A Hybrid Machine Learning Model for Malware Detection

BARTHOLOMEW IDOKO[1], FRANCISCA OGWUELEKA[2], STEVEN BASSEY[3], MONDAY ADENOMON[4]
[1,3,4]Centre for Cyber Space Studies, Nasarawa State University Keffi, NIGERIA.
[2]Computer Science Department, University of Abuja, Abuja-FCT, NIGERIA.

Abstract: The cyberspace is presently faced with many incidences related to malware compromising the cyber security goals of; confidentiality, integrity, availability, authenticity, non-repudiation and trust of system networks. Malware authors are breaking frontiers in developing new kind of malware such that anti-virus software cannot provide the level of protection that is being anticipated by users thereby resulting in unprecedented successful attacks being recorded. The study examine how existing machine learning models can be utilized in classifying malware infections and its propagation mechanism. This study adopts an experimental and quantitative research design using Machine Learning techniques to develop and evaluate a hybrid malware detection model. The approach includes the data collection, feature engineering, model development and performance evaluation. The hybrid model is designed using the combination of both malware (infected) files and benign (uninfected) files. The implementation of the system is done with the aid of deep learning structure (Inception v3) with SVM classifier by making use of the CNN learning method. The framework's input comes from instances of the explored malware dataset. These instances are supplied through the hybrid model (Inception v3) input layer. The instances are further forwarded to convolutional phase of Inception v3, where important features are extracted. The extracted features are utilized in form of support vector machine classifier's input. The incorporated support vector machine classifier uses the extracted features for classification where the output of the hybrid system is determined as either malware file or legitimate file. The hybrid model is made up of two different components; Inception v3 and SVM which combines to produce the hybrid model using the malware dataset to constructively detect malware. The final module, SVM, is used for classification, whereas Inception v3 is used for feature extraction. The simulation of all the tested models is carried out in order to analyze the data and answer the research questions. These models include; the developed hybrid model (Inception v3 + SVM), Convolutional Neural Network, CNN, Support Vector Machine, SVM, and Neural Network, NN. The aim is to deduce the best model for the malware dataset as well as to determine the best performing model. The accuracy rate of the hybrid system was obtained as 99.89%. The result shows significant convergence in both performance and learning. The result undoubtedly show how effective the proposed hybrid design is in comparison to other models developed to address the same issue.

Keywords: Malware, Detection, Model, Machine learning, Accuracy, Classification, Hybrid.

Received: March 7, 2025. Revised: May 11, 2025. Accepted: June 9, 2025. Published: July 30, 2025.

1. Introduction

The evolution of the initial term “virus” to today’s almighty “malware” reflects the evolution of threats over the past 20 years. The development and changes of such attacks have coincided with the development in the cyberspace. Therefore, every threat will target the vulnerable attack surface, hence the need to device new strategies to detect malware so as to prevent and mitigate its infection and spread. Malware can propagate through various mechanisms in order to deliver a specific payload. Due to the variation in malware propagation mechanism, malware can be classified based on the purposes and means they could interface with target computers (Yanfang, et al., 2017). A basic classification of malware includes file infectors (virus) and stand-alone malware (worms) (Idoko et al., 2025; Chandy, 2022). A different method of classifying malware is according to their specific actions and characteristics, examples; Trojans, Rootkits, Ransomware, Logic Bomb, Mobile code, Bot, Crypto- Malware, fileless-malware, Backdoors, etc (Chandy, 2022).

Malware exists in numerous forms hence a diversified dataset is needed in order to develop an effective malware detection system. Features must be collected from existing malware samples. The study of malware detection basically deals with analyzing executable files to identify behaviour that could be in the form of Indicators of Attack (IoA) and Indicators of Compromise (IoC) (Idoko & Bush, 2023). The emergence of anti-malware software has brought about the exponential increase in sophisticated malware with multiple polymorphic layers which are particularly designed to avoid detection by the anti-malware software (Damodaran et al., 2017). However, the practice has advanced the research in malware into the use of emerging technology for malware detection.

There are basically two approaches to malware analysis and detection namely, static and dynamic. Static Detection also referred to as code analysis or detection is a fast and simple detection method because the analysis and detection can be enabled with and without running the program. That is, static detection examines a sample without running it. This process uses reverse engineering principles which decompiles the malware and uses a number of tools to examine its source code (Sikorski & Honig, 2012; Chandy, 2022). Static detection has played a significant role as a preliminary detection technique throughout the history of malware investigation even though it is difficult to use this method to find complex and sophisticated malware. On the other hand, dynamic detection uses behaviour analysis while a malware is running to determine malicious intent (Jyothsna et al., 2011). Usually, this is done in a sandbox environment to ensure that the executable file does not cause any damage to the target computer. This detection method is capital intensive and difficult to manage in some instances. Programmers could be contacted to examine system calls or other behavioural trends which on the other hand, cannot be detected using black box testing (Jyothsna et al., 2011; Keragala, 2016).

Standard signature-based techniques for malware detection search for unusual activity using recognized digital indicators of malicious code. A breach can be detected using the lists of indicators of compromise (IoCs), which are mostly saved in a database (Rimon & Haque, 2023). Although IoCs are reactive in nature, they can be useful in detecting malicious activities. However, the need for the new detection methods arose from the exponential increase in polymorphic malware. To militate against this ugly trend, a heuristics-based approach in combination with machine learning techniques that provide more efficient detection accuracy is required (Gibert et al., 2020; Kumar et al., 2020). Researchers have over the years attempted to adopt Artificial Intelligence specifically Machine and Deep Learning Techniques as a better detective control mechanism since AI Algorithm has a major influence on the detection accuracy of the static malware detection tools (Venkatraman et al., 2019; Akbar & Ahmad, 2021). Most scholars have proposed ML Algorithm such as One-class classification, Support Vector Machine, Multi-class Support Vector Machine, Random Forest, Restricted Boltzmann Machine among others to achieve a near accurate results, their exists some gaps like false positive rate, poor classification accuracies, and low true positive rate. A Hybrid ML model could be used as a framework to integrate different ML techniques so as to better improve the detection and classification accuracy (Rimon & Haque, 2023).

This study proposed a hybrid machine learning algorithm referred to as Inceptron + v3 and considers CNN, SVM and NN as the base models which serve as the foundation through which independent predictions are carried out. They are generally referred to as the building blocks for the hybrid machine learning algorithm (Inceptron + v3) (Bush et al, 2023). CNN is one of the base models where the dataset is the input (hidden) layer of the CNN model, while the model's predictions are the output layer. One or more layers that carry out convolutions make up the hidden layers. This typically consists of a layer that uses the input matrix of the layer to do a dot product of the convolution kernel. It consists of four major layers which are; the convolutional, pooling, ReLU (Dense layer), and the fully connected (output) layers (Bush et al., 2018). These layers enable CNN to mimic how the human brain processes images in order to identify patterns and characteristics. Whereas, Figure 3.6, represents the architecture of CNN as applied to malware detection. The primary goal of the SVM model is to identify a hyperplane that divides the data into malicious and normal activities. The support vectors serve to define the decision boundaries and are the vectors closest to the hyperplane (Bush et al., 2023). The SVM transforms the input space into a higher dimensional space by using kernel methods. In order to enhance the margin between the classes, the SVM algorithm is taught to alter the hyperplane's position. The purpose of the NN function, f, is to minimize the discrepancies between its output and the y label of a given input x by using a set of labeled data. This cost function is frequently referred to as a loss function and an optimization procedure. To do this, the model is trained by adjusting the parameters, Q of f to be the best approximation function provided by:

$$\hat{Y} = f(x, \Theta) \qquad (1)$$

The standard optimization procedure for model training is called the backpropagation algorithm.

Inception v3 and SVM are two unique structures that are combined in the hybrid system to create a hybrid model that detects malware files in a constructive manner. The system is designed using the malware dataset, with Inception v3 being used for feature extraction and SVM, the final module, being used for classification.

2. Related Works

Malware detection in a critical environment is a difficult operation that requires advanced analysis and techniques for both identification and classification. To avoid detection and disrupt the analysis, malware developers use a variety of anti-malware detection techniques. Majority of malware analysts use Static and Dynamic analysis and techniques to analyze malwares similar to the proposed hybrid machine learning model that will integrate both the static and dynamic approach. However, the review of related studies has shown that there are advantages and disadvantages of using these analysis and detection techniques.

A proposal was made to use a machine learning strategy on static features to classify Windows PE files completely in a pipeline for the first time (Loi et al., 2021). The pipeline can distinguish between harmful and benign samples, and for those that are deemed dangerous, it offers a comprehensive classification based on behavior, malware family, and threat category. Despite known limitations such as the size of the training data, the imperfect labeling of the ground truth, and the semantic gap of models based on static features only, classification results were comparable to the state of the art for similar works while providing much more detailed information on malware characteristics. Finally, the extracted feature vector defining the raw PE and the specific ML model used generated an interpretable result, and the method is scalable to much larger datasets. As a result, the authors saw the work as a preliminary step towards a practical solution that would assist security analysts in managing new threats while cutting down on analysis time and expenses. In order to enhance the model's performance, the authors ought to have adjusted the ground truth, taken into account a larger dataset for training, explored the properties of the embedded features' space more, and thought about the possibility of a multi-label classification scheme. In the early phases of the pipeline, it could be intriguing to incorporate a detector for packed or encrypted samples as well.

Another author thought about creating ScaleMalNet, a highly scalable system for identifying, categorizing, and classifying zero-day malware. This study also evaluated deep learning architectures and conventional machine learning algorithms (MLAs) based on image processing, dynamic analysis, and static analysis methods (Vinayakumar et al., 2019). Initially, malware was categorized using a combination of static and dynamic analysis. In the second stage, malware were grouped together using image processing algorithms. In malware detection and classification using static, dynamic, and image processing, deep learning architectures performed better than standard MLAs. However, deep learning architectures are used on the domain knowledge extracted features in the malware detection study that is based on dynamic analysis. By gathering binary file memory dumps during runtime and mapping the memory dump file into a grayscale image, this can be prevented.

An internationally recognised machine learning model that can analyse events in a manner similar to that of human analysts was created by a group of US-based machine learning specialists as part of an ML-based malware detection system (Johnson & Grumbling, 2019). The model can identify if a specific instance of a given user accessing a given document is harmful or acceptable once it has been trained and applied to document-access events. The model however, receives two types of input; data about the user and data about the document accessed (Johnson & Grumbling, 2019). This standardized model can determine if a particular instance is benign or harmful given the identity of the user and the specific document after being trained on data from around the organization (Zheng, et. al. 2022). The group however, fed their model with the identical auxiliary processed data.

However, by using this method, Google, for instance, was able to identify instances of disgruntled employees looking for corporate documents for a rival company, as well as early reconnaissance actions of a red-team exercise. Nevertheless, there have also been some false positives from the model, such as when an employee was switching jobs or using outdated documents. And, so far, so good, the system has added new perspective to research and innovation. With extremely low false-positive rates, it has been able to detect real unique malware in the two years of its implementation by Google. By combining innovative approaches for handling the training data with established supervised learning techniques, this anomaly detection methodology proved effective (Johnson & Grumbling, 2019).

Previous study proposed a system for extracting selected features from the static and dynamic analysis techniques (Singh & Jain, 2017). An integrated method was developed employing the selected features in order to increase the classification and detection rate as compared to the use of simple static and dynamic approaches. To improve results of classification and detection, the researchers examined malwares that have anti-malware analysis features. The outcome demonstrates an accuracy of 63.30% for dynamic analysis, 69.72% for static analysis, and 73.47% for the integrated technique. The integrated technique yields better accuracy when compared to the static and dynamic approaches (Singh & Jain, 2017).

A proposed study that makes use of Application Package Interface (API) calls for malware detection and analysis was carried out (Salehi et al., 2021). According to the study, malware with comparable behaviors will make calls to the same set of parameters and APIs. The strategy was to use a dynamic analysis technique to extract feature vectors. Several different feature selection algorithms can be used to decrease the amount of features. The authors intercepted API calls using a VMware-based virtual machine and the WINAPIoverride32 program. The authors employed WEKA classifiers for classification.

The idea of analyzing malware utilizing the headers and import file sections of the Windows Portable Executable (WPE) file format was put up by another group of authors (Markel & Bilzor, 2020). The primary objective of the authors' work was that malicious executable files had different metadata than clean executables. After analyzing a number of PE32 header traits, only those that were best suited for categorization were chosen.

In a similar vein, a study on the use of ML for the detection of new dangerous executable files was conducted (Schultz et al., 2018). Using the static analysis approach, the authors primarily collected three static features: text information, byte-sequence n-grams, and portable executables (PE). The characteristics were taken from dynamic link libraries (dlls) inside the 32-bit executable. The n-gram technique, which extracts a string's n-byte sequence, was utilized to increase the detection rate. All of the text strings that were encoded in the executables were available in the string information. When the authors used machine learning instead of more conventional signature-based methods, they discovered that their detection rates were significantly higher. Nonetheless, a team of researchers applied the idea of feature extraction based on a quantity of bytes found in the malicious executable's source code (Tian et al., 2018). By employing this technique, they were able to extract several functions from malicious executables and use dynamic analysis to determine these functions' frequency of occurrence in order to detect the infected file. The executable codes for obfuscated files were concealed. Machine learning techniques from WEKA were used for classification.

To provide better categorization results, another set of researchers explored the use of the n-grams technique (Kotler & Maloof, 2020). To improve the detection rate, the authors worked with a variety of classification methods and a decision tree-based methodology. A related viewpoint was examined, in which the authors used dynamic analysis to extract characteristics by merging temporal and geographical data from run-time windows Application Programming Interface (API) (Ahmed et al., 2023). The authors claimed that examining temporal and spatial characteristics simultaneously can increase the likelihood of detecting malware. They went further to integrate the static and dynamic analysis of malware. The function length frequency and printed string information (PSI) vectors were obtained by static analysis. The log files of dynamic analysis tools such as HOOKAPI were used to extract dynamic feature vectors, which were made up of API functions and their arguments. The authors combined the feature vectors obtained from both static and dynamic analysis to construct the feature vector for the integrated approach. The findings indicate that when employing an integrated method, accuracy is higher than when utilizing a static and dynamic approach.

Finally, another study focused on the extraction of dynamic features from the CSV file generated from the cuckoo sandbox (Dhammi & Singh., 2019). Rather than concentrating solely on API calls, the authors also examined file information, registry modifications, and network analysis. Even though most authors made an effort to increase malware detection rates and classification, there is still room for improvement in analysis and detection methods, which has led to certain research gaps.

3. The Existing System and Knowledge Gap

Although, substantial body of literatures have addressed various facets of malware detection and analysis, however, there exist a noticeable gap in comprehensive studies that systematically compared the performance of the different machine learning model for malware detection. Attackers are able to continually breaching the system's defenses in a way that is unavoidable by taking advantage of an inherent asymmetry on these gaps (Johnson & Grumbling, 2019).

Traditional machine learning models could yield a high number of false positives because they don't witness many real attacks, which means there aren't enough samples to be used for training of an ML model. Defenders frequently place a high priority on lowering the number of false positives because they might be expensive. Nevertheless, this often has the effect of increasing the rate of false negatives, which implicitly raises the possibility of a successful attack (Johnson & Grumbling, 2019). The hybrid machine learning model that combines two different models is capable of constructively detecting malware with miss-classification rate kept to a minimum.

Security tools are frequently used by parties other than the resource owners themselves and operate on a number of interdependent system therefore, maintaining a low number of false positives without providing opportunities for attackers is typically a substantial issue (Johnson & Grumbling, 2019). However, the hybrid machine learning model make use of a LinearSVC or interpolator with a smooth/linear structure that cannot be easily adjusted.

Most Machine Learning tools recently designed for malware classification are widely based on accuracy, this simply means that the system is probabilistic in decision making. This is because ML tools for malware detection are not upgraded to meet the demands of either defensive or offensive cyber operations (Zheng, et. al., 2022). It is worthy to note that the parameters of hybrid ML systems for malware detection unlike the conventional ML tools can be upgrade and optimized to perform excellently without bias.

The glaring void inhibited as a result of limited access to malware dataset as well as obtaining more static and dynamic features for higher accuracy and detection (Singh & Jain, 2017). The inability to also access dataset for advanced malware samples such as metamorphic and polymorphic malwares whose detection and classification might be faster as compared to the ones that have been widely used by researchers (Singh & Jain, 2017). This glaring void and challenge has been tackled with the access to a large updated benign and malware (metamorphic and polymorphic) dataset comprises of 138,047 instances.

4. Methodology

This study adopts an experimental and quantitative research design using Machine Learning techniques to develop and evaluate a hybrid malware detection model. The approach includes the data collection, feature engineering, model development and performance evaluation. Figure 1 represents the machine learning development and classification scheme.

A hybrid learning model is developed with the potential to addressing the malware detection problems**.** The ML models implemented are; Inception + v3, CNN, SVM and NN. The final module, SVM, is used for classification, whereas Inception v3 is used for feature extraction. The general process is outlined in Figure 1.

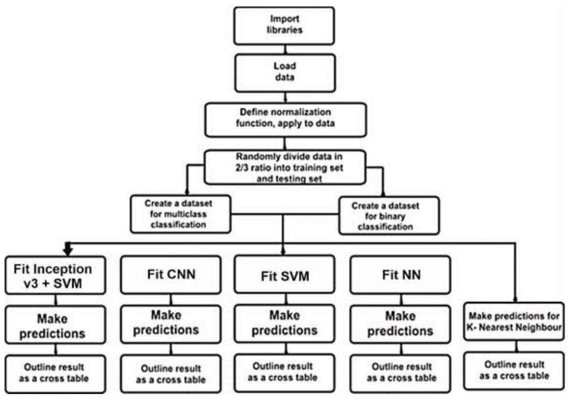

Import libraries
Load data
Define normalization function, apply to data
Randomly divide data in 2/3 ratio into training set and testing set
Create a dataset for multiclass classification
Create a dataset for binary classification
Fit Inception v3 + SVM
Fit CNN
Fit SVM
Fit NN
Make predictions
Make predictions
Make predictions
Make predictions
Make predictions for K- Nearest Neighbour
Outline result as a cross table
Outline result as a cross table
Outline result as a cross table
Outline result as a cross table
Outline result as a cross table

Figure 1: The Machine learning development and classification scheme

The packages that is used for the implementation of the algorithms are:

i. Support Vector Machines – Scikit-learn
ii. CNN – Keras
iii. Cross Table plotting – gmodels

The study considers the detection of the latest malicious code whose sample is obtained from a secondary data source (Virushare Machine Learning Repository) which provides the fundamental information that forms the basis of the investigation. The data is composed of the collection of multivariate attributes that characterizes instances of malware and benign data points that make up the dataset. The total sample of both benign and malware classes used for the experiment is 138,047 which comprises of 41,323 benign and 96,724 malware files. Furthermore, the dataset consist of 56 features used for training and generalization. An instance in the dataset is either classified into legitimate or malware file. The constructed hybrid model classifies these features into one of the two classes; benign = l or malware = m.

4.1 Data processing

The data is normalized before utilizing it. This is achieved by separating the data into training and testing sets for the training process, and splitting the data to be within a smaller range because the values of many attributes fall within a wide range of numbers. The normalization stages include, feature extraction, feature engineering and labelling.

4.1.1 Feature Extraction

The combining matrix, which comprises successful Application Programming Interfaces (APIs), unsuccessful APIs, and their return codes, is the feature extraction technique used in this study. The reports are analyzed and stored locally during the implementation phase. The feature extraction software then uses these reports as input to create the .csv file containing the combining matrix. The methodology specifies the minimum number of API calls; for example, all reports that generated fewer than five API calls is ignored. The extraction timestamp is included in the file, and logs detailing successful and unsuccessful operations is kept in a different file.

4.1.2 Feature Engineering

This is one of the crucial task of this study. The characteristics of the malicious codes that distinguishes them are called features. Since using every feature makes the model more complex for malware detection, it won't be necessary. However, the data is transformed by changing the original numerical representation of a quantitative value to another value, during which the nature, direction, and significance of the relationships of the variables used in the study were determined. Consequently, the characteristics of each feature are identified in order to choose more important features for the model's training.

In order to increase the detection accuracy, the feature engineering procedure which involved selecting, manipulating and transforming relevant features to eliminate features that are unnecessary or redundant is carried out. Because of the great size of the feature set in this study, feature selection is crucial. Python is applied in this case for the feature selection with the aid of Boruta package which provides a good and straightforward approach for feature selection in classification tasks. It is a wrapper technique that operates over ML algorithms (Kursa & Rudnicki 2020).

The algorithm is as follows:

i. Make scrambled (shadow) duplicates of every feature to increase the randomization.
ii. Using the new dataset, train an ML classifier and use the Mean Decrease Accuracy technique as a feature importance metric. Measure the importance of each feature and assign weights.
iii. Verify that the feature from the original feature set has a greater weight than the highest weight of this feature's shadow copy at the end of each iteration. At each iteration, eliminate the features that are deemed unnecessary.
iv. Stop whenever all features have been categorized as "selected" or "rejected," or when a predetermined number of ML iterations have been completed.

4.1.3 Labelling

The model is set up with hyper-parameters and specific kernel functions for malware detection. These setups are intended to capture the complexities of malware attack situations and are inspired by both exploratory experimentation and current research. The target value that the hybrid model can predict is either benign (1) or malware (m). The data is put through a unique machine learning process in four machine learning algorithms, namely; the hybrid model (Inception v3 + SVM), CNN, SVM and Neural Network (NN).

4.2 Techniques for Data Analysis

The models' performance is assessed by determining the model's reliability and qualities since the most accurate models perform well in real-time malware attacks. The behavioral characteristics of existing malware is assessed so as to identify the indices for developing the most efficient model for applied malware detection. Indicators such as obfuscated program, shelling program, overall time derived from existing and standard model serves as a guide for the analysis. The hybrid learning model is also assessed and compared with other models.

The results obtained from the four (4) models undergo a rigorous comparative analysis from the perspective of static, dynamic and hybrid using strategic evaluation methods and visualization. This analysis aimed at discerning the advantages and disadvantages of every model, offering subtle insights that help to improve and maximize malware detection techniques and frameworks.

Table 1: Strategies for the Evaluation of the Models

Features	Yardstick for Comparison
Strengths and limitations	Extensive analysis and visuals
Performance metrics	Using tables or graphs for comparison
Malware detection framework	Refine, integrate and optimize base on the comparative analysis

Visualizations like bar plots and radar plots are used to analyze the models' performance matrices. These visuals help to improve the results' interpretability and make it easier to see how well each model performs in comparison.

Table 2: Matrix comparison and Visuals

Visualization	Description
Bar Plots	Display each model's performance on a different matrix.
Radar Plots	Show the relative strength of each model

5. Implementation of the Proposed Model

In the real world, many malware strains make use of various flaws that may exist in particular software packages. As a result, the research includes a wide variety of services in the virtual systems that were created for stimulation.

The virtual machine is created by using VMcloak, an automated virtual machine generation and cloaking tool. The specifications for the virtual machine deployed for the experiment includes:

i. 1 CPU-CUDA core 1733 MHz
ii. 8 GB RAM
iii. Internet access
iv. NVIDIA GeForce GTX, 256-bit memory interface, supports NVIDIA Ansel, SLI
v. Adobe PDF reader 9.0
vi. Adobe Flashplayer 11.7.700.169
vii. Visual Studio redistributable packages 2005 - 2013.
viii. Java JRE 7
ix. .NET framework 4.0

The model utilizes deep learning structure (Inception v3) plus SVM classifier for its implementation by referencing the CNN algorithm. The framework's input comes from instances of the explored malware dataset. Here, these instances are fed into the hybrid model via the Inception v3 input layer. The instances are further forwarded to convolutional phase of Inception v3, where important features are extracted. The extracted features are utilized in form of support vector machine classifier's input. This incorporated support vector machine classifier uses the extracted features for classification. The output of the hybrid system is either malware file or legitimate file. The hybrid model comprises of Inception v3 and SVM which are the two unique structures that interact to form the hybrid model using the malware dataset to constructively detect malware files. The final module, SVM, is used for classification, whereas Inception v3 is used for feature extraction.
The proposed model is implemented in four different stages, namely; training and testing, data split, cross validation and matrices evaluation.

5.1 Model Training and Testing

The Model undergo training and testing so that the machine learning algorithm learns to recognize patterns and make predictions based on input data. It involves feeding the algorithm with the extracted malware dataset consisting of input-output pairs, where the algorithm learns to map inputs to corresponding outputs through iterative adjustments of the internal parameters. During training, the algorithm compares its predictions with the actual outputs, calculates the error, and updates its parameters using optimization techniques like gradient descent to minimize this error. This iterative process continues until the model's performance on a separate validation dataset reaches satisfactory levels, indicating that it has learned to generalize well to new, unseen data. The trained model can then be deployed to make predictions on new data it hasn't seen during training. The models are saved for later use after testing and exported into a file. The models can also be used to develop an antimalware program and to also conduct additional research.

The split sample in the training and testing is decompiled for the static part using HOOKAPI, then extract Permission as a static feature from the Window Portable Executable (WPE) file Manifest. The approach is implemented in a virtual environment to simulate computer network devices in the dynamic section. Next, is the extraction of API (Application Package Interface) as dynamic features from dynamic log files which is installed to run the PE samples. The task of determining if the sample is harmful or not is carried out by the detection as illustrated in Figure 2.

5.2 Dataset Split

The original dataset is divided using ten-fold validation techniques after which the

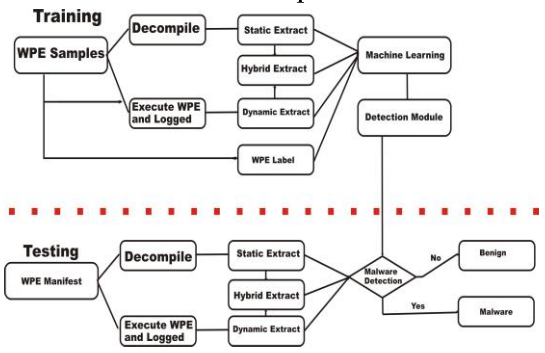


Figure 2: Model Training and Testing.

largest portion of the dataset is used to train the model, while the smaller portion is used for testing. The sample under investigation is divided into ten (10) equal parts. 9 of the 10 sections are utilized in the training phase, while the remaining portion is used in the test phase. By flipping the training and testing signals ten times, the training process is repeated. Additionally, the LinearSVC learning method is used to implement training with 150 training epochs.

5.3 Cross-Validation

Cross table charting of the four models is done in order to forecast how well the models perform on the new data. Cross-validation is employed in the design process of the hybrid model. This strategy aids in overcoming the limitation of the machine learning algorithms' accuracy evaluation techniques.

5.4 Metrics Evaluation

The variables for classification and detection such as Accuracy, Precision, Recall, F1- Score, TPR, FPR, F-measure and AUC-ROC is giving maximum attention to ensure their relevance to the research questions is met.

First, the actual target values is compared with the projected values, a confusion matrix which is a method for assessing the model's classification performance in machine learning is implemented to calculate the error and present a comprehensive picture of the model's performance.

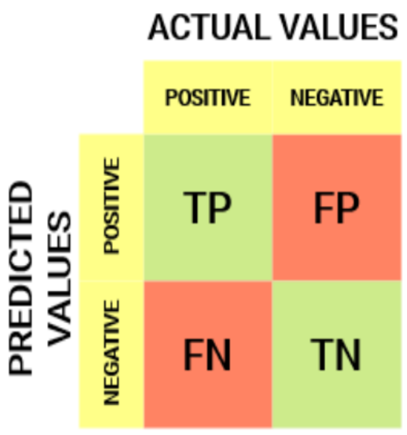


Figure 3: Confusion Matrix. (Bush & Abiyev, 2023).

There are two values in the target variables: positive and negative;

Where the True Positive = TP,

The True Negative = TN,

False Positive = FP,

False Negative = FN.

The values of the target variables ranges between 0 and 1 and are displayed within the python Google Collaboration environment during the prediction. Secondly, the total number of correctly classified samples is divided by the total number of samples in the test set, the accuracy is calculated. However, in the event that the dataset is unbalanced, accuracy is limited. The accuracy obtained makes it impossible to evaluate the models if any class or dataset output has a disproportionately high or low number of samples compared to the other classes (Moustafa, et al., 2017). The accuracy is calculated thus:

$$\text{Accuracy} = \frac{TP + TN}{FP + FN + TP + TN} \quad (2)$$

The receiver operating characteristics area under curve (ROC AUC) is one of the most widely used metrics, particularly for two-class imbalanced data. The trade-off between true positive and false positive rates is evaluated across various classification thresholds using the AUC-ROC metrics.

The F1-score is then applied as shown in equation 3.

$$\text{F1 score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (3)$$

Again, Recall (sensitivity), specificity, and accuracy were applied to gauge the models' unique capacities for identifying distinct output classes using the equations;

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (5)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (6)$$

In order to solve issues with imbalanced datasets and differing levels of model complexity, the analysis takes into account the context of malware detection. A transfer learning of the techniques with optimum performance is conducted. This entails integrating two of the best models in terms of performance to produce a hybrid model with a better performance. The results is compared with standard models and those of other researchers so as to recommend the best model for malware detection and classification.

6. Results and Analysis

In this section, we present the result of the study from simulation of all the tested models in order to analyze the data and answer the research questions. These models include; the developed hybrid model (Inception v3 + SVM), CNN, SVM, and NN. The aim is to deduce the best model for the malware dataset as well as to determine the best performing model.

6.1 Simulation Using Inception v3 + SVM (Hybrid Model)

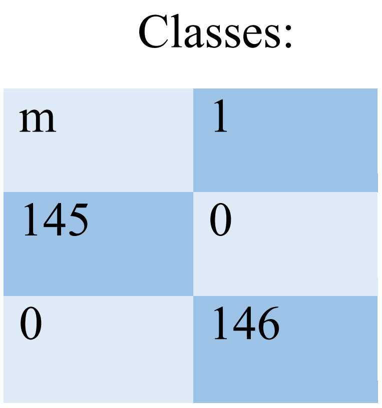
Four parameters define the Inception module: depth (D), height (H), width (W), and output class number. The input size is denoted by H and W. Input channels are represented by depth. W is 299 and H is 299 in the input size of 299x299x3. Lastly, D is 3, which is the RGB standard. As stated earlier, we performed factorization operations on this high dimensional input space, resulting in a significant reduction in dimension. The factorized low space is then used as the input for the support vector machine classifier. A classification report is shown in Table 3, with the corresponding confusion matrix of the hybrid model used to detect malware files as shown in Figure 4. These findings were from an experiment that used the cross-validation approach. Out of the two classes in the investigated dataset, miss-classification only happened once on class m (malware file), as seen in table 3. It is evident from these data that the hybrid model is efficient because the miss-classification rate is kept to a minimum.

Table 3: Classification report

Classes	Precision	Recall	F1-score	Support
m	1.00	0.99	1.00	146
1	1.00	1.00	1.00	146

The SVM class used in this investigation is the LinearSVC estimator. LinearSVC is essentially a linear interpolation. It has a smooth/simplified learning structure, and is much less adjustable. For categorization, it is connected with Inception v3. One of the parameters of the SVM classifier is loss='Squared_Hinge,' which stands for hinge loss square.

Penalty = "l2" specifies the penalization norm, C = 5 denotes the error term parameter C, and "ovr" = Multi_Class which determines the multiclass approach when y has more than two classes.



Classes:

m	1
145	0
0	146

Figure 4: Confusion matrix for hybrid ML model

Cross-validation is used to carry out the design process on the hybrid model that was presented in the initial simulation. The ten-fold cross-validation is used in this experiment. The sample under investigation has been divided into ten (10) equal parts. 9 of the 10 sections are utilized in the training phase, while the remaining portion is used in the testing phase. By flipping the training and testing signals ten times, the training process is repeated. Additionally, the LinearSVC learning method is used to implement training with 150 training epochs. The accuracy value that has been demonstrated is the average of 10 simulations. The average accuracy rate during the test phase was 99.89%, with an error of 0.0135.

Monte Carlo-style estimation is investigated in the second simulation using the same database, where the hybrid framework stops at 500 epochs. The dataset is randomly divided into 60% for training and 40% for testing at each epoch. Monte Carlo estimators are a broad category of techniques that rely on random sampling iterations to generate numerical results (Kroese et al., 2024). Using randomization to solve predictable tasks is the core notion. They are frequently used in physical and mathematical tasks. It is also useful in situations where using another methodology is impossible or very challenging. The Monte-Carlo estimator is mostly used to solve three types of problems: optimization, numerical integration, and producing solutions using a probability distribution. Fundamentally, every problem with a probabilistic interpretation can be solved using Monte Carlo estimators. As was previously indicated, Monte Carlo experiments were conducted using a malware database, and the hybrid model achieved an accuracy of 98.97% and an error rate of 0.0214. The results of simulations of the hybrid system using both cross-validation and Monte Carlo techniques are shown in Table 4.

Table 4: Results of the hybrid model's simulation

Methods	Accuracy (%)	RMSE
Monte Carlo Techniques	98.97	0.0214
Cross-Validation Techniques	99.89	0.0135

6.2 Convolutional Neural Network (CNN) Simulation

In order to detect malware, the CNN algorithm with a fully connected network is utilised at the initial stage. The CNN structure used for malware detection is shown in Table 5. It consists of 2 convolutional layers, max-pooling, and fully-connected layers.

Table 5: CNN Structure

Layer type	Output Shape	Param #
Conv2d_1 (Conv2D)	(None, 26,26,16)	160
Max_pooling2d_1	(None, 11,11,32)	0
Conv2d_2 (Conv2D)	(None, 11,11,32)	4640
Max_pooling2d_2	(None, 5,5,32)	0
Conv2d_3 (Conv2D)	(None, 3,3,64)	18496
Max_pooling2d_3	(None, 1,1,64)	0
Flatten_1 (Flatten)	(None, 64)	0
Dense_1 (Dense)	(None, 768)	49920
Dense_2 (Dense)	(None, 128)	98432
Dense_3 (Dense)	(None, 24)	3096

The data is split into two halves for CNN training: 80% and 20%. Twenty percent was used for testing and eighty percent for training. 60% and 40% of the 80% of data allocated for training are used for training and validation, respectively.

Equations (3–5) were employed to ascertain the CNN's output signals. Z-score normalization was used to scale each input signal during simulation, which improved the model's generalizability. The training is done using an RMSprop learning method. Furthermore, 150 epochs was used to train the CNN model. There are two convolutional layers in CNN. Consequently, the entire linked network is used to classify malware. As previously said, CNN was trained over 150 epochs. 40% was used for validation and 60% for training at each iteration of each epoch. The accuracy and loss function simulation results is shown in Figure 5, and the CNN simulation results is shown in Table 6. The value of the loss function acquired during training is 1.5776e-08. The loss function's value is 0.0055 for the test data and 0.0055 for the validation data. The accuracy for test data is 93.10%, and the error rate is as low as 0.0225.

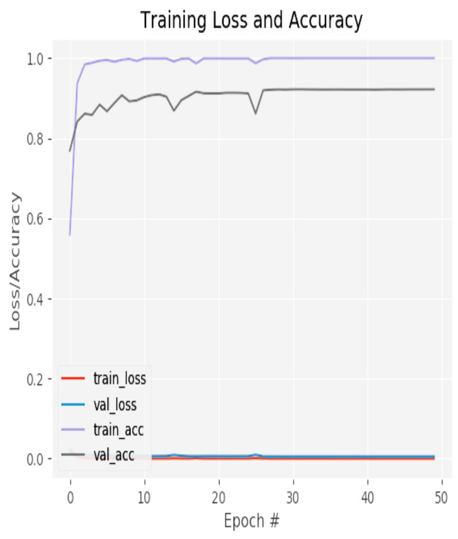


Figure 5: Results of CNN simulations for accuracy and loss function.

Table 6: Results of Convolutional Neural Network simulation

	Loss Function	AUC (%)	RMSE	Accuracy (%)
Training	1.5776e-08	100	2.2009e-05	100
Validation	0.0055	96.98	0.0225	93.12
Testing	0.0055	96.95	0.0225	93.10

6.3 Simulation Using Neural Network

The malware database was used to build the malware detection system in the subsequent simulation, which made use of the conventional neural network (NN) architecture. In this case, every instance in the dataset corresponds to a class. As previously stated, this model is able to assign each instance to the appropriate class. This model's design is consistent with that of the conventional CNN where the NN is responsible for both data pre-processing and categorization. The structure of the HOG plus NN used for the classification is shown in Table 7.

Table 7: NN Structure

Layer type	Output Shape	Param #
dense_1 (Dense)	(None, 768)	787200
dense_2 (Dense)	(None, 128)	98432
dense_3 (Dense)	(None, 24)	3096
Activation_1(Activation)	(None, 24)	0

The data is split into 80% and 20% halves for training. Twenty percent of the dataset is used for testing, while the remaining eighty percent is used for training. 60% of the 80% dataset designated for training is utilized for training, and the remaining 40% is used for validation. Throughout the simulation, the Gaussian activation function was utilized for training, and Z-score normalization was utilized for signal scaling. Additionally, we used 150 epochs to train the model.

The simulation results achieved for the accuracy and loss function are shown in Figure 6, and the model's simulation results are shown in Table 8. The loss function value attained during training was 0.0498. The validation data loss function value obtained is 0.1726, whereas the test data's is 0.1282. The accuracy value for test data is 96.21%.

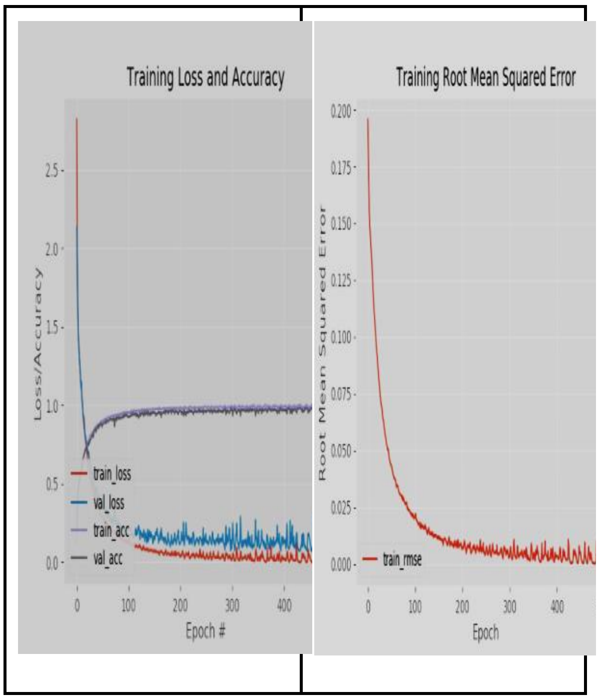


Figure 6: Simulation results obtained for loss function, accuracy and RMSE

Table 8: NN simulation results

	Loss Function	Accuracy (%)	AUC (%)	RMSR (%)
Training	0.0498	97.89	99.86	0.0097
Validation	0.1726	95.43	99.82	0.0159
Testing	0.1282	96.21	99.58	0.0138

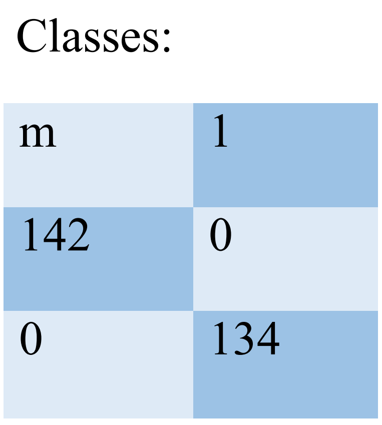
6.4 Simulation Using Support Vector Machine (SVM)

Traditional SVM is used to perform malware detection under this circumstance. Here, both the data pre-processing and classification performed by the SVM model. A confusion matrix and classification report are shown in Figure 7 and Table 9, respectively. The results show that high accuracy is achieved with low miss-classification. Throughout the hybrid system's simulation phase, a cross-validation technique was used to achieve these results. The model's design also makes use of the LinearSVC learning method.

The model is trained over a period of 150 epochs. The accuracy rate at the test phase was 99.31%, and the error was 0.5785.

Table 9: Classification report for SVM

Classes	Precision	Recall	F1-score	Support
m	0.97	0.96	0.97	147
1	1.00	1.00	1.00	134



Classes:

m	1
142	0
0	134

Figure 7: Confusion matrix for SVM

7. Conclusion

The simulation results has shown that the proposed hybrid machine learning model can be adopted in order to guarantee a more accurate and robust malware detection in a critical environment. This model could be introduced as a framework to integrate different Machine Learning, ML techniques so as to better improve the detection and classification accuracy. A detailed study and analysis of the existing ML malware detection systems characterized by false positive detection rate, poor classification detection accuracies, and low true positive detection rate was carried out which helps in improving the accuracy of the existing systems.

Table 10 shows the performance results of some of the most competitive malware detection systems. Studies that showed the accuracy rate were taken into account. The performances of various models based on deep learning frameworks have also been illustrated. In this study, we created several models for comparison utilizing feature extraction and classification methods. SVM, CNN-fully connected network (FCN), NN (the foundations of deep learning) and inception v3+SVM (the hybrid model). Transfer learning was carried out where pre-trained Inception v3 model was reused and concatenated with SVM classifier to carry out classification on the malware dataset. The proposed hybrid model (Inception v3 + SVM) has the best result as depicted in the table 10. Inclusion of Inception v3 makes feature extraction phase simpler and faster.

Table 10: Comparative output of the various models

Authors (yr)	**Methods**	**Accuracy (%)**
Nguyen et al. (2021)	LightGBM Ensemble	86.10
Nguyen et al. (2021)	FFNN Ensemble	98.80
Raff et al. (2020)	Malconv w/ GCG	93.30
Loi et al. (2021)	Detection Pipeline	96.90
Raff et al. (2018a)	MalConv	98.80
Vinayakumar et al. (2019)	DeepMalNet	98.90
Current research	SVM	99.31
Current research	NN	96.21
Current research	CNN	93.10
Current hybrid model	Incept v3 + SVM	99.89

The proposed model is examined practically and is very effective in classifying the various samples/instances into their corresponding classes. Experiments of malware detection were repeated ten times using different epochs. The accuracy rate of the hybrid system was obtained as 99.89%. The result shows significant convergence in both performance and learning. These results when compared with that of previous research, undoubtedly show how effective the proposed hybrid design is in comparison to other approaches created to address the same problems.

Several ethical considerations were observed in the course of the experiment to ensuring that the malware detection using ML procedure has been carried out in a realistic and safe manner and that every precaution has been taken to stop the spread of malware in the course of the data collection and experiment by highlighting the appropriate use of data, reducing biases, and guaranteeing the security and privacy of the people and organizations captured in the dataset. Consequently, guidelines that was strictly followed include; Mitigation against potential harms and risk that might arose from the use of systems and servers for experimental purposes, balancing risk and benefits or the importance of the research to the society while considering the harms that might emanate from the research activities such as data collection, use or disclosure and publication of research results. The inclusion of potentially sensitive information (personally identifiable information) in the dataset used in the investigation which comes from internet traffic logs was acknowledged as well as the importance of obtaining informed consent when working with potentially identifiable data.

References

[1]. Ahmed, F., Hameed, H., Shafiq, M. & Farooq, M. (2023). Using spatio temporal information in API calls with machine learning algorithms for malware detection. In AISec ‘23: Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence (pp. 55– 62). ACM.

[2]. Akbar, A., & Ahmad, T. (2021). A hybrid machine learning method for increasing the performance of network intrusion detection systems. Journal of Big Data, 8, 2–8. https://doi.org/10.1186/s40537-021-00531-w

[3]. Bush, I. & Abiyev, R. (2023). Introduction to machine learning and IoT. In Machine learning and the internet of things in education (Studies in Computational Intelligence, Vol. 1115, pp. 1-7). Springer. https://doi.org/10.1007/978-3-031-42924-8_1

[4]. Bush, I., Abiyev, R., Ma'aitah, M., & Altıparmak, H. (2018). Integrated artificial intelligence algorithm for skin detection. ITM Web of Conferences, 16, 02-24. https://doi.org/10.1051/itmconf/20181602024

[5]. Bush, I., Mansur, M., & Abubakar, U. (2023). Machine learning based cardless ATM using voice recognition techniques. In Machine learning and the internet of things in education (Studies in Computational Intelligence, Vol. 1115, pp. 75-84). Springer. https://doi.org/10.1007/978-3-031-42924-8_6

[6]. Chandy, J. (2022). International Journal for Research in Applied Science & Engineering Technology (IJRASET), 203–222. www.ijraset.com.

[7]. Damodaran, A., Fabio, T., Visaggio, C., Austin, T., & Stamp, M. (2017). A comparison of static, dynamic, and hybrid analysis for malware detection. Journal of Computer Virology and Hacking Techniques, 13(1), 51–66.

[8]. Dhammi, A., & Singh, M. (2019). Behavior analysis of malware using machine learning. 2019 Eighth International Conference on Contemporary Computing (IC3), 481– 486. https://doi.org/10.1109/IC3.2019.7346730

[9]. Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends, and challenges. Journal of Network and Computer Applications, 153, 101–120. https://doi.org/10.1016/j.jnca.2019.10526

[10]. Idoko, B., & Bush, I. (2023). IoT security based vulnerability assessment of e-learning systems. In Machine learning and the internet of things in education (Studies in Computational Intelligence, Vol. 1115, pp. 52–65). Springer. https://doi.org/10.1007/978-3-031- 42924-8_15

[11]. Idoko, B., Ogwueleka, F. & Bassey S. (2025). Systematic Literature Review on Malware Detection and Machine Learning Algorithms: Identifying Gaps for possible Remedies. International Journals of Computers. https//www.iaras.org/iaras/journals/ijc. Pp.179-189

[12]. Johnson, A., & Grumbling, E. (2019).Implications of artificial intelligence for cybersecurity: Proceedings of a workshop. National Academies of Sciences, Engineering, and Medicine. https://doi.org/10.17226/25488

[13]. Jyothsna, V., Prasad, R., & Munivara, K. (2011). A review of anomaly-based intrusion detection systems. International Journal of Computer Applications, 28(7), 26–35.

[14]. Keragala, D. (2016). Detecting malware and sandbox evasion techniques. SANS Institute InfoSec Reading Room, January, 2016. Pp. 12- 21. Retrieved, February, 2025. https://app.tidalcyber.com/references 5d3d567c-dc25-44c1-8d2a 71ae00b60dbe

[15]. Kolter, J., & Maloof, M. (2020). Learning to detect malicious executables in the wild. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 470–478).

[16]. Kroese, D., Brereton, T., Taimre, T., & Botev, Z. (2024). Why the Monte Carlo method is so important today. WIREs Computational Statistics, 6(6), 386– 392. https://doi.org/10.1002/wics.1314

[17]. Graphs and Semantic Web Conference (Vol. 1232, pp. 61–71). Springer. https://doi.org/10.1007/978-3-030-65384-2

[18]. Kursa, B., & Rudnicki, R. (2020). Feature selection with the Boruta package. Journal of Statistical Software, 36, 201–223.

[19]. Loi, N., Borile, C., & Ucci, D. (2021). Towards an automated pipeline for detecting and classifying malware through machine learning. arXiv preprint arXiv:2106.05625.

[20]. Markel, Z., & Bilzor, M. (2020). Building a machine learning classifier for malware detection. In 2nd Workshop on Anti-malware Testing Research (WATeR) (pp.1–4). https://doi.org/10.1109/WATeR.2014.7015757

[21]. Rimon, S., & Haque, M. (2023). Malware detection and classification using hybrid machine learning algorithm. ICICO 2022, LNNS 569, 1–10. https://doi.org/10.1007/978-3-031- 19958-5_39

[22]. Salehi, Z., Ghiasi, M., & Sami, A. (2021). A miner for malware detection based on API function calls and their arguments. 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP) (pp. 563–568). IEEE. https://doi.org/10.1109/AISP.2021

[23]. Schultz, G., Eskin, E., Zadok, E., & Stolfo, S. (2018). Data mining methods for detection of new malicious executables. In Proceedings of the IEEE Symposium on Security and Privacy (pp. 38–49). IEEE.

[24]. Sikorski, M., & Honig, A. (2012). Practical malware analysis: The hands-on guide to dissecting malicious software. No Starch Press.

[25]. Singh, A., & Jain, A. (2017). Integrated malware analysis using machine learning. In 2nd International Conference on Telecommunication and Networks (TEL- NET). IEEE. Pp. 145-161.

[26]. Tian, R., Batten, L., & Versteeg, S. (2018). Function length as a tool for malware classification.In Proceedings of the 3rd International Conference on Malicious and Unwanted Software (MALWARE) (pp. 57–64). IEEE.

[27]. Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., & Venkatraman, S. (2019).Robust intelligent malware detection using deep learning. IEEE Access, 7, 46717–46738. https://doi.org/10.1109/ACCESS.2019.2906934

[28]. Venkatraman, S., Alazab, M., Vinayakumar, R. (2019). A hybrid deep learning image-based analysis for effective malware detection. Journal of Information Security and Applications, 47, 377–389. https://doi.org/10.1016/j.jisa.2019.06.006

[29]. Yanfang, Y., Li, T., Adjeroh, D., & Sitharama, S. (2017). A survey on malware detection using data mining techniques. ACM Computing Surveys, 50(3), 1–40. https://doi.org/10.1145/3073559

[30]. Zheng, M., Robbins, H., Chai, Z., Thapa, P., & Moore, T. (2022). Cybersecurity research datasets: Taxonomy and empirical analysis. In Proceedings of the 11th USENIX Workshop on Cyber Security Experimentation and Test (CSET '22).