Static Malware Analysis using Image Transformation and Deep Neural Networks

Shreyanshpati Yadav, Department of Information Technology, Delhi Technological University, New Delhi, India email: yadavshreyansh09@gmail.com

Rahul Gupta, Department of Information Technology, Delhi Technological University, New Delhi, India email: rahulgupta100689@gmail.com

Abstract: A great deal of research and study is being done in the field of malware analysis and the classification of malware using different models and techniques. Recent years have seen an increase in the presence of malware code files, making it very time-consuming and tedious for malware analysts to manually analyze the large number of files. Malware detection is imperative to the functioning of anti-virus companies, and hence, it is essential to have efficient tools to quickly analyze a large number of source files and detect the presence of malware within them. Conventional malware analysis methods are primarily of two types: static analysis and dynamic analysis (with a hybrid approach also possible). In static analysis, tools are used to parse entire source files to detect malware. In contrast, dynamic analysis executes the files in a sample sandbox environment to examine the malware's behavior. However, in today's times, a popular up-and-coming technique is to represent the malware as images and use machine learning models (neural networks) to classify these images and malware files into families. After properly training the classifier, the model would be able to classify them accordingly. Researchers and academicians have been moving towards automatic malware classification due to the rise in the number and complexity of malicious code in modern software. Converting malware code into images and then classifying malware images using neural networks is an attempt to simplify the malware detection process. The overall objective of this research is to classify the identified malware dataset and train our model such that it can eventually classify new malware files into families on its own. The application of modern machine learning techniques, specifically convolutional neural networks (CNNs), has been employed. Using image processing techniques, an effective method for categorizing and visualizing malware has been proposed as the first step in this research. The proposed technique achieves an overall accuracy of 96.57 % with the combination of CNN and SVM as the classifier.

Keywords: Malware Classification, Binary Images, Convolutional Neural Networks, Deep Learning, SVM

1. Introduction

In the modern world, malware attacks are widespread and pose serious threats to various industries every day. Malware detection and analysis have seen significant research in the past decade due to the wide variety of obfuscation techniques and the rapid pace at which malware families have evolved, becoming increasingly dangerous. Several new variants of malware have emerged, and one of the primary steps in understanding and dealing with malicious code is to successfully classify them. Various techniques, such as packing, encryption, and the use of polymorphic malware, have posed new problems for software engineers and researchers.

To handle the ever-increasing evolution of malware, it is crucial to develop more robust and advanced malware classification methods. Our problem statement revolves around a pertinent issue: classifying malware files into families and studying the similarities among classes. Malware file samples that belong to the same family tend to share common characteristics and trends, making it easier to deal with them if their base class is known. The objective of analyzing existing machine learning malware classification models, comparing their effectiveness, and modifying existing approaches to yield different results forms the backbone of our problem statement.

Existing approaches to malware classification have employed conventional techniques, such as static and dynamic analysis, in conjunction with modern machine learning-based methods. The most successful of these include the use of neural networks to achieve higher effectiveness in malware classification tasks. In the domain of neural networks, significant success has been found by using image classification techniques to classify malware files. There has been considerable work in converting binary malicious file code to images and training models to detect malware files from their image representations.

To develop a thorough understanding and gain insights into the problem statement, we decided to holistically analyze existing techniques and methodologies by testing three different models on two popular malware datasets: the Microsoft BIG Malware Dataset and the MalImg dataset. Convolutional Neural Networks (CNN), K-Nearest Neighbour (KNN), and Convolutional Neural Networks amalgamated with Support Vector Machine (CNN+SVM) have been applied to understanding the approach towards malware classification.

Extensive experiments have been conducted on the two datasets to obtain tangible results and metrics, providing a comparative analysis between existing techniques. These are also

intended to serve as the foundation and bedrock for further development in malware classification models by extending them to evolutionary algorithms and exploring the potential application of genetic programming in this context. The application of the three models has yielded significant results that support the problem statement.

Malware analysis has traditionally been carried out to detect malware files using conventional techniques, such as static and dynamic analysis [1]. However, despite the promising nature of some of these techniques, especially in dynamic analysis, the fact that they require running malware files in a virtual environment, and the limited extent to which these techniques can be effectively scaled, makes them less efficient than several modern machine learning based alternatives [2]. Various malware families tend to exhibit similar patterns in their interaction and behavior within systems, and hence, techniques based on machine learning are now an insightful alternative to malware classification [3].

2. Related Work

The past 10-15 years have been marked by significant efforts in developing intelligent ML-based malware classification models. These use a variety of techniques used in machine learning, namely Support Vector Machines [4], Naïve Bayes classifiers [5], multidimensional classifiers [6], and so on, but a lot of success has been achieved using neural networks and models based on techniques such as CNN and KNN [7]. However, many CNN-based approaches have encountered issues with large malware samples and have required modifications to address problems in detection and classification. Consequently, there is a need for a more advanced CNN architecture to support malware classification [8]. In some cases, classification methods and approaches have required the execution of actual malware code as well. In contrast, other related research has limited the extent of viewing malware families as digital images and classifying them into families [9].

Modern methods in malware classification research have also included the use and implementation of feature extractors, such as recurrent neural networks, which have been used to extract features that have even been used to train the classifier to detect malicious malware [10]. In some techniques, methods have also been applied to reduce the dimensionality of the input data for the ML model. After this, a neural network is used to reduce the error rate by a significant margin to obtain the desired results (as seen Dahl et al.) [11]. Some other works in the domain have employed graph-based approaches built upon clustering techniques [12], while researchers have also attempted to utilize metadata and analyze import and export tables to aid in malware analysis [13].

However, among all the existing research in malware classification, neural network-based approaches, especially those built upon CNN, have been found to be most successful in building a generic framework for the purpose of classifying malware samples into their respective families [11]. Building upon the work of Kalash et. The Al, Maling, and Microsoft datasets have been utilized to develop an enhanced CNN model for malware family classification, thereby enhancing the effectiveness of the approach [14]. K-nearest neighbor-based classification techniques are also implemented to compare the accuracy and other key metrics of the technique's success.

3. Methodology

The following section will provide a more in-depth examination of the work. It demonstrates the CNN architecture used to generate images for the malware files. The necessary data for our problem domain is explained in Section 3.1. Our model architecture and its primary components are represented in the following parts. This also includes the model's implementation details and other related components.

3.1 Dataset Description

The popular Malimg dataset, provided by Natarajan et al. [15], is a database of 9,339 malware samples. These samples are visualized as images before being applied to a CNN. In our implementation, we have utilized images, and each malware sample in the MalImg dataset belongs to one of the reference malware families. There is a total of 25 families. The number of samples of malware images belonging to each family varies in MalImg, and the dataset is split into training and testing samples. In the implementation, 70 percent of the dataset, i.e., 6,538 malware samples, are part of training, and the remaining 30 percent, i.e., 2,801 malware samples, are used for testing. There are 25 families (classes) of malware present in the MalImg dataset.

Microsoft released a large dataset of malware samples, comprising 21,741 files, in 2015 as part of its malware classification challenge, which was conducted on Kaggle. The dataset was partitioned for ease of implementation, with 10,686 malware files divided into two parts: training and testing, in a 70:30 ratio (7,480 files for training and 3,206 files for testing). Like MalImg, the split and distribution of malware samples varied in different datasets. However, all the 10,686 files belong to one of 9 malware classes.

3.2 Executables Binay Files

One of the first steps in implementing our model was to create byte files (hexadecimal

representations) of the malware's Portable Executable (PE) portion. The malware files comprise several rows (records), and the dataset is divided into two sections for easy conversion of an executable file into a byte file. The two sections of the dataset are the instruction's memory address offset and the instruction itself. The conversion to byte files is performed using the IDE disassembler tool, which is essential because it serves as a mapping of the original assembly instructions present in assembly source code files.

Malware binaries used are taken in portable executable format, and this is usually recognized by their extensions such as .bin, .dll, .exe etc. Portable executable files are made of various components, which include the following:

1) .text- Code section, contains program instructions
2) .rdata- Read-only data
3) .data- Contains modifiable data
4) .rsrc- Contains resources used by malware

Hexadecimal pairs are combined to create pixel values, and malware binary files are converted to pixels in grayscale format by taking 8 bits at a time. These grayscale images display a range of textual patterns that serve as the basis for classifying malware into distinct families. The patterns are represented in Figure 1.

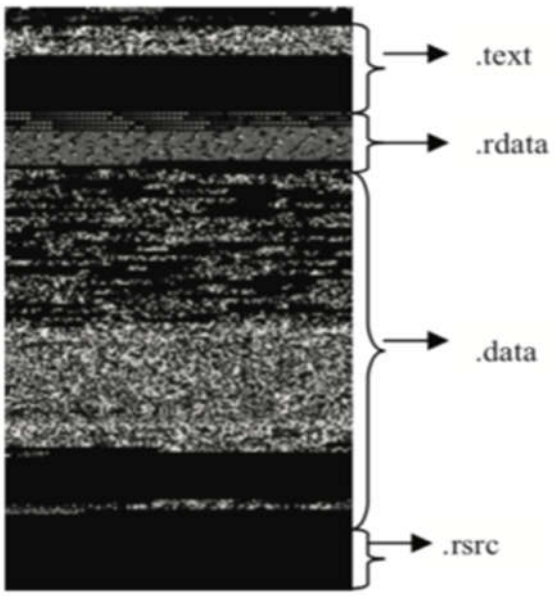

Figure 1: Representation of PE files as an image

3.3 Visual Representation of Malware

The first step in converting a malware sample into its equivalent image is to interpret every byte as a pixel. Visualizing a malicious executable file as an image makes it easier to differentiate different sections of the binary source code. Moreover, given the tendency of malware creators to make only minor variations in new malware, visual references to existing malware aid in detecting these subtle changes. At the same time, the structure of the malware samples remains the same, and it is only when observed in great depth and detail that distinct feature patterns can be noticed. Another reason to go for a visual representation of malware files is that zero-padding can be detected much more easily. Zero-padding is a technique used to align blocks and reduce overall entropy in executable files, which becomes more easily observable in images. The flow is shown in the figure. 2.

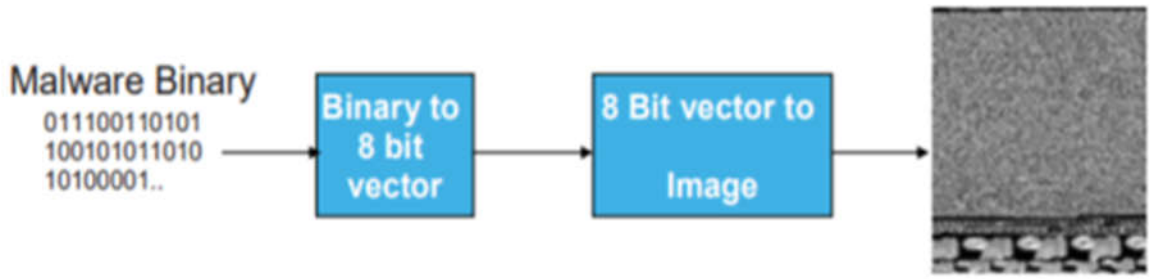

Figure 2. Representation of malware in image form

The process can be summarized as follows:

1. Read a binary malware file in the form of a vector containing 8-bit unsigned integers.
2. Convert every component of the vector from binary to decimal value.
3. Save the decimal value using another vector representing samples of malware files.
4. Reshape resultant vector to 2D/3D matrix and visualize as grayscale/ RGB image.
5. Determine the spatial resolution of the binary file as per its size

3.4 Training and Testing of Model

To check the performance of the proposed model, the dataset is split into training and test sets in a stratified way.

Experiments are conducted across the two datasets as follows:

1) MalImg malware dataset is a practical malware collection of 9339 malware files from 25 malware families, as indicated in Table 4.1 above. The characteristics of each family, including the number of samples in each family, are listed in Table 4.1. We divide the data in the same way as before: 70% in training and 30% in testing.
2) Microsoft (BIG 2015) is a practical malware collection that contains 10686 malware samples from 9 malware families. In Table 4.2, the data for each family, including the sample size, is listed. We divide the data in the same way as before: 70% in training

and 30% in testing. The 2 datasets used as part of this research were then tested by applying the following techniques:

- KNN
- CNN
- CNN+SVM

The implementation and results of the performed experiments are explained in the following section.

4. Results and Discussion

The experimental findings are presented in the following chapter. Section 4.1 includes the implementation of K-NN. Section 4.2 includes the implementation of CNN. Section 4.3 includes the implementation of CNN+SVM.

4.1 K-Nearest Neighbour

A key method for solving classification-based problems is to utilize the standard K-Nearest Neighbours (K-NN) algorithm. Owing to its simplicity, it is able to largely reduce the number of hyperparameters that are to be optimized. Furthermore, the absence of an explicit training phase facilitates the process and is referred to as the scoring phase. Upon input of the training data, the sample is scored across the entire set by determining the nearest neighbours, followed by a majority vote-based classification system.

Although the absence of complex parameters eases the training process, it is vital to choose an optimal value for ‘k’. Under-assigning the ‘k’ value can result in the data becoming overfitted, making it difficult to classify unknown and new data points, thereby reducing our accuracy. Over-assigning the ‘k’ value results in underfitting, which also leads to reduced accuracy. Hence, we ensure that we obtain a value of k in the model that yields the highest accuracy. As part of the current research, we plot the validation accuracy across a range of values for ‘k’ and determine the value of ‘k’ at which deviations from that value result in a decline in model performance.

The upper and lower bounds for the search space of k have been set at 1 and 100, respectively. Once we’ve fitted the K-NN model on both datasets, it was observed that the MalImg dataset yielded optimal performance with the model at k=21, whereas the corresponding optimal value for the Microsoft dataset was at k=3. This is as shown in the figure. 3 & 4.

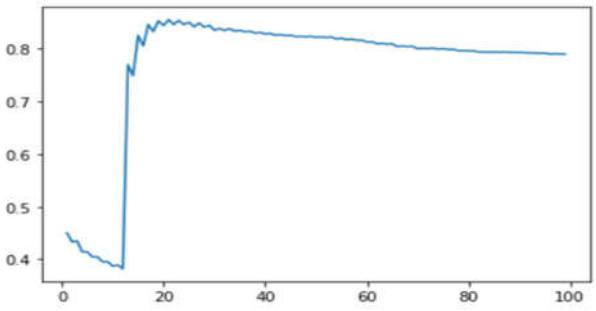

Figure 3 Optimal K-values for MalImg Dataset

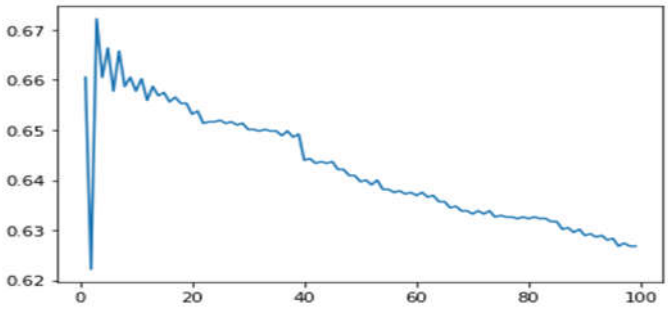

Figure 4 Optimal K-values for Microsoft Dataset

The results of training and testing our KNN model are represented by a confusion matrix shown in Figures 5 & 6 below. The first image displays the confusion matrix for MalImg, while the next one shows the confusion matrix for the Microsoft BIG Dataset. Tables 1 & Table 2 show the weighted average accuracy, precision, recall, and F1-score

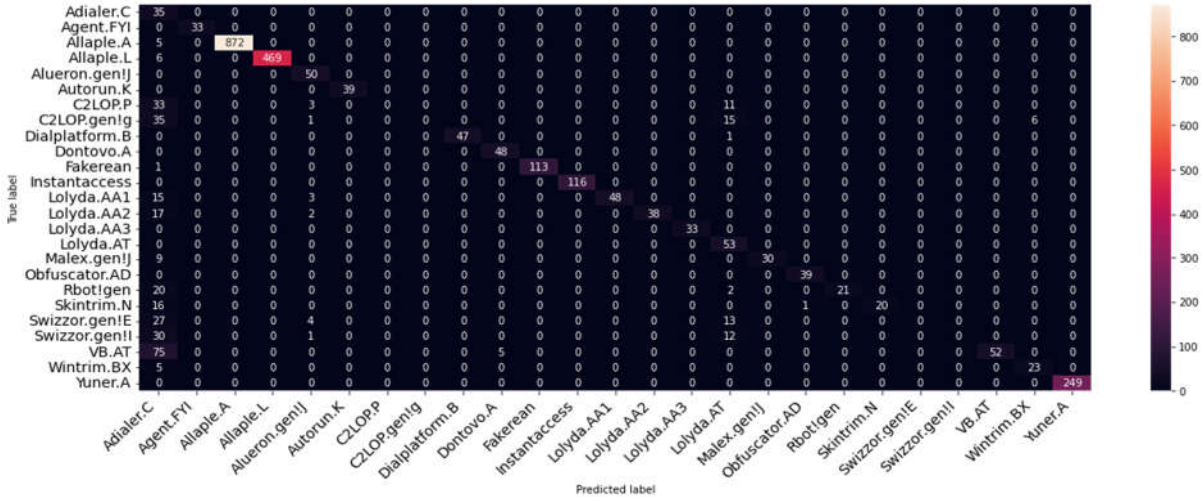

Fig. 5. KNN Confusion Matrix for MalImg Dataset.

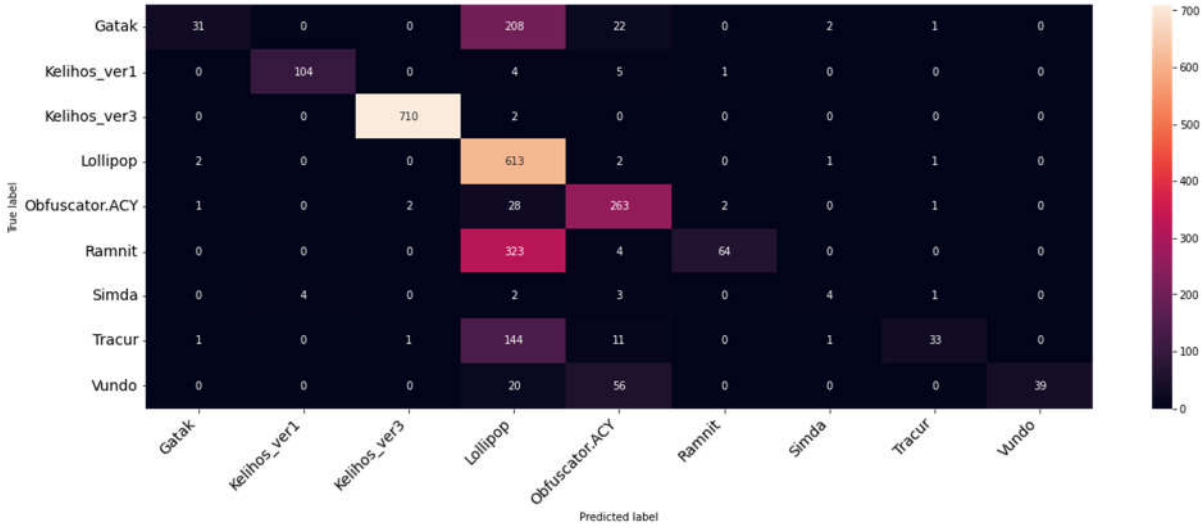

Fig. 6. KNN Confusion Matrix for Microsoft BIG Dataset.

Table 1 Results Metrics of KNN on MalImg Data Set & Microsoft Big Dataset.

Model	Accuracy	Precision	Recall	F1-Score
KNN (MalImg)	88.65 %	91%	87%	87%
KNN (Microsoft Big)	77.21 %	81%	68%	63%

4.2 CNN

Due to the visual representation of the malware object code, the model classification is inherently an image classification task performed with the assistance of the CNN models. The various malware visual representations have been classified into categories by the discrete extraction of patterns within them, as the binary image files obtained from a particular malware pattern family have a higher propensity to produce images of greater similarity. Upon making this assumption, we observe that feature extraction helps identify patterns based on the distribution of pixels on the screen.

A convolutional neural network takes input in the form of an image containing the parameters of the passed malware image. The input malware image is passed to the CNN model for classification and processing, which is then followed by multiple convolution layers, each with filters of 50, 30, and 15, respectively. Every single convolution layer is then backed by Max Pooling layers of pool size (2X2), which then reduces input signals for the image. Additionally, a Flatten layer is followed, which essentially normalizes the malware image into 128 neurons for a Fully Connected layer, preceded by a ReLu Activation Layer. This further classification is achieved with 9 neurons and 25 neurons for the Microsoft Big 2015 and MalImg datasets, respectively. The main structure of the CNN model can be represented as:

1. Convolutional Layer 1 with 50 filters and a kernel size of 3X3
2. ReLu as the Activation Layer
3. Max Pooling Layer with pool size of 2X2
4. Convolutional Layer 2 : 30 filters, (3 * 3) kernel size
5. ReLu as the Activation Layer
6. Convolutional Layer 3 with 15 filters and kernel size of 3X3
7. ReLu as the Activation Layer
8. Max Pooling Layer with pool size of 2X2
9. Dropout Layer which essentially drops the 25 percent of the neurons.
10. Flatten Layer
11. FC Layer 1 with 128 neurons and ReLu as the activation function
12. Dropout Layer which essentially drops the 50 percent of the neurons.
13. FC Layer 2 with 50 neurons and Softmax as the activation function
14. FC Layer 3 with 25 neurons and 9 neurons for Malimg and Microsoft Big 2015 dataset, respectively, using Softmax activation function.

We trained our model for 15 epochs. After training and testing our model on both the Microsoft Malware Dataset (BIG 2015) and the MalImg dataset, we achieved accuracies of 86.46% and

88.11%, respectively. The confusion matrices of the trained and tested models are represented in Figs. 7 & and 8 for MalImg and the Microsoft Big dataset, respectively.

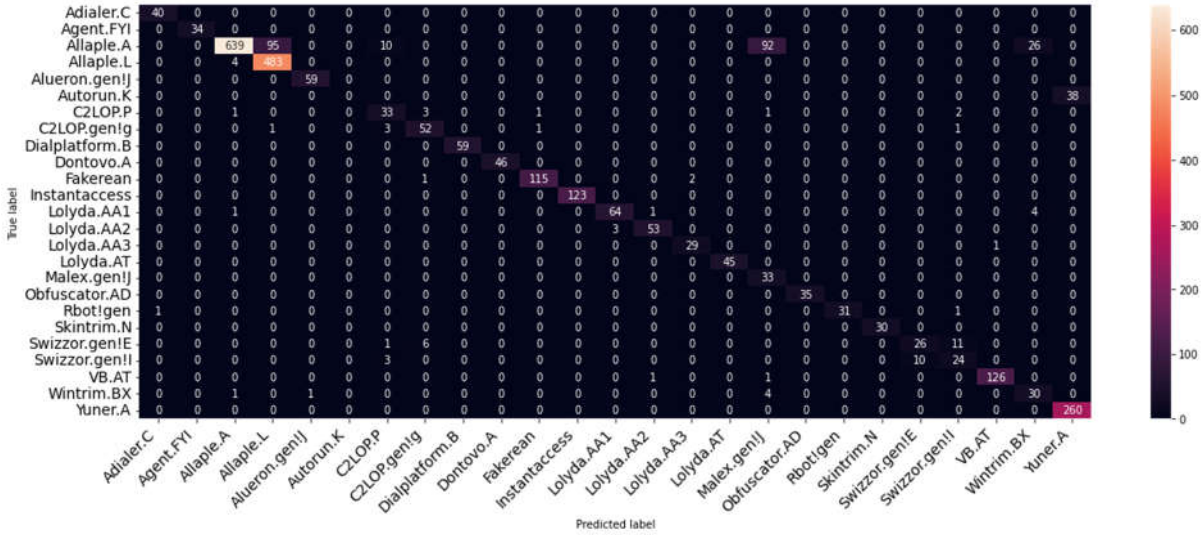

Fig. 7. CNN Confusion Matrix for MalImg Dataset.

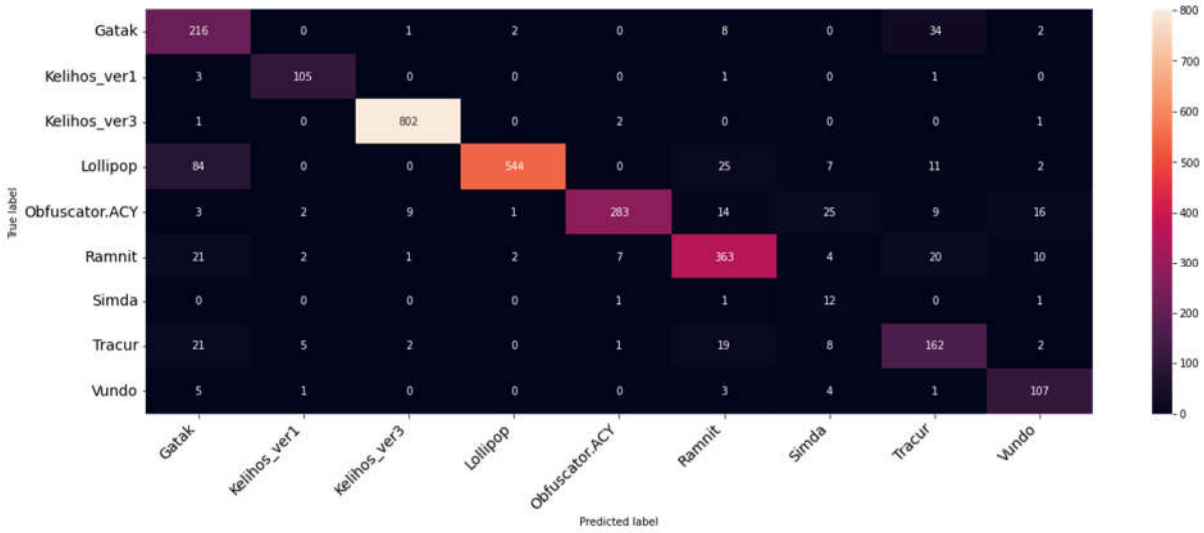

Fig. 8. CNN Confusion Matrix for MalImg Dataset.

The results of training and testing our CNN model are represented by accuracy, as well as calculations of parameters such as F1-score, recall, and precision in Table 2.

Table 2 Results Metrics of KNN on MalImg Data Set & Microsoft Big Dataset.

Model	Accuracy	Precision	Recall	F1-Score
CNN (MalImg)	86.46 %	91%	88%	88%
CNN (Microsoft Big)	88.11 %	89%	86%	87%

4.3 CNN and SVM

In this model, the characteristics and strengths of the two models- CNN and SVMs- were combined to create a hybrid multi-classifier. The Softmax layer in the CNN model was replaced

with a CNN layer. Moreover, a regularizer (L2) was employed, followed by the use of an optimizer. The range of weight values that the network could hold was constrained by the CNN to reduce overfitting. The overall structure can be understood as follows:

1. Convolutional Layer 1 with 50 filters, each with a kernel size of (3, 3)
2. ReLu as the Activation Layer and Strides = 2
3. Max Pooling Layer with the pool size of 2X2
4. Convolutional Layer 2 with 30 filters, each with a kernel size of (3, 3)
5. ReLu as the Activation Layer
6. Convolutional Layer 3 with 15 filters, each with a kernel size of (3, 3)
7. ReLu as the Activation Layer and Strides = 2
8. Max Pooling Layer with the pool size of 2X2
9. Flatten Layer
10. FC Layer with 128 neurons and ReLu as the activation function.
11. SVM Layer, which comprises the Dense layer with kernel regularizer as l2, Softmax as the activation function
12. Optimizer = 'adam' and loss as 'squared_hinge'

The CNN+SVM model yields the best results among all three models used. The accuracy is increased, and the computational requirements are also decreased. After training and testing our model on both the Microsoft Malware Dataset (Big 2015) and the MalImg dataset, we achieved accuracies of 94.80% and 96.57%, respectively. The results for the training and testing of our CNN+SVM model are represented by a confusion matrix in Figs. 9 & fig. 10 for the MalImg and Microsoft Big datasets, respectively. The accuracy and calculations of parameters, such as F1-score, recall, and precision, are presented in Table 3.

Table 3 Results Metrics of CNN+SVM on MalImg Data Set & Microsoft Big Dataset.

Model	Accuracy	Precision	Recall	F1-Score
CNN + SVM (MalImg)	94.80%	95%	97%	96%
CNN + SVM (Microsoft Big)	96.57%	95%	95%	95%

Tables 4 & 5 present the final comparison of accuracies between the three techniques on the two datasets.

Table 4 Comparison of accuracy for Malimg Dataset

MalImg Dataset	
Model	Accuracy
CNN	88.11 %
KNN (k=21)	88.65 %
CNN + SVM	96.57 %

Table 5 Comparison of accuracy for Microsoft BIG Dataset

Microsoft Big Dataset	
Model	Accuracy
CNN	86.46 %
KNN (k=21)	77.21 %
CNN + SVM	94.80 %

5. Conclusion and Future Scope

The increasing threat posed by malware has made it essential to research malware families and develop robust techniques for classifying and characterizing them. Our research involved converting malware file data into a visual representation in the form of grayscale images and then applying image classification techniques using CNN, K-NN, and CNN+SVM to assess the effectiveness of these methodologies. The endeavour has been benchmarked by comparing the results of these different models on two datasets: the Microsoft BIG Malware Dataset and the MalImg Dataset, and also by building upon existing research in these domains. Different down-sampling sizes, filter sizes in CNN, and other model specifications have been used to obtain various metrics for analysis. The details regarding the implementation of the models, the results obtained, and the comparative analysis between the techniques are described in Sections 3 and 4.

The next step in the research is to develop an evolutionary algorithm for malware family classification. Evolutionary algorithms are a modern machine learning concept inspired by biological evolution, which have shown great results with a high level of accuracy in various applications. Using such a technique in a domain like malware classification is likely to produce highly effective results. Among evolutionary algorithms, future work would attempt to apply genetic programming (GP), a subset of evolutionary algorithms, for malware classification. The adaptive nature of genetic programming helps resolve several drawbacks of traditional classification techniques, thereby forming more functional relationships between data and its categories.

6. References

[1] N. Idika and A. P. Mathur, “A survey of malware detection techniques,” Purdue University, West Lafayette, IN, USA, Tech. Rep., 2007.

[2] K. Rieck, T. Holz, C. Willems, P. Düsel, and P. Laskov, “Learning and classification of malware behavior,” in Proc. Int. Conf. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), Berlin, Germany: Springer, 2008, pp. 108–125.

[3] M. Siddiqui, M. C. Wang, and J. Lee, “A survey of data mining techniques for malware detection using file features,” in Proc. 46th Annu. Southeast Regional Conf., New York, NY, USA: ACM, 2008, pp. 509–510.

[4] W. Hardy, L. Chen, S. Hou, Y. Ye, and X. Li, “DLMD: A deep learning framework for intelligent malware detection,” in Proc. Int. Conf. Data Mining (DMIN), Las Vegas, NV, USA, 2016.

[5] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, “Data mining methods for detection of new malicious executables,” in Proc. IEEE Symp. Security and Privacy, Oakland, CA, USA, 2001, pp. 38–49.

[6] J. Z. Kolter and M. A. Maloof, “Learning to detect malicious executables in the wild,” in Proc. 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, New York, NY, USA: ACM, 2004, pp. 470–478.

[7] D. Gibert Llauradó, “Convolutional neural networks for malware classification,” M.S. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2016.

[8] Kaggle Blog, “Microsoft malware classification challenge (BIG 2015): First place team—Say no to overfitting,” May 2015. [Online]. Available: http://blog.kaggle.com/2015/05/26/microsoft-malware-winners-interview-1st-place-no-to-overfitting/

[9] G. Conti and S. Bratus, “Voyage of the reverser: A visual study of binary species,” in Proc. Black Hat USA, Las Vegas, NV, USA, 2010.

[10] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, “Malware classification with recurrent networks,” in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Brisbane, QLD, Australia, 2015, pp. 1916–1920.

[11] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, “Large-scale malware classification using random projections and neural networks,” in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada, 2013, pp. 3422–3426.

[12] J. Kinable and O. Kostakis, “Malware classification based on call graph clustering,” J. Comput. Virol., vol. 7, no. 4, pp. 233–245, Dec. 2011.

[13] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, “PE-Miner: Mining structural information to detect malicious executables in real time,” in Proc. RAID, 2009.

[14] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, “Malware classification with deep convolutional neural networks,” in Proc. 9th IFIP Int. Conf. New Technologies, Mobility and Security (NTMS), Paris, France, 2018, pp. 1–5.

[15] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” in Proc. 8th Int. Symp. Visualization for Cyber Security (VizSec), New York, NY, USA: ACM, 2011, pp. 1–8.