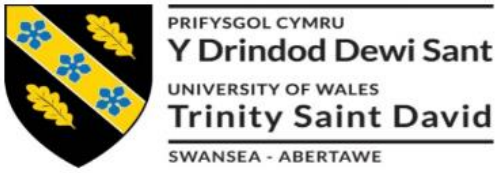

A malware classification method based on the multi-layer feature fusion of malware image representations and opcode Markov images

by

Jinwei Xu

Supervisor: Dr Michael Dacey

Project submitted as part of the requirements for the award of MSc Computer Networks and Cyber Security

August 2024

Declaration of Originality

I Jinwei Xu declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : .....

Date : .....

FORM OF CONSENT

I Jinwei Xu hereby consent that my Project, submitted in candidature for the degree MSC Computer Networks and Cyber Security, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : .....

Date : .....

Abstract

As the threat of malware to information security becomes increasingly severe, the study of efficient malware classification methods has become more urgent. This paper proposes a multilayer malware classification method based on the fusion of image representation and opcode features. By integrating the features of image-based malware representation and opcode Markov image, the classification performance is enhanced. Specifically, our model introduces two feature fusion modules: a cross-attention-based fusion module and a multiplication-based low-level feature fusion module. These modules achieve effective fusion of deep features, thereby improving the accuracy of malware classification. Experimental results show that the model combining malware image features extracted by ResNet18 and malware opcode Markov image features extracted by EfficientNetB0 performs the best. Comparative experiments with traditional feature fusion methods demonstrate that our approach has significant advantages in classification performance. Further experiments on another dataset validate the generalization ability of our method. This research provides an efficient and effective solution for malware classification.

Table of Contents

List of Figures

List of Tables

Acknowledgment

I would like to sincerely thank my advisor, Michael Dacey, for his guidance on my academic and research journey. I am also deeply grateful to my family for their encouragement and support throughout this process. Additionally, I want to express my appreciation to my friends for their assistance in my studies. Without your help, I would not have achieved the results I have today. Thank you all!

1. Introduction

1.1 Introduction

In recent years, with the rapid development of the internet industry, an increasing number of devices have connected to the internet, significantly transforming people's lifestyles and work habits. However, alongside the broad application of new technologies, information security issues have become increasingly prominent. Cybercriminals employ various methods to carry out attacks, causing severe damage to both individuals and businesses. Among these, malware attacks are one of the primary threats to current information security. Malware can spread through various channels, such as email attachments, malicious website links, and infected applications. If a user inadvertently clicks or downloads these, malware may be implanted in the device, leading to the theft of sensitive information, locking of system files, or even ransom demands. These attacks not only threaten users' privacy and security but also pose the risk of data breaches and business disruptions for enterprises, resulting in immeasurable losses.

Malware is not only increasing rapidly in quantity, but its complexity and diversity are also continuously evolving. As cyber defense mechanisms advance, attackers have adopted more sophisticated techniques to bypass security measures. By continuously improving and updating malware code, they make it more stealthy, destructive, and harder to detect and defend against. This rapid evolution poses significant challenges to the cybersecurity field. According to the SonicWall 2024 Mid-Year Threat Report[1], the total amount of malware increased by 30% in the first half of 2024, a significant rise compared to the same period last year. The growth was particularly notable between March and May, with a 92% increase in May alone. According to the

VirusTotal Malware Trends Report[2], malware is increasingly utilizing newer and more covert distribution methods. The use of traditional file formats such as Excel, RTF, CAB, and compressed files in malware distribution is gradually decreasing, being replaced by emerging file types and distribution methods. In 2023, OneNote files and JavaScript distributed through HTML quickly became mainstream distribution mediums.

In recent years, deep learning technology has made significant advancements across various fields, demonstrating its powerful capabilities in handling complex data and tasks. Particularly in the domain of malware classification, deep learning has emerged as a highly regarded and widely applied approach.

The image-based representation of malware has become a significant approach in the field of deep learning-based malware classification. This method involves converting malware bytecode into images, enabling deep learning models to automatically learn and extract useful features, thereby enhancing the accuracy and efficiency of malware detection. The image-based representation of malware refers to the process of converting malware bytecode data into grayscale or color images. This approach offers several advantages. Firstly, The image-based approach allows deep learning models to automatically learn and extract features by transforming malware bytecode into images, thereby reducing the reliance on manual feature engineering. Secondly, The image-based method effectively addresses code obfuscation by distinguishing malware from legitimate software through visual features, without depending on specific code structures or patterns. Thirdly, The image-based method is highly cross-platform adaptable, enabling unified application across different operating systems without the need to develop separate detection algorithms for each platform.

The Markov image of malware opcodes is another commonly used feature for malware classification based on deep learning methods. The use of the Markov image of malware opcodes is justified because opcodes represent the sequence of instructions that an application executes during its runtime, reflecting the application's low-level operations. By analyzing these opcode sequences, it is possible to capture the behavioral characteristics of the application, which is crucial for distinguishing different types of malware. Moreover, compared to higher-level features in the code (such as API calls), opcodes are closer to machine code and are less affected by techniques such as code obfuscation and compression[3]. Therefore, extracting features from opcodes enhances the capability to detect malware, particularly when dealing with obfuscated malware.

Traditional classification methods typically rely on the analysis of a single feature, such as image-based malware representation or opcode Markov image analysis. However, as malware technology continues to evolve, these single-feature analysis methods face certain limitations when dealing with complex and diverse malware. Therefore, it is particularly necessary and promising to propose a joint analysis method that combines malware's image-based presentation with opcode Markov images. The image-based presentation of malware intuitively displays the overall structural characteristics of the malware, making patterns and features that are difficult to detect in binary data more apparent. Meanwhile, the opcode Markov image captures subtle behavioral differences by analyzing the statistical characteristics of the malware’s instruction sequence. The combination of these two approaches not only enhances the diversity of malware feature extraction but also improves the robustness and accuracy of detection. However, effectively integrating these two features still presents several challenges. First, a key issue is how to fully leverage the advantages of both

approaches in feature extraction and representation. Additionally, capturing and representing the connections between image-based features and opcode Markov features to improve the classifier's ability to recognize malware is also a major difficulty.

In response to these challenges, this research will explore different fusion strategies with the aim of proposing a malware classification method that fully utilizes the advantages of both features.

1.2 Research Aim and Objectives

The purpose of this study is to propose a malware classification method based on a multi-level feature fusion of image-based presentations and opcode Markov images of malware. The specific objectives of this research include: conducting a comprehensive literature review to understand the current malware detection techniques using image-based presentations and opcode Markov images, as well as their developments; designing a malware classification method that incorporates multilayer feature fusion; testing the proposed method on publicly available datasets to validate its effectiveness; and evaluating the method using various metrics to comprehensively assess its performance and practicality. Through these steps, this study aims to provide a novel and effective malware classification solution for the field of cybersecurity.

1.3 Research Questions

- Which deep learning model is optimal for extracting image representations of malware and opcode Markov image features?

In the study of malware classification models, selecting an appropriate model to extract image representations of malware and opcode Markov image features is a crucial question. Therefore, this study will investigate and validate the performance of

commonly used models in extracting malware image representations and opcode Markov image features.

- Can an effective technique for malware feature fusion be developed?

Feature fusion techniques have been widely applied in the field of malware classification. However, existing research primarily focuses on simple feature fusion methods, which often overlook the complex relationships and potential complementarity between features, leading to limited improvements in classification performance. Thus, the critical question of this research is: How can a more effective feature fusion method be designed to deeply explore and utilize the relationships between features, thereby significantly improving the accuracy and robustness of malware classification? To address this issue, this study will investigate various feature fusion strategies and integrate advanced machine learning techniques, aiming to develop a new method that fully leverages the power of features.

- Can a robust and generalizable malware classification algorithm be developed?

In the field of malware classification, robustness and generalizability are key criteria for evaluating the quality of classification algorithms. Malware comes in many forms, and attack methods are constantly evolving. Therefore, an effective classification algorithm should maintain efficient and accurate classification capabilities even when faced with diverse datasets. This study will validate the proposed method's classification performance across different malware datasets to assess its adaptability and stability in various environments.

1.4 Structure of The Project

In the first chapter, Introduction, the research content is introduced, providing an overview of the study and its objectives. The second chapter, Literature Review, presents a review of related literature, discussing the background and context of the research. The third chapter, Methodology, details the methods employed in this study. The fourth chapter, Design, describes the design of a classification method based on the multi-layer feature fusion of malware image representations and opcode Markov images. The fifth chapter, Implementation, explains how the algorithm was implemented. The sixth chapter, Experiment, involves conducting experiments to answer the research questions posed in the study. The seventh chapter, Discussion, explores the answers to the research questions, comparing the findings with existing literature. Finally, the eighth chapter, Conclusion, offers conclusions and provides insights for future research directions.

2. Literature Review

2.1 Image-based Malware Representation

In the field of malware analysis, traditional approaches are increasingly facing complex challenges. To more effectively identify and classify malware, researchers have proposed an innovative method: representing malware by converting its binary data into image form. This image-based malware representation approach not only reveals the unique patterns of malware but also opens up new possibilities for utilizing computer vision and deep learning techniques in malware detection. In the following, we will review the relevant research achievements in this area.

The research by Nataraj et al.[4] was the first to visualize malware into images. The core idea of the study is to visualize the binary files of malware as grayscale images, noting that images belonging to the same malware family often exhibit similar layouts and textures. Based on this visual similarity, the research introduces a method for classification using standard image features, without the need for code disassembly or execution. The main experimental results show that this method achieved a classification accuracy of 98% in a database containing 25 different malware families. Additionally, the technique demonstrated a certain level of resilience against common obfuscation techniques, such as partial encryption. The innovation of this research lies in the use of image features for malware analysis, opening new avenues for future malware analysis based on computer vision techniques.

Tekerek et al.'s research[5] proposes an algorithm called B2IMG, which is designed to convert byte files into image format for the purpose of malware classification. The specific steps of this algorithm include reading the byte files, data processing, image generation, and image conversion. By directly converting byte data into image data,

the B2IMG algorithm avoids the information loss often encountered in traditional analysis methods, thereby improving the accuracy of malware classification. To address the issue of data imbalance, the study also introduces CycleGAN (Cycle-Consistent Generative Adversarial Network) for data augmentation. Finally, the study employs DenseNet to classify the image-based malware representations. Experimental results demonstrate that using the image data converted by the B2IMG algorithm, combined with data augmentation via CycleGAN, can significantly improve malware classification accuracy. Notably, the classification accuracy reached 99.86% on RGB images.

The study by Shaukat et al.[6] proposes an innovative malware detection method based on deep learning. The proposed method first visualizes executable files (PE files) as color images, then uses a fine-tuned model to extract deep features from these images. Finally, it employs a Support Vector Machine (SVM) to detect malware based on these deep features. Experimental results show that this method outperforms existing methods on multiple benchmark datasets, achieving an accuracy of 99.06% on the Malimg dataset. Additionally, the study introduces data augmentation techniques to address the issues of data imbalance and the scarcity of publicly available malware detection datasets, significantly enhancing detection performance.

Chaganti et al.[7] explored a malware classification method based on image representation. The study proposed using the EfficientNetB1 model for classifying malware families, leveraging byte-level image representation techniques of malware. After comparing the performance of various CNN pre-trained models, the authors found that EfficientNetB1 achieved a classification accuracy of 99% while requiring significantly fewer network parameters than other pre-trained models. Additionally, various visualization techniques were employed in the study to compare the

performance of different CNN models. The research demonstrated that EfficientNetB1 not only effectively improves accuracy in malware classification but also reduces the consumption of computational resources.

The study by Acharya et al.[8] proposes a malware classification framework based on the EfficientNet-B1 model. The malware samples in the study are represented as byte code grayscale images and classified using the EfficientNet-B1 model. The experimental results demonstrate that the model achieved a classification accuracy of 98.57% on a dataset comprising 10,868 samples from 9 different malware families, significantly outperforming other pretrained deep learning models.

The research by Yadav et al.[9] proposes using deep learning methods for automated malware detection. The research compares the performance of 26 convolutional neural network models in Android malware detection and proposes a detection method based on the EfficientNet-B4 model. This method involves converting Android's DEX files into images, extracting features from these images using the EfficientNet-B4 model, and finally performing binary classification to distinguish between malware and benign software through a Softmax classifier. Experimental results demonstrate that the proposed model achieves a 95.7% accuracy rate in the binary classification task, outperforming other comparative models.

In the study conducted by Lojain et al.[10], the core components of APK files, such as classes.dex, resources, manifest, and certificates, were utilized. These binary data were converted into 8-bit vectors and then transformed into grayscale images. These grayscale images were subsequently used to train and test the model. The study employed the ResNet-50 model, replacing its softmax classification layer with an SVM model (using a Gaussian kernel) to enhance detection performance. After conducting

experiments on the DREBIN dataset, the research results showed that the grayscale image model, which combined Certificates (CR) and Android Manifest (AM), achieved a classification accuracy of 97%. Additionally, the model performed exceptionally well on other metrics such as precision, recall, and F1-score, all exceeding 95%.

The work by Asam et al.[11] involves detecting and classifying malware variants using deep learning and machine learning techniques. The research introduces two novel malware classification frameworks: Malware Classification based on Deep Feature Space (DFS-MC) and Malware Classification based on Deep Boosted Feature Space (DBFS-MC). In the DFS-MC framework, a custom Convolutional Neural Network (CNN) architecture is employed to generate deep features, which are then input into a Support Vector Machine (SVM) algorithm for malware classification. In the DBFS-MC framework, an enhanced feature space is generated by combining the deep feature spaces of two custom CNN architectures, aiming to improve classification discrimination. On the MalImg malware dataset, an accuracy of 98.61% was achieved.

The study by Ahmed et al.[12] proposed the use of a transfer learning approach with the Inception V3 model to classify malware samples from the BIG15 dataset. The research also compared the performance of several other machine learning and deep learning models, including Logistic Regression (LR), Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Long Short-Term Memory networks (LSTM). In the experiments, the transfer learning approach using the Inception V3 model achieved a classification accuracy of 98.76% on the test dataset, while the accuracy on the training dataset reached 99.6%.

The research by Mallik et al.[13] proposes a convolutional recurrent-based malware classification technique that leverages visual recurrent features in grayscale malware images for classification. Initially, the malware samples are converted into grayscale images, and convolutional neural networks (CNNs) are used to extract structural similarity features. To balance the dataset and reduce class bias, data augmentation is applied. Subsequently, visual features are extracted using the VGG16 feature extractor, and these features are processed through two stacked Bidirectional Long Short-Term Memory (BiLSTM) layers. Finally, the processed features are fused for the final malware family classification. The authors tested the model's performance on two benchmark datasets, demonstrating that this approach is both practical and effective for malware family classification.

Currently, classification methods based on image-based representation of malware have become a common and effective technical approach. These methods typically incorporate convolutional neural networks (CNNs) such as VGG, ResNet, EfficientNet, and other models, demonstrating outstanding classification performance. However, selecting the appropriate convolutional neural network model remains a topic that requires further research. Additionally, solely relying on image-based representation techniques may sometimes fail to fully realize their potential. It might be necessary to combine them with other analytical methods or more sophisticated feature extraction and fusion techniques to further enhance classification accuracy and robustness.

2.2 Opcode

Opcode, short for operation code, is a portion of a machine language instruction that specifies the operation to be performed. While obtained from a decompiled .asm file, opcodes provide a detailed view of the low-level instructions executed by a program. Using opcodes for malware classification offers several benefits. They reveal the specific behaviors and operational details of the malware, allowing for precise identification of its functions and intents. This granularity aids in distinguishing between different types of malware and understanding their underlying mechanisms.

The research by Singh et al.[3] develops and evaluates a multimodal deep learning framework called SHIELD, designed for detecting malware within Android systems. The framework integrates opcode Markov images and dynamic API calls, utilizing a Multimodal Autoencoder (MAE) to minimize the reliance on feature engineering and to autonomously discover relevant features for malware detection. SHIELD demonstrated strong performance on two benchmark datasets, CICandMal2020 and AMD, achieving detection rates of 94% and 87%, respectively.

The study by Deng et al.[14] aims to enhance the effectiveness of malware detection through a novel three-channel visualization approach. The deep learning model employed in this research includes a Convolutional Neural Network (CNN) for feature extraction and classification. The study utilizes a publicly available malware dataset from Microsoft, which contains multiple malware families, to evaluate the effectiveness of the proposed method. The feature extraction techniques involve generating images from assembly instructions and creating three distinct channels using Markov transition matrices, which retain the essential information required for malware classification. The three channels correspond to Letter Markov Images, Opcode Initial Markov Images, and Opcode Markov Images. The research findings indicate that

MCTVD exhibits an extremely high accuracy rate (99.44%) in malware classification, along with significant precision, recall, and F1 scores, demonstrating the effective integration of multi-channel data.

The study by Gao et al.[15] proposes an anti-obfuscation Android malware analysis method named CorDroid. The authors propose a method that combines various features to counteract code obfuscation, and they develop CorDroid based on two new features: the Enhanced Sensitive Function Call Graph (E-SFCG) and the Opcode-based Markov Transition Matrix (OMM). E-SFCG describes the relationships between sensitive function calls, while OMM reflects the transition probabilities between opcodes. The authors validate the complementarity of E-SFCG and OMM in the face of different obfuscation techniques through experiments and demonstrate CorDroid's high execution efficiency. Experimental results show that algorithm outperforms existing detection methods.

The study by Zhao et al.[16] proposes a deep learning-based method for classifying malware families through visualization techniques. By converting binary files into images and utilizing the texture features within these images for clustering, the researchers employed a deep convolutional neural network (CNN) to perform feature fusion and classification on Markov images generated from bytecode and opcode. Specifically, the bytes and opcodes in malware binary files were transformed into Markov images based on transition probability matrices. Experiments conducted on Microsoft's malware dataset demonstrated that the method, which fuses image features from both bytecode and opcode, achieved an accuracy of 99.76% and an F-score of 98.91%.

The research by Mai et al.[17] proposes a malware detection method based on Markov images and the MobileNet model, emphasizing the generation of Markov images from opcode sequences and the subsequent classification of these images using the lightweight MobileNet model. This method achieves good detection performance while maintaining low computational resource consumption. Experimental results indicate that classifying the generated Markov images with the MobileNet model can effectively detect malware in IoT scenarios.

The opcode Markov images have been widely applied in the field of malware classification, demonstrating exceptional performance in handling complex malware detection tasks, particularly when dealing with obfuscation techniques and unknown malware, thereby overcoming the limitations of traditional detection methods. Various convolutional neural networks, as commonly used feature extractors, have enhanced detection accuracy. However, research on combining opcode Markov images with image-based representations of malware for classification is relatively limited. Future work could further explore new feature extraction methods and model architectures to improve the system's robustness and adaptability.

2.3 Feature Fusion in Malware Classification

Feature fusion has been widely applied in the field of malware classification. By integrating different features, it effectively enhances the classification performance of models.

The paper by Chen et al.[18] proposes an innovative approach to Android malware detection by utilizing Graph Attention Networks (GAT) and the deep fusion of multimodal features. This paper introduces a novel type of call graph, named the Class-Set Call Graph (CSCG), designed to effectively extract both structural and semantic features of Android applications. Furthermore, the paper presents a feature fusion network that integrates CSCG features with permission features to enhance malware detection. In this network, features are progressively fused through a three-layer deep network. Experimental results demonstrate that this method achieves detection accuracy ranging from 97.28% to 99.54% across three constructed datasets, outperforming existing approaches.

The paper by Xuan et al.[19] proposes a malware classification method combining Bidirectional Temporal Convolutional Network (BiTCN) and Transfer Learning Atrous Spatial Pyramid Pooling EfficientNet (TAEfficientNet), named BiTCN-TAEfficientNet. This method enhances classification accuracy by fusing multiple features, utilizing malware assembly data and API sequences as features, and introducing a bidirectional temporal convolutional network to capture bidirectional temporal features. Additionally, the paper employs a fusion classifier and Quantum Particle Swarm Optimization (QPSO) algorithm to optimize BiTCN-TAEfficientNet, which further enhances the algorithm's accuracy and robustness while reducing the impact of adversarial techniques. The fusion classifier achieves feature fusion through concatenation. Experimental results show that this method achieves classification

accuracies of 99.461% and 97.92% on the Kaggle and DataCon datasets, respectively, representing improvements of 0.38% and 0.87% compared to other methods.

The study by Li et al.[20] proposes a method for classifying malware families based on multimodal fusion and weight self-learning. Firstly, the study extracts multidimensional features of malware through static analysis, including byte, format, statistical, and semantic features, which are then fused during the feature engineering phase through concatenation. In the model construction phase, a weight self-learning mechanism is introduced to automatically learn the weights of different features within each family. This approach demonstrates great classification performance on imbalanced malware datasets.

The study by Kumar et al.[21] proposes a novel architecture for malware classification based on image visualization. This approach utilizes a VGG16 model as a feature extractor, combined with three convolutional neural network models to obtain varied feature maps. The extracted features are concatenated to form a feature map, which is then trained using six classifiers. The experiments were conducted using the MalImg dataset, which contains 9,339 images from 25 families, as well as real-world packed malware to validate the method's generalization ability. The results show that the MLP model achieved an accuracy of 98.55% on the MalImg dataset and 94.78% on the real-world malware dataset.

The research of Dib et al.[22] proposes an innovative multi-dimensional deep learning framework aimed at enhancing cybersecurity by analyzing the classification of Internet of Things (IoT) malware. The research focuses on utilizing strings extracted from malware executables and image-based features. In the "feature fusion and classification" step, these features learned from different data representations are

concatenated to form a shared multimodal representation. This concatenated multimodal representation is then input into a neural network with fully connected layers for final, efficient classification. The study analyzed over 70,000 recently detected IoT malware samples, using Convolutional Neural Networks (CNN) and Long Short-Term Memory Networks (LSTM) to process image and string data, respectively. Experimental results indicate that this multi-layered deep learning framework significantly outperforms traditional single-layer classifiers in classification accuracy, achieving an accuracy rate of 99.78%.

The research of Chen et al.[23] proposes a novel method for detecting Android malware by integrating various features of Android applications. First, the paper introduces a new Class Set Call Graph (CSCG), which uses Java class sets as nodes and designs a CSCG construction method that can determine node size based on the application's scale. Then, a topic model is used to mine semantic features from the source code. Next, a Graph Attention Network (GAT) is employed to extract CSCG features. Finally, the study constructs a deep learning-based multimodal feature fusion network. This network enhances the accuracy and robustness of malware detection by concatenating CSCG features with permission features at multiple fusion points and classifying the fused features using a deep neural network model. Experimental results show that this method achieves a detection accuracy of 97.28% to 99.54% across three constructed datasets, outperforming existing methods.

Yang et al.[24] introduced a hybrid attention network model for malware detection that enhances accuracy by aligning and integrating multiple features, specifically combining binary file and opcode features. The model initially extracts temporal sequences and jump characteristics from binary files using stacked convolutional networks while employing a triangular attention algorithm to extract opcode features

from assembly code. Subsequently, a cross-attention mechanism is used to align and fuse these two distinct sources of features, resulting in more stable and representative feature representations. The literature emphasizes the crucial role of the cross-attention mechanism in this process, as it establishes deep connections between different modal features, enabling the model to better understand and learn the relationships between binary files and assembly code, thereby significantly improving malware detection performance. Experimental results demonstrate that this multi-feature fusion strategy, based on mutual attention, outperforms existing benchmark methods across multiple datasets, showcasing its advantages and effectiveness in malware detection tasks.

The work by Snow et al.[25] proposes an end-to-end multi-model deep learning framework aimed at directly extracting features from malware data to enhance classification accuracy and generalization ability. The model integrates three distinct deep neural network architectures to process different attributes of malware data. The model concatenates various features and then classifies them using a Multi-Layer Perceptron (MLP). Experimental results demonstrate that the proposed model excels in both classification accuracy and training time. The model achieves an average classification accuracy of 98.35% in 4-fold cross-validation, with a best classification accuracy of 99.23%, and its training time is lower compared to other methods.

Currently, in the field of malware classification research, feature fusion methods have been widely applied. These methods significantly enhance classification accuracy by extracting and integrating structural features, semantic features, permission features, and image-based features. However, the current feature fusion methods still primarily rely on traditional approaches such as concatenation and addition, which to some extent limit the potential of feature fusion technology in malware classification.

Moreover, the classification methods that integrate malware image features with opcode Markov image features have not been sufficiently explored, and further research and optimization are required.

2.4 Research Gap

Although image-based malware classification techniques have shown outstanding performance, many studies still rely on a single feature representation, such as images generated only from bytecode or opcode. These methods perform well on specific datasets; however, they exhibit significant limitations when dealing with data imbalance, adversarial attacks, and unknown malware variants. Therefore, research on combining image representations of malware with Opcode Markov images for classification is not only necessary but also has great potential.

Most current feature fusion methods rely on traditional operations like concatenation and addition, failing to fully exploit and utilize the potential of multiple features. Therefore, proposing a more effective feature fusion method is crucial for improving malware classification performance.

In research on classification based on malware image representations and Opcode Markov images, convolutional neural networks are commonly used as feature extraction tools. However, the actual performance of different convolutional neural networks in malware classification still requires further in-depth study.

Moreover, the robustness of malware classification algorithms across different datasets is equally critical and urgently needs further validation and improvement.

3. Research Methodology

3.1 Research Philosophy

The philosophical foundation of this research is positivism. Positivism emphasizes the validation of hypotheses through objective data, which aligns with the experimental approach of classifying malware based on real-world data in this study.

3.2 Research Process

Figure 2 illustrates our research process, which comprises five key steps: data collection, data preprocessing, model design, model training, and model evaluation. These steps form the core of our research methodology.

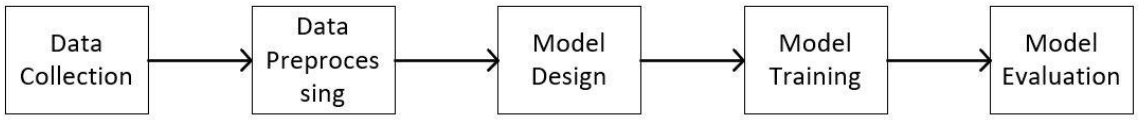

Figure 2-Research Process

Data Collection: In the research of malware classification, collecting a high-quality malware dataset is crucial. Common methods for data collection include honeypots, third-party sharing sites, and open-source datasets. Each of these methods has its own advantages and disadvantages.

Data Preprocessing: In deep learning-based malware classification research, data preprocessing is essential, as this step involves extracting features from malware data that can be processed by deep learning algorithms. In our research, we utilized two types of features: image based malware representations and malware opcode Markov images.

Model Design: This study employs a fusion of malware images and malware opcode Markov images for malware classification. Therefore, it is necessary to select an appropriate deep learning algorithm to extract features from these two types of images. Additionally, a feature fusion method needs to be designed to effectively combine these features.

Model Training: Model training is the process by which the model learns from the data and updates its parameters. This includes forward propagation, loss calculation, and backpropagation. Furthermore, training a deep learning model requires setting a loss function and an optimization algorithm. The loss function measures the accuracy of the model's predictions, while the optimization algorithm helps the model update its parameters.

Model Evaluation: After completing model training, it is necessary to evaluate its performance in the malware classification task. Common evaluation metrics include accuracy, precision, recall, F1-score, and confusion matrix.

3.3 Data Collection

In malware research, collecting high-quality datasets is a crucial step in the study. Currently, commonly used methods for collecting malware datasets include honeypots, third-party malware sharing websites, and open-source datasets.

3.3.1 Honeypot

Honeypot technology is a widely used method for collecting malware, designed to deceive attackers to capture malicious behavior[26]. This is achieved by configuring vulnerable network services on certain decoy hosts to attract and capture attack behaviors. Honeypots are categorized into low-interaction[26] and high-interaction honeypots[27].

3.3.2 Third-party malware sample sharing websites

Third-party malware sample sharing websites are another commonly used method for collecting malware. Users can upload and download various malware samples for research and analysis purposes.

3.3.3 Open-source datasets

Open-source datasets are an important resource frequently used in malware research. Open-source malware datasets often contain a large number of labeled malware samples, providing convenience for researchers.

3.4 Data Pre-processing

3.4.1 Image-based Malware Representations Extraction

Nataraj et al.[4]were the first to propose a method for mapping malware into images and utilizing these images for malware classification. In this method, the binary file of the malware is first read into a one-dimensional array of 8-bit unsigned integers. This one-dimensional array is then reshaped into a two-dimensional array, generating the corresponding grayscale image. The width of the image is adaptively adjusted based on the file size: smaller files result in narrower image widths, while larger files correspond to wider image widths. The processing workflow is illustrated in Figure 1,

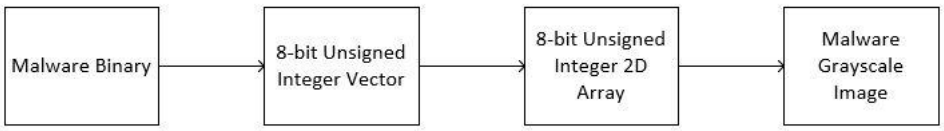

Figure 1 Malware visualization algorithm by Nataraj et al.

Tekerek et al.[5] proposed an algorithm for mapping malware into grayscale and color images. Unlike the method introduced by Nataraj et al., this algorithm is capable of

generating color images with richer textures and organizing them into square images that are better suited for deep learning processing.

The image generation process, whether for grayscale or color images, shares the following common steps: First, the binary file of the malware, represented as hexadecimal characters, is read. Then, based on the predetermined image type, the dimensions of the corresponding image matrix are calculated. Subsequently, the malware's numeric data is populated into the matrix to generate the corresponding image.

The key difference lies in the fact that grayscale images use only a single channel to represent pixel intensity, whereas color images utilize multiple channels, thereby capturing more complex textures. Additionally, to enhance the feature representation of the images, the method proposed by Tekerek et al.[5] specifically excludes meaningless zero values, thereby optimizing the image generation process. The processing workflow is illustrated in the accompanying Figure 2.

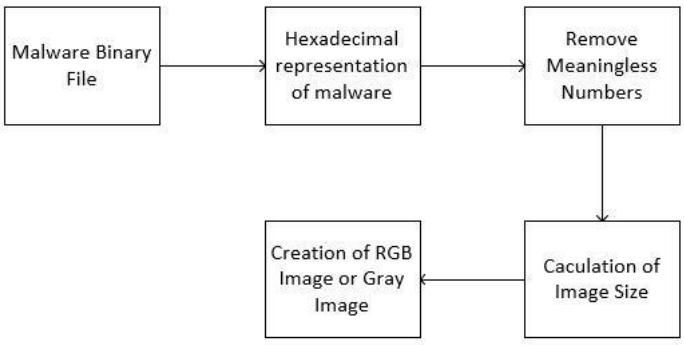

Figure 2 Malware visualization algorithm by Tekerak et al.

3.4.2 Malware Opcode Markov Image Extraction

The Markov image of malware is a method used to represent and analyze malware characteristics by converting the statistical features of malware opcode sequences into images. This approach visualises the transition probability matrix of byte pairs (or opcode pairs) in the malware as an image, thereby capturing its statistical properties. This image format can be used as input for deep learning models for malware detection and classification.

The generation of Markov images is based on Markov chain theory[17]. A Markov chain assumes that the future state of a system depends only on its current state, independent of previous states. In the context of malware analysis, this implies that the occurrence probability of an opcode depends solely on the preceding opcode. By calculating the transition frequencies of all adjacent opcode pairs within the entire malware sample, a transition probability matrix can be constructed. Each element of this matrix represents the probability of transitioning from one opcode to another. Finally, by visualizing the transition probability matrix as an image, a Markov image is generated.

According to research by Zhao et al.[16], the steps to generate a Markov image from malware opcodes include the following: opcode sequence extraction, opcode pair statistics, transition probability matrix generation and markov image generation.

- Opcode Sequence Extraction: Extracting opcode sequences from malware.
- Opcode Pair Statistics: Counting the frequency of each opcode pair in malware samples.

- Transition Probability Matrix Generation: Calculating the transition probabilities based on the statistics of opcode pairs and generating the transition probability matrix.
- Markov Image Generation: Multiplying the values in the transition probability matrix by 255 to meet pixel requirements and storing the results as grayscale images.

3.5 Malware Classification Models Design

This study proposes a malware classification method based on multi-level feature fusion, incorporating both image-based malware representation and malware opcode features. To achieve this, we selected deep learning models to extract features from these two types of images. In the current field of malware classification, convolutional neural networks (CNNs) are widely used for image feature extraction, with commonly employed models including ResNet, VGG, and EfficientNet. Therefore, this study will utilize these models for feature extraction. Additionally, to effectively fuse these two types of features, we will select appropriate existing feature fusion methods and propose a novel low-level feature fusion approach.

3.5.1 Convolutional Neural Network

3.5.1.1 ResNet

ResNet (Residual Network)[28] was proposed by Microsoft Research as a convolutional neural network architecture that employs residual connections. These residual connections, which allow the input data to be directly passed to subsequent layers, address the issues of vanishing and exploding gradients in deep convolutional neural networks. This enables deeper networks to effectively learn and significantly improves their performance.

3.5.1.2 VGG

The VGG model[29] is a classic convolutional neural network originally proposed by the Visual Geometry Group (VGG) at the University of Oxford in 2014. Due to its simple yet effective structure, the VGG model has been widely applied in image recognition and computer vision tasks. The core design principle of the VGG model is to construct a deep network by stacking multiple small 3x3 convolutional kernels, which enables the network to capture more image features. The most common VGG networks are VGG-16 and VGG-19, which consist of 16 and 19 trainable layers, respectively. This deep structure allows the model to learn more complex and rich feature representations.

3.5.1.3 EfficientNet

EfficientNet, proposed by Google[30] in 2019, is a convolutional neural network architecture renowned for its higher efficiency and superior performance. The core idea behind EfficientNet is the use of a method called "compound scaling," which simultaneously balances the network's depth, width, and resolution. This approach enables EfficientNet to maintain high accuracy while significantly reducing computational costs.

3.5.2 Feature Fusion

3.5.2.1 Feature Concatenate

The feature fusion method based on Feature Concatenation is a technique that directly concatenates multiple feature vectors column-wise.

3.5.2.2 Multilayer Deep Network based Feature Fusion

Chen et al.[18] proposed a feature fusion method based on a multilayer deep network. In this approach, two feature sets of different lengths are processed independently through separate network branches. The features are then fused at an intermediate layer to form a new feature representation, which is subsequently passed through the output layer to generate the final result. The multilayer deep network is shown in Figure 3.

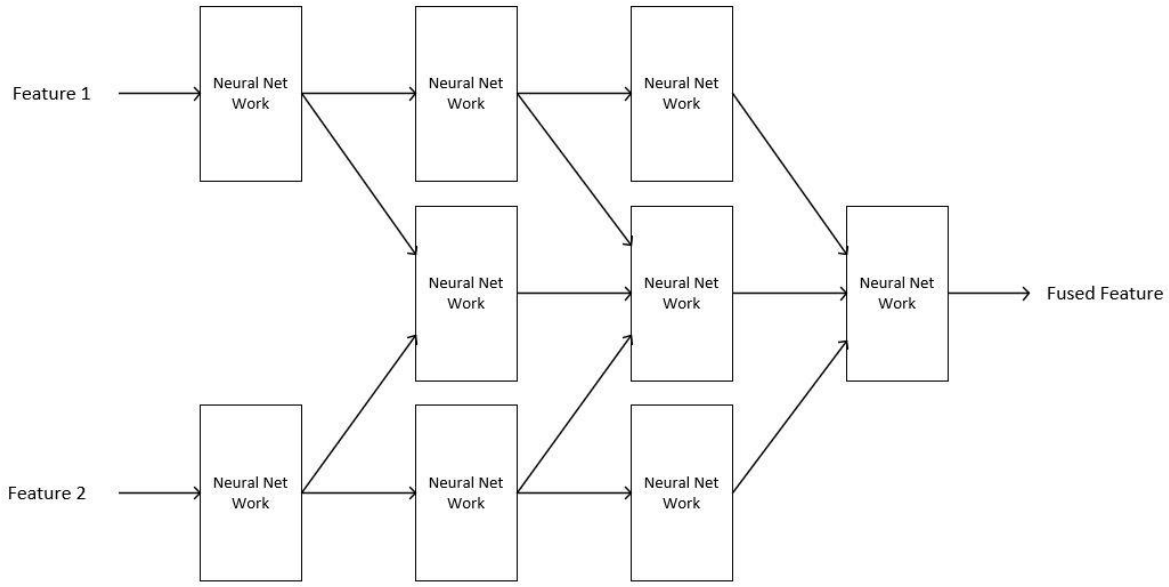

Figure 3 Multilayer deep network

3.5.2.3 Cross-Attention Mechanism based Feature Fusion

Yang et al.[24] proposed a feature fusion method known as the cross-attention mechanism, which is an improvement based on the self-attention mechanism. This mechanism enhances the fusion and interaction of information by exchanging or sharing keys, queries, or values between different features.

The self-attention mechanism is a technique that can establish dynamic weighting relationships between different positions within the same sequence. Specifically, in a self-attention mechanism, each element in the input sequence is treated as a query, key, and value. The attention weights are obtained by calculating the dot product between the queries and keys, which are then applied to the corresponding values to generate a weighted representation of the input sequence. The advantage of the self-attention mechanism lies in its ability to capture global dependencies within the input sequence, even if these dependencies are far apart in the sequence.

While the self-attention mechanism performs well when processing a single feature sequence, its limitation is that it operates only within the same feature space. This means it can only compute self-correlations for a single input feature sequence and cannot directly handle interaction information between multiple features.

The cross-attention mechanism is an extension and improvement of the self-attention mechanism. Unlike the self-attention mechanism, which performs correlation calculations within a single feature space, the cross-attention mechanism aims to capture complex relationships across different feature spaces. In the cross-attention mechanism, keys, queries, and values between different features are exchanged or shared, enabling the model to capture associative information across feature spaces.

The core idea of the cross-attention mechanism is to establish associations between different feature representations. Through this exchange or sharing, the model can capture richer associative information across different feature spaces, thereby enhancing the final representational capability.

3.5.2.4 Multiplication based Low-level Feature Fusion Method

This research will design a multiplication based low-level feature fusion method to integrate the high-level features of opcode Markov images with the low-level features of image-based malware representations through multiplication.

3.6 Model Training

In this section, we will introduce the key components involved in the model training process, including the loss function, optimization algorithm, learning rate, batch size, training epochs, data augmentation, and transfer learning.

3.6.1 Loss Function

A loss function is a mathematical function used to quantify the difference between the predicted values generated by a model and the actual target values; the primary goal in training a model is to minimize the value of this function, thereby reducing prediction errors.

3.6.1.1 Cross-entropy Loss

Cross-entropy loss is a loss function commonly used in deep learning for classification tasks. It evaluates the performance of a model by measuring the difference between the true class distribution and the predicted probability distribution.

$$L(\boldsymbol{y}, \boldsymbol{p}) = -\sum_{i=1}^{C} y_i \log(p_i) \quad (1)$$

Here, y_i represents the true class label, and p_i denotes the predicted probability that the sample belongs to class .

3.6.2 Optimization Algorithm

An optimization algorithm is a method used to adjust model parameters in order to minimize the loss function value, thereby improving model performance.

3.6.2.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a commonly used optimization algorithm in deep learning model training. This algorithm calculates the gradient of the loss function with respect to a single sample, and then iteratively updates the model parameters. The update rule is defined as follows:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J\left(\theta; x^{(i)}, y^{(i)}\right) \tag{2}$$

where θ represents the model parameters, η is the learning rate, and $\nabla_\theta J\left(\theta; x^{(i)}, y^{(i)}\right)$ denotes the gradient of the loss function with respect to the sample. Due to its high computational efficiency and low memory requirements, SGD is particularly well-suited for training on large-scale datasets. However, since the direction of the updates may experience significant fluctuations, it can lead to slower convergence rates and even potential entrapment in local minima, thereby affecting the overall training effectiveness of the model.

3.6.2.2 Adam Algorithm

The Adam algorithm is a widely used optimization method in deep learning training. This algorithm employs an adaptive learning rate mechanism, which can automatically adjust the learning rate based on the variation in gradients. By estimating the first and second moments of the gradients, Adam dynamically scales the learning rate, allowing for more precise parameter updates. This mechanism enables the algorithm to balance the update rates of different parameters during training, thereby improving convergence speed and overall learning performance. However, in certain specific scenarios, the convergence of the Adam algorithm may not meet expectations. The Adam algorithm is shown in Table 1.

Adam algorithm process

Required inputs: Initial parameter θ, momentum Variable v, global learning rate α, momentum factor β_1, accumulated squared variable s, accumulated gradient squared factor β_2

1. Randomly select a sample
2. Calculate the loss function:

$$\nabla_\theta J(\theta)$$

3. Update the momentum term v and the squared gradients accumulation s:

$$v = \beta_1 v + (1 - \beta_1)\nabla_\theta J(\theta)$$

$$s = \beta_2 s + (1 - \beta_2)\left(\nabla_\theta J(\theta)\right)^2$$

4. Bias correction:

$$\hat{v} = v/(1 - \beta_1^t)$$

$$\hat{s} = s/(1 - \beta_2^t)$$

5. Update the parameters:

$$\theta = \theta - \alpha\hat{v}/(\sqrt{\hat{s}} + \epsilon) \quad \epsilon = 10^{-8}$$

6. Return the updated parameter θ

Table 1 Adam algorithm process

3.6.3 Learning Rate

Learning rate is a hyperparameter that controls the step size of each update to the model parameters during the optimization process.

3.6.4 Batch Size

Batch size is the number of training examples processed simultaneously before updating the model's parameters in one iteration.

3.6.5 Training Epochs

Batch size is the number of training examples processed simultaneously before updating the model's parameters in one iteration.

3.6.6 Data Augmentation

Data augmentation is a technique that artificially increases the diversity of a training dataset by applying random transformations, such as rotations or flips, to the input data.

3.6.6.1 Random Cropping and Horizontal Flip

Random cropping and horizontal flip were first employed in the work of Krizhevsky et al.[31] to increase the diversity of training data and thereby enhance the model's generalization ability.

3.6.7 Transfer Learning

Transfer learning is a machine learning technique that leverages knowledge gained from a pre-trained model on one task and applies it to a new, related task, reducing the need for extensive training data and time on the new task.

3.7 Model Evaluation

In this section, we will introduce several key metrics for evaluating the performance of the proposed classification model, including accuracy, precision, recall, F1 score, and confusion matrix.

3.7.1 Accuracy

Accuracy is one of the fundamental metrics used to evaluate the performance of classification models. It represents the proportion of correctly predicted samples out of the total number of samples.

The equation for calculating accuracy is as equation3:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{3}$$

Where:

- TP (True Positive): The number of positive samples correctly classified as positive.
- TN (True Negative): The number of negative samples correctly classified as negative.
- FP (False Positive): The number of negative samples incorrectly classified as positive.
- FN (False Negative): The number of positive samples incorrectly classified as negative.

3.7.2 Precision

Precision is one of the key metrics used to evaluate the performance of a classification model. It represents the proportion of actual positive samples among all samples that the model has predicted as positive.

The equation for calculating precision is as equation 4:

$$\text{Accuracy} = \frac{TP}{TP+FP} \tag{4}$$

3.7.3 Recall

Recall is an important metric for evaluating the performance of classification models. It represents the proportion of actual positive samples that the model correctly identifies as positive.

The equation for calculating recall is as equation 5:

$$\text{Accuracy} = \frac{TP}{TP+FN} \tag{5}$$

3.7.4 The F1 Score

The F1 Score is a comprehensive metric for evaluating the performance of classification models. It is the harmonic mean of Precision and Recall. The F1 Score aims to balance Precision and Recall, making it particularly useful in scenarios with imbalanced classes.

The equation for calculating the F1 Score is as equation 6:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

3.7.5 The confusion matrix

The confusion matrix is one of the essential tools for evaluating the performance of classification models. It presents the relationship between the model's predictions and the actual labels in a tabular format, thereby aiding in the analysis of the model's performance across various categories.

4. Design

4.1 Data Collection

In malware research, common data collection methods include honeypots, third-party sharing websites, and open-source datasets. We chose to use open-source datasets because, compared to the other two methods, they offer clear classification labels, lower costs, and do not involve complex security and legal issues.

The Microsoft Malware Classification Challenge dataset originates from the Microsoft Malware Classification Challenge and contains samples of nine types of malware. The dataset is divided into training and testing sets, with labeled samples in the training set. The number of samples for each type is shown in Table 2. Each malware sample includes two files: a .bytes file, which represents the binary content of the malware in hexadecimal format, and a decompiled .asm file. The reason for selecting this dataset lies in its widespread use in malware classification research, facilitating comparison with various methods. Additionally, the decompiled files provided by this dataset allow for the extraction of opcode sequences, further enhancing the depth of the research.

Malware Family	Number of Samples
Ramnit	1541
Lollipop	2478
Kelihos_ver3	2942
Vundo	475
Simda	42
Tracur	751
Kelihos_ver1	398
Obfuscator.ACY	1228
Gatak	1013

Table 2 Microsoft Malware Classification Challenge dataset

The CCF BDCI 2021 Malware Dataset is sourced from the CCF BDCI 2021 Digital Security Competition—AI-based Malware Family Classification Contest. This dataset provides samples of ten types of malware, similarly divided into training and testing sets, with the types and quantities of training samples shown in Table 3. Each malware sample includes two files: a PE file without the PE header and an .asm file generated using IDA Pro. The reason for selecting this dataset is that it offers a large number of diverse samples and also provides decompiled files, making it convenient to extract opcode sequences.

Malware Family	Number of Samples
0	428
1	746
2	20
3	261
4	321
5	181
6	776
7	1350
8	594
9	1164

Table 3 CCF BDCI 2021 dataset

4.2 Data Pre-processing Design

4.2.1 Image-based Malware Representations Extraction

Inspired by the study conducted by Tekerek et al.[5], the process of visualizing malware in this research includes the following steps: reading the binary file, adjusting the array length and calculating the image dimensions, and generating the image (as shown in Figure 4). The specific steps are as follows:

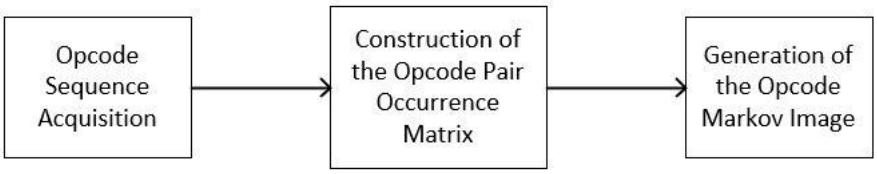

Figure 4 Image-based malware representations extraction

- Reading the Binary File: Extract the binary data of the malware sample, which serves as the foundation for subsequent processing.
- Adjusting the Array Length and Calculating the Image Dimensions: Based on the size of the read data, determine the appropriate image dimensions. If the data length is insufficient, padding is performed to match the required image dimensions.
- Generating the Image: The adjusted array is reshaped into a square matrix and saved as an image file for further analysis.

4.2.2 Opcode Markov Image Extraction

The process of extracting an Opcode Markov Image, inspired by the study of Zhao et al.[16], involves three steps, as illustrated in Figure 5.

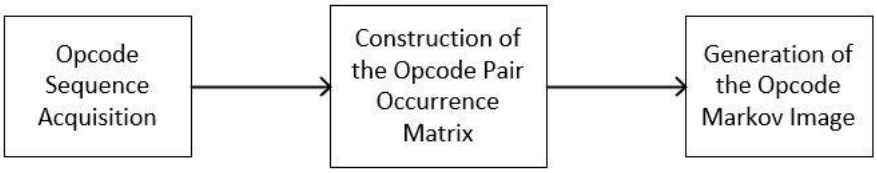

Figure 5 Process of extracting an Opcode Markov Image

- Opcode Sequence Acquisition: Opcode sequences are extracted from decompiled malware samples. The extraction process is based on a commonly used set of opcodes from the x86 instruction set, identified and extracted from the decompiled files through string matching. To enhance data purity, irrelevant content such as line numbers and comments beginning with a semicolon are filtered out, ensuring that the extracted opcode sequences remain undisturbed.
- Construction of the Opcode Pair Occurrence Matrix: The frequency of each opcode and its subsequent opcode in the extracted sequences is counted. Rarely occurring opcodes are categorized as one type. The resulting occurrence matrix is structured in a 224x224 format.
- Generation of the Opcode Markov Image: Each element in the opcode pair occurrence matrix is divided by the sum of the elements in its row to calculate the transition probability, which is then multiplied by 255 to generate pixel values. Finally, the transition frequency matrix is converted into a grayscale image.

4.3 Model Design

In this study, a deep learning model based on multilayer feature fusion is proposed to enhance the accuracy and robustness of malware detection. Specifically, the model integrates two different data representations: image-based malware representation and malware opcode Markov images, leveraging a multi-level feature fusion mechanism to fully exploit their complementary information in the malware classification task.

The multilayer feature fusion mechanism includes a feature fusion module based on cross-attention and a low-level feature fusion module based on multiplication. In the cross-attention-based feature fusion module, the high-level features of the image-based malware representation and opcode Markov image are integrated. Meanwhile, in the multiplication-based low-level feature fusion module, the low-level features of the image-based malware representation are combined with the high-level features of the opcode Markov image. Through this multilayer feature fusion, the two types of features can interact more comprehensively and deeply, enabling the model to more effectively extract and identify malware characteristics.

The model primarily consists of five modules: an image-based malware representation feature extraction module, a malware opcode Markov image feature extraction module, a cross-attention based feature fusion module, and a multiplication-based low-level feature fusion module, with the final output of malware categories achieved through a fully connected layer. The architecture of the proposed model is illustrated in Figure 6.

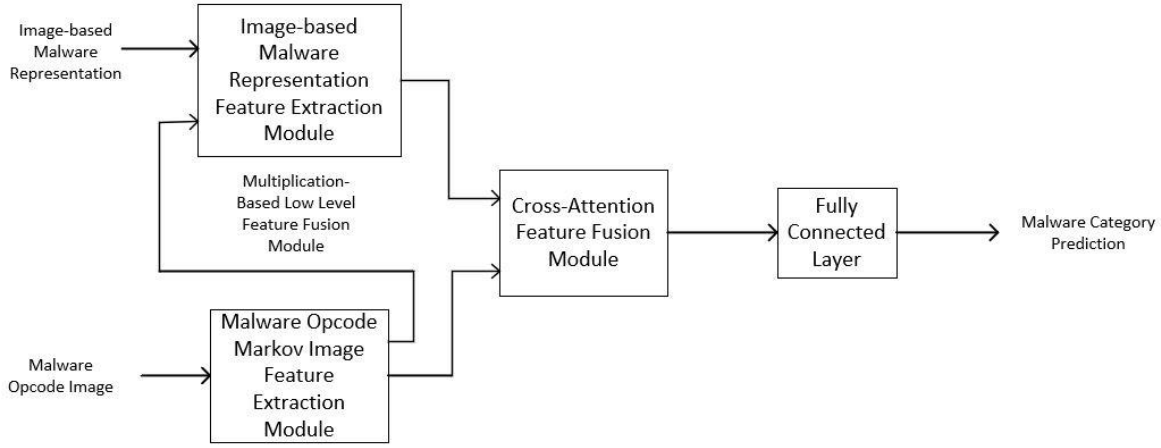

Figure 6 Model architecture

- Image-based Malware Representation Feature Extraction Module: This module extracts deep features from image-based malware representation using convolutional neural networks (CNNs). In this study, VGG16, ResNet18, and EfficientNetB0 are employed as feature extractors.
- Malware Opcode Markov Image Feature Extraction Module: Similarly, this module employs another convolutional neural network to extract features from the malware opcode Markov images. Likewise, VGG16, ResNet18, and EfficientNetB0 are used as feature extractors in this study.
- Cross Attention Feature Fusion Module: This module designs a cross attention mechanism for fusing the features extracted by the first two modules.
- Multiplication-based Low Level Feature Fusion Module: In this module, the features from the Markov images are first upsampled to match the size of the low level image-based malware representations features extracted by the convolutional neural network, and then the two are fused through a multiplication operation.

- Fully Connected Layer: The fully connected layer receives the features fused through the mutual attention mechanism and ultimately outputs the prediction of the malware category.

4.3.1 Image-based Malware Presentation Feature Extraction Module

The study employs VGG16, ResNet18, and EfficientNetB0 as the feature extraction modules for Image-based Malware Presentation. The performance of these models will be evaluated in subsequent experiments through comparative analysis. To adapt these models for the malware image feature extraction module, necessary modifications were made, focusing primarily on two aspects: the fully connected layers of the convolutional neural networks (CNNs) and the addition of a Multiplication Layer in the lower convolutional layers, aimed at integrating low-level features.

In the original models, the fully connected layers classify the output features extracted by the CNNs. However, in this study, the output size of the fully connected layers was adjusted to 1 × 512 to align with the dimensions of the Image-based Malware Presentation features and Opcode Markov image features, facilitating subsequent processing.

Although the structures of these CNNs differ, this study introduced a Multiplication module after the fourth or fifth convolutional layer in the lower layers of these networks. This modification aims to integrate the Opcode Markov image features with the low-level features of the Image-based Malware Presentation, thereby enhancing the model's classification performance by combining multi-level feature information.

The modified structures of the VGG16, ResNet18, and EfficientNetB0 models are illustrated in Figures 7, Figure 8, and Figure 9, respectively.

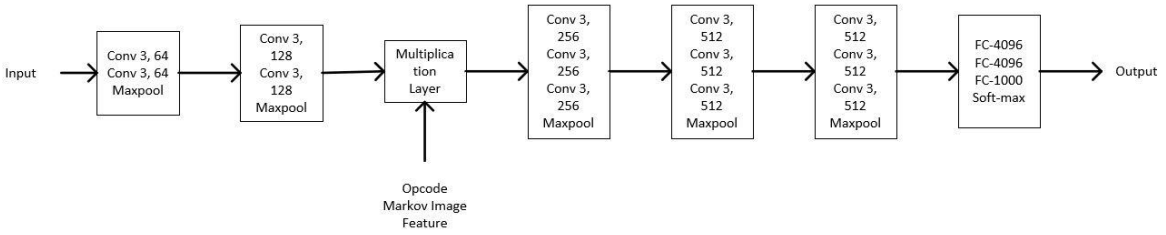

Figure 7 Modified VGG16

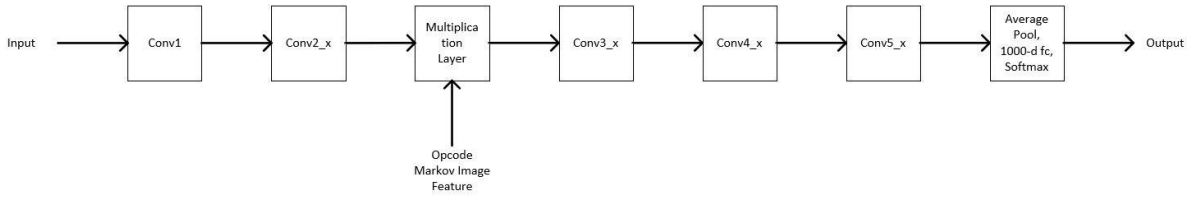

Figure 8 Modified ResNet18

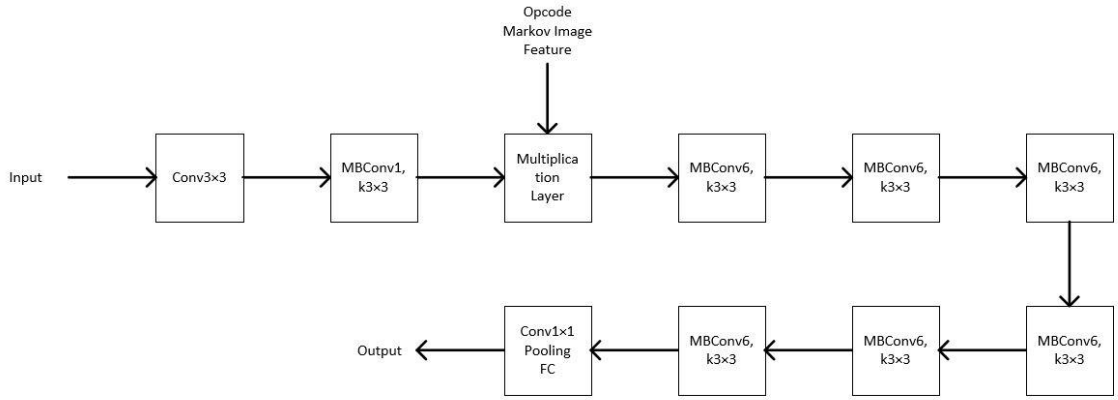

Figure 9 Modified EfficientNetB0

4.3.2 Opcode Markov Image Feature Extraction Module

In this study, VGG16, ResNet18, and EfficientNetB0 were selected as the Opcode Markov Image Feature Extraction Modules. Subsequent experiments conducted a detailed evaluation of the extraction performance of these different models through comparative analysis.

To effectively apply these models to malware image feature extraction, two key modifications were made. First, the fully connected layer of the convolutional neural networks (CNNs) was adjusted. Second, the features from the last convolutional layer of each model were extracted and utilized as the high level feature output for the Opcode Markov images.

In the original models, the output from the fully connected layer is typically used for classification tasks. However, in this study, the output dimension of the fully connected layer was adjusted to 1 × 512 to align with the feature dimensions of the image-based malware representation, thereby simplifying the subsequent feature fusion process. Additionally, the features extracted from the final convolutional layer were used as additional outputs, which were then fused with the low-level features of the image-based malware representation.

The architectures of the modified VGG16, ResNet18, and EfficientNetB0 models are illustrated in Figures 10, Figure 11, and Figure 12, respectively.

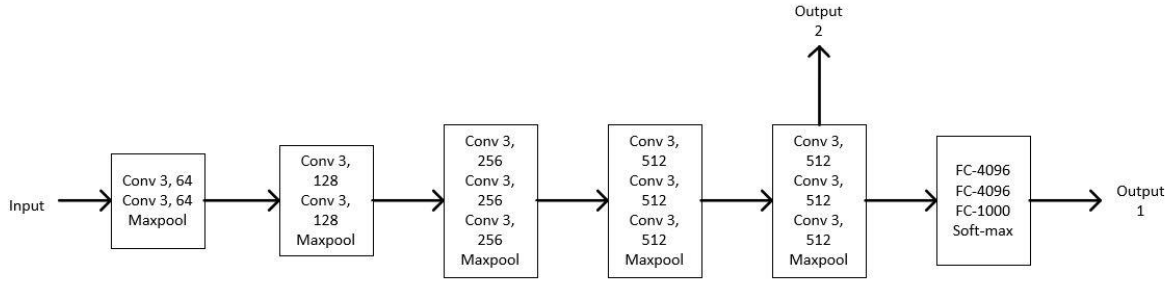

Figure 10 VGG16

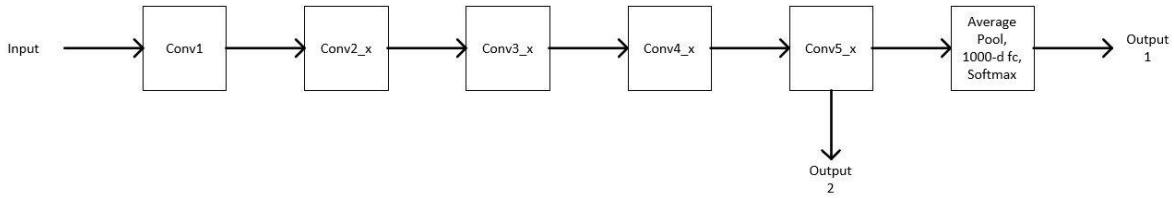

Figure 11 ResNet18

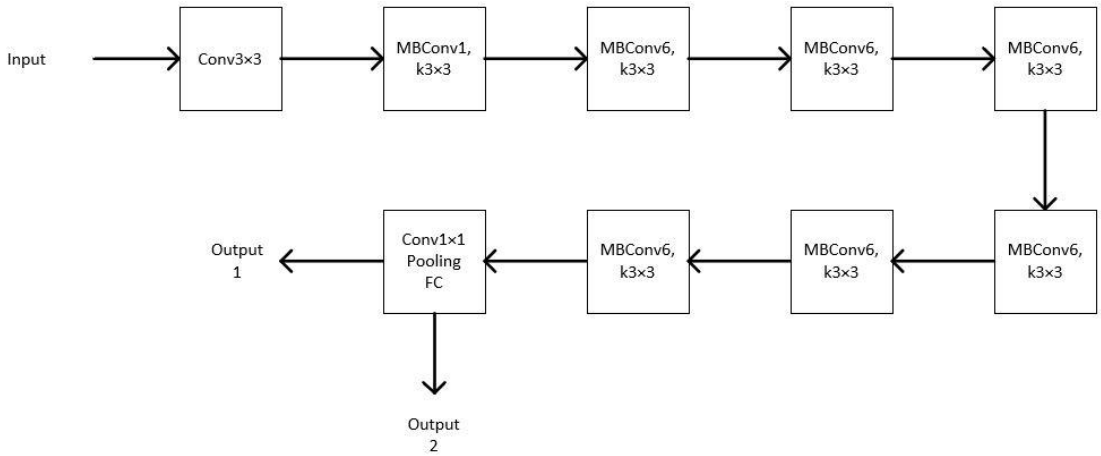

Figure 12 EfficientNetB0

4.3.3 Cross-attention Feature Fusion Module

The cross-attention-based feature fusion module demonstrates superior performance in current technologies. Compared to simple concatenation-based feature fusion methods, the cross-attention mechanism allows for interaction between features from different modalities. In contrast to layer-by-layer fusion methods based on multi-layer neural networks, the cross-attention mechanism can dynamically adjust weights based on the relevance of input features. As a result, the model can dynamically capture the most meaningful associations between different inputs, rather than simply assigning fixed feature weights. This dynamic capability makes the model more flexible and effective in capturing relationships between various features when dealing with multiple types of features.

The cross-attention-based feature fusion model proposed in this study consists of four modules: the feature transformation module, the attention computation module, the residual module, and the concatenation module, as illustrated in Figure 1. In the feature transformation module, the query, key, and value for the image-based malware representation features and the malware opcode image features are calculated using Equations 7 through 12. In the attention computation module, attention weights are first computed, followed by a weighted sum of the values. The attention weights are calculated using Equations 13 and 14, and the final attention output is obtained using Equations 15 and 16. In the residual module, the features before and after cross-attention processing are summed to compensate for information loss. Finally, the features are fused through a concatenation operation.

The cross-attention-based feature fusion module is shown in Figure 13.

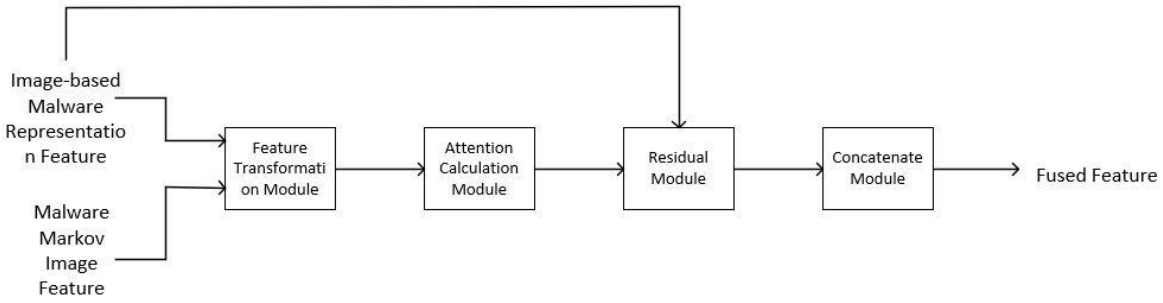

Figure 13 Cross-attention based feature fusion module

$$Q_{\text{malimg}} = X_{\text{malimg}} W_{Q_{\text{malimg}}} \tag{7}$$

$$K_{\text{malimg}} = X_{\text{malimg}} W_{K_{\text{malimg}}} \tag{8}$$

$$V_{\text{malimg}} = X_{\text{malimg}} W_{V_{\text{malimg}}} \tag{9}$$

$$Q_{\text{opimg}} = X_{\text{opimg}} W_{Q_{\text{opimg}}} \tag{10}$$

$$K_{\text{opimg}} = X_{\text{opimg}} W_{K_{\text{opimg}}} \tag{11}$$

$$V_{\text{opimg}} = X_{\text{opimg}} W_{V_{\text{opimg}}} \tag{12}$$

$$A_{\text{malimg}} = \text{Softmax}\left(\frac{Q_{\text{malimg}} K_{\text{opimg}}^{\text{T}}}{\sqrt{d_k}}\right) \tag{13}$$

$$A_{\text{opimg}} = \text{Softmax}\left(\frac{Q_{\text{opimg}} K_{\text{malimg}}^{\text{T}}}{\sqrt{d_k}}\right) \tag{14}$$

$$Z_{\text{malimg}} = A_{\text{malimg}} V_{\text{malimg}} \tag{15}$$

$$Z_{\text{opimg}} = A_{\text{opimg}} V_{\text{opimg}} \tag{16}$$

4.3.4 Multiplication Based Low Level Feature Fusion Module

The feature fusion module based on cross-attention integrates the image-based malware representations with the high-level features of the opcode Markov images. However, to better extract the texture features of the image-based malware representations, it is necessary to fuse the high-level features of the opcode Markov images with the low-level features of the image-based malware representations. Since image-based malware representations contain rich texture information, enhancing the extraction of their low-level features through the high-level features of the opcode Markov images is beneficial. Low-level features of convolutional neural networks typically capture local information such as edges and textures, while high-level features primarily represent global semantic information. By fusing high-level and low-level features, the model can leverage features at different levels simultaneously, enhancing its ability to capture complex patterns. Moreover, as the number of layers in the convolutional neural network increases, some local details may be lost in the high-level features. Therefore, relying solely on cross-attention-based feature fusion may not fully exploit the relationships between these features, thereby limiting the improvement in classification performance. To address this, this paper proposes a multiplication-based low-level feature fusion method. This method achieves dynamic weight adjustment by performing a multiplication between the high-level features of the opcode Markov images and the low-level features of the image-based malware representations, allowing the model to more flexibly capture the relationships between features, thereby enhancing classification performance.

In this method, the high-level features are extracted from the last convolutional layer of the opcode Markov image feature extractor, while the low-level features are obtained from the fourth or fifth convolutional layer of the image-based malware

representations feature extractor. Since the dimensions of high-level features and low-level features are usually different, with high-level features typically having smaller width and height, it is necessary to upsample the high-level features. In this study, upsampling is achieved through a transposed convolutional layer, as the transposed convolutional layer can learn an upsampling method more suitable for specific tasks. The stride, padding, and other parameters of the transposed convolutional layer are finely tuned to achieve effective upsampling of features with different dimensions.

The structure of this module is shown in Figure 14.

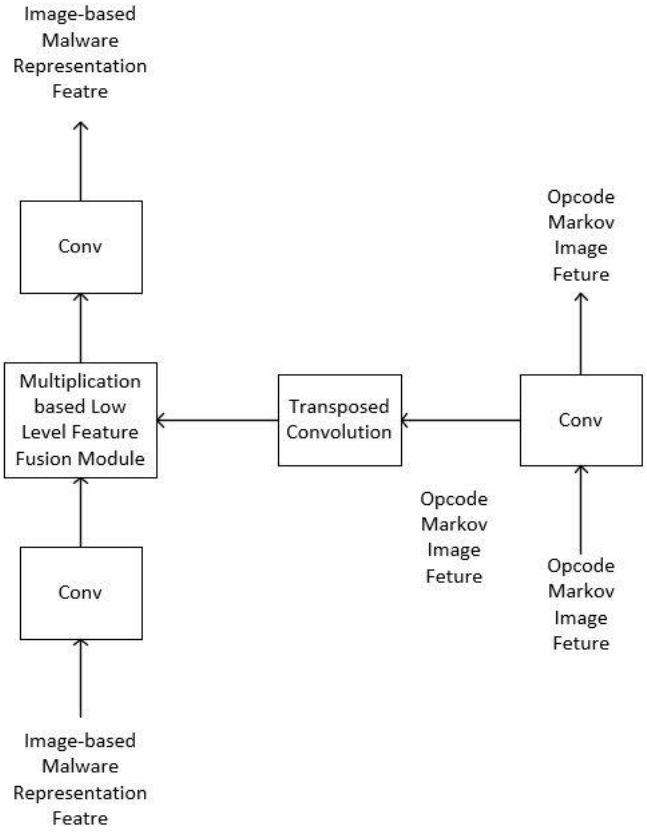

Figure 14 Multiplication Based Low Level Feature Fusion Module

4.4 Model Training

In this study, the cross-entropy loss function, a commonly used loss function, was employed. The Adam optimizer was chosen over Stochastic Gradient Descent (SGD) due to its advantages in avoiding local optima, which can negatively impact the model's classification performance. The training process was conducted with 100 epochs, a batch size of 32, and a fixed learning rate of 0.001. Additionally, data augmentation techniques, including random cropping and horizontal flipping, were applied. Transfer learning was also utilized to further enhance the model's training efficiency and effectiveness.

4.5 Model Evaluation

In this study, the performance of the model is evaluated using accuracy, precision, recall, F1-score, and the confusion matrix.

5. Implementation

5.1 Hardware and Software Resource

5.1.1 Hardware Resource

This study was primarily conducted on Colab. Colab instances are typically equipped with around 12GB to 25GB of RAM and 50GB of virtual hard disk space. Google Drive was mounted on Colab to extend storage capacity. An A100 GPU was used for deep learning training.

5.1.2 Software Resource

The primary programming language used in this study is Python. The Pandas library was utilized for reading .csv files, the Numpy library for matrix manipulation, the PIL library for image storage, and the Torch library for creating and training deep learning models.

5.2 Data Pre-processing Implementation

5.2.1 Opcode Sequences Extraction

In this project, I developed a Python script to extract opcode sequences from assembly code files. The script accomplishes the opcode extraction through the following steps:

- Loading the Opcode Set: First, the script loads the opcode set from a CSV file. Each opcode is stored as an element in a set, allowing for quick matching during subsequent processing.
- Processing the Assembly Code File: The script reads the assembly code file (.asm) line by line. For each line, it uses the .split(';') method to remove any potential comments (i.e., content starting from the semicolon ;). Then, it applies the .split() method to split the remaining code into a sequence of independent strings based on whitespace characters.

- Opcode Matching: For each split string, the script checks whether it exists in the opcode set. If a match is found, the corresponding opcode is saved into a list.
- Storing the Results: All matched opcodes are eventually written to a new CSV file, facilitating subsequent analysis and processing.

5.2.2 Opcode Markov Image Extraction

In this project, I developed a Python script to extract opcode Markov images from sequences of opcodes. The script accomplishes the extraction of opcodes through the following steps:

- Loading Opcode Sequences: First, the script loads the opcode sequences from a CSV file.
- Obtaining the Opcode Pair Frequency Matrix: The code traverses adjacent pairs in the opcode sequences and counts their occurrences.
- Generating the Probability Transition Matrix: Each row of the frequency matrix is divided by the sum of the respective row to obtain a probability transition matrix.
- Converting the Matrix into an Image: The normalized matrix is multiplied by 255 to convert it into a grayscale image. Finally, the matrix is saved as a PNG image using the plt.imsave function.

5.2.3 Image-based Malware Representations Extraction

In this project, to extract image-based malware representations from malware binary data, I developed a Python script. The script accomplishes the extraction of opcodes through the following steps:

Reading the Binary File: We use Python's open function to open the file in binary mode and read its contents into a byte array.

Filtering Invalid Data: To enhance the usability of the image, we filter out all byte values of 0 from the array, as these typically represent invalid information.

Adjusting Array Length: To convert the byte array into an RGB image, we need to ensure that the array length is a multiple of 3. If the length is insufficient, we pad zeros at the end of the array.

Calculating Image Dimensions: We calculate the side length of the square image that the byte array can represent and pad additional zeros if necessary to ensure the array length equals the square of the side length multiplied by 3.

Generating the Image: We reshape the adjusted array into a numpy array with the shape (side_length, side_length, 3) and use the PIL library to convert it into an RGB image.

To extract the binary data of malware from the .bytes files in the Microsoft Malware Classification Challenge dataset, we have adjusted the script accordingly. The script reads the file content line by line, skips the line numbers, and excludes meaningless symbols such as "00" and "??." It then reads and retains the remaining valid data.

5.3 Model Implementation

In this study, the model implementation was based on the PyTorch framework. The torch.nn module provided various neural network components, while torchvision.models offered pre-trained classical models. The image feature extraction module and the opcode-based Markov image feature extraction module utilized these

pre-trained models with corresponding modifications. The cross-attention-based feature fusion module and the multiplication-based low-level feature fusion module were custom-designed according to specific requirements. Finally, all modules were integrated into the MultilayerFeatureFusionModel class, achieving multilayer feature fusion for malware classification.

5.4 Model Training Implementation

First, the preprocessed training and test sets are loaded into memory from the specified paths. Then, the training set is split into a training set and a validation set in an 8:2 ratio.

In the implementation process, the PyTorch library is used to build and train the model. PyTorch provides the DataLoader class for loading data in batches, and by loading the data onto the GPU, it significantly improves the efficiency of data processing. During model training, PyTorch's automatic differentiation feature is utilized, with the torch.autograd module automatically computing gradients, enabling backpropagation and parameter updates in each epoch.

The model training process is divided into several epochs. In each epoch, the model first enters training mode by calling the model.train() method, which performs forward propagation, loss calculation, backpropagation, and parameter updates using an optimizer (e.g., Adam) on the training data. Then, the model enters evaluation mode, where it is evaluated on the validation set. The model.eval() method is used to ensure that gradients are not calculated during inference, saving computational resources and improving inference speed.

After training is complete, the model weights that performed best on the validation set are used for final evaluation on the test set. Finally, the torch.save() function is used to save the model weights for use in future research or applications.

5.5 Model Evaluation Implementation

The study uses the sklearn.metrics library to calculate the accuracy, precision, recall, and F1 score of the model, and the confusion matrix is visualized through sns.heatmap.

6. Experiments

6.1 Evaluation of Various Convolutional Neural Networks for Malware Classification

6.1.1 Experiment Settings

This experiment aims to compare the performance of different convolutional neural networks (CNNs) in the task of malware classification. To evaluate the classification effectiveness, multiple malware classification models will be used. These models employ VGG16, ResNet, or EfficientNet as the feature extractors for image-based malware representation and Opcode Markov Image. All models incorporate a multi-level feature fusion mechanism proposed in this study. A total of nine models are evaluated in the experiment, with their names and configurations detailed in Table 4.

Model Name	Feature Extractor for Image-based Malware Representations	Feature Extractor for Opcode Markov Image
Model 1	VGG16	VGG16
Model 2	VGG16	ResNet18
Model 3	VGG16	EfficientNetB0
Model 4	ResNet18	VGG16
Model 5	ResNet18	ResNet18
Model 6	ResNet18	EfficientNetB0
Model 7	EfficientNetB0	VGG16
Model 8	EfficientNetB0	ResNet18
Model 9	EfficientNetB0	EfficientNetB0

Table 4 Model configurations

During the experiment, the ResNet18 and EfficientNetB0 models were initialized using PyTorch's pretrained models to accelerate training and improve classification accuracy. To enhance the generalization ability of the models, data augmentation techniques such as random cropping and random flipping were applied to the malware images in the training set. The experimental dataset is sourced from the Microsoft Malware Classification Challenge. During the training phase, cross-entropy loss was used as the loss function, Adam optimizer as the optimization algorithm, with a learning rate set at 0.001, a batch size of 32, and a total of 100 epochs. The evaluation metrics for the experiment include accuracy, precision, recall, and F1-score.

6.1.2 Experiment Results

Model	Accuracy	Precision	Recall	F1 Score
Model1	0.9640	0.9649	0.9640	0.9644
Model2	0.9668	0.9674	0.9668	0.9671
Model3	0.9825	0.9836	0.9825	0.9827
Model4	0.9709	0.9713	0.9709	0.9711
Model5	0.9732	0.9735	0.9732	0.9734
Model6	0.9949	0.9950	0.9949	0.9949
Model7	0.9862	0.9863	0.9862	0.9860
Model8	0.9843	0.9848	0.9843	0.9843
Model9	0.9931	0.9933	0.9931	0.9931

Table 5 Experiments results

The experiment results are show on Table 5.

6.1.3 Analysis

Based on Table 5, Model 6 demonstrates the best performance across all metrics. The model achieves an accuracy of 99.49%, a precision of 99.50%, a recall of 99.49%, and an F1 score of 99.49%. Model 6 utilizes ResNet18 and EfficientNetB0 as feature extractors, indicating that the combination of these two feature extractors is highly

effective in the task of malware classification. Following closely is Model 9, which achieves 99.31% in all metrics, also performing exceptionally well. Model 9 also employs EfficientNetB0 as a feature extractor, further demonstrating the superiority of EfficientNetB0.

The comparison reveals that model performance significantly improves when EfficientNetB0 is used for feature extraction. This improvement could be due to EfficientNetB0's ability to balance model complexity and accuracy effectively. Notably, even though Model 9 uses dual EfficientNetB0 as feature extractors, Model 6 outperforms it. This could be because the deep residual network of ResNet18 excels at extracting complex texture features in image-based malware representations, while EfficientNetB0 is more effective in extracting opcode Markov image features with fewer texture details.

Overall, models using VGG16 as a feature extractor (Model 1, Model 2, Model 3, Model 4) perform relatively poorly, especially compared to models combining EfficientNetB0 and ResNet18. This could be attributed to the relatively older architecture of VGG16, which, despite having a large number of parameters, is less efficient than more recent networks. ResNet18, on the other hand, performs quite well, with its deep residual networks effectively capturing malware features, particularly in the complex patterns found in image representations.

Based on the experimental results, Model 6, which uses ResNet18 to extract image-based malware representation features and EfficientNetB0 to extract opcode Markov image features, is the optimal model.

6.2 Comparison of Feature Fusion Techniques for Malware Classification

6.2.1 Experiment Settings

This experiment will compare the performance of different feature fusion methods in the task of malware classification, utilizing multiple malware classification models to assess their effectiveness. The feature extractors are configured based on the optimal settings identified in previous experiments. In each model, the image features of the malware and the opcode Markov image features will be integrated using various fusion methods, including feature concatenation, a multilayer deep network-based fusion method, a cross-attention-based feature fusion method, and the multilayer feature fusion method proposed in this study. The experiments will be conducted on the Microsoft Malware Classification Challenge dataset. A total of four models are evaluated in the experiment, with their names and configurations detailed in Table 6.

Model Name	Feature Fusion Method
Model 1	Feature Fusion based via Concatenation
Model 2	Feature Fusion based via Multilayer Deep Network
Model 3	Feature Fusion based via Cross Attention
Model 4	Feature Fusion based via Multilayer Feature Fusion

Table 6 Model configurations

6.2.2 Experiments Results

6.2.2.1 Results for Model 1(Feature Fusion based via Concatenation)

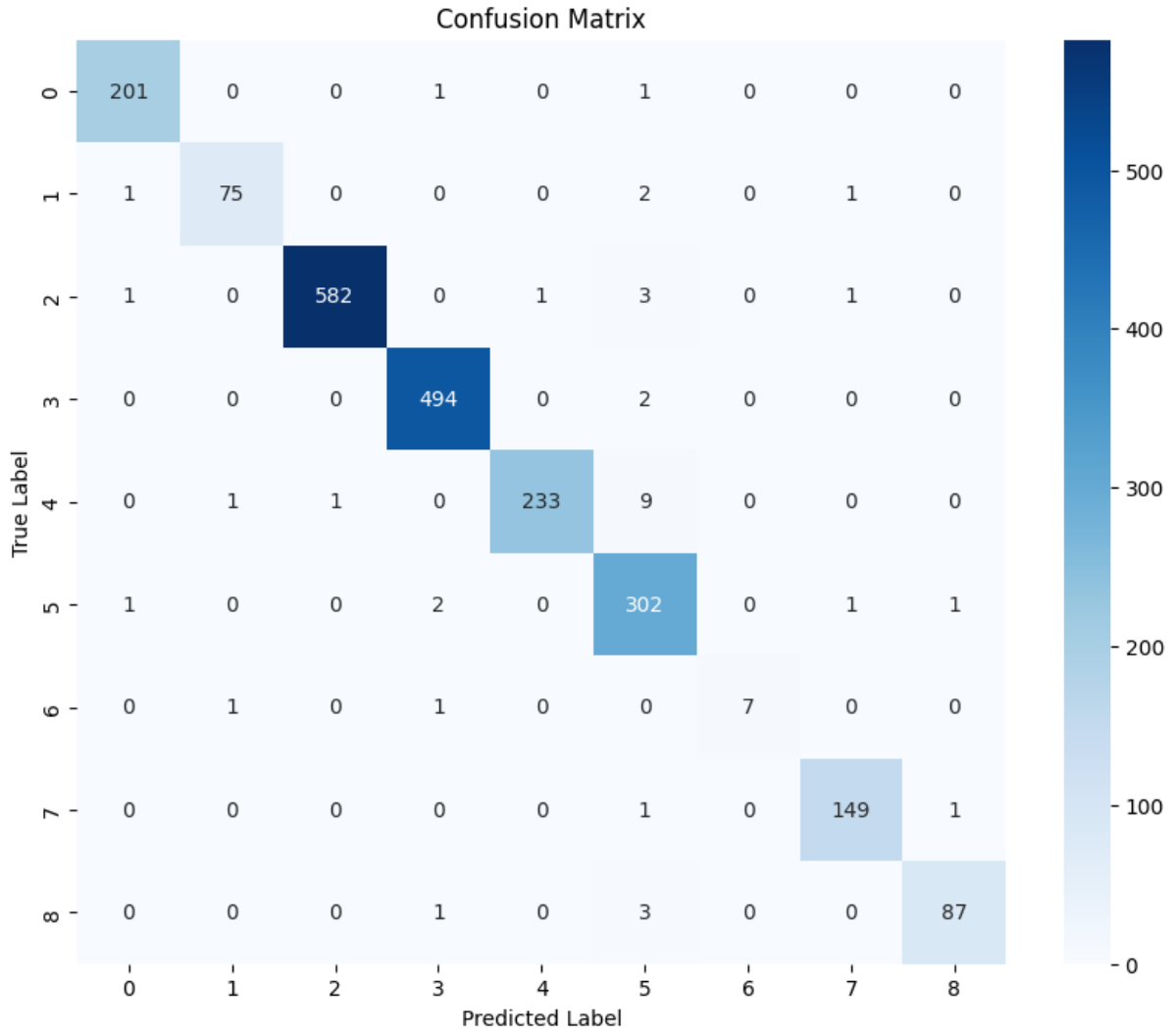

Figure 15 Confusion matrix of model 1

Accuracy	Precision	Recall	F1 Score
0.9825	0.9829	0.9825	0.9825

Table 7 Results of model 1

Model 1 employs the concatenate method proposed in this research for feature fusion. The confusion matrix shown in Figure 15 and evaluation metrics in Table 7 present

the classification results. According to the confusion matrix, the model correctly classified 2,130 malware samples and misclassified 38 samples. It can be observed that the model performs best on class 0 and class 7, with only two misclassified samples in each of these classes. This could be because the features of class 0 and class 7 are more distinct and have a greater degree of separation from other classes, allowing the model to classify these classes more accurately. The model performs worst on class 4, with 11 samples misclassified. This may be due to the higher internal feature diversity within class 4, meaning that the malware samples in this class have significant differences, making it challenging for the model to learn a unified pattern for correct classification. Additionally, 21 samples from other classes were misclassified as class 5. This could be because the features of class 5 significantly overlap with other classes in the feature space. The model struggles to distinguish samples from these overlapping regions, leading to a tendency to classify them as class 5.

In terms of evaluation metrics, the model achieved an accuracy of 0.9825, a precision of 0.9829, a recall of 0.9825, and an F1 score of 0.9825.

6.2.2.2 Result for Model 2(Feature Fusion based via Multilayer Deep Network)

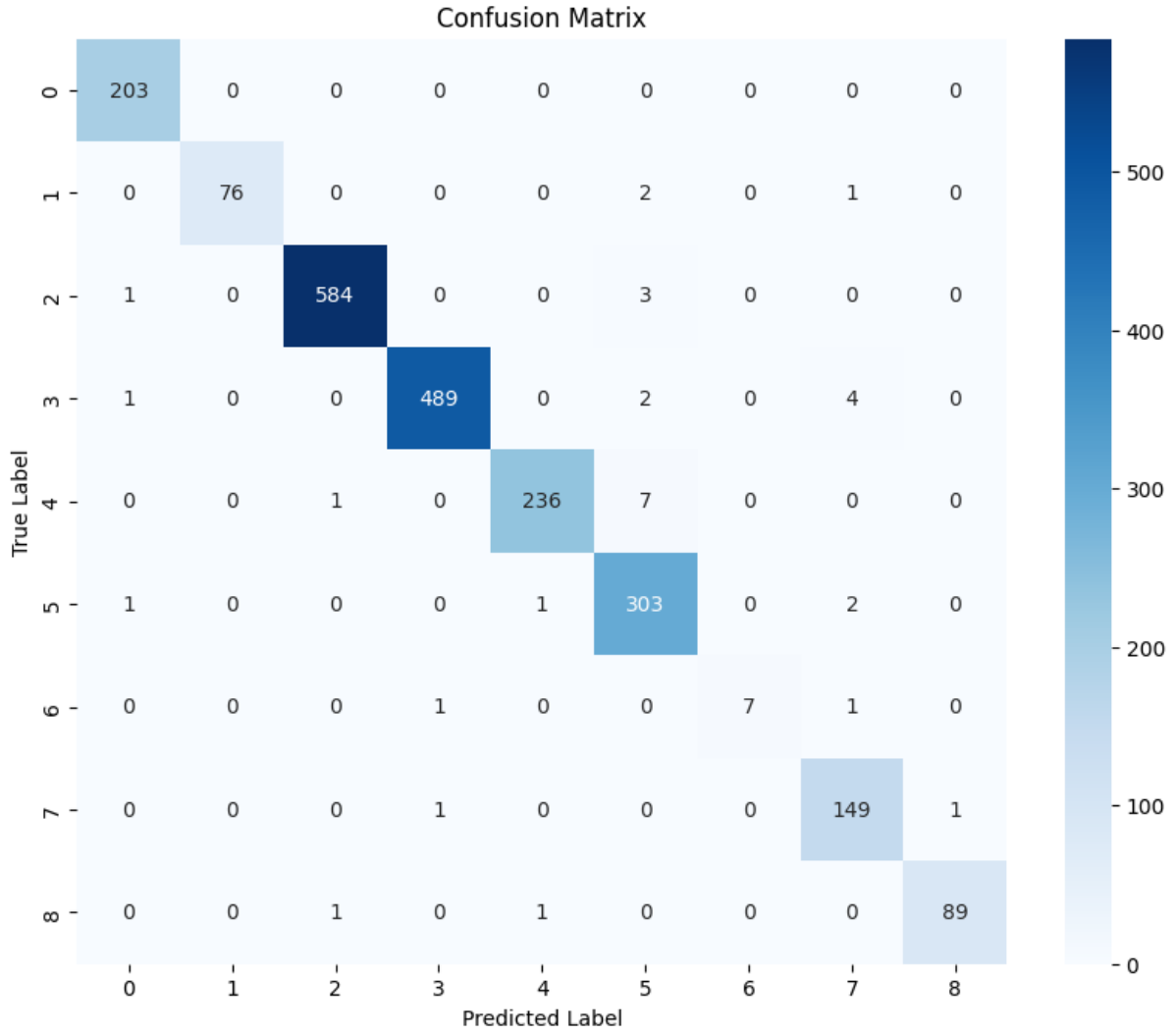

Figure 16 Confusion matrix for model 2

Accuracy	Precision	Recall	F1 Score
0.9852	0.9856	0.9852	0.9852

Table 8 Results for model 2

Model 2 employs the multilayer deep network method for feature fusion. The confusion matrix shown in Figure 16 and evaluation metrics presented in Table 8 demonstrate the classification results. According to the confusion matrix, the model correctly classified 2,136 malware samples and misclassified 32 samples. It is evident that the

model performed best in classifying Category 0, with all Category 0 samples being accurately identified. This could be attributed to the more distinct features of Category 0, which have greater differentiability from other categories, allowing the model to classify these samples more accurately. The model performed worst in classifying Category 4, with 8 Category 4 malware samples being misclassified, of which 7 were incorrectly identified as Category 5. This may be due to the similar internal features between Category 4 and Category 5. Additionally, it can be observed that 14 malware samples from other categories were erroneously classified as Category 5.

Regarding the evaluation metrics, the model achieved a classification accuracy of 0.9852, a precision of 0.9856, a recall of 0.9852, and an F1 score of 0.9852.

6.2.2.3 Result for Model 3(Feature Fusion based via Cross Attention)

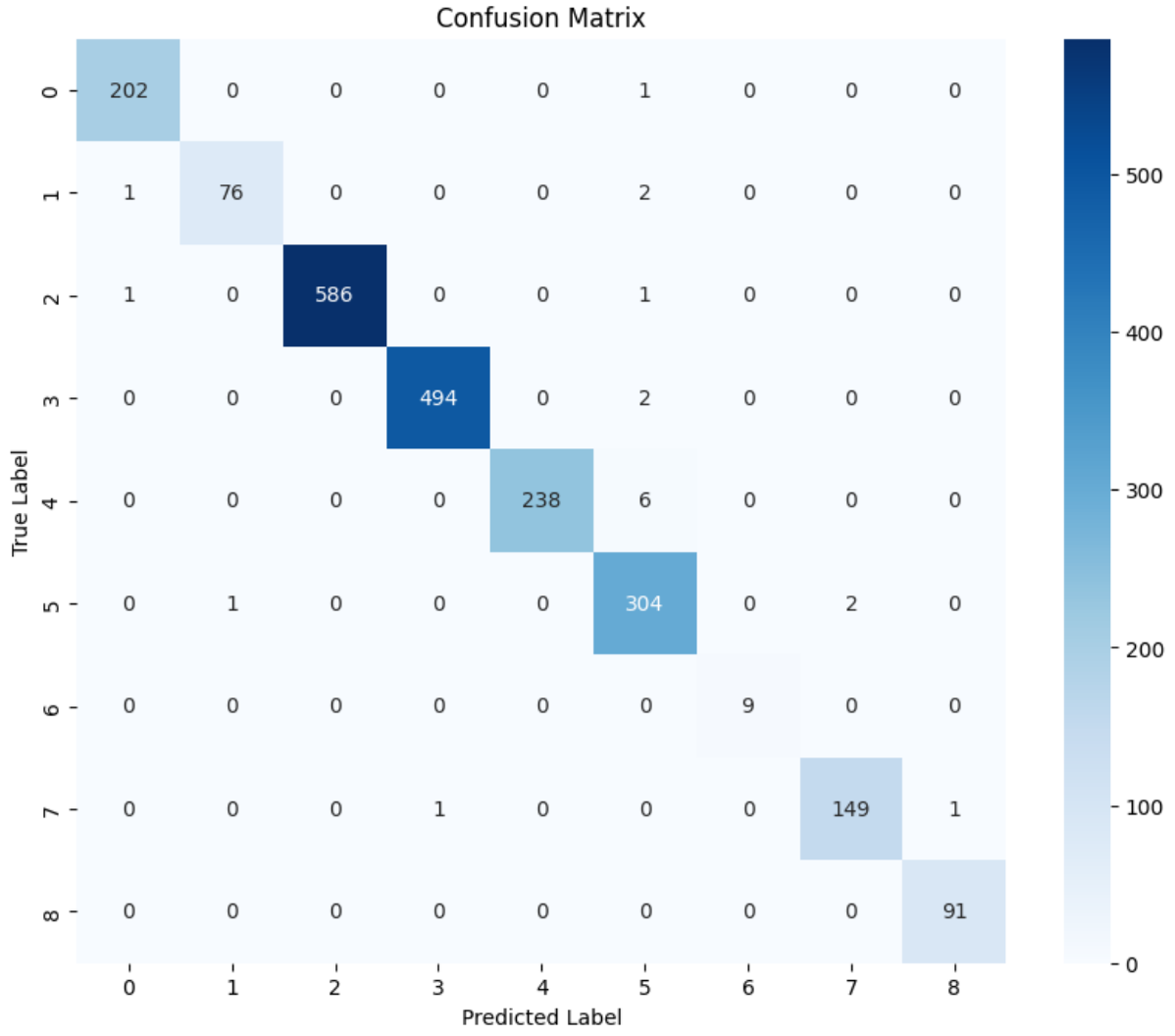

Figure 17 Confusion matrix for model 3

Accuracy	Precision	Recall	F1 Score
0.9912	0.9914	0.9912	0.9913

Table 9 Results for model 3

Model 3 employs the cross attention method for feature fusion. The confusion matrix and evaluation metrics presented in Table 9 illustrate the classification results. According to the confusion matrix shown in Figure 17, the model correctly classified 2,149 malware samples and misclassified 19 samples. It is evident that the model

performs best in classifying categories 0 and 8, with only one sample from category 0 being misclassified, and category 8 being entirely correctly classified. The model performs worst in classifying category 4, where six malware samples from category 4 were misclassified, all of which were incorrectly identified as category 5. This misclassification may be due to the internal similarity between categories 4 and 5. Additionally, 12 samples from other categories were misclassified as category 5.

In terms of evaluation metrics, the model achieved an accuracy of 0.9912, a precision of 0.9914, a recall of 0.9912, and an F1 score of 0.9913.

6.2.2.4 Result for Model 4(Feature Fusion based via Multilayer Feature Fusion)

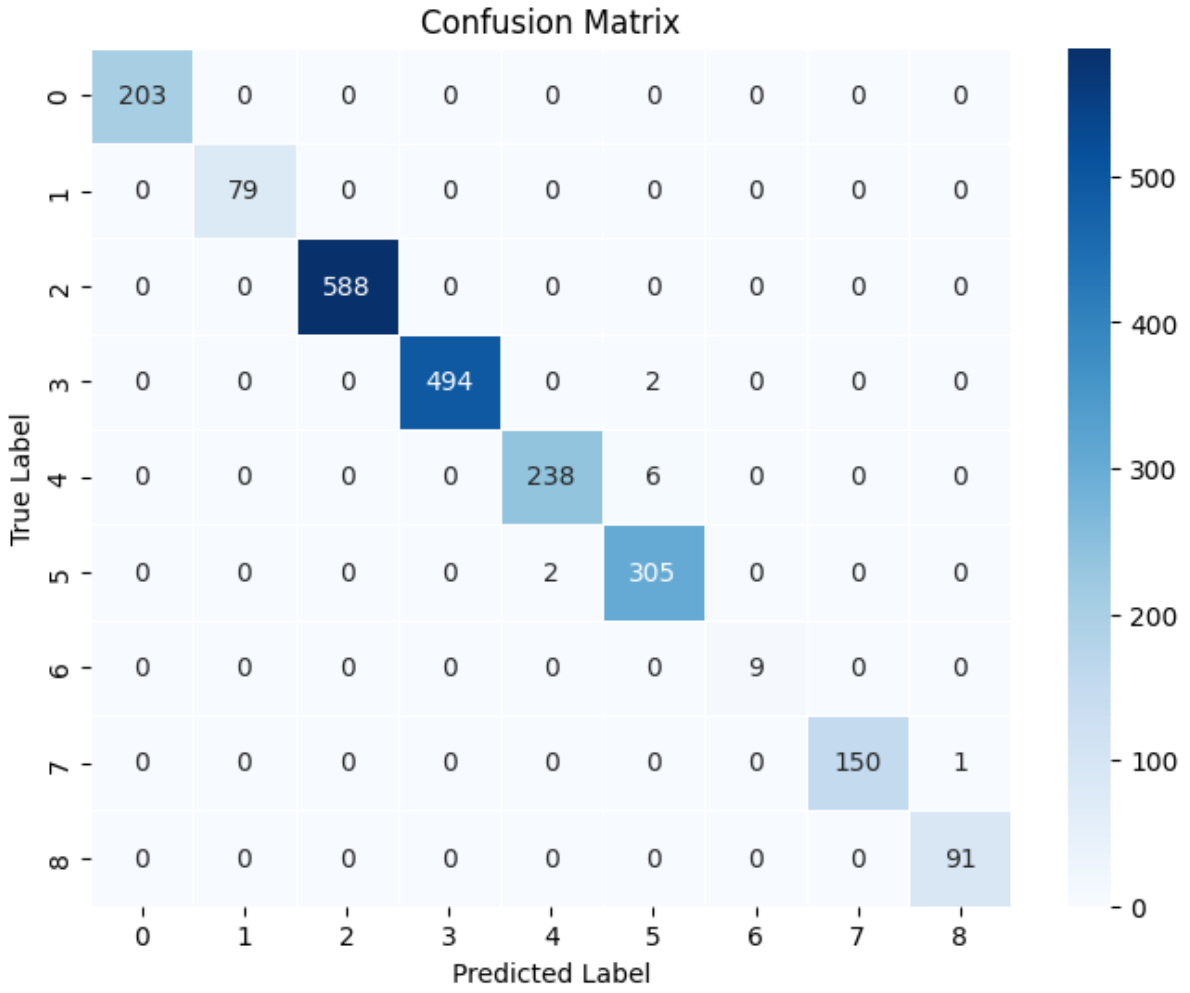

Figure 18 Confusion matrix for model 4

Accuracy	Precision	Recall	F1 Score
0.9949	0.9950	0.9949	0.9949

Table 10 Results for model 4

Model 4 employs the multilayer feature fusion method proposed in this research for feature fusion. The confusion matrix and evaluation metrics shown in Table 10 illustrate the classification results. According to the confusion matrix shown in Figure 18, the model correctly classified 2,157 malware samples and misclassified 11 malware samples. It can be observed that the model performs best in classifying

categories 0, 1, 2, 6, and 8, with these categories being completely correctly classified. The model performs worst in classifying category 4, with 6 samples from category 4 being misclassified, all of which were incorrectly identified as category 5. This may be due to the similarity in internal features between categories 4 and 5. Additionally, it is evident that 8 malware samples from other categories were misclassified as category 5.

In terms of evaluation metrics, the model achieves an accuracy of 0.9949, a precision of 0.9950, a recall of 0.9949, and an F1 score of 0.9949.

6.2.3 Analysis

In this experiment, we compared the performance of several commonly used feature fusion methods in the context of malware classification. The methods evaluated include feature fusion based on concatenation, multilayer deep network based feature fusion, cross-attention based feature fusion, and the novel multilayer feature fusion method proposed in this research. The experimental results demonstrate that the proposed multi-level feature fusion method outperforms the others across all metrics. Specifically, this method achieved an accuracy of 99.49%, a precision of 99.50%, a recall of 99.49%, and an F1 score of 99.49%. The experimental results indicate that the proposed multi-level feature fusion method successfully integrates features from image-based malware representation and Opcode Markov Image, effectively capturing inter-feature relationships and highlighting critical information within key features. Therefore, the proposed multi-level feature fusion method is a reliable feature fusion approach.

6.3 Performance Analysis of Malware Classification Models Across Different Datasets

6.3.1 Experiment Settings

This experiment employed the optimal configuration validated in previous experiments and was trained and evaluated on the CCF BDCI malware classification dataset.

6.3.2 Experiment Results

The experiment results are shown on Table 11 and Figure 19.

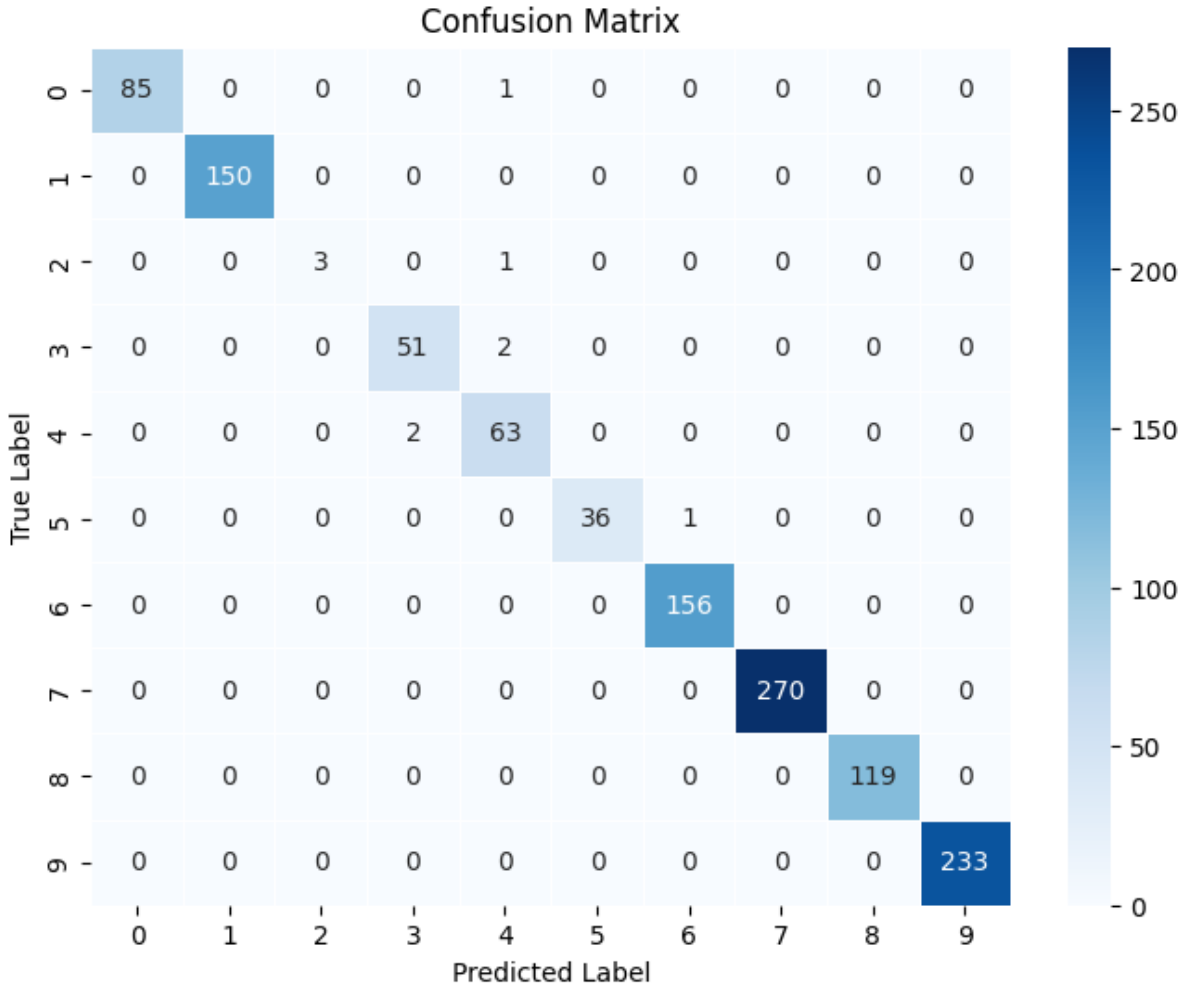

Figure 19 Confusion matrix

Accuracy	Precision	Recall	F1 Score
0.9940	0.9941	0.9940	0.9940

Table 11 Experiment results

6.3.3 Discuss

The experimental results show that the proposed method in this study achieved an accuracy of 99.40%, a precision of 99.41%, a recall of 99.40%, and an F1 score of 99.40%. This indicates that the method proposed in this research is effective in classifying different types of malware. From the confusion matrix, it can be observed that misclassifications occurred in malware categories 0, 2, 3, 4, and 5. Specifically, categories 3 and 4 each had two misclassifications: two samples of category 3 were incorrectly identified as category 4, and two samples of category 4 were misidentified as category 3. This may be due to the similarity between categories 3 and 4. Additionally, a total of four samples were misclassified as category 4. Overall, the method proposed in this study performs well in classifying samples from the CCF BDCI malware classification dataset.

7. Discussion

7.1 Discussing Results for each Research Question

- Which deep learning model is optimal for extracting image representations of malware features and opcode Markov image features?

To address this issue, we conducted an experiment to evaluate the performance of different models in the malware classification method proposed in this study. Three deep learning models, widely used in the field of image processing, were selected for this research: VGG16, ResNet18, and EfficientNetB0. These three convolutional neural networks were tested in combination during the experiment. The results indicated that the model performed best when ResNet18 was used as the feature extractor for image-based malware representations, and EfficientNetB0 was used as the feature extractor for opcode Markov images.

- Can an effective technique for malware feature fusion be developed?

To validate that our proposed multilayer feature fusion method outperforms commonly used methods in existing research, we compared it with feature fusion methods based on concatenation, multilayer deep network, and cross-attention. The experimental results on the Microsoft Malware Classification Challenge dataset showed that our method achieved the best performance across all metrics, with an accuracy of 0.9949, a precision of 0.9950, a recall of 0.9949, and an F1 score of 0.9949. These results indicate that our method enables the model to more deeply learn the correlations between the two types of features, highlighting critical knowledge through information interaction.

- Can a robust and generalizable malware classification algorithm be developed?

To validate the accuracy of the proposed method for malware classification across different datasets, this study trained and evaluated the model using the CCF BDCI Malware Classification Dataset. The experimental results demonstrate that the proposed method achieves an accuracy of 99.40%, a precision of 99.41%, a recall of 99.40%, and an F1 score of 99.40% on this dataset. These results indicate that the proposed method is effective on other type of malwares.

7.2 Comparison with Existing Literature

Ahmed et al.[12] proposed a method for malware classification based on transfer learning using Inception V3, where the byte data of malware is converted into image features. On the Microsoft malware dataset, they achieved an accuracy of 98.76% and a recall of 94.8%. However, our approach achieved even better metrics, with an accuracy of 99.49% and a recall of 99.54%. This indicates that our method, which integrates malware image features with malware opcode features, can identify a broader range of malware category information. In comparison, the superior performance of our model can be attributed to the additional semantic information provided by the opcode features, enabling the classifier to capture deeper behavioral patterns of the malware.

Mallik et al.[13] proposed a method for classifying malware grayscale images using a convolutional recurrent network. Experimental results on the Microsoft Malware Classification Challenge dataset show that their method achieved an accuracy of 0.9836, a precision of 0.9940, a recall of 0.9688, and an F1 score of 0.9812. In contrast, our method achieved an accuracy of 0.9949, a precision of 0.9950, a recall of 0.9949, and an F1 score of 0.9949 on the same dataset. Overall, our method outperforms

theirs across all metrics. While the difference in precision between the two methods is minimal, our approach surpasses theirs by more than one percentage point in the other metrics. This indicates that when relying solely on malware grayscale images for classification, the model captures less informative content compared to when the features from malware images are combined with opcode features for classification.

Deng et al.[14] proposed a novel approach for generating malware images using assembly instructions and Markov transition matrices. Based on this, they designed a convolutional neural network (CNN) for malware classification. Their method achieved a test accuracy of 99.44% on the Microsoft malware dataset. In comparison, our feature fusion method achieved an accuracy of 99.49%, slightly surpassing their result, indicating a certain advantage in accuracy. Furthermore, their precision was 99.44%, while our precision reached 99.50%, suggesting that a higher proportion of files predicted as a certain type of malware indeed belonged to that category in our approach. Additionally, their recall rate was 99.13%, whereas ours was 99.54%, implying that our method was more effective in identifying various types of malware. Finally, the F1-score of Deng et al.'s method was 99.29%, while our method achieved 99.52%, demonstrating that our approach better handles minority classes in the imbalanced Microsoft malware dataset.

Zhao et al.[16] proposed a visualization-based method for malware family classification utilizing deep learning. They convert binary files into images and cluster the malware based on texture features within these images. The researchers employed a deep convolutional neural network to fuse and classify features from Markov images generated from both bytecode and opcode. Experimental results on the Microsoft Malware Classification Challenge dataset demonstrated that their model achieved an accuracy of 0.9976, precision of 0.9901, recall of 0.9881, and an F1 score

of 0.9891. In comparison, our method achieved an accuracy of 0.9949, precision of 0.9950, recall of 0.9949, and an F1 score of 0.9949 on the same dataset. Although our method slightly underperformed Zhao et al.'s method in terms of precision, our model excelled in other metrics. It is noteworthy that while both methods utilized Markov images generated from opcode for malware classification, Zhao et al. further incorporated Markov images generated from bytecode. Our method, on the other hand, is based on an image-based malware representation generated directly from binary files. Theoretically, this image-based malware representation should encompass more malware information than Markov images generated from bytecode, but our experimental results did not outperform their method. This could be attributed to the fact that ResNet18 may have inferior feature extraction capabilities compared to the convolutional neural network used in their approach. In future research, we will consider optimizing the convolutional neural network architecture to further improve classification performance.

Yang et al.[24] proposed a hybrid attention network based on multi-feature alignment and fusion for malware detection. This method first utilizes a 1D convolutional neural network to extract time series features of binary files and applies a triangular attention algorithm to extract opcode features from assembly code. Then, a cross-attention module is used to align and fuse the binary file features with the assembly code features, and finally, a deep neural network is employed to detect malware. Experimental results on the Microsoft Malware dataset demonstrate that this method achieves outstanding performance in terms of accuracy, precision, recall, and F1-score, reaching 99.54%, 99.40%, 99.41%, and 99.40%, respectively. In our study, although both methods employ cross-attention-based feature fusion strategies, our model exhibits better balance across performance metrics, achieving 99.49%

accuracy, 99.50% precision, 99.49% recall, and 99.49% F1-score. Specifically, while Yang et al.'s method slightly outperforms in accuracy, our model shows superior performance in precision, recall, and F1-score. This may indicate that the introduction of low-level feature fusion components in our approach further enhances the overall performance of the model.

Snow et al.[25] proposed an end-to-end multi-model deep learning framework for addressing the problem of malware classification. Their approach utilizes a fully connected network to process metadata, a convolutional neural network (CNN) to handle grayscale images derived from malware bytecode, and an LSTM network to process opcode sequences within malware files. On the Microsoft malware dataset, their method achieved an accuracy of 98.35%. In comparison, our approach reached an accuracy of 99.49%, significantly outperforming theirs. This indicates that, although their method also employs a multi-feature fusion mechanism, our method demonstrates more pronounced advantages in the comprehensiveness of feature selection and the effectiveness of the fusion mechanism.

8. Conclusion

8.1 Conclusion

Nowadays, malware has become a significant threat to information security, making the study of malware classification highly urgent. Traditional methods typically rely on single-feature approaches for malware classification, but these methods often struggle to cope with obfuscation techniques employed by malware to evade detection. Therefore, research on malware classification based on feature fusion is both necessary and promising.

We propose a malware classification method based on the fusion of malware images and malware opcode features, effectively classifying malware by fusing features from both image-based malware representation and opcode sources. Specifically, our model employs two feature fusion modules: a cross-attention-based feature fusion module and a low-level feature fusion module based on multiplication. These modules enable deep feature fusion, thereby classifying malware more effectively.

To determine the most suitable model for extracting image-based malware representation features and malware opcode Markov image features and accurately classifying malware, we conducted comparative experiments. We selected three common convolutional neural networks to extract the corresponding opcode information. Experimental results indicate that the best model combination is to use ResNet18 for extracting malware image features and EfficientNetB0 for extracting malware opcode Markov image features.

To further validate the effectiveness of our method, we compared the classification performance of traditional feature fusion methods with our approach. The results show that our method outperforms traditional methods in metrics such as accuracy and

recall. Moreover, to evaluate the classification effectiveness of our method on different types of malware, we tested it using the CCF BDCI 21 dataset. The experimental results demonstrate that our method can effectively classify malware on this dataset.

In conclusion, our research provides a reliable and effective new approach for malware classification and lays a solid foundation for future research on malware classification based on multi-feature fusion. Moving forward, research on malware classification methods based on multi-feature fusion will continue to address the increasingly complex malware threats.

8.2 Future Work

In future work, we will continue to delve deeper into malware classification methods based on multi-level feature fusion. Although we have already implemented a feature fusion method based on mutual attention, there remains significant room for further exploration in this area. Additionally, we plan to incorporate other deep learning models for feature extraction and investigate the potential application of additional features in malware classification.

9. Reflection

Throughout the research process, I encountered several challenges related to both technical aspects and time management. Initially, file operations and string processing posed significant difficulties for me. Due to the frequent need to handle files and accurately process strings during data processing, I spent a considerable amount of time on these tasks, which slowed down the progress of my research. However, through continuous practice and repeated trials, I gradually became proficient in these technical areas, leading to a significant improvement in processing efficiency.

In terms of time management, I noticed that my efficiency was relatively low at the beginning of the project, primarily because I did not plan the allocation of time between experiments and writing effectively. To address this issue, I decided to start writing the dissertation while conducting experiments. This approach of simultaneously conducting experiments and writing not only improved my work efficiency but also helped me maintain a consistent train of thought when documenting the research process. Additionally, communication with my supervisor played a crucial role. Through multiple discussions with my supervisor, I received valuable advice that helped me better plan the research timeline and overcome the challenges I faced during the project.

By overcoming these challenges, I not only enhanced my technical skills but also learned to manage my time more effectively. These experiences will have a profound impact on my future research and learning endeavors.

10. Reference

[1] "SonicWall 2024 Mid-Year Threat Report," SonicWall, American, 2024. [Online] Available: https://www.sonicwall.com/resources/white-papers/mid-year-2024-sonicwall-cyber-threat-report [Accessed Aug. 20, 2024].

[2] "VirusTotal Malware Trends Report: Emerging Formats and Delivery Techniques," VirusTotal, Spain, 2023. [Online] Available: https://blog.virustotal.com/2023/07/virustotal-malware-trends-report.html [Accessed Aug. 20, 2024].

[3] N. Singh, S. Tripathy and B. Bezawada, "SHIELD : A multimodal deep learning framework for Android malware detection ", Proc. Int. Conf. Inf. Syst. Secur., pp. 64-83, 2022.

[4] L. Nataraj, S. Karthikeyan, G. Jacob and B. S. Manjunath, "Malware images: Visualization and automatic classification", Proc. 8th Int. Symp. Vis. Cyber Secur., pp. 4, Jul. 2011.

[5] Tekerek and M. M. Yapici, "A novel malware classification and augmentation model based on convolutional neural network", Computers Security, vol. 112, pp. 102515, 2022.

[6] K. Shaukat, S. Luo and V. Varadharajan, "A novel deep learning-based approach for malware detection", Eng. Appl. Artif. Intell., vol. 122, Jun. 2023.

[7] R. Chaganti, V. Ravi and T. D. Pham, "Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification", J. Inf. Secur. Appl., vol. 69, Sep. 2022.

[8] V. Acharya, V. Ravi and N. Mohammad, "EfficientNet-based Convolutional Neural Networks for Malware Classification", International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-6, 2021.

[9] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan and T. D. Pham, "EfficientNet convolutional neural networks-based android malware detection", Comput. Secur., vol. 115, 2022.

[10] N. Lojain, A. Marwan, A. Abdullah and J. Anca, "Android Malware Detection Using ResNet-50 Stacking", Computers Materials & Continua, vol. 74, no. 2, pp. 3997-4014, 2023.

[11] M. Asam, S. J. Hussain, M. Mohatram, S. H. Khan, T. Jamal, A. Zafar, et al., "Detection of exceptional malware variants using deep boosted feature spaces and machine learning", Appl. Sci., vol. 11, no. 21, pp. 10464, Nov. 2021.

[12] M. Ahmed, N. Afreen, M. Ahmed, M. Sameer and J. Ahamed, "An inception v3 approach for malware classification using machine learning and transfer learning", Int. J. Intell. Netw., vol. 4, pp. 11-18, 2023.

[13] A. Mallik, A. Khetarpal and S. Kumar, "ConRec: malware classification using convolutional recurrence", J Comput Virol Hack Tech, 2022.

[14] H. Deng, C. Guo, G. Shen, Y. Cui and Y. Ping, "MCTVD: A malware classification method based on three-channel visualization and deep learning", Comput Secur, vol. 126, Mar. 2023.

[15] C. Gao et al., "Obfuscation-resilient Android malware analysis based on complementary features", IEEE Trans. Inf. Forensics Security, vol. 18, pp. 5056-5068, 2023.

[16] Z Zhao et al., "Malware classification based on visualization and feature fusion", 2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC). IEEE, 2021: 53-60.

[17] Mai, Changren et al., "MobileNet-Based IoT Malware Detection with Opcode Features." Journal of Communications and Information Networks 8.3 (2023): 221-230.

[18] S. Chen, B. Lang, H. Liu, Y. Chen and Y. Song, "Android malware detection method based on graph attention networks and deep fusion of multimodal features" in Expert Systems with Applications, Elsevier, vol. 237, pp. 121617, 2024.

[19] Xuan, Bona, Jin Li, and Yafei Song, "BiTCN-TAEfficientNet malware classification approach based on sequence and RGB fusion." Computers & Security 139 (2024): 103734.

[20] S. Li, Y. Li, X. Wu, S. A. Otaibi and Z. Tian, "Imbalanced malware family classification using multimodal fusion and weight self-learning", IEEE Trans. Intell. Transp. Syst., Oct. 2022.

[21] Sanjeev Kumar and Kajal Panda, "Sdif-cnn: Stacking deep image features using fine-tuned convolution neural network models for real-world malware detection and classification", Applied Soft Computing, vol. 146, pp. 110676, 2023.

[22] M. Dib, S. Torabi, E. Bou-Harb and C. Assi, "A multi-dimensional deep learning framework for IoT malware classification and family attribution", IEEE Trans. Netw. Service Manage., vol. 18, no. 2, pp. 1165-1177, Jun. 2021.

[23] S. Chen, B. Lang, H. Liu, Y. Chen and Y. Song, "Android malware detection method based on graph attention networks and deep fusion of multimodal features" in Expert Systems with Applications, Elsevier, vol. 237, pp. 121617, 2024.

[24] Xing Yang, Denghui Yang and Yizhou Li, "A hybrid attention network for malware detection based on multi-feature aligned and fusion", Electronics (Switzerland), vol. 12, 2 2023.

[25] E. Snow, M. Alam, A. Glandon and K. Iftekharuddin, "End-to-end multimodel deep learning for malware classification", Proc. IEEE Int. Joint Conf. Neural Netw., pp. 1-7, 2020.

[26] S. Madan, S. Sofat and D. Bansal, "Tools and techniques for collection and analysis of Internet-of-Things malware: A systematic state-of-art review", J. King Saud Univ.-Comput. Inf. Sci., vol. 34, no. 10, pp. 9867-9888, Nov. 2022.

[27] B. Wang, Y. Dou, Y. Sang, Y. Zhang and J. Huang, "IoTCMal: Towards a hybrid IoT honeypot for capturing and analyzing malware", Proc. IEEE Int. Conf. Commun. (ICC), pp. 1-7, 2020.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep residual learning for image recognition", Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016.

[29] Simonyan K. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[30] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks", Proc. Int. Conf. Mach. Learn., pp. 6105-6114, 2019.

[31] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks", Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

Appendice

Declaration of Originality

I Jinwei Xu declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : .....

Date : .....

FORM OF CONSENT

I Jinwei Xu hereby consent that my Project, submitted in candidature for the degree MSC Computer Networks and Cyber Security, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : .....

Date : .....