A BI-MODAL DEEP LEARNING TECHNIQUE FOR MALWARE CLASSIFICATION

Samik Bhattarai[1] *, **Samir Shrestha**[1], **Dinesh Ghemosu**[1]

[1] Department of Computer and Electronics Engineering, Khwopa College of Engineering, Tribhuvan University

Abstract

In recent times, there has been a notable surge in the prevalence of intrusive malicious programs infiltrating our devices unbeknownst to us. The identification and categorization of such malware have commonly employed methodologies like static analysis, dynamic analysis, and hybrid analysis. With the abundance of extensive data and advances in deep learning models, a multitude of techniques have emerged for the detection and classification of malware. This paper introduces a bimodal approach for malware classification based on static features using the Microsoft Malware Classification Challenge (BIG 2015) dataset. It incorporates two input modes, one utilizing malware images and the other employing malware metadata. The methodology involves the transformation of raw byte files of malware into visually interpretable grayscale images. Additionally, a meticulous feature engineering process utilizes .asm files of malware to extract metadata. The proposed method employs various Convolutional Neural Network (CNN) layers for processing malware images derived from byte files, and a Deep Neural Network (DNN) to handle malware features extracted from .asm files. A hybrid feature map is generated by fusing the output of CNN and DNN, which is then passed to the classification layer. The model presented in this paper achieves an accuracy of 98.62%, precision of 98.65%, f1-score of 98.63%, and recall of 98.60%.

Keywords: Malware Analysis, Malware Classification, Bi-Modal, Convolutional Neural Network, Deep Neural Network

1. Introduction

Malicious software, or malware, is deliberately crafted software aimed at causing harm to computers or computer networks, posing a significant threat in today's computing landscape. In 2022, the global tally of malware attacks reached 5.5 billion, indicating a two percent increase compared to the previous year. The highest recorded number of malware attacks in recent years occurred in 2018, with a global count of 10.5 billion reported incidents (Petrosyan, 2023a). As illustrated in Figure 1, ransomware attacks impacted over 72 percent of businesses worldwide in 2023. The primary catalyst for the surge in malware attacks is the widespread use of obfuscation, a common technique employed by malicious software developers to create new iterations of their harmful programs. To effectively counteract the escalating threat of malware, a robust classification technique is imperative for managing variants belonging to the same malware family. Understanding a malware's behavior and its propensity to infect devices is crucial for devising effective countermeasures against its proliferation.

To minimize the potential impact of malware, a variety of techniques have been employed over the years for classifying and analyzing malicious software. The predominant methods for malware analysis include static analysis and dynamic analysis. Static analysis, the most widely utilized approach, does not involve the execution of actual malware samples. Instead, it focuses on extracting metadata through the disassembly of opcodes, API calls, segment information, and similar elements. Unlike dynamic analysis, static analysis does not require a live environment for running malware samples, resulting in lower system resource consumption. However, static analysis has drawbacks, as it

*Corresponding author: Samik Bhattarai
Department of Computer and Electronics Engineering, Khwopa College of Engineering, Tribhuvan University
Email: samikbhattarai010@gmail.com
(Received: Jan. 15, 2025, Accepted: April 23, 2025)
https://doi.org/10.3126/jsce.v12i1.82360

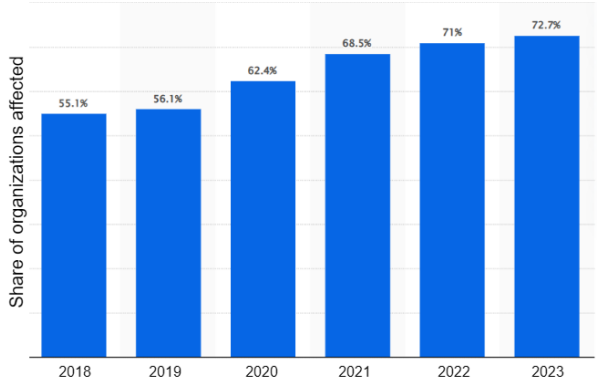

Figure 1. Annual share of organizations affected by ransomware attacks worldwide from 2018 to 2023. Source: (Petrosyan, 2023b)

struggles to effectively analyze malware codes employing obfuscation techniques and is limited in its ability to address polymorphic and metamorphic malware.

Conversely, dynamic analysis entails running malware samples and scrutinizing their trends and patterns within a secure sandbox environment. This methodology facilitates the examination of malware behavior in real-time using a debugger, enabling the identification of polymorphic malware and a thorough comprehension of the malware's actions. Dynamic analysis focuses on the runtime actions that are challenging to obfuscate. However, due to the need to execute the code, this approach demands more system resources and a lengthier duration to observe and analyze dynamic behavior. Additionally, precautions must be taken to ensure that the malware's execution during analysis does not lead to the infection of the system.

Traditional methods for classifying malware rely on machine learning algorithms that necessitate feature extraction and subsequent feature selection, limiting their applicability to smaller datasets and their ability to discern meaningful patterns in contemporary malware with obfuscation and polymorphic behavior. The advent of deep learning has simplified malware classification by alleviating the need for manual feature engineering.

In a study (Nataraj et al., 2011), a novel approach was proposed, converting malware binaries into grayscale images for visualization and classification using image processing techniques. Subsequently, various deep learning models have been explored, with convolutional neural networks (CNNs) initially designed for image processing finding frequent application in malware classification. For instance, Llauradó et al. (2018) presented a method utilizing CNNs to extract local and invariant features from malware files, representing them as images.

Transfer learning on pretrained models has also been employed in malware classification with introduction of an inception V3 approach using transfer learning for this purpose (Ahmed et al., 2022). More recently, researchers have ventured into addressing malware classification tasks through Natural Language Processing (NLP) approaches. A hybrid static classifier integrating CNN and bidirectional LSTM, where CNN automates feature selection and extraction was proposed (Abdullah et al., 2023). Extracted features are then passed to the bidirectional LSTM to capture long-term dependencies and sequential information in the malware data. Recently, the trend of using transformer architecture for malware classification has also risen. In a recent study (Bavishi & Modi, 2024), it has been shown that the use of a CNN along with a lightweight Vision Transformer yielded performance almost as equal to the state-of-the art models but with faster inference speed.

This present work introduces a hybrid technique that combines a feature extractor based on Convolutional Neural Network (CNN) and Multi-Layer Perceptron (MLP). The proposed method involves the automatic extraction of certain features using CNN, and some are extracted manually using feature engineering process. Both sets of features are combined to enhance the classification ability. In this approach, malware bytes are transformed into malware images, and malware metadata is extracted from .asm files through meticulous feature engineering. The CNN processes the malware images, while the MLP handles the metadata. Subsequently, the feature maps from both layers are concatenated, and the combined result is passed to the classification layers comprising MLP and a Softmax layer.

The main contributions of this study are summarized as follows:

1. Implementation of manual feature engineering process by using malware .asm file.
2. Conversion of raw malware binary into 2D array, depicting pixel intensities to visualize malware as image.
3. Use of Convolution Neural Network for automatic feature extraction using the image representation of malware.
4. Implementation of bi-modal architecture, creating hybrid feature map by using both manual and automatic feature extraction in order to create a robust malware classifier.

2. Related Works

2.1. Static Analysis

In a study (Llauradó et al., 2018), a malware classification technique was suggested, employing convolutional neural networks (CNNs) to capture local and invariant features from malware files by transforming them into image representations. The research noted that visually, images of executable software within a specific family exhibited similarities and were visually distinct from those of other fami-

lies. K-fold cross-validation was employed to gauge generalization performance, resulting in accuracies of 97.3% and 97.5% for 5-fold and 10-fold cross-validation, respectively.

Le et al. (2018) presented a Deep Learning-based malware classification approach that requires no expert domain knowledge and complex feature engineering process and is based on a purely data-driven approach for pattern and feature identification. The authors proposed a one-dimensional representation of the raw malware binary file. Furthermore, they applied recurrent neural network layers, LSTM, on top of the convolution layers before feeding the output of the recurrent layer to the output layer. The idea behind this approach was to capture dependencies between different pieces of code in a binary file. Finally, they evaluated CNN, CNN-UniLSTM and CNN-BiLSTM configurations, highlighting the efficiency of their final model, CNN-BiLSTM which achieved an accuracy of 98.2%.

In a study, Rafique et al. (2019) used a deep learning-based malware detection technique based on static methods for classifying different malware families. The technique used a simple CNN and a CNN autoencoder to extract features from the byte file and a wrapper based technique that utilizes SVM with RBF kernel to select important opcodes as features from ASM file. A hybrid feature space was generated from these features. Multilayer Perceptron (MLP) trained on this feature space was used for classification of malware. It achieved a log loss of 0.09 and accuracy up to 97.6383% when tested on BIG15 dataset.

A study (Gibert et al., 2019) proposed a Hierarchical Convolutional Network (HCN) for malware classification to extract features from both mnemonic level and function level. Files contained function calls and jump instructions which transfers the control of the program into another address in memory. When representing executable files only as a sequence of instructions, we lose its hierarchical information. This approach takes into account the hierarchical structure of Portable Executable files by treating them as a sequence of sequences, where each sequence contains mnemonics describing a particular function. This approach obtained a log loss of 0.0419 when tested on the Microsoft BIG15 dataset.

Çayır et al. (2021) proposed an ensemble method of capsule network model, RCNF based on the bootstrap aggregation technique for mitigating the challenge of imbalanced malware type classification. The authors suggest that capsule network architecture proposed in the paper reduces the complexities in feature extraction of malware images by eliminating the pooling components and creating a bagging ensemble CapsNet Model increases the performance on predicting rare malware classes. The authors have evaluated results on the BIG 2015 dataset and obtained an F1-Score of 0.9820 with an ensemble of 10 Capsule Networks.

A spatial attention and Convolutional Neural Network (SACNN) based on deep learning framework for image-based malware classification was proposed in a study (Awan et al., 2021). A frozen VGG19 model was used as feature extractor and a CNN model enhanced by attention was used for the classification. To generate attention dynamic spatial convolution was utilized. It utilized class weighting technique to balance the classes in the dataset. It utilized Malimg benchmark dataset and achieved an accuracy of 97.68% during testing.

Dang et al. (2021) proposed the use of LSTM models along with NLP techniques like word embeddings using opcode sequences extracted from disassembled executables. The paper highlights the inclusion of Convolution Neural Network (CNN) layers with word embedding and bidirectional LSTM (BiLSTM) that greatly improves the efficiency in classifying large sets of malware families. The results demonstrate that the best model which includes embedding layer, CNN Layer and BiLSTM layer was able to classify samples from 20 different malware families with an average accuracy of 81.06%.

In their study, Ahmed et al. (2022) used an inception V3 approach for malware classification using transfer learning. The approach converts byte files into image data which is passed as input to the model. For the transfer learning, the InceptionV3 pre-trained on ImageNet weights were taken. The model was trained and tested on Microsoft BIG15 dataset. The train and test accuracy of 99.6% and 98.76% respectively were obtained.

An article (Lin & Yeh, 2022) presented a method to explore informative features from one-dimensional structure of binary executables by using bit-level and byte-level 1D CNN model. It implemented 1D resizing on the byte vector obtained from malware file to get one dimensional byte sequence and applied bit expansion on it to get bit level sequence. The accuracies obtained for Malimg dataset were 98.81% and 98.7% for byte-level and bit-level approach respectively. It indicated that considering one-dimensional sequences will benefit both classification ability and computational cost.

Chen et al. (2022) proposed a method for malware classification that is based on Graph Neural Networks (GNNs). The call relationship and the potential semantics of the malware function is represented by a graph structure in order to depict the behavior of malware. GNNs aggregate the nodes on the graph and update features of specified nodes before the readout function outputs a fixed sized graph level embedding. Furthermore, they introduced a Siamese network-based similarity model to measure the distance between two samples in the feature space, determining whether they belong to the same malware family. The proposed similarity model is evaluated using the Malware Bazaar dataset, demonstrating accuracy and precision of 92% and 92.9% respectively with a test dataset.

A novel approach for malware classification using NLP was considered in a recent work (Mehta et al., 2023). The study consider a hybrid architecture, where Hidden Markov Models(HMMs) are trained on opcode sequences whose subsequent hidden states are fed to a Random Forest (RF) classifier as feature vectors. Here, the extraction of HMM hidden state sequences is viewed as a form of feature engineering for the malware classification problem domain. This unique hybrid architecture is compared to traditional machine learning and deep learning techniques which demonstrates that the proposed HMM-RF outperforms other models on a challenging malware dataset achieving accuracy of 97.58%.

A recent research (Arrowsmith et al., 2025) proposed an innovative approach to Android malware classification by integrating multimodal deep learning techniques. They utilized a hybrid architecture where convolutional neural networks (CNNs) were employed to analyze binary images of Android applications, and graph neural networks (GNNs) were trained on function call graphs (FCGs). To combine the outputs of these models, a multilayer perceptron (MLP) meta-classifier was used for late fusion, effectively integrating predictions. This hybrid system was evaluated against traditional unimodal deep learning methods and demonstrated superior performance, achieving an accuracy of 90.6% when DenseNet (a CNN) and GIN (a GNN) models were combined. This research highlighted the advantages of leveraging complementary data modalities and late fusion strategies to enhance classification accuracy and address gaps in existing malware detection techniques.

2.2. Dynamic Analysis

Kumar et al. (2019) described an approach for malware classification which utilized a hybrid approach, static approach to classify malware files and dynamic approach to provide useful insights when malware is obfuscated. They used information from file header and section header for extracting static features. Using Cuckoo sandbox, they extracted network, API and process based dynamic features. The classifier selected for classification was random forest. The approach achieved an accuracy of 96.73% and 96.31% on packed and obfuscated malware samples, respectively.

A classification system Malscore based on probability scoring and machine learning was used in Xue et al. (2019). CNN with SPP layer is used to analyze static features and variable n-grams and machine learning is used to analyze dynamic features. The static features are grayscale images and dynamic features are native API call sequences. Both static and dynamic classifiers are trained in the training phase. During the test phase, probability scoring is used to decide whether to perform dynamic analysis or not to reduce dynamic analysis overhead. The dataset was collected from the VX Heaven website. Random forest algorithm was chosen for dynamic analysis. Malscore achieved 98.82% accuracy with recall and precision of 100% in 52 malware families.

A research work (Tang & Qian, 2019) introduced a visualization and deep learning method for malware classification. Here, the authors extract sequences of API calls using dynamic analysis and then use color mapping rules to create feature images. These feature images act as visual fingerprints for malware variants that represent malware behavior. Finally, a Convolutional Neural Network is trained in order to classify different feature images with 9 malware families, and 1000 variants in each family, achieving TPR, precision, recall and F1-Score all above 99%, while the FPR below 0.1%, showing that visualization and CNN are effective for malware classification using dynamic API call sequence.

A behavioral malware detection method based on Deep Graph Convolutional Neural Networks (DGCNNs) to learn directly from API call sequences and their associated behavioral graphs was proposed (Schranko de Oliveira & Sassi, 2019) along with a model consisting of graph Convolutional Layers, fully connected layers, and a Sigmoid Layer for binary classification. The experimental results demonstrate the effectiveness of the proposed method with an accuracy of 92. 44% and an F1 score of 92. 01% for the balanced data set. Additionally, for the imbalanced dataset, the proposed method achieves an F1-Score of 99.42%, showcasing comparable performance to LSTM networks. According to the authors, the results show that the graph structure of the API call sequences plays an essential role in the problem of malware detection.

A research (Zhang et al., 2023) introduced Mal-ASSF, which fuses the semantic and sequence features of the API calls for detecting malware of different families. The authors argue that simply mapping APIs to numerical values often ignores the semantic information of functions. To overcome this, they employ dimensionality reduction through word embedding and utilize Balts to extract behavioral features of sequential segments. Additionally, the API semantic chain of function names is transferred into a vectorized representation containing operation type encoding and operation object encoding by the embedding layer, and the subsequent semantic features are extracted by the BiLSTM layer. The fusion of these sequential and semantic features is then processed by the attention module, followed by the MLP classifier module, achieving an accuracy of 94.49%, a precision of 94.01%, a recall of 94.19%, and an F1-score of 94.02% on the test set, in accurately classifying malicious code families.

3. Dataset

The experiments were carried out using the (Alessandro Panconesi & WWW BIG - Cup Committee, 2015) dataset (Microsoft malware classification challenge, BIG 2015),

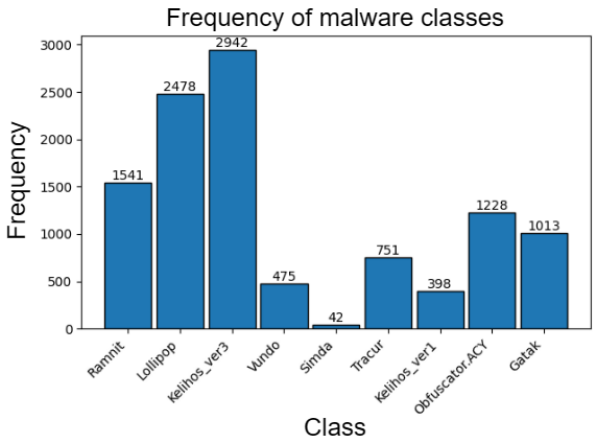

Figure 2. Distribution of malware families

which consists of 10,868 malware samples. For each malware sample, there are .byte and .asm files. ASM files contain the malware assembly code, including information related to assembly language code, function calls, and variable allocation. Similarly, Byte files consist of the hexadecimal representation of the portable executable (PE) of the malware. The dataset comprises nine different malware families:

- Ramnit: Ramnit is a versatile family of malware that consists of viruses, worms, and Trojans. They have the ability to infect EXE, DLL, and HTML files.
- Lollipop: Lollipop is a malware that is involved in online fraudulence by demanding ransom from the victim. Such malware will instruct its victim to make a fund transfer in order to reduce the damage done by the Trojan in victim's device.
- Kelihos_ver3, Kelihos_ver1: The Kelihos versions are mostly involved in spamming and phishing. Once the phishing link is clicked, various malicious activities like information theft and distribution of other malware occur.
- Vundo: Vundo, also known as Virtumonde is a large family of trojans that mainly infects windows operating system. It mainly spreads through malicious websites, corrupted software programs and infected email attachments.
- Simda: Simda is a large family of malware which is popular for countering the security of windows based system. Once installed on a machine, attacker can remotely control it to perform malicious activities, information theft, financial fraud and so on.
- Tracur: Tracur is a malignant Trojan program that redirects our web searches to a different link, most probably to a fraud advertisement link. It basically takes over our search link from the search engines and redirects to a different link, whose admin provides revenue to the malware authors.
- Obfuscator.ACY: Obfuscator is a type of malware that tends to hide its behaviour and purpose to bypass security system installed in a device. These malware may exhibit any kind of behaviours underneath its obfuscation and are relatively hard to detect.
- Gatak: Gatak is a backdoor trojan programmed to spread rapidly after the first infection. After the trojan is injected into a system, it will either alter the data present in victim's device or obstruct correct functioning of the device.

During the experiment, we used 75% of the malware samples for training and 25% for testing purposes. From the training set, 75% of the samples were used for actual training, and 25% were used for validation."

The distribution of the dataset can be illustrated in the Figure 2. From the histogram, we can see the significant imbalance in the data, which was a major challenge in our research.

4. Methodology

4.1. Malware Images from Byte Files

As the dataset consists of byte files and asm files, we have utilized byte file of each malware sample to extract images from them. Firstly, the hex sequence is read line by line and each hex value is converted into an integer ranging from 0-255. As there are significant '??' sequence in the byte file, we treated it as 0. Finally, the resulting array is converted into 2-D array with each field consisting of integers ranging from 0-255, which is treated as a malware image as shown in Figure 3.

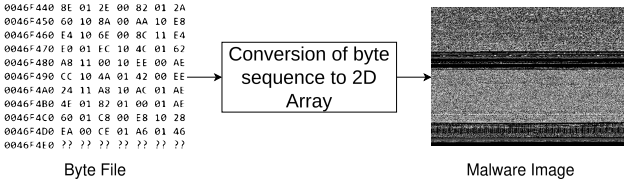

Figure 3. Conversion of byte file to malware image

4.2. Metadata of Malware from asm Files

Traditional malware classification methods use careful feature engineering processes to extract the most important malware features. For this research purpose, we have also utilized the asm file of each malware sample in order to create a hybrid feature map. From the asm file, following feature sets are extracted:

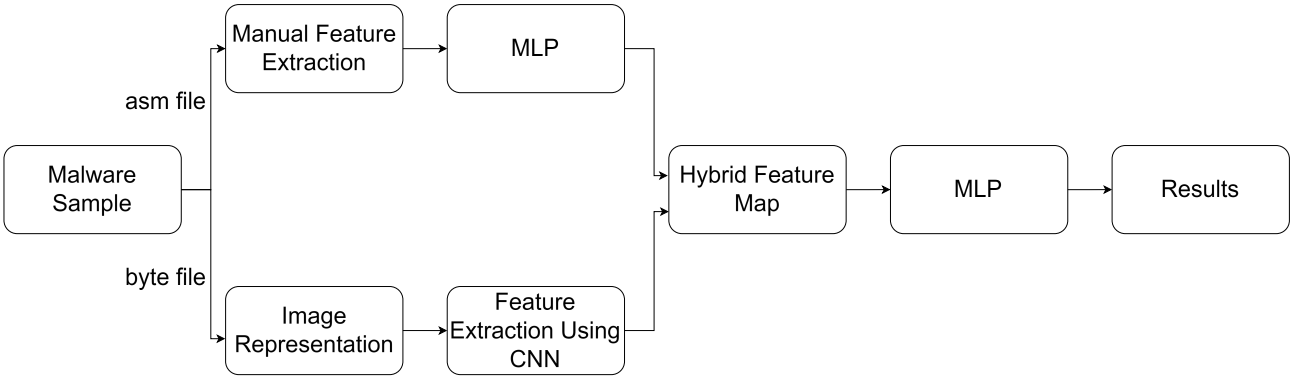

Figure 4. Block diagram of proposed method

- Size: Consists of asm file size and number of lines of code.
- 1-Gram opcode features: Consists of frequency of opcodes used in assembly code.
- Registers features: Consists of frequency of registers used in assembly code.
- Data define directives features: Consists of memory allocation, data structures definition, and initial memory contents features.
- Symbols frequency features: Consists of frequency of symbols used in assembly code.
- Section features: Consists of sectional features of assembly code like .text, .data, .bss, .reloc and so on.
- Miscellaneous features: Consists of miscellaneous features like offset, loc, import, bool, long and so on.

From the above 7-feature sets, we have extracted 277 total features for each malware sample.

4.3. Proposed Bi-Modal Architecture

After the extraction of malware images and metadata from malware samples, the metadata is processed by Multi-Layered Perceptron. At the same time, the image representation of malware is processed by CNN layers. The output of both layers is then used to create a hybrid feature map which is then used by the classification layer. Figure 4 shows the block diagram of our proposed multi-modal approach.

Our proposed model uses both malware images, extracted from byte files and malware metadata, extracted from asm files. A 3 layered CNN architecture is used for image input whereas a 2 layered Dense Network is used for metadata input. The output of last CNN layer is flattened and concatenated with the output of last dense layer before passing to a 2 layered Dense Network. Finally, its output is passed to the Softmax Layer which categorizes malware into 9 families. The architecture of our proposed model is shown in Figure 5.

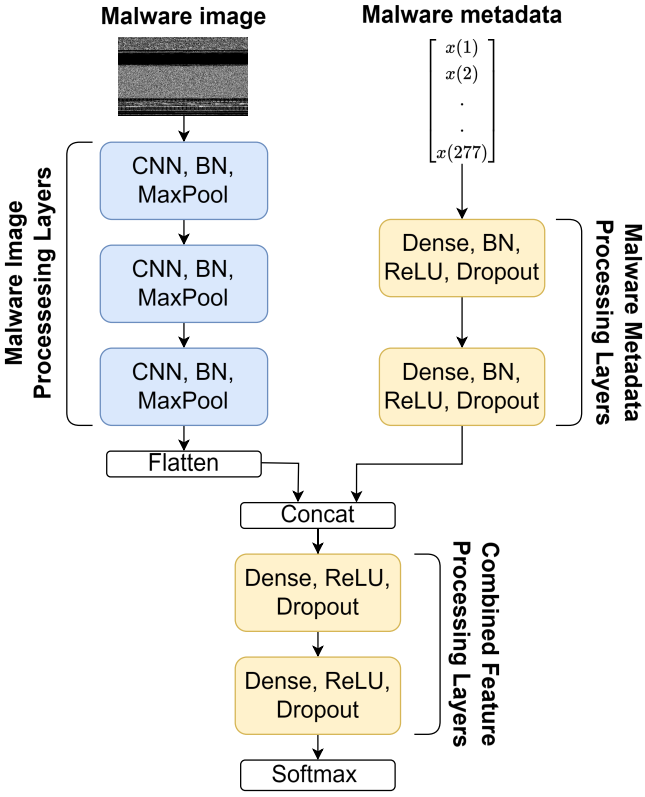

Figure 5. Bi-Modal architecture for malware classification

Since the malware images are of varying shapes, we first resized the malware images to 100x100 pixels. The detailed architecture of our proposed model is explained below:

4.3.1 Malware Image Processing Layer

A 3-layered CNN architecture is used for processing the pixel data of malware images (Table 1 which consist of fol-

lowing layers:

- Convolution layer:
  - The first Convolutional Layer comprises 32 filters, each having a kernel size of 3x3 with activation function as Rectified Linear Unit (ReLU). It uses the 'valid' padding, which means no zero-padding is added to the input images.
  - The second Convolutional Layer consists of 64 filters and the same kernel size and padding. It operates on the output of the previous layer.
  - The third Convolutional Layer consists of 128 filters and the same kernel size and padding.

Table 1. CNN layers for malware image processing

Conv. Layer	Filter	Kernel Size	Padding
CNN1	32	3x3	valid
CNN2	64	3x3	valid
CNN3	128	3x3	valid

- Batch Normalization Layer:
  - Batch Normalization is applied after each Convolutional Layer. It helps stabilize and accelerate the training of deep neural networks.
- Max Pooling Layer:
  - After each normalization block, Max Pooling is applied to reduce the spatial dimensions of the feature maps. The pooling size is 2x2, and it uses a stride of 2, which means the pooling window moves by 2 pixels in both the horizontal and vertical directions.
  - There are three Max Pooling Layers corresponding to the three Convolutional blocks each with same specifications.

4.3.2 Malware Metadata Processing Layer

Before passing the metadata to the dense layer, we first normalized the metadata using Z-Score normalization, in which values are concentrated around mean with the standard deviation of one. We then passed normalized data to a 2-layered Dense Network consisting of following layers:

- Dense Layer:
  - The first Dense Layer consists of 128 neurons (units) and a ReLU activation function. It operates on the normalized metadata.
  - The second Dense Layer consists of similar architecture as that of first dense layer.
- Batch Normalization Layer:
  - Batch Normalization is applied after every Dense Layer. This helps stabilize and accelerate the training of Deep Neural Networks by normalizing the inputs to each layer.
- Dropout Layer:
  - After both normalization layers, a Dropout Layer (with p=0.2) is applied. Dropout is a regularization technique that randomly sets a fraction (in this case, 20%) of input units to zero during training. This helps prevent overfitting by introducing some level of noise and promoting robustness.

4.3.3 Combined features processing layer

The output of last CNN Layer is flattened and concatenated with last Dense Layer (Table 2 of metadata processing layer, which is then passed to the combined feature processing layer. This network consists of following layers:

- Dense Layer:
  - The first Dense Layer consists of 128 neurons and ReLU activation function.
  - The second Dense Layer consists of 64 neurons and ReLU activation function.

Table 2. Dense layers for combined feature processing

Dense Layer	Neurons	Activation
Dense3	128	ReLU
Dense4	64	ReLU

- Dropout Layer:
  - Dropout Layer is added after every Dense Layer with a dropout rate of 30% or 0.3
- Output Layer:
  - The final layer is a dense layer with 9 neurons and a Softmax activation function. Softmax is used for converting the network's raw output into probability values for 9 malware classes.

The detailed architecture of our proposed model can be shown in Figure 6.

5. Result and discussion

We used train data and validation data for the experimentation whereas test data to evaluate the model. During the experimentation, we mainly considered accuracy as our evaluation metric. Our focus was to increase validation and

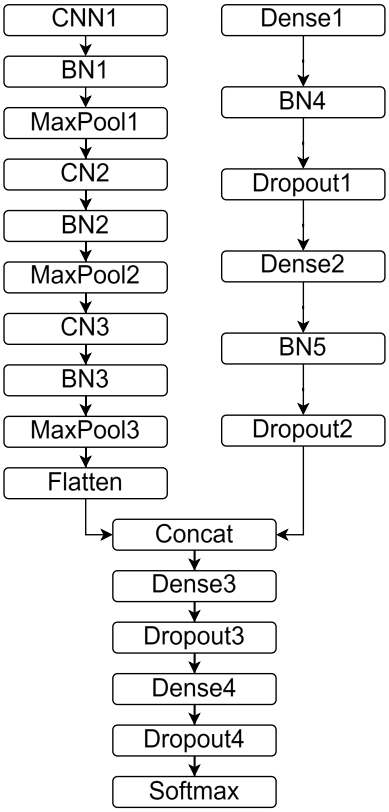

Figure 6. Bi-Modal architecture for malware classification

test accuracy as much as possible so that our model could effectively classify unseen malware families. Due to the reason that our dataset was quite imbalanced, we considered precision as another evaluation metric since, for an imbalanced dataset we cannot just rely upon accuracy, Furthermore, we also analyzed recall and f1-score of our model.

The first experimentation was performed with fixed learning rate of 0.01 throughout 55 epochs with optimizer as 'adam'. The results of which are shown in Table 3 and Figure 7.

Table 3. Evaluation of proposed method with fixed learning rate

Metrics	Score
Accuracy	97.67%
Precision	97.58%
Recall	97.60%
F1-Score	97.54%

Secondly, we experimented with learning rate scheduling where the learning rate varies as the epoch increases. We used following formula in order to vary the learning rate:

$$\text{lr} = \text{initial\_lr} \times \text{drop}^{\lfloor (1+\text{epoch})/\text{epochs\_drop} \rfloor}$$

where:

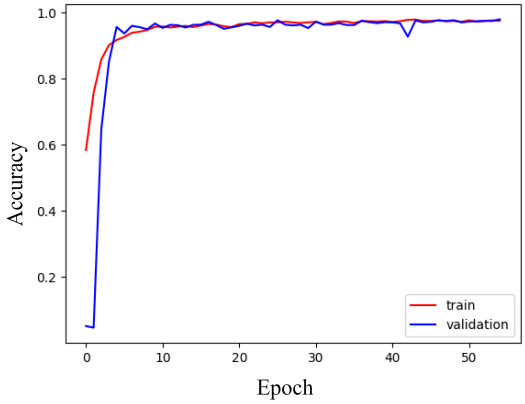

Figure 7. Accuracy curve of proposed method with fixed learning rate

- **lr:** Learning rate.
- **initial_lr:** Initial learning rate.
- **drop:** Factor by which the learning rate is reduced.
- **epoch:** Current epoch.
- **epochs_drop:** Number of epochs after which the learning rate is reduced.

Table 4. Evaluation of proposed method with learning rate scheduling

Metrics	Score
Accuracy	98.62%
Precision	98.64%
Recall	98.60%
F1-Score	98.63%

We used initial_lr as 0.001, drop as 0.75 and epochs_drop as 5 so that for every 5 epochs the learning rate drops by a factor of 0.75. The results of varying learning rate with similar other hyperparameters as that of first experiment are shown in Table 4 and Figure 8.

We were able to increase accuracy from 97.67% to 98.62% by using learning rate scheduler to our proposed method. Similarly, Precision, Recall and F1-Score also increased by around 1%.

During the experiment we observed that combining inputs from different modalities enabled the model to be more robust to imbalanced data, allowing model to generalize better in different situations even when the distribution of malware classes fluctuates. Our results indicate that a multimodal approach enables a classifier to a more complete understanding of malware families. Comparison of the proposed technique with existing state-of-the-art methods in Table 5 shows that the multi-modal approach is more stable and efficient in malware classification, indicating a promising future for upcoming research.

Table 5. Comparison to previous works

Techniques	Accuracy	F1-Score
CNN (Llauradó et al., 2018)	97.50%	94.00%
CNN+BiLSTM (Le et al., 2018)	98.20%	96.05%
CNN+SVM+MLP (Rafique et al., 2019)	97.55%	-
VGG16+CNN+Attention (Awan et al., 2021)	97.62%	97.20%
CNN+BiLSTM (Dang et al., 2021)	81.06%	-
HMM+RF (Mehta et al., 2023)	97.58%	97.32%
CNN+Vision Transformer (Bavishi & Modi, 2024)	96.60%	-
Bi-Modal CNN+MLP (Present work)	**98.62%**	**98.63%**

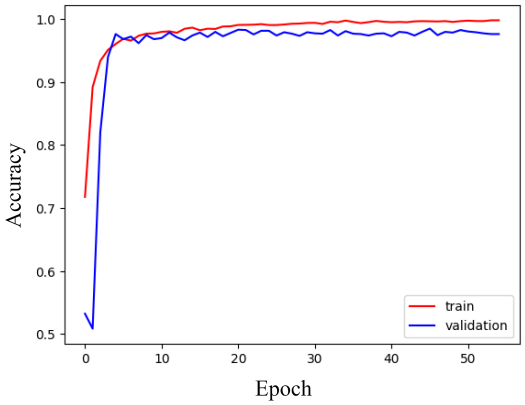

Figure 8. Accuracy curve of proposed method with learning rate scheduling

6. Conclusion and Future Work

In this paper, we present a novel method of malware classification that incorporates a bimodal approach which learns recurring malware trends and patterns through multiple information sources. The malware images extracted from byte files are fed to a 3-layered CNN and the malware metadata extracted from asm files are fed to a 2-layered dense network. The output of both is then fed to a classification layer. The experimental results demonstrate the effectiveness of our proposed method than state-of-the-art method in effectively classifying malware families. The results suggest that underlying malware trends and patterns can be easily learned using a multi-modal approach that incorporates learning from multiple information sources.

There are several avenues in terms of future work for malware classification. Our dataset consists of 10,868 malware samples which exhibit a high level of imbalance. Working with larger and balanced data sample is always beneficial to create efficient and robust classifiers. Additionally, since malware samples consist of long-term dependencies and sequential information, LSTM and BiLSTM layers could be tried within the same bi-modal architecture. While our approach still achieves remarkable results, future work could also be done using NLP approach. In general, NLP is still in a developing phase and its potential could hugely impact malware classification problems.

References

Abdullah, M. A., Yu, Y., Adu, K., Imrana, Y., Wang, X., & Cai, J. (2023). HCL-Classifier: CNN and LSTM based hybrid malware classifier for Internet of Things (IoT). *Future Generation Computer Systems*, *142*, 41–58. https://doi.org/https://doi.org/10.1016/j.future.2022.12.034

Ahmed, M., Afreen, N., Ahmed, M., Sameer, M., & Ahamed, J. (2022). An inception V3 approach for malware classification using machine learning and transfer learning. *International Journal of Intelligent Networks*, *4*, 11–18. https://doi.org/https://doi.org/10.1016/j.ijin.2022.11.005

Alessandro Panconesi, W. C., Marian, & WWW BIG - Cup Committee. (2015). Microsoft malware classification challenge (big 2015). https://kaggle.com/competitions/malware-classification

Arrowsmith, J., Davis, E., & Chen, M. (2025). Multimodal deep learning ensembles for android malware classification. *Journal of Cybersecurity Engineering*, *12*(4), 543–563. https://doi.org/10.1234/jsce.v12i4.12345

Awan, M. J., Masood, O. A., Mohammed, M. A., Yasin, A., Zain, A. M., Damaševičius, R., & Abdulkareem, K. H. (2021). Image-based malware classification using vgg19 network and spatial convolutional attention. *Electronics*, *10*(19). https://doi.org/10.3390/electronics10192444

Bavishi, S., & Modi, S. (2024). Accelerating malware classification: A vision transformer solution. https://arxiv.org/abs/2409.19461

Çayır, A., Ünal, U., & Dağ, H. (2021). Random capsnet forest model for imbalanced malware type classification task. *Computers & Security*, *102*, 102133. https://doi.org/https://doi.org/10.1016/j.cose.2020.102133

Chen, Y.-H., Chen, J.-L., & Deng, R.-F. (2022). Similarity-based malware classification using graph neural

networks. *Applied Sciences*, *12*(21), 10837. https://doi.org/10.3390/app122110837

Dang, D., Troia, F. D., & Stamp, M. (2021). Malware classification using long short-term memory models.

Gibert, D., Mateu, C., & Planes, J. (2019). A hierarchical convolutional neural network for malware classification. *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN.2019.8852469

Kumar, N., Mukhopadhyay, S., Gupta, M., Handa, A., & K. Shukla, S. (2019). Malware classification using early stage behavioral analysis. *2019 14th Asia Joint Conference on Information Security (AsiaJCIS)*, 16–23. https://doi.org/10.1109/AsiaJCIS.2019.00-10

Le, Q., Boydell, O., Mac Namee, B., & Scanlon, M. (2018). Deep learning at the shallow end: Malware classification for non-domain experts. *Digital Investigation*, *26*, S118–S126. https://doi.org/10.1016/j.diin.2018.04.024

Lin, W.-C., & Yeh, Y.-R. (2022). Efficient malware classification by binary sequences with one-dimensional convolutional neural networks. *Mathematics*, *10*(4). https://doi.org/10.3390/math10040608

Llauradó, D. G., Mateu, C., Planes, J., & Vicens, R. (2018). Using convolutional neural networks for classification of malware represented as images. *Journal of Computer Virology and Hacking Techniques*, *15*, 15–28. https://api.semanticscholar.org/CorpusID:52273131

Mehta, R., Jurečková, O., & Stamp, M. (2023). A natural language processing approach to malware classification. *Journal of Computer Virology and Hacking Techniques*. https://doi.org/10.1007/s11416-023-00506-w

Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011). Malware images: Visualization and automatic classification. *Proceedings of the 8th International Symposium on Visualization for Cyber Security*. https://doi.org/10.1145/2016904.2016908

Petrosyan, A. (2023a). Number of malware attacks per year 2015-2022. *Statista*. https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/

Petrosyan, A. (2023b). Annual share of organizations affected by ransomware attacks worldwide from 2018 to 2023. *Statista*. https://www.statista.com/statistics/204457/businesses-ransomware-attack-rate/

Rafique, M. F., Ali, M., Qureshi, A. S., Khan, A., & Mirza, A. M. (2019). Malware classification using deep learning based feature extraction and wrapper based feature selection technique. *ArXiv*, *abs/1910.10958*. https://api.semanticscholar.org/CorpusID:204851828

Schranko de Oliveira, A., & Sassi, R. J. (2019). Behavioral malware detection using deep graph convolutional neural networks. https://doi.org/10.36227/techrxiv.10043099.v1

Tang, M., & Qian, Q. (2019). Dynamic api call sequence visualisation for malware classification. *IET Information Security*, *13*(4), 367–377. https://doi.org/10.1049/iet-ifs.2018.5268

Xue, D., Li, J., Lv, T., Wu, W., & Wang, J. (2019). Malware classification using probability scoring and machine learning. *IEEE Access*, *7*, 91641–91656. https://doi.org/10.1109/ACCESS.2019.2927552

Zhang, S., Wu, J., Zhang, M., & Yang, W. (2023). Dynamic malware analysis based on api sequence semantic fusion. *Applied Sciences*, *13*(11), 6526. https://doi.org/10.3390/app13116526