International Journal of Safety and Security Engineering
Vol. 15, No. 1, January, 2025, pp. 141-150
Journal homepage: http://iieta.org/journals/ijsse

Optimizing Malware Detection and Classification in Real-Time Using Hybrid Deep Learning Approaches

Yaseen Ahmed Mohammed Alsumaidaee[1*], Mustafa Mahmood Yahya[2], Abdulelah Hameed Yaseen[3]

[1] Department of Computer Science, Al Imam Al Adham University College, Kirkuk 36015, Iraq
[2] Department of Technical Engineering Mechatronics, Technical College of Engineering, Al-Kitab University, Kirkuk 36015, Iraq
[3] Department of Petroleum Engineering, Technical College of Engineering, Al-Kitab University, Kirkuk 36015, Iraq

Corresponding Author Email: eng.yassin.ahmed@imamaladham.edu.iq

https://doi.org/10.18280/ijsse.150115

Received: 6 November 2024
Revised: 27 December 2024
Accepted: 18 January 2025
Available online: 31 January 2025

Keywords:
malware, 1D-CNN, 2D-CNN, LSTM, hybrid analysis, dynamic analysis, static analysis, deep learning

ABSTRACT

Malware detection and classification are critical for ensuring system security in real-time applications. Conventional approaches may not be optimized to combine precise results with low time consumption and become a problem when it comes to processing large volumes of different malware samples in a real-time setting. The general framework for this paper is to introduce a new detection and classification method that uses deep learning (DL) models to detect and classify malware. We developed and tested two models: the static convolutional neural network-long short-term memory (CNN-LSTM) model and the dynamic CNN 1D-LSTM model in this work. The models achieved an accurate rate of 99%. Static-CNN-LSTM was able to classify the malware based on static analysis. At the same time, the proposed dynamic (1D-CNN-LSTM) model got the best results, with a 100% success rate, by gathering behavioral data. This means that it can accurately classify even new and complicated dynamic malicious program variants. Therefore, this study's results show that using a hybrid approach raises the rate of detection while also meeting the real-time processing needs of systems with a lot at stake that need to perform well. Our approach represents a substantial improvement in malware detection, delivering a more efficient and versatile response to contemporary cyber threats.

1. INTRODUCTION

This rapid growth of the interconnected systems and devices in contemporary society means that malware and other cyber threats move fast across the domains of individuals and enterprises as well as state-critical facilities [1, 2]. Malware, viruses, worms, ransomware, and trojans are programs designed to cause harm, steal information, and give unauthorized access to data [3, 4].

Therefore, as these threats evolve, the landscape of antimalware defense becomes increasingly complex [5]. The previous techniques of threat detection are constantly failing to cope with the modern malicious programs leveraging methods such as polymorphism, obfuscation, and dispersing mechanisms to avoid identification [6-8].

That's why the limitations of traditional approaches to combating malware have led to the emergence of new methodologies [9, 10].

Essentially, signature-based techniques are highly efficient in identifying viruses, but they become ineffective when new or modified viruses are present [11].

Heuristic approaches try to solve this problem by concentrating on the behavior indicators but many of them produce numerous false alarms and the extensive time demands when real-time monitoring and control are needs across a large network. Automated recognition of malware patterns and behavior by incorporating Machine learning (ML) into the existing methods has advanced the method's ability to identify malware [12, 13].

Nevertheless, it has become apparent that highly sophisticated malware and the constant requirement for immediate responses necessitate the application of deep learning (DL) approaches for pattern identification concerning extraordinary patterns and variability to other categories, which could not previously be applied [14-16].

CNN and long-short-term memory (LSTM) networks are commonly used DL models that appear to possess high levels of efficiency in the detection of malware. CNNs excel at extracting spatial features, making them more suitable for analyzing static malware features like binary file patterns or opcode sequences. LSTMs concentrate on sequential data, offering a temporal perspective on the dynamic behavior of malware over time [17, 18].

But building a system that entails only one type of DL model may not fully provide an overall view of the nature and behaviors of the malware-especially in real-life detection since the rate of accuracy and speed are both of essence [19, 20].

Therefore, malware detection has been one of the main areas in cybersecurity studies for a long time already, because traditional approaches like signature-based detection and heuristic methods are not able to respond to the emerging challenges of modern malware effectively [21].

Given that modern threats are complex and can avoid fresh detection, scholars conduct ML and DL to enhance the accuracy and flexibility of approaches [22]. In this section we will present a critical discussion of the previous research with a focus on recent developments in the paradigm of ML and DL, hybrid models, as well as their application in real-time malware detection.

1.1 Machine learning for malware detection

The initial attempt at employing ML for malware detection was categorized by the use of supervised learning algorithms such as decision trees (DT), support vector machines (SVMs), and k-nearest neighbors (KNN). These methods attempted to classify malware according to traits derived from its source, binary form, or execution profile [23, 24].

But while used often with success for detection of known threats, they were far less effective at detecting polymorphic malware and displayed high false positive rates.

Wadkar et al. [25] investigated the evolution of malware families by employing feature ranking with linear SVMs weights to analyze the change in malware features over different time points. Though this method of classification is very efficient, it involves mostly feature selection, and it tends to overcomplicate data because of linear SVMs [26].

Furthermore, Wong et al. [27] implemented transfer learning on Shuffle Net and DenseNet-201, incorporating features from all layers and freezing the network to minimize overfitting. The last layer for classification is an optimal error correction output coding (ECOC) SVMs ensemble where three different configurations can be used, such as: one-against-one.

The One vs. One (OVO) and All (OVA) configurations aim to address issues related to computational complexity and data separation with non-linear margins. As for parameter searching, the grid search method with discrete values is introduced. The performance of the proposed model is evaluated on various datasets (Malimg, MaleVis, VirusMNIST, Dumpware (10). However, the use of SVM for the final classification can occasionally increase the complexity of the model and incur computational overhead when turning the SVM parameters.

Moreover, Babbar et al. [28] proposed a ML framework for Android malware detecting IoT device attacks. Therefore, the aforementioned framework utilizes static analysis techniques to identify disguised Android malicious apps and suggests restricting them due to their potential threat to users' privacy and system integrity. Training and testing are performed over more than ten thousand mobile Android applications using ML algorithms like KNN models, Naive Bayes, decision trees, and support vector machines.

The given KNN model results in a prediction accuracy of 93% with a precision 95% recall 90% and an F1 score of 92%. However, it could be relatively slightly effective against thorough malwares that use a number of small tricks, like obfuscating their code, since it makes the model not wholly tough."

On the other hand, proposes ML algorithms for Malware detection particularly, KNN enhanced with Firefly Optimization Algorithm (FOA) [29]. In this study, the author uses the developed MalMem-2022 dataset to investigate effects of parameter tuning and feature selection on classification. The results show the significance of the proposed technique, which enhances the accuracy, the recall, the precision, and the F1-score with up to 3.59% value in KNN with FOA.

The results presented in this study stress the necessity of the choice of the KNN parameters and use of FOA for the improved feature selection., Nevertheless, the approach may prove to experience difficulties with more intricate malware types, which may hinder its application in general.

Qasem et al. [30] described a detection system for distinguishing PDF files that are harmless from those that are malicious, in the current security threat by hidden malware in PDF files. From a precise AdaBoost decision tree system trained with best hyperparameters, the performance of the system is tested on the Evasive-PDFMal2022 dataset. The findings illustrate higher levels of model prediction accuracy of about 98.84%.

However, the approach of the certain dataset's usage may be inadequate in terms of stimulating diversification or uncertainty of new threats in malware.

Mat et al. [31] explained that the proliferation of mobile malware is a growing concern, prompting this work to advance an Android malware detection system that utilizes permission features under the Bayesian classification framework. The permission features are extracted using static code analysis from a pool of 10,000 samples from the AndroZoo and Drebin databases. Possible choices of feature selection depending on different algorithms are excluded, though information gain and chi-square are used to improve detection rates.

For malware detection using permission features, the proposed system reaches the maximum detection rate of 91.1%. However, the described approach has potential drawbacks, especially because the system can rely only on static analysis, so more advanced malware that tends to bypass permission checks may become a significant issue.

Additionally, the existing static ML based model faced difficulties in identifying new versions of malware that had some ability to transform themselves into other forms in an effort to avoid being identified, and the dynamic ML models were often large and time-consuming in terms of performance. Since the ability of models to calculate static and dynamic parameters was imperative but with comparatively fewer false positives, the researchers turned to DL mechanisms proficient in feature extraction and generalization.

1.2 Deep learning approaches

The adoption of DL models, specifically Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) such as LSTM, VGG16, and Gated Recurrent Unit (GRU) networks, marked a new phase in malware detection research [32, 33]. CNNs have proven effective for static malware analysis due to their strength in identifying patterns within structured data, such as binary files or opcode sequences.

Bakhshinejad et al. [34] described a malware detection system, where they use a parallel CNN model that performs feature extraction, and thus does not require a complex feature selection process. Due to the work with raw bytes of executables, the proposed model demonstrates high accuracy in recognizing malware and exemplifies the superiority of a DL approach over the ML one.

However, it uses raw byte data which, although offers flexibility and simplicity, may reduce the ability to analyze intermediated patterns and adapt to changing malware patterns.

On other side, Yeboah et al. [35] proposed an automatic nature feature extraction by using the 1D CNN-the approach is applied to Android malware detection. The model operates on n-grams of static opcode sequences which is semantically embedded and aimed to detect the existing malware in to- from binary files.

It applies several feature sets of n-gram opcode sequences and their predictions are obtained through weighted voting. the model can with a 98% positive predictive value and 97% sensitivity. Although the 1D CNN has low complexity, detection performance relies on the dataset's similarity; it might not generalize to a diverse set of malwares.

Furthermore, Jha et al. [36] introduced an effective RNN model that uses network temporal sequences to classify malware. This work evaluates the effectiveness of various hyperparameters by determining which step sizes have a greater impact on the classification of the RNN-based malware detector than their input sizes.

Three feature vectors are considered, of which one is hot encoding, the second one is random, and the third one is word2Vec. The performance metrics used are AUC and variance. The presented results show that the proposed algorithm with Word2Vec in the form of the feature vector based on the Skip-gram model of context words has the maximum AUC and optimal stability and is therefore the most effective. However, using specific feature vectors can be problematic when applying the model across various datasets of malware.

Moreover, Zhou et al. [37] introduced a more advanced GRU model known as SIMGRU targeting Android malware detection based on static analysis. They suggest three versions: InputSimGRU, HiddenSimGRU, and InputHiddenSimGRU based on the clustering similarity. Experimental outcome indicates that all three models outperform the basic GRU model and other detection techniques, proving that similarity in static analysis is useful. However, this approach can be unfavorable for complicated and evolutive malware that should modify functionalities in the course of their execution.

Deepa et al. [38] presented a method for the identification and categorization of malware utilizing DL and image-based characteristics. Traditional signature-based techniques can't handle new types; therefore, the authors use the Malimg dataset, which contains images of malware. The authors first convert malware samples into byteplot grayscale images, after which they extract features using a VGG16 pre-trained model. These features shape classifiers such as SVM,

XGBoost, DNN, and Random Forest to detect malware families. The results prove the high accuracy of the work; they used samples from 25 families and analyzed 9,339 samples in total. However, the use of image-based representations may not be efficient in handling non-confirming malware or obfuscation techniques.

1.3 Hybrid deep learning models

As it was mentioned, only one type of model may not be sufficient enough to cover all the features of malware; therefore, many recent works introduced a combination of CNNs and LSTMs for both static and dynamic analyses. They both enhance both the accuracy and the robustness of the detection mechanisms, and a combination of the two-a 'hybrid' approach-allows for doing a thorough analysis of file structure combined with behavioral data [39].

For example, Jeon et al. [40] presented HyMalD, a hybrid IoT malware detection framework that combines Bi-LSTM and SPP-Net, using both static and dynamic analysis to target obfuscated malware. HyMalD gathers static features from opcode sequences and dynamic aspects from Application programming interface (API) calls, achieving a detection accuracy of 92.5% and a false-negative rate of 7.67% when compared to solely static approaches. Yet, its dual-analysis approach raises the computational load, which can be problematic in constrained Internet of Things (IoT) equipment.

On the other hand, the general approach to ransomware detection that Manavi and Hamzeh [41] introduced includes the consideration of executable file headers through the LSTM network with PE headers. Analyzing byte sequences within headers, the model is able to clearly separate ransomware from benign files with 93.25% accuracy, which would allow for the program's rapid detection without the need to execute files. However, ransomware variants that contaminate header data may pose a threat to this approach.

Surendran et al. [42] presented an effective approach of a Dual/Two-Layered Android Malware Detection System. To overcome the interrelationships between static features, such as API calls and permissions, and dynamic features like system calls, they present a Tree Augmented Naive Bayes (TAN) model, and the detection accuracy is 97%. Nevertheless, the fact that the method targets specific features may also provide lower coverage against the new generation of malware utilizing other obfuscation techniques.

Moreover, Taher et al. [43] proposed a novel method of analyzing the Android programs for the identification and categorization of destructive programs, which overcomes the restrictions of past methods against highly intelligent obfuscation methods. Their framework includes three phases: normalization and rough preprocessing for feature extraction of static and dynamic data; feature selection using two methods; and a proposed model based on a neural network and improved Harris Hawk Optimization (HHO) algorithm.

The experiments reveal the effectiveness of the concept over isolated static and dynamic analysis. However, the method digitalizes the search process, which heavily relies on the quality of feature extraction and selection, and may encounter issues due to the rapid development of malware strategies.

Last but not least Dabas et al. [44] proposed a new effective malware detection for Windows based on API calls, features, and ML to curb new malware attacks. API call details are gathered in three features: use, frequency, and calls as sequences, resulting in multiple feature sets for the data set preprocessed using the TF-IDF method. Algorithms, despite researching and testing this integrated feature set, report better than 99.6% precision.

The study downsizes the feature set by 9% and bounds the accuracy from the upper and lower sides as 99.6% to 99.9%. However, use of feature selection may reduce the capability of detecting the malware because of these techniques that are employed by the malware.

In our research, the idea is to use the DL identification approach that integrates both the static and dynamic features of the malware while striving to retain the ability to work in real time. The model framework includes two complementary components: They tested two models that we have named a Static-CNN-LSTM model and a Dynamic-CNN 1D-LSTM model.

The four primary static models extract characteristics captured from the binary structure and code pattern of the malware, where the CNN and LSTM layers are sensitive and prove to give high accuracy in studying the static attributes. The Static-CNN-LSTM model that has been built in this paper was able to achieve an accuracy rate of 99%, which is indicative of high reliability in the identification of known and structured malware types.

On the other hand, the proposed Dynamic-CNN 1D-LSTM model targets runtime behavioral analysis, representing sequences of actions or events that are initiated by malware at runtime within a controlled environment. This model responds to the problem of accurately identifying and targeting insidious and sneaky malware that may adapt its actions in order not to be caught by certain detecting parts. Using CNN and 1D-LSTM layers together, the suggested dynamic model for telling malware apart based on how it acts is 100% accurate.

These models are complementary; that is, static analysis identifies known viruses, while dynamic analysis is able to catch new ones, especially those that are intended to avoid static analysis. The proposed hybrid model approach has several important benefits for the purpose of real-time malware detection. First, it enables pure and fast handling of the static and dynamic characteristics of the object. This will help the system to identify and classify threats in a highly loaded mode.

Second, the proposed system integrates CNN and LSTM; it allows the system to consider both spatial and temporal characteristics and is less sensitive to noise. Last but not least, the proposed approach's high level of detection accuracy reduces the dangers associated with false positives and false negatives, which are highly sensitive in fields like cybersecurity, where unnoticed threats can have disastrous consequences. The research endeavours to achieve the following objectives and contributions:

1) We created two new datasets, each containing 8513 malware samples, and 1000 non-malware or benign samples categorized into 30 different types. The first dataset comprises 64 by 64 grayscale images, which were converted, while the second dataset is in the format of a CSV file, which contains the sequences of API calls for each sample.
2) The detection of malware occurs in real time through two distinct methodologies: first, through the 1D-CNN, used for analyzing the malware images; second, let use long short term memory (LSTM) networks for analyzing the API call sequences that was extracted by forming the signature using the Python Pefile module for every sample.
3) Since we have 50 samples of each malicious and benign program, we only got results from the samples that the Python Pefile module analyzed as valid. We omitted samples that returned "none," indicating no extractable data, to accelerate the detection regime.
4) Importantly, most of the samples of malware in these two sets were obtained in the year 2024, which means that many samples will represent the most relevant threats in the modern world of malware.

2. METHODOLOGY

2.1 Data collection

in this paper, a dataset of 8,513 malicious Portable Executable (PE) files was acquired from VirusShare, a source commonly utilized by cybersecurity researchers for malware samples.

The virus was classified into 29 families utilizing classification labels supplied by Microsoft virus Protection. Furthermore, 1,000 benign executable (EXE) files were obtained from the studies [45-47], resulting in a balanced dataset including 30 categories, as seen in Table 1.

This study employed two separate datasets. In the first dataset, each PE file was transformed into a grayscale picture utilizing the methodology proposed by Nataraj et al. [48].

The conversion entailed converting binary malware data into an 8-bit vector, subsequently transferred to pixel values ranging from 0 to 255, with 0 denoting black, 255 denoting white, and intermediate values indicating various shades of gray. This grayscale depiction facilitated the visual examination of discrete binary structures inside the virus, as seen in Figure 1.

Table 1. Types of malware families

No.	Family	Type	Samples
1	Benign	Benign	1000
2	Ako	Ransomware	260
3	Autorun.NE	Virus	249
4	Banker.LY	TrojanSpy	260
5	Delf.DU	Backdoor	260
6	Drolnux.B	Worm	259
7	Eggnog.A	Worm	300
8	GandCrab.AE	Ransomware	220
9	Ganelp.E	Worm	260
10	Linkury.RS!MTB	Adware	244
11	Neconyd.A	Trojan	259
12	Nemucod	TrojanDownloader	260
13	Neojit.A	TrojanDownloader	300
14	OpenInstaller	PUA	260
15	Playtech	PUA	260
16	QQPass.GP	PWS	260
17	Qukart	TrojanSpy	260
18	Resur.A!epo	Virus	258
19	Shodi.A	Virus	220
20	Simda.D	PWS	159
21	Sivis.A	Virus	260
22	Small.M	TrojanSpy	260
23	Soltern!rfn	Worm	260
24	Trickbot.GML!	MTB Trojan	300
25	Unruy.F	TrojanDownloader	260
26	Upatre.A	TrojanDownloader	300
27	Urelas.AA	Trojan	260
28	Wabot.A	Backdoor	260
29	Yoof.E	Worm	289
30	Zombie!rfn	Trojan	256

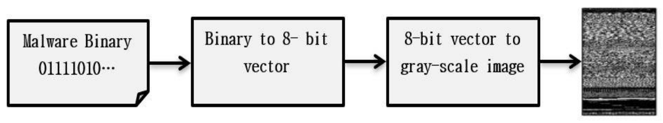

Figure 1. Process for transforming malware into image representations

For the second dataset, the first 50 API call sequences were removed from each file to reduce complexity and accelerate identification. The extraction procedure utilized the Pefile module, a Python program designed for handling PE files. To optimize the data, all "none" API requests were omitted. Every distinct API call in all samples was allocated a particular index, substituting the API call names with their respective index numbers.

The processed sequences were stored in CSV format, maintaining the critical structure for pattern detection and analysis while considerably decreasing the data's dimensionality.

2.2 Proposed methodology

The suggested malware detection technique utilizes two DL models, each specifically designed for distinct data types inside the malware detection framework, as seen in Figure 2.

A CNN was developed in the initial model to examine pictures produced from a dataset comprising both malicious and benign software samples. We converted the samples into the Portable Executable (PE) format, which encompasses 30 different malware categories. We standardized each image to 150×150 pixels and normalized it to ensure uniformity in feature scaling, preparing the picture data for CNN processing.

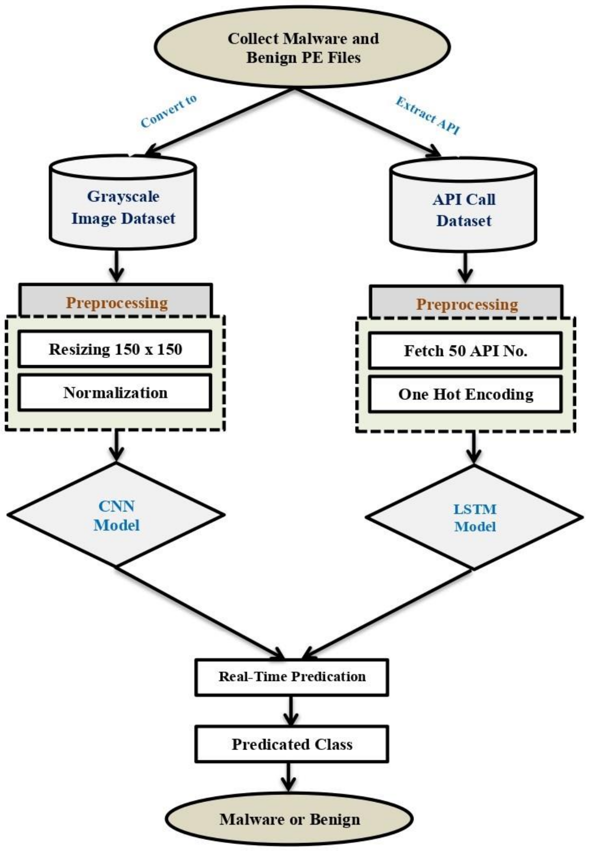

Figure 2. Two-model CNN-LSTM framework for improved detection

We divided the dataset into three subsets: 80% for training, 10% for testing, and 10% for validation, and further shuffled it to enhance the accuracy of the model. First, an embedding layer (is a way to convert high-dimensional input sequences into lower-dimensional vector representations) turns high-dimensional API sequences into 50-dimensional vector representations. Next, a SpatialDropout1D layer (is a dropout layer that randomly drops input features to prevent overfitting) with a 0.1 dropout rate stops the process from fitting too well. We employ a Conv1D layer with 32 filters and a kernel size of 2 to capture local patterns in the API sequence, using ReLU activation and'same' padding. MaxPooling1D, with a pool size of 5, succeeds in this by reducing the sequence length while maintaining essential characteristics.

An LSTM layer with 32 units further processes the sequence to capture long-term dependencies in the API call patterns. We employ a dense layer with a SoftMax activation function to categorize the sequence into one of 30 malware classifications. We configure the model using the Adam optimizer, categorical cross-entropy loss, and accuracy as the performance evaluation metric.

In second model employed LSTM networks on API call sequences extracted from the second dataset. We divided the dataset into 70% for training, 15% for testing, and 15% for validation. The second model employs a Time Distributed CNN-LSTM architecture to examine grayscale photos of malware specimens.

The architecture begins with a TimeDistributed (is a layer that applies operations across each time step in a sequence independently, preserving temporal relationships) Conv2D layer, equipped with 64 filters and a (2,2) kernel. This is followed by two TimeDistributed MaxPooling2D layers, which are designed to extract spatial information and reduce dimensionality. We deploy a further Conv2D layer with 128 filters and additional MaxPooling2D layers to enable deeper feature abstraction.

A TimeDistributed Flatten layer flattens the outputs of these layers, allowing an LSTM layer with 100 units to sequentially analyze and identify patterns across various regions of the malware picture. A concluding dense layer employing a SoftMax activation mechanism categorizes the malware picture into one of thirty classifications.

The model uses the Adam optimizer, sparse categorical cross-entropy loss, and accuracy as its main metrics to accurately group malware into different groups based on image data. Figure 3 explains the 1D-2D CNN-LSTM Architecture for Dual-Model Integration.

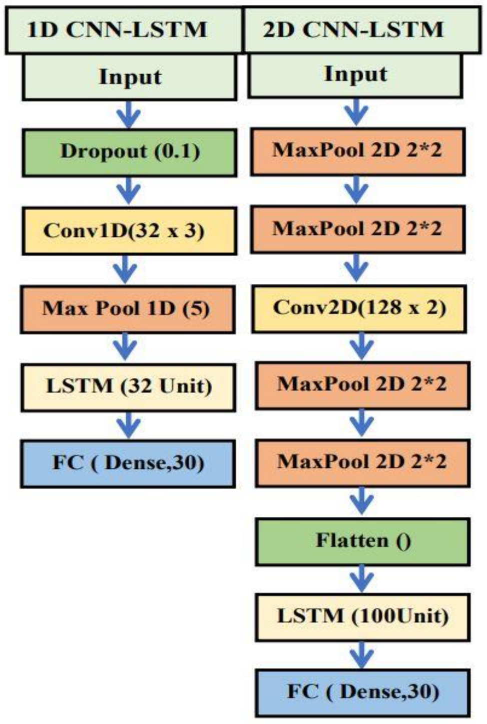

Figure 3. 1D-2D CNN-LSTM Model: A two-stage architecture

Additionally, the suggested method uses two separate datasets for the two models. One dataset is for grayscale images of malware, for which a model was created for static analysis, and the other is for the sequences of API calls for dynamic analysis. We split the datasets into training, testing, and validation sets separately to assess the stability of both models. Table 2 displays the summary of the sampling split for both models.

Table 2. Dataset splits for static and dynamic analysis models

Dataset Type	Total Samples	Training Split	Validation Split	Testing Split
Grayscale Images (PE)	8,513	80% 6,810 samples	10% 852 samples	10% 851 samples
API Call Sequences	8,513	70% 5,959 samples	15% 1,277 samples	15% 1,277 samples

The Rationale Behind of Dataset Splits for Grayscale Images (Static Analysis) and API Call Sequences (Dynamic Analysis), where within it the data was divided into 80% for training, 10% for testing, and 10% for validation. This split also keeps enough samples to test how well the model works while giving the CNN enough data to learn features from binary structures without overfitting. The 10% set aside for testing and validation ensures the model performs well on similar data most of the time.

So, for the API call sequence dataset, a 70:15:15 data split was used. This is because more evaluation, testing, and validation data should be used to test this model's real-world performance. The higher percentage of spending on testing and validation is also important for the dynamic model because API sequences change, and it is expected that more similarities and differences between types of malicious code will be found.

2.2.1 Rationale behind the chosen preprocessing methods

As mentioned before, the chosen preprocessing techniques that include converting PE images to grayscale and extracting API call sequences are intended to harmonize with each other and reflect both the static and dynamic attributes of malware.

Converting images to grayscale and using static analysis After the PE files are made, they are resampled to 150×150 grayscale images. This makes the feature space less complicated without getting rid of the structural information in the malware binaries. Since the images are grayscale, the model captures spatial features of malware organization that set the malicious samples apart from the benign ones. For instance, Deepa et al. [38] have explored the use of grayscale image representation in malware detection. This method gets rid of dependencies on preprocessing and a number of raw data features while keeping the most important ones so that VGG16 can learn from patterns that make sense.

2.2.2 API call sequence extraction for dynamic analysis

API call sequences capture the dynamic nature of malware and can reveal different activities of malware, such as file operations, network interactions, and resource misuse. The rationale behind selecting API calls stems from the research conducted by Zhou et al. [37] and Surendran et al. [42], which demonstrated the importance of API-based dynamic features in identifying obfuscated malware samples. Furthermore, considering the temporal characteristics of networks, the dependencies in the API sequences are captured in the long term, giving them the ability to identify new tactics used by malware.

2.2.3 Complementary roles of preprocessing techniques

Using both structural and behavioral properties in the preprocessing stage makes sure that data from both is taken into account. Static features, such as binary patterns, give information about known malware types, whereas dynamic features, for example, API sequences, identify new and constantly evolving malware types. This method uses both CNNs and LSTMs to make the best features of both even better, making it more reliable, accurate, and adaptable for real-time use. Thus, by incorporating these preprocessing methods, the given framework enables multi-featured malware analysis with consideration of various comprehensive characteristics in a relatively short time.

3. RESULTS AND DISCUSSION

All the experiments were conducted in the Python language and in an environment of the Jupyter Notebook, which allowed for the analysis and visualization of results and training of models in real-time. The first model trained was the dynamic 1D CNN-LSTM model, which was trained with health, malware, and benign image datasets.

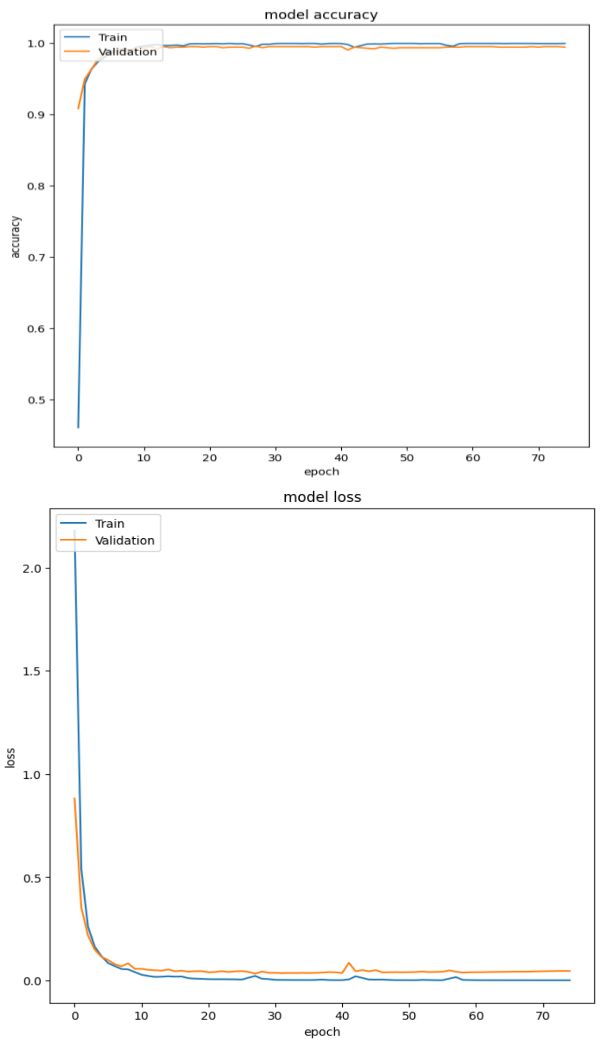

Figure 4. The accuracy and loss curves for the Dynamic-1D-CNN-LSTM malware detection model

We trained this model with a broad dataset, which involved converting image files from exe files. The CNN layers learned spatial properties, while the LSTM layer learned the dependency among sequences of API calls. We trained the model for a total of 70 epochs with a batch size of 32, achieving an accuracy of 100%. This result confirms that the chosen model can correctly classify all samples in the training and validation datasets and indicates the sample's quality and representativeness.

The model's ability to achieve the highest possible score may indicate its ability to distinguish between malicious and non-malicious software, but further experiments on larger datasets are necessary to confirm the model's stability.

Figure 4 displays the aforementioned diagrams, illustrating the model's accuracy and training curve during the training process. The model's accuracy looks very good, gradually increasing with each epoch to ensure effective learning and convergence.

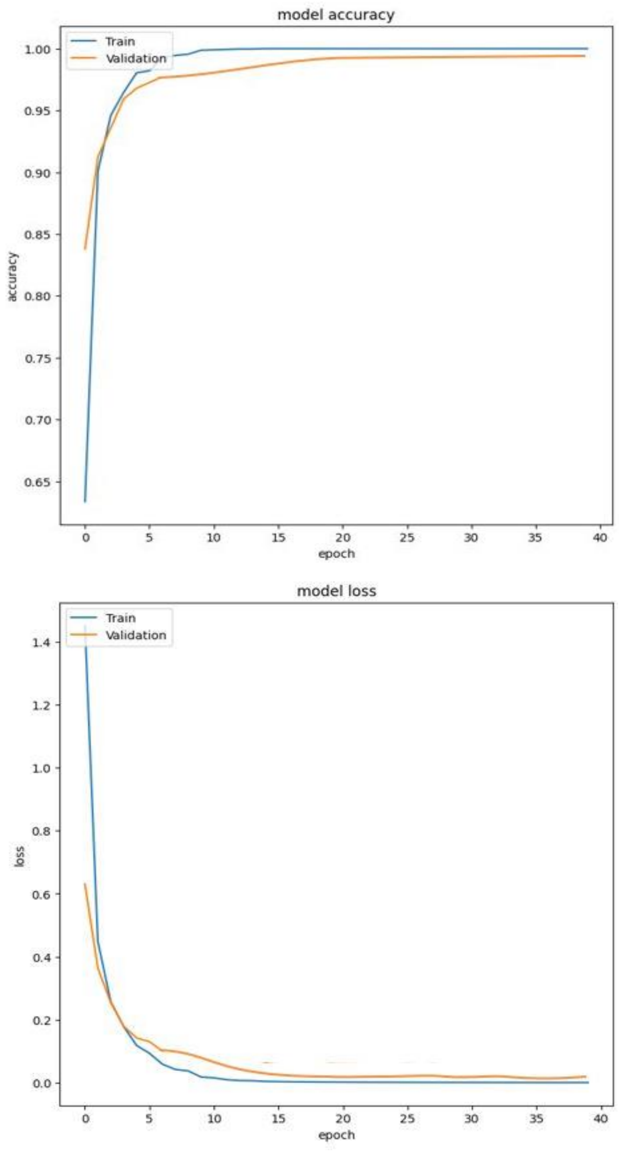

Figure 5. The accuracy and loss curves for the Static-2D-CNN-LSTM malware detection model

The second model used was a Static CNN-LSTM model, which like the previous model was trained with batch size of 32 but training was done only for 40 epochs. This model has shown 99% accuracy, thus outcompeting other malware detection models reported in the literature.

The high accuracy achieved by this model is evident to show its capability of identifying trends in the static features derived from the API call sequence. Evidently, Figure 5 shows that the accuracy of this model is fairly stable and reliable throughout the different epochs performed in the training process.

After training, both models saved to use in restoration of future samples that were not included in training set but belonged to four families of malware. This was important in determining the extent to which the models could be generalized. Given the results of the previous sections, the real-time testing indeed revealed that these models would be capable of classifying these unseen samples into either the benign or specific malware family.

The external test samples also prove how just fine the models are and that the proposed algorithms can be employed with actual cases where they would be useful in early identification of new and constantly arising malware threats.

Furthermore, the suggested method is very accurate-up to 100% with the dynamic CNN 1D-LSTM model-but there are some assumptions and limits that need to be thought about at the same time:

- Potential Overfitting: Because the dynamic CNN 1D-LSTM model is so accurate, it's possible that it's too effective at fitting the dataset used in this study. Overfitting refers to the model's inability to develop with high generality. The division of training, testing, and validation sets in this case results in a certain degree of cross-validation. However, the issue of data overfitting emerges when working with small, well-defined datasets. To reduce such a risk, regularization approaches like dropout and proper selection of the learning rate were used, but outside validation is needed.
- Impact of Dataset Size and Diversity: The datasets that were used in this study contain modern threats but have 8,513 samples and 30 categories of malware. The samples may be confined to a relatively small range of malware types, and their differentiation might not properly reflect a broad spectrum of possibilities that meet user experiences in practice. The study's strength lies in its extensive use of samples from various malware families, each with varying levels of obfuscation, across various contexts and regions. But a bigger set of data with a lot of different malware families, different obfuscation heuristics, and different settings or situations would have been a much better test for the model. Furthermore, data from which sample sequences were derived were collected in controlled settings only, and their performance in usual, uncontrolled, real-time conditions has not been tested yet.
- As well as the Assumptions Made During Model Development The preprocessing steps of casting PE files into grayscale images and using API call sequences rely on the premise that the image and sequence representations are rich enough to encode the features that set malware apart. As prior research backs these techniques, other possible preprocessing strategies may further boost the detection outcomes.
- Future Work: To address the limitations identified above, the following directions are proposed for future work:

- Testing on Larger and More Diverse Datasets: Adding an openly available or competitively sourced more extensive malware type dataset in conjunction with more generic malware type and/or obfuscation types to improve model robustness testing.
- Evaluating Generalization in Real-World Scenarios: The training of the models in actual scenarios to determine their performance when detecting approaches and methods deployed by adversaries.
- Exploring Lightweight Architectures: Identifying more efficient hybrid models' settings with low computational demand that will not lessen detection efficacy.

3.1 Comparative analysis with state-of-the-Art models

We conducted a comparison with other malware detection studies to contextualize the effectiveness of the proposed models. We compared the proposed models using performance metrics that include precision, recall, and F1-scores with other studies to provide a more comprehensive view of their performance. The results are shown in Table 3.

In terms of accuracy as well as F1-score, precision, and recall, the proposed hybrid approach outperforms the other recent models with remarkable differences. Consequently, through static and dynamic analysis, the models are capable of identifying traditional and continuously evolving malware with high accuracy and stability.

Table 3. Comparison our model of different models

Study	Models	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
[49]	CNN	97.48	98.71	96.22	97.45
[50]	LSTM	98.38	98.9	98.8	98.9
[51]	TextCNN and LSTM	99	99	98	99
[52]	CVAE-GAN	97.30	92.34	97.44	94.82
[53]	DNN	94.38	94.82	94.82	94.82
[54]	TL-GNN	98.87	99.55	97.30	99.42
Our Study	Static-CNN-LSTM	99	99	99	99
	Dynamic-CNN-1D LSTM	100	100	100	100

4. CONCLUSIONS

This paper introduces two novel datasets derived from the same samples but analyzed through different methodologies: one dataset consists of malware and benign images for static analysis classification, while the other encompasses API call sequences for dynamic malware classification. To facilitate the multi-classification of these samples, we employed two distinct DL models.

The first model, a dynamic 1D CNN-LSTM, utilized the dataset of malware and benign images, achieving an impressive accuracy of 100%. The second model, which implemented a static CNN-LSTM, leveraged the API call sequences from the second dataset and attained a high accuracy of 99%.

We retained both models after the training phases and used them for real-time malware detection and classification. The study's results show that the static CNN-LSTM model is more accurate than the other two. This is similar to how well the model works when using API call sequences to find malware.

Furthermore, our results show the benefits of using two models instead of one, which might be better than the other one. This ensures that if one model fails to identify malware, such as those using encryption or advanced masks, another model will complement it, thereby enhancing the overall detection capability.

Also, the comparison of malware images and the sequences of API calls obtained during the experiment also showed some correlation, which can be directly linked to the specific types of malwares and will help to systematize the knowledge of malware behaviors and to suggest improvements in the classification algorithms used in further studies.

REFERENCES

[1] Aslan, Ö.A., Samet, R. (2020). A comprehensive review on malware detection approaches. IEEE Access, 8: 6249-6271. https://doi.org/10.1109/ACCESS.2019.2963724

[2] SL, S.D., Jaidhar, C.D. (2019). Windows malware detector using convolutional neural network based on visualization images. IEEE Transactions on Emerging Topics in Computing, 9(2): 1057-1069. https://doi.org/10.1109/TETC.2019.2910086

[3] Chakkaravarthy, S.S., Sangeetha, D., Vaidehi, V. (2019). A survey on malware analysis and mitigation techniques. Computer Science Review, 32: 1-23. https://doi.org/10.1016/j.cosrev.2019.01.002

[4] Alsumaıdaee, Y.A. (2019). Automatic malware detection using data mining techniques based on power spectral density (PSD). Master's Thesis, Altınbaş Üniversitesi.

[5] Botacin, M., Ceschin, F., Sun, R., Oliveira, D., Grégio, A. (2021). Challenges and pitfalls in malware research. Computers & Security, 106: 102287. https://doi.org/10.1016/j.cose.2021.102287

[6] Akhtar, M.S., Feng, T. (2022). Malware analysis and detection using machine learning algorithms. Symmetry, 14(11): 2304. https://doi.org/10.3390/sym14112304

[7] Ngo, F.T., Agarwal, A., Govindu, R., MacDonald, C. (2020). Malicious software threats. The Palgrave Handbook of International Cybercrime and Cyberdeviance, Palgrave Macmillan, Cham, pp. 793-813. https://doi.org/10.1007/978-3-319-78440-3_35

[8] Kurnaz, S., Ahmed, Y. (2019). Automatic malware detection using data mining techniques based on Power Spectral Density (PSD). International Journal of Computer Science and Mobile Computing, 8(3): 27-30.

[9] Shaukat, K., Luo, S., Varadharajan, V. (2023). A novel deep learning-based approach for malware detection. Engineering Applications of Artificial Intelligence, 122: 106030. https://doi.org/10.1016/j.engappai.2023.106030

[10] Al-Tahee, M., Easa, H.K., Jabbar, K.A., Hussein, L.,

Alatba, S.R., Mohammed, M.A. (2024). Artificial intelligence assisted cyber-Physical systems with intelligent cyber security using deep learning. In 2024 International Conference on Smart Systems for Electrical, Electronics, Communication and Computer Engineering (ICSSEECC), Coimbatore, India, pp. 689-694. https://doi.org/10.1109/ICSSEECC61126.2024.10649428

[11] Aurangzeb, S., Aleem, M. (2023). Evaluation and classification of obfuscated Android malware through deep learning using ensemble voting mechanism. Scientific Reports, 13(1): 3093. https://doi.org/10.1038/s41598-023-30028-w

[12] Ahmed, S.R., Mohamed, S.J., Aljanabi, M.S., Algburi, S., Majeed, D.A., Kurdi, N.A., Al-Sarem, M., Tawfeq, J.F. (2024). A novel approach to malware detection using machine learning and image processing. In Proceedings of the Cognitive Models and Artificial Intelligence Conference, pp. 298-302. https://doi.org/10.1145/3660853.3660931

[13] Alsmadi, T., Alqudah, N. (2021). A survey on malware detection techniques. In 2021 International Conference on Information Technology (ICIT), Amman, Jordan, IEEE, pp. 371-376. https://doi.org/10.1109/ICIT52682.2021.9491765

[14] Gopinath, M., Sethuraman, S.C. (2023). A comprehensive survey on deep learning based malware detection techniques. Computer Science Review, 47: 100529. https://doi.org/10.1016/j.cosrev.2022.100529

[15] Faruk, M.J.H., Shahriar, H., Valero, M., Barsha, F.L., Sobhan, S., Khan, M.A., Whitman, M., Cuzzocrea, A., Lo, D., Rahman, A., Wu, F. (2021). Malware detection and prevention using artificial intelligence techniques. In 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, pp. 5369-537. https://doi.org/10.1109/BigData52589.2021.9671434

[16] Naseer, M., Rusdi, J.F., Shanono, N.M., Salam, S., Muslim, Z.B., Abu, N.A., Abadi, I. (2021). Malware detection: Issues and challenges. Journal of Physics: Conference Series, 1807(1): 012011. https://doi.org/10.1088/1742-6596/1807/1/012011

[17] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8: 1-74. https://doi.org/10.1186/s40537-021-00444-8

[18] Thakur, P., Kansal, V., Rishiwal, V. (2024). Hybrid deep learning approach based on LSTM and CNN for malware detection. Wireless Personal Communications, 136(3): 1879-1901. https://doi.org/10.1007/s11277-024-11366-y

[19] Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., Ghayvat, H. (2021). CNN variants for computer vision: History, architecture, application, challenges and future scope. Electronics, 10(20): 2470. https://doi.org/10.3390/electronics10202470

[20] Karat, G., Kannimoola, J.M., Nair, N., Vazhayil, A., Sujadevi, V.G., Poornachandran, P. (2024). CNN-LSTM hybrid model for enhanced malware analysis and detection. Procedia Computer Science, 233: 492-503. https://doi.org/10.1016/j.procs.2024.03.239

[21] Aboaoja, F.A., Zainal, A., Ghaleb, F.A., Al-Rimy, B.A.S., Eisa, T.A.E., Elnour, A.A.H. (2022). Malware detection issues, challenges, and future directions: A survey. Applied Sciences, 12(17): 8482. https://doi.org/10.3390/app12178482

[22] Ngo, Q.D., Nguyen, H.T., Le, V.H., Nguyen, D.H. (2020). A survey of IoT malware and detection methods based on static features. ICT Express, 6(4): 280-286. https://doi.org/10.1016/j.icte.2020.04.005

[23] Shhadat, I., Hayajneh, A., Al-Sharif, Z.A. (2020). The use of machine learning techniques to advance the detection and classification of unknown malware. Procedia Computer Science, 170: 917-922. https://doi.org/10.1016/j.procs.2020.03.110

[24] Al-Janabi, M., Altamimi, A.M. (2020). A comparative analysis of machine learning techniques for classification and detection of malware. In 2020 21st International Arab Conference on Information Technology (ACIT), Giza, Egypt, pp. 1-9. https://doi.org/10.1109/ACIT50332.2020.9300081

[25] Wadkar, M., Di Troia, F., Stamp, M. (2020). Detecting malware evolution using support vector machines. Expert Systems with Applications, 143: 113022. https://doi.org/10.1016/j.eswa.2019.113022

[26] Easa, H.K., Saber, A.A., Hamid, N.K., Saber, H.A. (2023). Machine learning based approach for detection of fake banknotes using support vector machine. Indonesian Journal of Electrical Engineering and Computer Science, 31(2): 1016-1022. https://doi.org/10.11591/ijeecs.v31.i2.pp1016-1022

[27] Wong, W.K., Juwono, F.H., Apriono, C. (2021). Vision-Based malware detection: A transfer learning approach using optimal ECOC-SVM configuration. IEEE Access, 9: 159262-159270. https://doi.org/10.1109/ACCESS.2021.3131713

[28] Babbar, H., Rani, S., Sah, D.K., AlQahtani, S.A., Kashif Bashir, A. (2023). Detection of android malware in the internet of things through the K-nearest neighbor algorithm. Sensors, 23(16): 7256. https://doi.org/10.3390/s23167256

[29] Al Saaidah, A., Abualhaj, M.M., Shambour, Q.Y., Abu-Shareha, A.A., Abualigah, L., Al-Khatib, S.N., Alraba'nah, Y.H. (2024). Enhancing malware detection performance: Leveraging K-Nearest neighbors with firefly optimization algorithm. Multimedia Tools and Applications, 1-24. https://doi.org/10.1007/s11042-024-18914-5

[30] Abu Al-Haija, Q., Odeh, A., Qattous, H. (2022). PDF malware detection based on optimizable decision trees. Electronics, 11(19): 3142. https://doi.org/10.3390/electronics11193142

[31] Mat, S.R.T., Ab Razak, M.F., Kahar, M.N.M., Arif, J.M., Firdaus, A. (2022). A Bayesian probability model for Android malware detection. ICT Express, 8(3): 424-431. https://doi.org/10.1016/j.icte.2021.09.003

[32] Yerima, S.Y., Alzaylaee, M.K., Shajan, A.P.V. (2021). Deep learning techniques for android botnet detection. Electronics, 10(4): 519. https://doi.org/10.3390/electronics10040519

[33] Amin, M., Tanveer, T.A., Tehseen, M., Khan, M., Khan, F.A., Anwar, S. (2020). Static malware detection and attribution in android byte-code through an end-to-end deep system. Future Generation Computer Systems, 102: 112-126. https://doi.org/10.1016/j.future.2019.07.070

[34] Bakhshinejad, N., Hamzeh, A. (2020). Parallel-CNN

network for malware detection. IET Information Security, 14(2): 210-219. https://doi.org/10.1049/iet-ifs.2019.0159

[35] Yeboah, P.N., Baz Musah, H.B. (2022). NLP technique for malware detection using 1D CNN fusion model. Security and Communication Networks, 2022(1): 2957203. https://doi.org/10.1155/2022/2957203

[36] Jha, S., Prashar, D., Long, H.V., Taniar, D. (2020). Recurrent neural network for detecting malware. Computers & Security, 99: 102037. https://doi.org/10.1016/j.cose.2020.102037

[37] Zhou, H., Yang, X., Pan, H., Guo, W. (2020). An android malware detection approach based on SIMGRU. IEEE Access, 8: 148404-148410. https://doi.org/10.1109/ACCESS.2020.3007571

[38] Deepa, K., Adithyakumar, K.S., Vinod, P. (2022). Malware image classification using vgg16. In 2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS), Kochi, India, pp. 1-6. https://doi.org/10.1109/IC3SIS54991.2022.9885587

[39] Salih, A., Zeebaree, S.T., Ameen, S., Alkhyyat, A., Shukur, H.M. (2021). A survey on the role of artificial intelligence, machine learning and deep learning for cybersecurity attack detection. In 2021 7th International Engineering Conference "Research & Innovation amid Global Pandemic"(IEC), Erbil, Iraq, pp. 61-66. https://doi.org/10.1109/IEC52205.2021.9476132

[40] Jeon, J., Jeong, B., Baek, S., Jeong, Y.S. (2021). Hybrid malware detection based on Bi-LSTM and SPP-Net for smart IoT. IEEE Transactions on Industrial Informatics, 18(7): 4830-4837. https://doi.org/10.1109/TII.2021.3119778

[41] Manavi, F., Hamzeh, A. (2021). Static detection of ransomware using LSTM network and PE header. In 2021 26th International Computer Conference, Computer Society of Iran (CSICC), Tehran, Iran, pp. 1-5. https://doi.org/10.1109/CSICC52343.2021.9420580

[42] Surendran, R., Thomas, T., Emmanuel, S. (2020). A TAN based hybrid model for android malware detection. Journal of Information Security and Applications, 54: 102483. https://doi.org/10.1016/j.jisa.2020.102483

[43] Taher, F., AlFandi, O., Al-kfairy, M., Al Hamadi, H., Alrabaee, S. (2023). DroidDetectMW: A hybrid intelligent model for android malware detection. Applied Sciences, 13(13): 7720. https://doi.org/10.3390/app13137720

[44] Dabas, N., Ahlawat, P., Sharma, P. (2023). An effective malware detection method using hybrid feature selection and machine learning algorithms. Arabian Journal for Science and Engineering, 48(8): 9749-9767. https://doi.org/10.1007/s13369-022-07309-z

[45] Al-Musawi, H.S.H. (2022). Hybrid malware detection and classification in real-time by deep learning techniques. Doctoral Dissertation.

[46] Bruzzese, R. (2024). Building visual malware dataset using virusshare data and comparing machine learning baseline model to CoAtNet for malware classification. In Proceedings of the 2024 16th International Conference on Machine Learning and Computing, pp. 185-193. https://doi.org/10.1145/3651671.3651735

[47] Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A., Wang, G. (2021). BODMAS: An open dataset for learning based temporal analysis of PE malware. In 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, pp. 78-84. https://doi.org/10.1109/SPW53761.2021.00020

[48] Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S. (2011). Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, pp. 1-7. https://doi.org/10.1145/2016904.2016908

[49] Zhang, S., Hu, C., Wang, L., Mihaljevic, M.J., Xu, S., Lan, T. (2023). A malware detection approach based on deep learning and memory forensics. Symmetry, 15(3): 758. https://doi.org/10.3390/sym15030758

[50] Alomari, E.S., Nuiaa, R.R., Alyasseri, Z.A.A., Mohammed, H.J., Sani, N.S., Esa, M.I., Musawi, B.A. (2023). Malware detection using deep learning and correlation-based feature selection. Symmetry, 15(1): 123. https://doi.org/10.3390/sym15010123

[51] Zhang, S., Gao, M., Wang, L., Xu, S., Shao, W., Kuang, R. (2025). A malware-Detection method using deep learning to fully extract API sequence features. Electronics, 14(1): 167. https://doi.org/10.3390/electronics14010167

[52] Huang, Y., Liu, J., Xiang, X., Wen, P., Wen, S., Chen, Y., Chen, L., Zhang, Y. (2024). Malware identification method in industrial control systems based on Opcode2vec and CVAE-GAN. Sensors, 24(17): 5518. https://doi.org/10.3390/s24175518

[53] Li, H., Xu, G., Wang, L., Xiao, X., Luo, X., Xu, G., Wang, H. (2024). MalCertain: Enhancing deep neural network based android malware detection by tackling prediction uncertainty. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pp. 1-13. https://doi.org/10.1145/3597503.3639122

[54] Raza, A., Qaisar, Z.H., Aslam, N., Faheem, M., Ashraf, M.W., Chaudhry, M.N. (2024). TL-GNN: Android malware detection using transfer learning. Applied AI Letters, 5(3): e94. https://doi.org/10.1002/ail2.94