Huang, Y. and Sandhu, R., 2018, July. Mal- ware detection in cloud infrastructures using convolutional neural networks. In 2018 IEEE 11th international conference on cloud computing (CLOUD) (pp. 162 -169). IEEE.

9. Sharma, A., Malacaria, P. and Khouzani, M. H. R., 2019, June. Malware de- tection using 1- dimensional convolutional neural networks. In 2019 IEEE European Symposium on Security and Privacy Workshops ( Euro S&PW) ( pp. 247 - 256). IEEE.

10. Catak, F. O., Ahmed, J., Sahinbas, K. and Khand, Z. H., 2021. Data augmen- tat ion-based malware detection using convolutional neural networks. Peer J Computer Science, 7, p. e346.

11. Vasan, D., Alazab, M., Wassan, S., Safaei, B. and Zheng, Q., 2020. Image- Based malware classification using ensemble of CNN architectures ( IMCEC). Computers & Security.

12. Jerlin, M. A. and Marimuthu, K., 2018. A new malware detection system using machine learning techniques for API call sequences. Journal of Applied Security Research, 13( 1), pp. 45-62.

13. Koli, J.D., 2018, March. RanDroid: Android malware detection using ran- dom machine learning classifiers. In 2018 Technologies for Smart -City Energy Security and Power ( ICSESP) ( pp. 1 - 6). IEEE.

14. Hyrum S. Anderson, Phil Roth., 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. _32_bit_St