# Development of a Hybrid Machine Learning Model (Inception V3 +SVM) for Malware Detection

PATRICIA BOSSAH[1], BARTHOLOMEW IDOKO[2]
[1,2]Department of Computer Science, Federal Polytechnic Ohodo, Enugu, NIGERIA.

*Abstract:* The malware landscape is constantly evolving. The tools used for malware detection and threat elimination in critical environments must evolve too. Discovering threats before they can do harm presents a continuous challenge to the world of cybersecurity. By gaining a better understanding of malware and how it functions, we can take proper steps to eliminate these threats from the systems we want to protect. The research considers; the practice of Malware analysis with threat hunting, Indicator of Attack (IoA) and Indicator of Compromise (IoC) and the Identification and prevention of threats when they would otherwise bypass the traditional detection systems using Machine Learning (ML) model. In this paper, a hybrid model is proposed as a framework that can integrate different ML techniques so as to better improve the detection and classification accuracy. The system is implemented using deep learning structure (Inception v3) plus SVM classifier. The framework's input comes from instances of the explored malware dataset where 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 classification report of inception v3 + SVM model yields a better accuracy when compared to that of the SVM model. The advantage of this framework is that it can be applied to train and partition a very large data set over and over. This framework will guarantee a more accurate and robust detection system that could outperformed the conventional malware detection models.

*Key-words*: Malware, detection; Machine Learning; accuracy; classification; support vector machine, inception v3.

Received: March 23, 2024. Revised: March 12, 2025. Accepted: April 15, 2025. Published: July 30, 2025.

## 1. Introduction

Researchers have over the years attempted to adopt machine learning Techniques as a better detective control mechanism for malware analysis and detection since ML Algorithm has a major influence on the detection accuracy of the static malware detection tools. Even as most scholars proposed a single 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.

Attackers continue to develop new malware codes of existing malware and distribute them throughout the internet shadow economy (Distler, 2013). The cyberspace is presently faced with many incidents related to malware compromising the cyber security goals of; confidentiality, integrity, availability, authenticity, non-repudiation and trust of system networks (Chandy, 2022). The number of malware attack will continue to rise as hackers keep improving on their tactics. In 2016, the total exposed identities to malware attacks were 429 million in over nine mega breaches with over 10 million records (Symantec Internet Security Threat Report, 2016).

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 to identify 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's (Damodaran, *et al*., 2017).

Another problem identified in the study is the use of conventional preventive measures for malicious code detection such as signature-based techniques or recognizing abnormal behavior specified by a rule or a function (Jacob & Wanjala, 2017; Jyothsna, *et al*., 2011; Akbar & Ahmad, 2021). Nevertheless, the shortcomings and difficulties associated with conventional approaches make it hard to get efficient and successful outcomes when conducting an inquiry (Idoko et al., 2022). A new, structured method for information gathering, malware analyses and mitigation is required in light of the sharp rise in malware attacks that compromises data availability, confidentiality, and integrity. (Heureux *et al*., 2017; Akbar & Ahmad, 2021).

It is pertinent to note that several reports have highlighted various problems that needed to be properly addressed by cyber security experts and researchers attempting to work on malware detection and prevention (Idoko et al., 2025). The problem of collecting and evaluating huge amount of dataset for obfuscated malicious software. This method consumes much time and it is highly expensive as well as involves applying multiple procedures (Eid *et al*., 2013; Akbar & Ahmad, 2021). There is often complication arising from the detection when a malware attack succeeded with the aid of sophisticated tools and Techniques (Akbar & Ahmad, 2021). Today's ML-based classification systems are often accuracy-based, which means they only use a calculated likelihood to determine whether something is an attack. This is due to the fact that machine learning techniques in cybersecurity are not tailored to the requirements of attackers or defenders. The uncertainty caused by the same data being spread across multiple computer systems or a vast network, particularly when the detection is carried out using pre-defined malware detection tools as well as hurriedly completed in a workstation (Eid *et al*., 2013).

The selection of hybrid machine learning for this study is due to its innate abilities in complex data processing, pattern recognition, and feature extraction. Because each model has distinct advantages, they are all appropriate options for dealing with the complex problems that advanced cyber threats possess. The objective for utilizing the many advantages of these methods is to have a thorough grasp of their suitability and efficacy in view of malware analysis.

## 2. Empirical Review

The static and dynamic approaches are combined to create the hybrid model. The Malware framework employs a debugging-based behaviour pattern technique to monitor and review data, ultimately confirming the presence of malware (Mukamurenzi, 2018; Bayer *et al*., 2010). The analyst gains more advantage by combining the important features of both methods. It makes sense to combine the static and dynamic approaches to identify malware because the outcome is more consistent and efficient. This will make the proposed new framework for malware analysis and detection a plausible foundation. However, the major setback could be that the method is very slow during malware investigation and analysis but nevertheless, this wouldn't be a limitation to users for whom efficiency overcomes the delay in the time of investigation (Shijo & Salim, 2015).

A proposed study of detecting and analyzing malwares based on Application Package Interface (API) calls 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 number of features. The authors intercepted API calls using a VMware-based virtual machine and the WINAPIoverride32 program. The authors employed WEKA classifiers for classification (http://www.cs.waikato.ac.nz/ml/weka/). 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). The authors extracted mainly three static features using the static analysis approach, these features include, string 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 scientists 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. Static analysis was used to obtain the function length frequency and printed string information (PSI) vectors. API functions and their parameters made up the dynamic feature vectors that were extracted from the log files of dynamic analysis tools such as HOOKAPI. The feature vector for the integrated approach was created by the authors by combining the feature vectors gathered from both static and dynamic analysis. 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. Methodology

The method used in the study involves developing a Support Vector Machine (SVM) model and inception v3+SVM model and compare the model performance with each other and that of other researchers.

## 3.1 Research Design

A hybrid detection model that combines dynamic and static features represents the system architecture shown in Figure 1. The system architecture described the structure of the design stages and how each component is related to one another. From the point of data collection through processing, training, testing as well as the detection module, their exist synergy between two or more components or stages. The system is designed in multiple steps as shown in Figure 1. The following elements will be considered during the system design stage:

**i. Collection of Malware Sample:** The study focuses on the detection of all malicious code, the malware samples is collected from VirusShare, which is an updated private malware repository that provides the latest version of malware samples.

**ii. Dataset Creation:** This aspect is seen as one of the most challenging tasks of this research. An existing dataset is used to update the latest malware dataset. The study uses Portable Executable (PE) file module with python to extract the malware's static characteristics, change the hexadecimal data values to decimal and add them to a CSV file to create the dataset.

**iii. Feature Analysis:** Feature analysis is one of the crucial tasks 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. Therefore, in order to select more significant features for the training of the model, the attributes of each feature is determined.

**iv. Data Processing:** There is a wide range of values for the various features. Thus, a narrower range of data is processed. Before being used, data are normalized. For the training process, the data is subsequently separated into training and testing sets.

**iv. Model Training and Testing:** Various ML techniques can be used for model training. The training set is used to train the model, and the testing set is used to test it. Different models have different advantages and disadvantages. A superior metrics analysis is guaranteed when all the methods are available for use.

**v. Models Performance Evaluation:** Lastly, the models' performance is assessed. Performance determines the model's reliability and qualities since the most accurate models perform well in real-time malware attacks. Similarly, the SVM model developed for the study and other models developed by other researchers are assessed and contrasted with the hybrid learning model.

## 3.2 Model Development

The strategies and techniques for developing both SVM and Inception v3 + SVM are in line with the research design. To develop a hybrid model that can effectively detect polymorphic malware and zero-day malicious code, experimental and quantitative research techniques was used in the study.

**3.2.1 SVM Model Development Strategy:** The SVM model 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. In order to enhance the margin between the classes, the SVM algorithm is taught to alter the hyperplane's position. After training, the SVM determines if a new, unknown data point is harmful by analyzing which side of the hyperplane it falls on. The predict_proba method and the scikit-learn package are used by the system to obtain probability estimates for every class. For a better comprehension of the outcomes, print statements are used to retrieve the dimensions of both outputs. Classification: After training, the SVM uses the side of the hyperplane to identify fresh, unseen data items and categorize them as either benign or harmful.

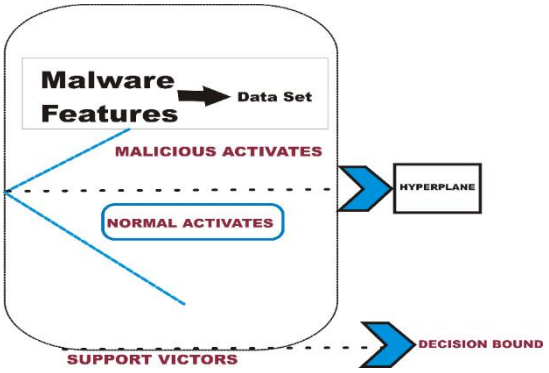


Figure 2: SVM Classification and Data Representation in Malware Detection

**3.2.2 Inception v3 + SVM Model Strategy:** 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 as shown in Figure 3. 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.

A fundamental component of the hybrid system, Inception v3, is built on CNN and is used to extract features for further categorization. Neurons perform convolutions with the convolution filters on input data by employing the weight-sharing technique. The ReLU layer receives the output characteristics from the convolution layers. The acquired signals are then fed into the pooling layer after the generated feature map has been activated using f(x) =max (0, x) as the activation function. More complex feature extraction is carried out after the feature map has been reduced following multiple layers of convolution and pooling.

A sufficiently tiny feature map is then used to convert the contents into a one-dimensional vector, which is then supplied into the classification module (SVM). The

convolutional layer of the CNN is obtained from $I$ = input map, $b$ = biases, and a filter bank (Bush & Abiyev, 2023). The renewed input vector + bias is given by:

$$x_{i,j}^{l} = \sum_{m} \sum_{n} \omega_{m,n}^{l} o_{i+m,j+n}^{l-1} + b^{l} \quad (1)$$

The equation that can determine the convolutional and pooling layers' output is:

$$o_{i,j}^{l} = pool(f(\sum_{m} \sum_{n} \omega_{m,n}^{l} o_{i+m,j+n}^{l-1} + b^{l})) \quad (2)$$

Loss function minimization is used to train CNN parameters. The parameters' values are established during training. The Adam optimizer (adaptive moment estimation) is used to update the network parameters (Abiyev & Adepoju, 2023). In order to match the amount of CNN parameters, this strategy also lessens the relative scarcity of data. Inception v3 is utilized in this study as an SVM module for classification and a CNN model for feature extraction.

## 4. Model Implementation

The Models undergo training and testing so that the machine learning algorithm has the capacity to identify trends and forecast outcomes from input data. The algorithm is fed with the extracted malware dataset consisting of input-output pairs, where inputs are mapped to corresponding outputs by the algorithm 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. The repetition procedure keeps running 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 is deployed to carry out 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 variables for classification and detection; 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 are compared with the projected values, a confusion matrix which is a method for assessing the classification model's performance in machine learning is performed to calculate the error and present a comprehensive picture of the model's performance.

| +ve | -ve |
|---|---|
| TP | FP |
| FN | TN |

Figure 4: Confusion Matrix

There are two values in the target variables: positive (+ve) and negative (-ve);
Where the True Positive = TP, True Negative = TN, False Positive = FP and False Negative = FN
The values of the target variables range between 0 and 1 and are displayed within the python Google Collaboration environment during the prediction. Secondly, By dividing the total number of samples in the test set by the number of successfully identified samples, 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:

$$\text{Accuracy} = \frac{TP + TN}{FP + FN + TP + TN} \quad (3)$$

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 categorization thresholds using the AUC-ROC statistic.

The F1-score is then applied as shown in equation 4.

$$\text{F1 score} = \frac{TP}{} \quad (4)$$

$$TP + ½ (FP + FN)$$

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

$$\text{Recall} = \frac{TP}{TP + FN} \quad (5)$$

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (6)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (7)$$

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 are compared with standard models and those of other researchers so as to recommend the best model for malware detection and classification.

## 5. Results and Analysis of Findings

In this section, simulation of both SVM and Inception V3 + SVM is carried out in order to analyze the data and answer the research questions. The aim is to determine the best performing model from the experiment.

**5.1 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 are performed by the SVM model. A confusion matrix and classification report are shown in Figure 5 and Table 1, respectively. The results show that high accuracy is achieved with low miss-classification. As explained throughout the hybrid system's simulation phase, a cross-validation technique was used to achieve these results.

The model is trained over a period of 160 epochs. The accuracy rate at the test phase was 99.43%, and the error was 0.5522.

Table 1: Classification report for SVM

| Classes | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **m** | 0.97 | 0.97 | 0.98 | 158 |
| **1** | 1.00 | 1.00 | 1.00 | 145 |

Classes:

| | |
|---|---|
| m | 1 |
| 158 | 0 |
| 0 | 145 |

Figure 5: SVM Confusion matrix

**5.2 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 289 and H is 289 in the input size of 289x289x3. 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 2, with the corresponding confusion matrix of the hybrid model used to detect malware files as shown in Figure 6. 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 2. It is evident from these data that the hybrid model is efficient because the miss-classification rate is kept to a minimum.

Table 2: Hybrid model Classification Report

| Classes | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **m** | 1.00 | 0.98 | 1.00 | 158 |
| **1** | 1.00 | 1.00 | 1.00 | 158 |

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.

Classes:

| m | 1 |
|---|---|
| 158 | 0 |
| 0 | 158 |

Figure 6: Confusion matrix for hybrid ML model

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 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 160 training epochs. The accuracy value that has been demonstrated is the average of 10 simulations. The average accuracy rate during the test phase was 99.90%, with an error of 0.0122.

Monte Carlo-style estimation is investigated in the second simulation using the same database, where the hybrid model stops at 600 epochs. The dataset is randomly divided into 60% for training and 40% for testing at each epoch. As was previously indicated, Monte Carlo experiments were conducted using a malware database, and the hybrid model produced an accuracy of 98.99% and an error rate of 0.0192. The results of simulations of the hybrid system using both cross-validation and Monte Carlo techniques are shown in Table 3.

Table 3: Hybrid model simulation Results

| Methods | Accuracy (%) | RMSE |
|---|---|---|
| **Monte Carlo Techniques** | 98.99 | 0.0192 |
| **Cross-Validation** | 99.90 | 0.0122 |

**5.3 Comparison of the results:** In this section the comparison of the proposed hybrid model and SVM, and others by three different researchers is carried out and it was discovered that the proposed hybrid model outperformed all the other techniques as seen in table 4.

Table 4: Model Performance Comparison

| Sources | Methods | Performances (Accuracy) |
|---|---|---|
| Markel & Bilzor, 2020 | SVM | 91.00 |
| Salehi *et al*., 2021 | WEKA-KNN | 86.00 |
| Ahmed *et al*., 2023 | Hybrid ML | 90.00 |
| Proposed model (current study) | SVM | 99.43 |
| Proposed model (current study) | Inception v3+SVM | 99.90 |

## 6. Conclusion

The choice of hybrid machine learning for this study is supported by its innate capacities for handling complex and high-dimensional data, as well as for pattern identification, feature extraction, and sophisticated data processing. Additionally, the methods are thought to have a more stable algorithm. Because each model has distinct advantages, they are all appropriate options for dealing with the complex problems

that advanced cyber threats provide. The reasoning stems from utilizing the many advantages of these methods to have a thorough grasp of their suitability and efficacy in the detection of malware.

SVMs are generally able to produce good accuracy, especially for large datasets with a lot of noise or overlapping classes, it could be more effective. However, integrating two different models; a deep learning (Inception V3) model and a conventional machine learning (SVM) model produce a better result that has contributed to the body of knowledge in the field of malware detection and Artificial Intelligence. The results shows that Inception v3 + SVM achieves 99.90 accuracy, SVM has 99.43 accuracy when compared with Dooms, et al., 2018; Patro et al., 2021 and Salem et.al., 2022 with accuracies of 91.00, 86.00 and 90.00 respectively.

**Acknowledgment**

The authors would like to express their sincere appreciation to TETFUND in Nigeria for providing the funding for this Institution Based Research (IBR).

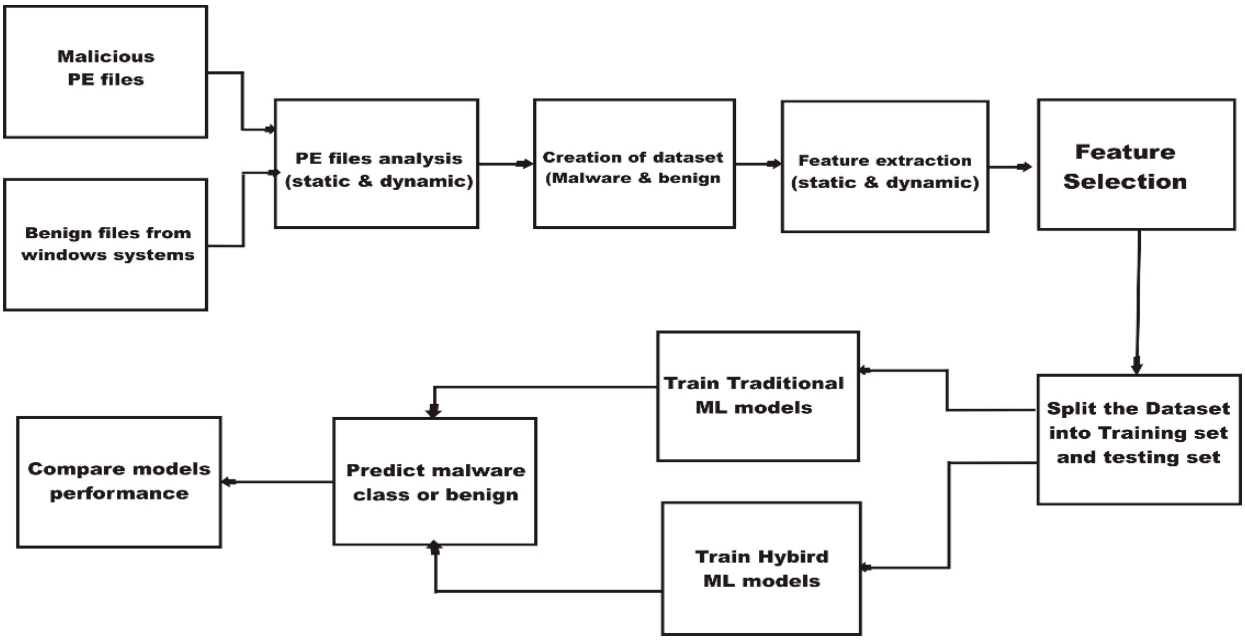


Figure 1: Research Design

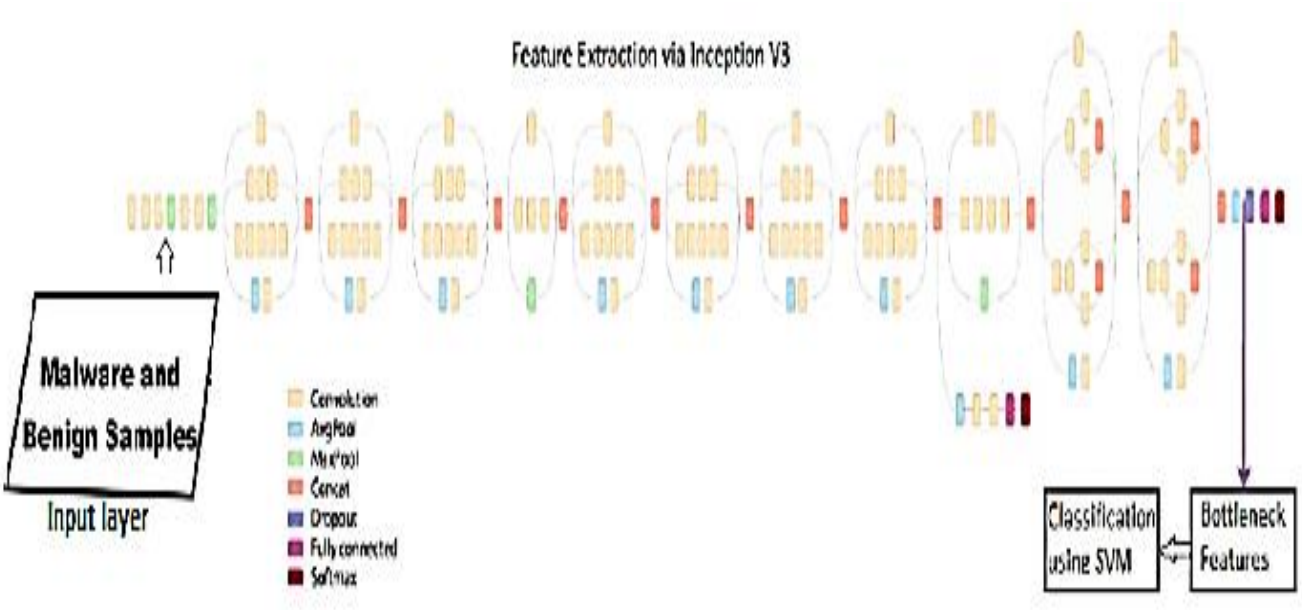


Figure 3: Proposed model Architecture

*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]. Abiyev, R., & Adepoju, J. (2023). Deep convolutional network for food image identification. In Machine learning and the internet of things in education (Studies in Computational Intelligence, Vol. 1115). Springer. https://doi.org/10.1007/978-3-031-42924-

[3]. 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/s40537021-00531-w

[4]. Bayer, U., Kirda, E., & Kruegel,C. (2010). Improving the efficiency of dynamic malware analysis. In Proceedings of SAC '10 (pp. 112–131). ACM.

[5]. 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 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

[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]. Distler, D. (2013). Malware analysis: An introduction. SANS Institute InfoSec Reading Room. https://www.sans.org/readingroom/white

[10]. Eid, H., Hassanien, A., Hoon, T.,& Banerjee, S. (2013). Linear correlation- based feature selection for network intrusion detection model. In Communication, Computing, and Information Science (Vol. 381, pp. 240– 248).

[11]. Heureux, L., Grolinger, K., Elyamany, H., & Capretz, M. (2017). Machine learning with big data: Challenges and approaches. IEEE Access, 5, 776–779.

[12]. 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

[13]. Idoko, B., JB Idoko, YZM Kazaure, YM Ibrahim, FA Akinsola, AR Raji. (2022): IoT Based Motion Detector Using Raspherry Pi Gadgetry. 2022 5th Information Technology for Education and Development (ITED), 1-5. 978-6654-9373-3/22 $31.00 (c) 2022 IEEE

[14]. 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

[15]. Jacob, N., & Wanjala, M. (2017). A review of intrusion detection systems. Global Journal of Computer Science Technology, 17(3), 11–14.

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

[17]. 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).

[18]. 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

[19]. Moustafa, N., HuJillSlay, J., & Creech, G. (2017). Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. IEEE Transactions on Big Data, 65–81.

[20]. Mukamurenzi, M. (2018). Storm Worm: A P2P botnet. NTNU (Norwegian University of Science and Technology) white paper.

[21]. 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

[22]. 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.

[23]. Shijo, P., & Salim, A. (2015). Integrated static and dynamic analysis for malware detection. Computer Science, 46, 804–811. https://www.researchgate.net/publication/276109044_integrated_static_and_dynamic_analysis_for_malware_detection Symantec Security Response. (2016). Locky ransomware on aggressive hunt for victims. https://www.symantec.com/connect/blogs/locky-ransomware-aggressive-hunt-victims

[24]. 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. University of Waikato. (n.d). WEKA version 3.8.1 Retrieved, May 15, 2025. Pp 28-37 from https://www.cs.waikato.ac.nz/ml/weka/