Article

A Novel Deep Learning Framework for IoT Malware Classification Integrating Feature Fusion and Attention Mechanisms

HAMZA JAVED *, ZENG FENG, SIDRA JAVED, Muhammad Shaheer

Posted Date: 29 January 2025

doi: 10.20944/preprints202501.1944.v1

Keywords: deep learning; feature fusion; malware attacks; IoT; attacks classification

A Novel Deep Learning Framework for IoT Malware Classification Integrating Feature Fusion and Attention Mechanisms

Hamza Javed [1,*], Feng Zeng [1], Sidra Javed [2] and Muhammad Shaheer [1]

[1] School of Computer Science and Engineering, Central South University, Changsha 410017, Hunan, China
[2] School of Software Technology, Dalian University of Technology, Dalian Liaoning 116024, China
* Correspondence: hamzajaved@csu.edu.cn

Abstract: The detection of malware attacks remains a significant challenge due rapid increase in variety of malicious files. An efficient system is crucial to ensure robust malware protection and to support post-attack recovery systems. In response to this challenge, we propose a novel deep learning-based framework which is designed to improve the accuracy and effectiveness of malware attacks detection. The framework employes two advanced pre-trained models including InceptionV3 and MobileNetV2, which are known for their robust feature extraction capabilities. To make the models computationally more efficient, we implement a truncation and compression process to eliminate redundant information, thereby refining the feature extraction workflow. Following this, we perform feature fusion process by combining the strengths of both models to create a more robust feature set. To further refine the combined features, we integrate a Squeeze and Excitation attention block, which enhances the model's ability to focus on the most relevant features for classification. This work addresses the complexities of malware classification in an evolving threat landscape. By effectively leveraging pre-trained models and enhancing them with feature fusion and attention mechanisms, our framework proves to be a robust tool for both binary and multi-class malware classification, making a significant contribution to cybersecurity. Our proposed framework was tested on two datasets. The first is an IoT malware dataset designed for binary classification, where the model achieved an accuracy of 97.09%. The second is the MALIMG dataset, which includes 25 distinct malware classes. On this dataset, the model achieved an accuracy of 97.47%. These results demonstrate the effectiveness of our approach in accurately classifying malware across different types and classes. We assessed the robustness of our model through a comprehensive analysis, including confusion matrix evaluations, ROC curve assessments, and class-wise performance analysis. These methods demonstrated the model's accuracy and reliability across different malware classes, further validating its effectiveness in real-world scenarios.

Keywords: deep learning; feature fusion; malware attacks; IoT; attacks classification

1. Introduction

Malware is also known as malicious software, which is designed to damage systems without the user's knowledge or consent. Malware attacks pose a significant risk to IoT devices because of their extensive usage and inadequate security protections. This list of vulnerabilities is an essential tool for developers and security experts to identify and address the most demanding security challenges facing IoT devices. It typically compromises a system's confidentiality, integrity, and availability, revealing it to illegal access and the variation of sensitive information[1] .Recent research highlights an alarming rise in malware attacks, emphasizing the urgent need to address these threats. The popularity of Internet of Things (IoT) devices has surged, enhancing human life quality and projected to grow to 43 billion units by 2023. IoT technology enables the transformation of physical objects into virtual ones with unique addresses, which can be controlled via widely used open-source Android devices [2]. This growing connectivity has made IoT devices particularly vulnerable to malware attacks. To enhance security and protect network infrastructure and its data, it is crucial to implement various essential security techniques, devices, and strategies throughout the entire infrastructure [3].McAfee Labs reported in 2019 that ransomware attacks experienced a significant increase of 118% during the first three months of the year, with new varieties of these threats also being identified [4]. This highlights a troubling trend in the cybersecurity landscape. Similarly, Kaspersky Security Network (KSN) documented in their findings that malware compromised the data of more than 70% of users, specifically targeting the covert collection of user data [5]. As reliance on digital technologies grows, individuals, institutions, and organizations are increasingly utilizing computers and databases to store not only essential but also sensitive information. This shift underscores the critical need to control and significantly reduce the escalating rate of malware attacks to protect these valuable digital assets.

IoT malware analysis is classified into static and dynamic approaches. Static detection typically involves signature-based methods, but these can be easily bypassed through obfuscation and may miss runtime vulnerabilities. In contrast, dynamic malware detection involves running applications in isolated environments like simulators or virtual machines. This secure and controlled setting allows for monitoring the behavior of a suspicious file to determine if it is normal or malicious. Traditional detection methods depend heavily on pre-existing signature libraries and require significant human intervention, making it challenging to keep pace with the rapidly expanding variety of malware[6].In academic discussions, two prevalent malware classification methods are Signature-based and Heuristic-based techniques. Signature-based classification involves checking files against a database with known malware signatures and identifying matches as small as 8-bit code patterns [7]. Despite its straightforward approach, it falls short against new malware variations due to its reliance on existing malware knowledge, which authors often avoid . On the other hand, Heuristic-based methods compare code against a database of known threats, flagging similar patterns. However, the scalability of this method is hindered by the exponential growth in malware, increasing database management challenges [8]. After discussing conventional methods like Signature-based and Heuristic-based malware classification, it's clear that their limitations make them ineffective against new and evolving threats. This inadequacy highlights the need for advanced techniques. As a result, the adoption of Machine Learning (ML), Deep Learning (DL), and advanced Intrusion Detection Systems (IDS) has become essential to address the complex landscape of cybersecurity threats effectively.

The deployment of an IDS is crucial to protect the three key aspects of network security. An IDS is designed to quickly detect various types of malware, which are unable to do by traditional firewalls [9]. Although there is a potential need for further enhancement. The corporations are actively improving network security by deploying advanced and intelligent intrusion detection solutions. Researchers have developed various machine learning-based methods to enhance the effectiveness of IDS for detecting a broad array of attacks. Given the complexity of networks such as the Internet of Things, traditional ML algorithms often struggle with the increasing load of data [10]. In recent years, the adoption of DL approaches has escalated due to their ability to process massive datasets through neural networks with multiple hidden layers. DL extracts insights and knowledge effectively across various disciplines to tackle a wide range of issues [11]. Unlike traditional methods like support vector machines and decision trees, deep learning offers a robust solution to diverse challenges. Specifically, researchers have recently implemented sophisticated deep learning techniques, such as Convolutional Neural Networks (CNN), in the field of cybersecurity.

2. Motivation and Major contribution

In the context of cybersecurity, the accurate detection and classification of malware are essential to protecting digital systems and ensuring data security. However, current methods face several challenges, including the rapidly increasing volume and complexity of malware, resource-heavy models, and the lack of comprehensive research on integrating feature fusion techniques in malware detection. To address these challenges, this study introduces a novel and efficient deep learning-based framework designed to improve the detection and classification of malware. Our approach not only enhances the accuracy and efficiency of malware detection but also offers a practical solution that can be effectively deployed in environments with limited resources. By applying model compression techniques followed by feature fusion from two deep learning models, our framework significantly enhances the performance of malware detection systems. This approach ultimately strengthens cybersecurity defenses and reduces the risk of malicious attacks. Our major contributions to this work are as follows:

- We introduce a novel and more efficient framework to enhance the accuracy and computational efficiency of malware attack detection. This framework leverages advanced feature fusion approach to effectively classify different types of malwares.
- Our approach involves the utilization and optimization of two advanced pre-trained models named InceptionV3 and MobileNetV2. By truncating and compressing these models, we eliminate redundant information, thereby enhancing the feature extraction process. This refinement significantly reduces computational overhead while maintaining high accuracy.
- We combine the feature extraction capabilities of both InceptionV3 and MobileNetV2 through a feature fusion process, creating a robust and comprehensive set of features. Additionally, the incorporation of a Squeeze and Excitation attention block allows the model to focus on the most critical features, thereby improving the precision of malware classification.
- The effectiveness of proposed method is validated using extensive experiments on two diverse datasets: an IoT malware dataset and the MALIMG dataset. Our model achieves a remarkable accuracy of 97.09% on the IoT malware dataset and 97.47% on the MALIMG dataset, demonstrating its robustness in both binary and multi-class malware classification tasks.
- These contributions not only advance the field of malware detection but also offer a practical solution for real-world cybersecurity applications, providing enhanced protection against a wide range of malware threats.

3. Related work

Malware detection involves identifying if an executable file is malicious and subsequently classifying it into its corresponding malware family. Increasingly, machine learning (ML) and deep learning (DL) techniques are enhancing Intrusion Detection Systems (IDS) by automating the detection process and enabling adaptability across various disciplines. These advanced techniques proceed through stages that include dataset construction, feature engineering, model training, and performance evaluation.

Several researchers [12–14] have adopted ML techniques such as SVM, AdaBoost, and Decision Trees in malware detection and classification. Specifically, Schultz et al. [15] have developed a static malware analysis algorithm that extracts features from programs, which are executables, strings, and byte n-grams. This algorithm utilizes a classifier named as Multinomial Naïve Bayes(MNB) for classification and has achieved an accuracy of 97.11%. The increasing incidence of malware attacks on IoT devices demonstrates the inadequacies of traditional methods, which depend heavily on signature libraries and the expertise of malware analysts. In contrast, Deng et al. [16] highlight that ML and DL techniques provide a more effective and automatically adaptable solution. Deep learning has been particularly instrumental in developing robust IDS systems capable of detecting a variety of malicious network assaults using diverse algorithms. Vishwakarma et al. [11] developed a deep learning-based anomaly-IDS model for IoT devices to classify various types of attacks. This model amalgamated four distinct datasets including BoT-IoT, ToN-IoT, CSE-IDS-2018, and UNSW-NB15, forming a single dataset with 21 attack classes and nine common features. It includes five hidden layers with batch normalization implemented in the second and fourth layers, and it was evaluated using both binary and multiclass classifications. Zhang et al. [13] introduced an advanced algorithm specifically designed for classifying ransomware. This algorithm employs n-grams of opcodes in order to analyze the static of malware. Extensive testing on real-world datasets has proven the effectiveness of this method, achieving an impressive accuracy rate of 91.43%. Angelo et al.[17]

developed a malware classification method using association rules, API call subsequences, and a Markov chain, enhanced by an S-DCNN with an automatic feature extractor. This approach achieved 93.93% accuracy with a Decision Tree on the MCD detector in dynamic analysis scenarios, , effectively handling evasion techniques that disrupt API call sequences. Jain et al.[18] explored malware classification by visualizing malware as images to avoid costly feature extraction processes. They compared Convolutional Neural Networks (CNNs) with Extreme Learning Machines (ELMs), finding that ELMs provided comparable accuracy to CNNs but required significantly less training time. This efficiency was observed with both one-dimensional and two-dimensional data. Kasongo et al.[19] developed a deep learning-based IDS model using a feature selection approach with the Extra-Trees classifier to rank features on the UNSW-NB15 and AWID datasets. The top-ranked features were used in a feed-forward deep neural network (FFDNN) for attack classification. This model outperformed traditional models like decision trees, SVMs, and random forests in both binary and multiclass classification. Naeem et al. [20] introduced an effective Malware Image Classification System framework for IoT structure. This system basically converts the malware files into grayscale images and then extracts both the global and local features from these images for final predictions. Vasan et al.[21] developed IMCFN, which is a CNN-based model and used to change the binary malware files into colored images for classification. This method employs data augmentation technique to fine-tune a DCNN which is already trained on the ImageNet vast dataset. In other study, Vasan et al. [22] introduced an ensemble models for malware classification. This architecture includes ResNet50 and VGG16 models, which are fine-tuned on malware images. Kumar et al.[23] proposed IMCNN, a deep CNN for malware detection using pre-trained models like VGG16, VGG19, InceptionV3, and ResNet50 for feature extraction. The method was tested on real-world malware datasets, achieving remarkable accuracy of 92.11% on the real-world dataset.

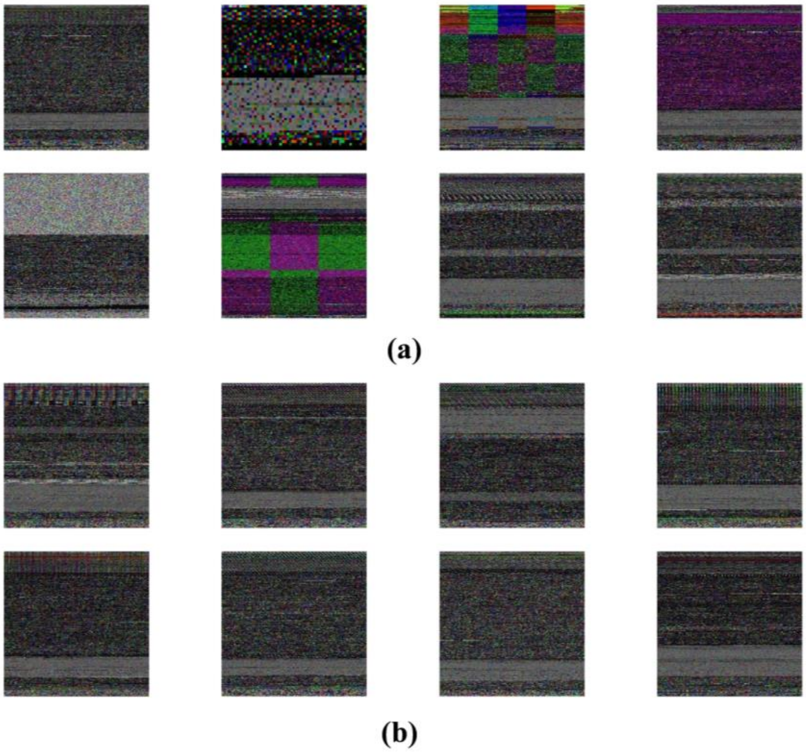
4. Proposed Methodology

4.1. Dataset Description

In our study on malware detection within IoT networks using deep CNN, we transform network packets into image representations to leverage the effective processing capabilities of CNNs. This method enhances the detection of malware intrusions by allowing the model to analyze data as images. Our study employs the IoT_Malware dataset [6]. The IoT_Malware dataset is based on byte sequences from executable files and is divided into two main classes. It contains 2,483 images representing benign files and 2860 images representing malware. This dataset facilitates precise classification by organizing data into distinct categories. Table 1 presents the distribution of these classes, and Figure 1 illustrates the visual representations of the images. The images are resized to a uniform shape of 224 × 224 pixels to standardize the input for the CNN model.

Table 1. Data distribution of IoT_Malware.

Classes	Train images	Test images	Total images
benign	1989	494	2,483
malware	2290	570	2860



(a)

(b)

Figure 1. IoT_Malware dataset samples (a) malware (b) benign images.

4.2. Pre-trained models for feature extraction

To develop a rapid and efficient detection method, we chose InceptionV3 and MobileNetV2 as the base models for extracting frame-level features. These models were selected due to their unique architectural designs, which emphasize computational efficiency while maintaining high accuracy. Inception-V3 employs an advanced architecture that efficiently captures intricate patterns without excessive computational demands. MobileNet-V2 is designed to deliver high performance in resource-constrained environments by using depthwise separable convolutions. These architectural characteristics make both models ideal for our approach. Below, we provide a detailed exploration of the distinguishing features of these models.

- **InceptionV3**

InceptionV3 is a powerful convolutional neural network architecture known for its efficiency and accuracy in image classification tasks [24]. We chose InceptionV3 for its ability to balance computational cost and performance, making it ideal for applications requiring rapid processing without sacrificing accuracy. This model incorporates several innovations, such as factorized convolutions and aggressive dimensionality reduction, which help to reduce the computational load while capturing complex patterns in data. InceptionV3 uses multiple parallel convolutional paths with different filter sizes, allowing it to recognize patterns at various scales within the same layer. This multi-scale processing capability enhances its ability to generalize across different image features. Additionally, its architectural design includes auxiliary classifiers that improve convergence during training and help in combating the vanishing gradient problem. These features make InceptionV3 a robust and versatile choice for extracting frame-level features, providing a comprehensive understanding of the data while maintaining high efficiency.

- **MobileNet-V2**

MobileNetV2 is a neural network architecture specifically designed to perform well on lightweight devices such as smartphones and embedded systems [25]. This model improves upon the original MobileNet by incorporating advanced techniques. These techniques enhance both computational efficiency and accuracy. One of the key innovations in MobileNetV2 is the use of depthwise separable convolutions (DSC), which significantly reduce the number of parameters and computational cost. Depthwise separable convolutions work by splitting a standard convolution operation into two separate processes including depthwise convolution and pointwise convolution. Depthwise convolution applies a single filter to each input channel and pointwise convolution combines these outputs using a 1x1 convolution. This factorization significantly reduces the computational demands, resulting in a network that is both lighter and faster while maintaining high performance. Additionally, MobileNetV2 introduces a novel architectural element known as the inverted residual block. This block enhances the model's capacity to capture features while maintaining a low computational cost. The inverted residual structure allows the model to preserve information across layers more effectively, enabling faster convergence during the training process and achieving enhanced accuracy in comparison to original architecture of MobileNet. Overall, MobileNetV2 is a highly effective and precise model that is ideally suitable for deployment in environments with limited computational resources. Its design makes it particularly advantageous for mobile applications, where it can provide high performance without draining battery life or requiring extensive processing power. This balance of efficiency and accuracy makes MobileNetV2 an excellent choice for a wide range of applications, from real-time image classification to other machine learning tasks on embedded devices.

4.3. Models' truncation and compression

The Inception-V3 and MobileNet-V2 models were modified through model truncation to make them more compact and reduce their parameter size. This process aimed to create lighter models without losing their core functionality or design principles. Even though these models became simpler, but they maintained their main design and functional capabilities. Research has demonstrated that these truncated models can still deliver strong performance when trained on smaller datasets and do not experience significant overfitting .To achieve this reduction in size, the classifier from the top of these DL backbone models was eliminated. Additionally, certain blocks within the models were eliminated, leading to a decrease in the number of trainable parameters. For instance, the original architecture of InceptionV3 model consists of approximately 21.8 million parameters before applying truncation. After applying truncation to this model, this number was significantly reduced to 474,528 parameters. Similarly, the original MobileNetV2 model had 2.5 million parameters and 16 blocks, but the parameters were reduced to 139,040 parameters by using only 7 core blocks. This reduction in parameters and blocks made the models much smaller while still retaining their important features. This process allows the models to maintain good performance and efficiency, making them suitable for use in environments with limited computational resources or for tasks that require faster processing times. Despite the reduction in complexity, these compressed models continue to provide effective solutions for machine learning tasks. These truncated models are topped with several layers, including SeparableConv2D, AveragePooling2D, and AlphaDropout. The SeparableConv2D layer performs efficient depthwise separable convolutions. This process extracts spatial features while reducing computational complexity. Subsequently, the AveragePooling2D layer down-samples the feature maps. This helps in retaining significant features while reducing dimensionality. Finally, AlphaDropout is applied to prevent overfitting. It randomly deactivates some neurons during training while maintaining the network's mean and variance for stability. These layers enhance the model's ability to generalize and improve overall performance.

4.4. Squeeze and Excitation Block

The Squeeze-and-Excitation (SE) block is a crucial architectural component that significantly enhances the representational power of convolutional neural networks (CNNs) [26]. It improves the network's performance by dynamically adjusting the weights of each feature map. This adjustment allows the network to recalibrate features on a channel-wise basis, which is achieved through two primary operations: squeeze and excitation, as illustrated in Figure 2.The squeeze operation uses global average pooling to condense each feature map into a single value per channel. This process captures global spatial information across the entire image. The excitation operation follows by employing a small neural network to learn dependencies across channels. It generates a set of modulation weights that are used to scale the original feature maps. This scaling process highlights important features while reducing the influence of less significant ones.

One of the key benefits of SE blocks is their ability to enhance model performance with minimal additional computational cost. By explicitly modeling the interdependencies between channels, SE blocks enable the network to focus on more informative features while suppressing irrelevant ones. This selective attention leads to improved accuracy in image classification tasks. SE blocks are preferred over other attention mechanisms due to their simplicity and effectiveness. They can be easily integrated into existing architectures such as ResNet, Inception, and MobileNet with minor modifications. The adaptive recalibration provided by SE blocks enhances the network’s ability to discriminate between different features, increasing robustness and improving feature distinction. This increased robustness contributes to achieving state-of-the-art performance on benchmarks like ImageNet, demonstrating the effectiveness of SE blocks in a wide range of tasks. The integration of SE blocks into CNN architectures not only boosts performance but also helps to maintain efficiency, making them a popular choice for enhancing deep learning models.

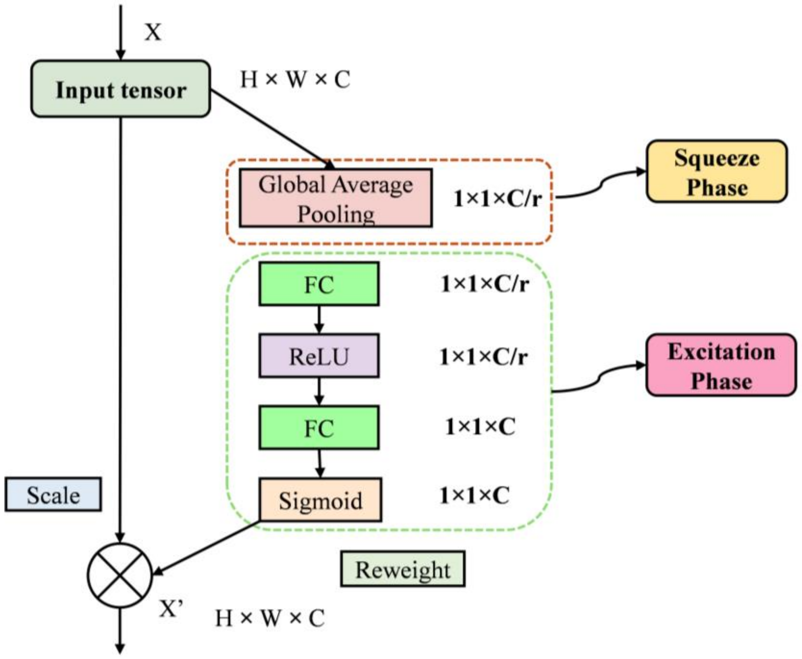


Figure 2. Workflow architecture of Squeeze and Excitation Block.

4.5. Proposed Feature Fusion Architecture

In our research, we introduced a Feature Fusion (FF) approach that utilizes MobileNetV2 and InceptionV3 as backbone models. To achieve a balance between performance and efficiency, these models are strategically compressed, thereby reducing their complexity without compromising their overall effectiveness. This approach leverages the unique strengths of MobileNetV2 and InceptionV3, capitalizing on their complementary capabilities to represent features more effectively. By combining feature maps from these two distinct architectures, our technique integrates a wide range of diverse features, enhancing the model's capability to capture detailed and intricate patterns within the input data. This fusion process generates a robust and comprehensive feature set, significantly improving the model's ability to understand and learn complex patterns from the data it processes. Consequently, the combined power of both architectures contributes to superior performance in recognizing and interpreting complex structures, leading to more accurate and reliable outcomes in various applications.

Following the feature fusion, we incorporate a Squeeze-and-Excitation (SE) block to refine the fused features and enhance the overall performance of the network. The SE block significantly boosts the representational power of the model by recalibrating the feature channels in a dynamic manner. This recalibration is achieved through a series of steps involving the squeeze and excitation operations, each playing a crucial role in optimizing the importance of feature maps. The squeeze operation begins by applying global average pooling across each feature map, effectively reducing the spatial dimensions of the feature maps and summarizing the information into a single scalar value for each channel. This operation captures the global context of the input data, enabling the model to consider broader spatial information when assessing the significance of each feature. By compressing the spatial information, the squeeze operation provides a compact and efficient representation that serves as the foundation for the subsequent stage. The excitation operation follows the squeeze step, where a small neural network learns inter-channel dependencies and relationships. This is achieved by using a two-layer fully connected network, which first reduces the dimensionality of the squeezed features to capture critical dependencies and then expands them back to the original number of channels. This network effectively generates modulation weights for each channel, allowing the SE block to selectively emphasize the most relevant features and suppress those that contribute less to the task. The resulting channel-wise attention mechanism fine-tunes the feature maps, enabling the network to prioritize critical information and discard noise, thus improving its capacity to recognize important patterns. Once the SE block refines the features through this recalibration process, the model proceeds with a series of layers designed to produce the final output. A flatten layer is employed to transform the multi-dimensional feature maps into a one-dimensional vector, preparing the data for subsequent fully connected layers. To mitigate the risk of overfitting during training, a dropout layer is introduced, randomly deactivating neurons while preserving valuable feature representations. This dropout mechanism ensures that the model does not become overly dependent on specific features, advancing generalization to unseen data.

The final stage of the architecture involves a dense classification layer, which contains neurons equal to the number of target classes. This layer acts as the decision-making component of the network, processing the refined and flattened feature vector to generate the final predictions. The entire process is meticulously designed to ensure efficiency, accuracy, and resilience to variations in input data. The process begins with feature fusion, where distinct feature sets are combined to enhance the representational power of the model. Subsequently, the Squeeze-and-Excitation (SE) block refines these features by selectively emphasizing informative attributes while suppressing less useful ones. The workflow concludes at the final classification stage, where the processed features are used to make accurate predictions. Each phase is structured to build upon the previous, ensuring comprehensive robustness and optimized performance throughout the model. The architecture systematically incorporates several key components such as feature fusion, SE block recalibration, flattening, dropout regularization, and dense classification to make a robust framework tailored for image classification tasks. This approach effectively balances complexity with performance, offering a solution that can adeptly handle diverse and complex data. The overall feature fusion architecture is illustrated in Figure 3.

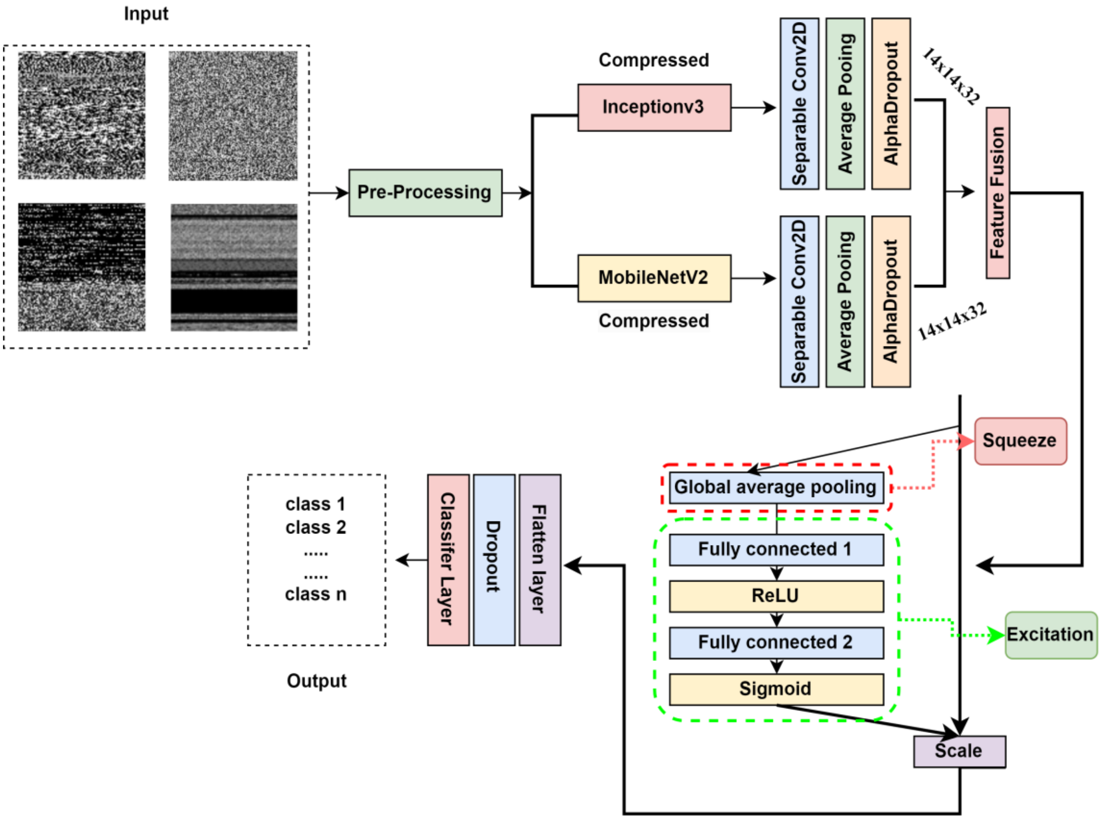


Figure 3. Overall feature fusion frameworks.

5. Experimental Results and Analysis

This section outlines the implementation details and evaluation results of the proposed feature fusion architecture, which is specifically designed for the classification of attacks. We detail the steps involved in constructing the fusion model and the methodologies used for training and testing. The effectiveness of our fusion model is assessed by comparing its performance against existing models, highlighting its superior accuracy and robustness in classifying various attack types. This comparison demonstrates the advantages of our approach in enhancing classification accuracy and improving overall model reliability.

5.1. Implementation Details

In this section, we outline the implementation details utilized for training the proposed model. Our research leverages the Keras framework and Python to conduct experiments on the Feature Fusion model. All experiments were performed within a Python environment, making full use of GPU runtime to enhance computational efficiency. We utilized an Nvidia Tesla K80 GPU, equipped with 16 GB of RAM and 512 GB of storage, to conduct our experiments. This setup enabled efficient processing and management of large datasets, which is essential for the successful training and evaluation of our model.

5.2. Hyperparameters details

We enhanced the reliability of our proposed method by carefully selecting optimal hyperparameters for training. The model was trained with a batch size of 64 and over 20 epochs. The learning rate was set to 1.0e-3 to ensure optimal performance. This configuration was chosen to make efficient use of computational resources while ensuring effective convergence. The improved performance of the model can be largely attributed to the Adam optimizer. Adam is well-regarded for its ability to provide higher accuracy and efficient memory usage. It is particularly effective

doi:10.20944/preprints202501.1944.v1

because it optimizes the behavior of sparse gradients and dynamically adjusts the learning rate for better and faster convergence. Additionally, our approach heavily relies on the categorical cross-entropy loss function. This function is essential for assessing how accurately the model predicts categorical labels. It plays a crucial role in our training process by guiding the model to learn accurately and generalize well to new, unseen data by minimizing the loss.

5.3. Performance Evaluation

Evaluating the performance of learning models is essential to determine their effectiveness. In our study, we use key metrics such as accuracy, precision, recall, and the F1 score to assess performance. These metrics are derived from the confusion matrix, which provides valuable insights into the classifier's performance on test data. The metrics are calculated using values from true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), with each value ranging from 0 to 1. Accuracy measures the model's ability to correctly predict both positive and negative classes. Precision is the ratio of true positives to the sum of true positives and false positives. Recall measures the proportion of true positives against all actual positives, including false negatives. The F1 score, which is the harmonic mean of precision and recall, ranges between 0 and 1, offering a balanced view of these metrics. These calculations help us accurately gauge the model's performance. Mathematically, these metrics are used to precisely measure the model's performance.

$$\text{Accuracy: } \frac{tp+tn}{tp+fp+fn+tn} \tag{1}$$

$$\text{Precision: } \frac{tp}{tp+fp} \tag{2}$$

$$\text{Recall: } \frac{tp}{tp+fn} \tag{3}$$

$$\text{F1-Score}=2\times \frac{(Precision\times Recall)}{(Precision+Recall)} \tag{4}$$

5.4. Performance Analysis

We evaluated the robustness of our model using the IoT-Malware attacks dataset, as described in section 3.1. The model's performance was analyzed using a confusion matrix which is a powerful method for detecting misclassifications in the test data. A confusion matrix is a valuable tool for evaluating the performance of classification models by showing the number of correct and incorrect predictions for each class. It categorizes predictions into four groups: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). True positives and true negatives are instances correctly identified by the model, while false positives and false negatives represent errors where the model has incorrectly labeled the observations. We conducted a detailed evaluation of three distinct models to determine their effectiveness in classifying samples. Figure 4(a) displays the confusion matrix of the base model InceptionV3, which correctly classified 994 out of 1064 total test samples. It misclassified 70 samples, indicating robust performance with room for further improvement. Figure 4(b) illustrates the confusion matrix for the MobileNetV2 model, which shows a slight improvement in classification accuracy. This model correctly identified 999 samples and misclassified 65, effectively lowering the error rate compared to InceptionV3.Figure 4(c) depicts the confusion matrix for the Proposed Model, which demonstrates the best performance among the three. It correctly classified 1033 samples, significantly minimizing the misclassifications to only 31. This performance indicates a high level of accuracy and efficiency, highlighting the Proposed Model's advanced capability to accurately differentiate between classes. Such high accuracy and a low misclassification rate highlight the model’s advanced capability to handle complex classification tasks, making it a robust choice for practical applications where precision is critical.

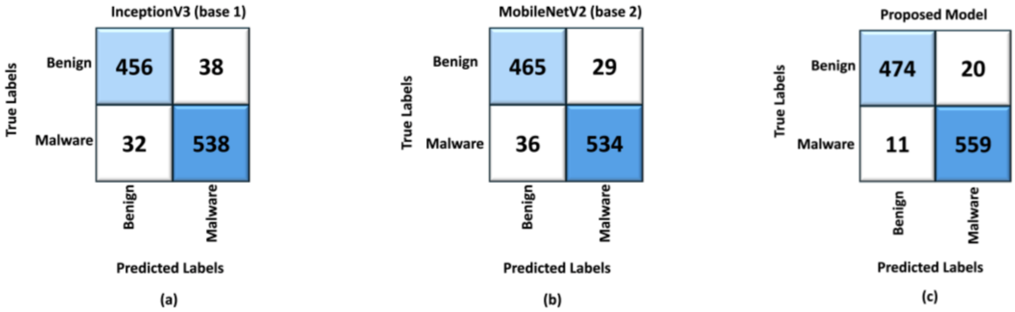


Figure 4. Confusion matrix of base models and proposed models on test set.

Table 2 illustrates the class-wise performance metrics for three models including inceptionV3 (base 1), mobilenetV2 (base 2) and the Proposed Model. These metrics include precision, recall, F1-score, and overall accuracy for classifying benign and malware samples. When comparing the two base models, both show similar performance with subtle variations across the metrics. InceptionV3 (Base 1) generally achieves slightly higher recall values for malware, signifying it is marginally better at identifying all actual malware samples. Conversely, mobilenetV2 (Base 2) exhibits higher precision for malware, indicating fewer false positives in its predictions. The Proposed Model outperforms both base models across all metrics for both classes. The Proposed Model demonstrates substantially improved precision and recall for the benign class, leading to an exceptionally high F1-score. This high score indicates well-balanced accuracy, effectively combining precision and recall. In the case of the malware class, the Proposed Model surpasses the base models in both precision and recall, with a particularly notable enhancement in precision. This improvement emphasizes the model's capability to significantly reduce false positives, which is crucial for reliable malware detection. This enhanced accuracy makes the Proposed Model highly effective and dependable for identifying malware threats.

Our model achieves superior performance primarily due to the integration of a feature fusion approach and the inclusion of a Squeeze-and-Excitation (SE) block. The feature fusion method effectively combines the strengths of various pre-trained models, enabling the extraction of a robust set of features. This approach ensures that critical information from various aspects of the data is captured and utilized, enhancing the ability of model to distinguish subtle differences between benign and malware images. Additionally, the Squeeze-and-Excitation block further refines the feature representation by performing channel-wise recalibration. This process selectively emphasizes important features and suppresses less useful ones, improving the model's focus and sensitivity to relevant patterns. The SE block's ability to adaptively adjust feature importance based on the data contributes significantly to the model's overall accuracy and robustness, making it particularly effective in complex classification tasks such as malware detection.

Table 2. Classwise performance of proposed model.

Models	Classes	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)
InceptionV3 (Base 1)	Benign	93.44	92.31	92.87	93.42
	Malware	93.40	94.39	93.89	
MobileNetV2 (Base 2)	Benign	92.81	94.13	93.47	93.89
	Malware	94.85	93.68	94.26	
Proposed Model	**Benign**	**97.73**	**95.95**	**96.83**	**97.09**
	Malware	**96.55**	**98.07**	**97.30**	

Figure 5 illustrates the Receiver Operating Characteristic (ROC) curves for various pretrained models and the Proposed Model. The ROC curve is a graphical illustration that demonstrates the indicative ability of a binary classifier system as its discrimination threshold is varied. The curve illustrates the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) across different threshold settings. The Area Under the Curve (AUC) serves as a measure of each model's effectiveness in distinguishing between benign and malware attacks. The Proposed Model achieves the highest AUC of 0.9953, indicating exceptional performance and an excellent balance between sensitivity and specificity. InceptionV3 follows closely with an AUC of 0.9815, while Xception also shows strong performance with an AUC of 0.9835. MobileNet and MobileNetV2 display slightly lower AUC values of 0.9829 and 0.9801 respectively, but still indicate very good classification capabilities. The steep rise of each curve towards the upper left corner reflects a high true positive rate with a low false positive rate, ideal for effective classification models. The Proposed Model's curve is closest to the top-left corner, signifying its superior performance in minimizing false positives and maximizing true positives compared to the other models tested.

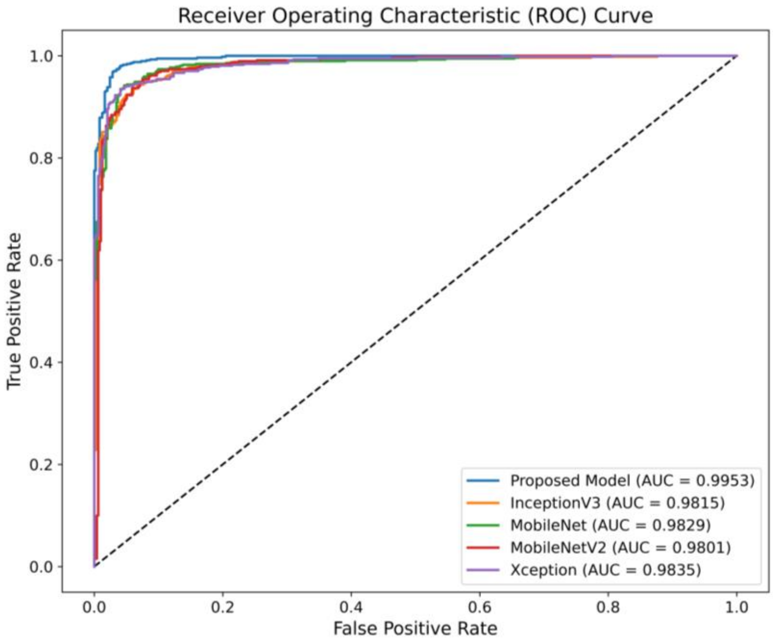


Figure 5. Roc curve of proposed model and base models.

5.5. Ablation study

Table 3 illustrates the impact of incorporating an attention block into the Proposed Model on its performance metrics across two classes: Benign and Malware. The table compares models with and without the attention block using precision, recall, F1-score, and overall accuracy. For the benign class, the Proposed Model without the attention block achieves a precision of 97.92%, a recall of 95.14%, an F1-score of 96.51%.When the attention block is added, the precision slightly decreases to 97.73%, but the recall improves to 95.95%, resulting in a slightly better F1-score of 96.83% .In the case of the malware class, the model without the attention block records a precision of 95.89% and an impressive recall of 98.25%, with an F1-score of 97.05%. With the attention block, the precision increases to 96.55%, and the recall slightly decreases to 98.07%, leading to a higher F1-score of 97.30% and maintaining high overall accuracy. The overall accuracy for the model without the attention block stands at 96.80%, which increases to 97.09% when the attention block is added, indicating an enhancement in performance across both classes. These results demonstrate the insertion of attention block improves the ability to model to focus on the most crucial relevant features, slightly improving both the precision and the F1-scores for both classes. The role of attention block in refining feature processing contributes to a balanced improvement in the model's diagnostic capabilities, confirming its effectiveness in achieving more accurate and reliable classifications.

Table 3. The impact of attention block on proposed model.

Models	Classes	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)
Proposed Model w/o attention block	Benign	97.92	95.14	96.51	96.80
	Malware	95.89	98.25	97.05	
Proposed Model with attention block	Benign	97.73	95.95	96.83	97.09
	Malware	96.55	98.07	97.30	

5.6. Additional experiments on Malimg dataset

To further assess the robustness of the proposed model, we incorporated the Malimg dataset into our evaluation. This additional dataset allowed us to verify the model's effectiveness across a broader range of samples. Additionally, we employed the Malimg Dataset [27], which includes 9,339 image-based representations of malware binaries across 25 classes, each corresponding to a malware family. This dataset structure allows for comprehensive analysis and classification. Figure 6 provides visual samples for each malware family of Malimg dataset.

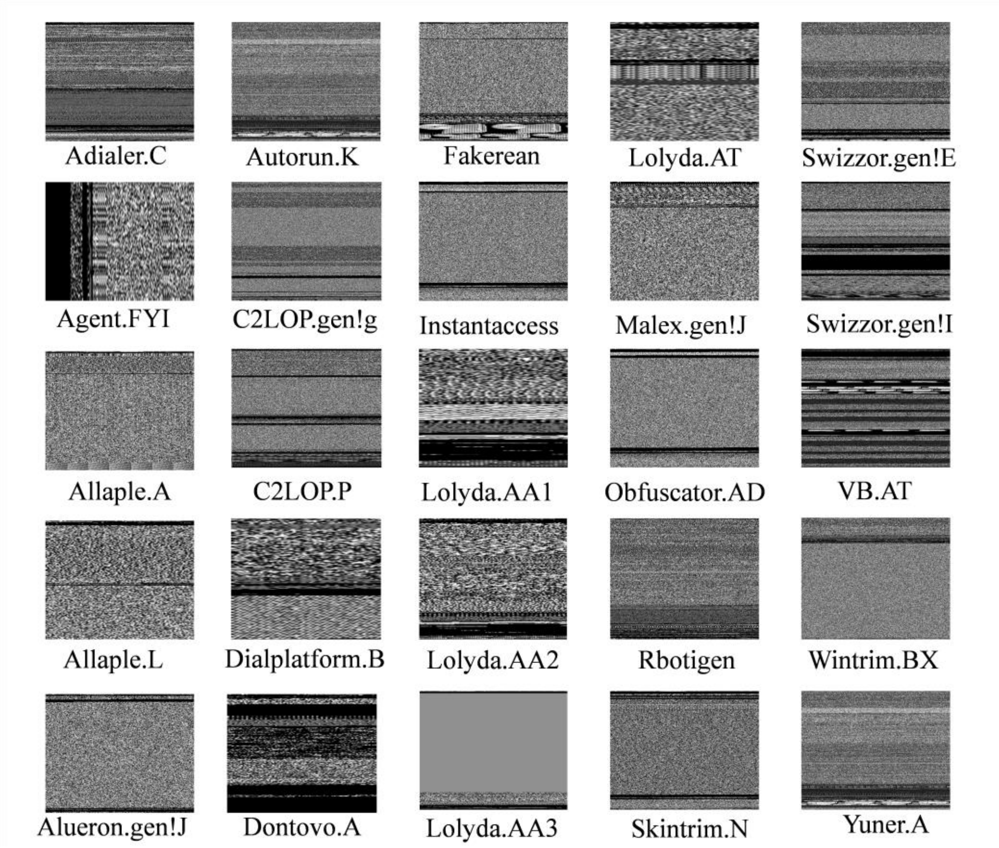


Figure 6. visual samples from Malimg dataset from each class.

5.7. Performance Analysis on Malimg dataset

To evaluate the effectiveness of the proposed model, we conducted additional experiments using the Malimg dataset. The model demonstrated impressive performance, as outlined in Table 4 below. It consistently outperformed other approaches across various classes, showcasing its robustness and accuracy in malware classification tasks. Several classes, including Adialer.C, Agent.FYI, Allaple.L, Alueron.gen!J, Autorun.K, Dialplatform.B, Dontovo.A, Instantaccess, Lolyda.AA1, Lolyda.AA2, Lolyda.AA3, Lolyda.AT, Malex.gen!J, Obfuscator.AD, Rbotgen, Skintimr.N, VB.AT, and Yuner.A, demonstrate perfect scores across all these metrics. For these classes, the model achieved a Precision, Recall, and F1-Score of 1.000, indicating that the model correctly identified every sample of these classes without any false positives or false negatives. Consequently, the Accuracy for these classes is also 1.000, showing that every sample was accurately classified by the model.

However, not all classes recorded this level of performance such as Allaple.A exhibits a slight drop in Precision, with a score of 0.9921, while maintaining a perfect Recall of 1.000. This indicates that while the model correctly identified all instances of this class, there were a few false positives, leading to an F1-Score of 0.9960. C2LOP.gen!g presents a more notable drop in Precision at 0.8621, though it still retains a perfect Recall, resulting in an F1-Score of 0.9259. This suggests a higher rate of false positives, despite the ability of model to correctly identify all images of this class. The class C2LOP.P shows a Precision of 0.9565 and a Recall of 0.8800, leading to an F1-Score of 0.9167. This reflects the presence of both false positives and false negatives, indicating challenges in the model's ability to correctly classify this class.

The class Fakerean achieved a perfect Precision of 1.000 but shows a slightly lower Recall of 0.9867 resulting in an F1-Score of 0.9933. This outcome shows that the model effectively avoided false positives, meaning it did not wrongly classify non-malicious instances as attacks. However, the model missed some actual attacks, leading to a few false negatives where real threats were not identified. Although the number of missed attacks was small, it indicates that the model did not detect every malicious instance perfectly. Similarly, Wintrim.BX shows a slight drop in Precision at 0.9615, though its Recall remains perfect at 1.000, resulting an F1-Score of 0.9804. This indicates some false positives but no missed instances of this class. The classes Swizzor.gen!E and Swizzor.gen!I are particularly challenging, with Swizzor.gen!E achieving a Precision of 0.6538 and a Recall of 0.6800, resulting in an F1-Score of 0.6667, and Swizzor.gen!I achieve even lower metrics, with a Precision of 0.6667, a Recall of 0.5600, and an F1-Score of 0.6087. These low scores reflect significant difficulties in correctly classifying these classes, with both high false positive and false negative rates.

In overall comparison, the proposed model shows a strong performance, with an overall accuracy of 0.9747, indicating that the model performs very well across the dataset. However, the challenges presented by classes like Swizzor.gen!I and Swizzor.gen!E highlight specific areas where the model struggles, potentially due to the similarities of these classes to others or the inherent difficulty in distinguishing their features. Despite these challenges, the model excels in identifying many other classes with perfect precision and recall, demonstrating its robustness and effectiveness in general. To further enhance the model's overall effectiveness, addressing the classification challenges of the more difficult classes could be a focus for future improvements. Our model demonstrates outstanding performance largely due to its carefully designed architecture that strategically integrates multiple advanced techniques. Our model architecture leverages the strengths of two powerful models, Inception V3 and MobileNetV2, which are known for their ability to capture a wide range of features. Inception V3 excels at handling complex patterns and varying scales in the data, while MobileNetV2 offers a lightweight, efficient architecture ideal for mobile and embedded applications. After extracting features using these models, we apply truncation and compression techniques to reduce dimensionality and retain only the most relevant information. This process ensures that the model is both efficient and focused on the most critical aspects of the data, which helps reduce computational complexity without sacrificing accuracy. Subsequently, feature fusion is performed to combine the strengths of both models, creating a robust and comprehensive feature set. This fusion step is crucial as it enhances the model's ability to make well-informed decisions by incorporating a richer set of features that capture various aspects of the input data. Additionally, the incorporation of a Squeeze and Excitation block further refines the model's focus. This block works by adaptively recalibrating the feature maps, enabling the model to prioritize the most informative channels while suppressing less relevant ones. This selective attention mechanism significantly boosts the model's ability to distinguish between subtle differences in the input data, leading to more accurate predictions. Finally, the refined features are passed through a SoftMax classifier, which outputs the probability distribution over the possible classes. This step ensures that the model makes precise and confident predictions, contributing to its overall superior performance. The combination of these carefully selected components and processes results in a model that not only performs exceptionally well but also generalizes effectively across various tasks.

Table 4. Classwise performance analysis on Malimg dataset.

Model	Classes	Precision	Recall	F1-score	Accuracy
	Adialer.C	1.0000	1.0000	1.0000	
	Agent.FYI	1.0000	1.0000	1.0000	
	Allaple.A	0.9921	1.0000	0.9960	
	Allaple.L	1.0000	1.0000	1.0000	
	Alueron.gen!J	1.0000	1.0000	1.0000	
	Autorun.K	1.0000	1.0000	1.0000	

	C2LOP.gen!g	0.8621	1.0000	0.9259	
	C2LOP.P	0.9565	0.8800	0.9167	
	Dialplatform.B	1.0000	1.0000	1.0000	
	Dontovo.A	1.0000	1.0000	1.0000	
	Fakerean	1.0000	0.9867	0.9933	
Proposed Model	Instantaccess	1.0000	1.0000	1.0000	0.9747
	Lolyda.AA1	1.0000	1.0000	1.0000	
	Lolyda.AA2	1.0000	1.0000	1.0000	
	Lolyda.AA3	1.0000	1.0000	1.0000	
	Lolyda.AT	1.0000	1.0000	1.0000	
	Malex.gen!J	1.0000	1.0000	1.0000	
	Obfuscator.AD	1.0000	1.0000	1.0000	
	Rbotigen	1.0000	1.0000	1.0000	
	Skintrim.N	1.0000	1.0000	1.0000	
	Swizzor.gen!E	0.6538	0.6800	0.6667	
	Swizzor.gen!I	0.6667	0.5600	0.6087	
	VB.AT	1.0000	1.0000	1.0000	
	Wintrim.BX	0.9615	1.0000	0.9804	
	Yuner.A	1.0000	1.0000	1.0000	

5.8. Limitations and future work

5.8.1. Limitations

- Although the use of lightweight models like MobileNetV2 aims to reduce complexity, the combination of multiple models, feature fusion, and the Squeeze and Excitation block still demands considerable computational power.
- The model's performance might not be consistent across various datasets, especially when the data distribution differs significantly from that of the training set.
- The model may struggle to meet the requirements for real-time applications due to the additional processing steps involved, such as feature fusion and attention mechanisms.

5.8.2. Future work

- Future research can focus on optimizing the model to reduce computational overhead, making it more suitable for deployment on devices with limited resources.

- Expanding the evaluation of the model to include a broader range of datasets from different domains will help in assessing and improving its generalizability.
- Developing more efficient processing techniques or simplifying the model's architecture could enhance its ability to function effectively in real-time scenarios.

6. Conclusion

Malware attacks represent a significant threat to cybersecurity, with malicious software programs designed to damage computer systems. These attacks can lead to severe consequences such as data breaches, financial losses, and compromised security infrastructures. As cyber threats evolve and the diversity and volume of malware continue to increase, the detection and classification of these threats become increasingly complex. Traditional methods frequently fail to keep up with the complexity of modern malware, highlighting the demanding need for advanced detection techniques. Deep learning presents a promising solution due to its capability to learn intricate patterns and make predictions from extensive datasets, thereby enhancing the accuracy of malware detection and classification. To address these challenges, our study introduces a deep learning-based framework specifically designed to enhance the accuracy and efficiency of malware detection. This framework integrates two pre-trained models such as InceptionV3 and MobileNetV2 due to their strong feature extraction capabilities. We refined these models by applying truncation and compression techniques to eliminate redundant information, thereby reducing computational costs and optimizing the feature extraction process. The framework advances through a process of feature fusion by combining the features from both backbone models to form a comprehensive and enriched feature set. This fusion is designed to capture a wide array of malware characteristics, ensuring that both fine-grained and broad-spectrum features are effectively represented. To further optimize this feature integration, a Squeeze and Excitation attention block is employed. This block operates by compressing the feature maps into a compact representation, which is then used to recalibrate the original feature maps. This recalibration enhances the model's focus on the most critical and relevant features, significantly improving the precision and robustness of the malware classification process. The proposed framework was evaluated using two datasets: an IoT malware dataset for binary classification and the MALIMG dataset, which includes 25 different malware classes. The model achieved high accuracy rates with 97.09% on the IoT malware dataset and 97.47% on the MALIMG dataset, demonstrating its effectiveness across different types of malware. Furthermore, the robustness of the model was assessed through detailed analyses including confusion matrices, ROC curves, and class-wise performance evaluations. These evaluations confirmed the reliability and practical applicability of the framework in real-world cybersecurity scenarios, making it a valuable tool for enhancing malware detection and classification efforts.

References

1. Yuanming, L. and R. Latih, A Comprehensive Review of Machine Learning Approaches for Detecting Malicious Software. International Journal on Advanced Science, Engineering & Information Technology, 2024. **14**(3).
2. GOUIZA, N., H. JEBARI, and K. REKLAOUI, INTEGRATION OF IOT-ENABLED TECHNOLOGIES AND ARTIFICIAL INTELLIGENCE IN DIVERSE DOMAINS: RECENT ADVANCEMENTS AND FUTURE TRENDS. Journal of Theoretical and Applied Information Technology, 2024. **102**(5).
3. Kannari, P.R., N.S. Chowdary, and R.L. Biradar, An anomaly-based intrusion detection system using recursive feature elimination technique for improved attack detection. Theoretical Computer Science, 2022. **931**: p. 56-64.

4. Seymour, W., Examining Trends and Experiences of the Last Four Years of Socially Engineered Ransomware Attacks. 2022.
5. Bolat, P. and G. Kayişoğlu, Cyber Security. Security Studies: Classic to Post-Modern Approaches, 2023: p. 173.
6. Asam, M., et al., IoT malware detection architecture using a novel channel boosted and squeezed CNN. Scientific Reports, 2022. **12**(1): p. 15498.
7. Odii, J., et al., COMPARATIVE ANALYSIS OF MALWARE DETECTION TECHNIQUES USING SIGNATURE, BEHAVIOUR AND HEURISTICS. International Journal of Computer Science and Information Security (IJCSIS), 2019. **17**(7).
8. Yunmar, R.A., S.S. Kusumawardani, and F. Mohsen, Hybrid Android Malware Detection: A Review of Heuristic-Based Approach. IEEE Access, 2024. **12**: p. 41255-41286.
9. Khraisat, A., et al., Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2019. **2**(1): p. 1-22.
10. Saheed, Y.K., et al., A machine learning-based intrusion detection for detecting internet of things network attacks. Alexandria Engineering Journal, 2022. **61**(12): p. 9395-9409.
11. Vishwakarma, M. and N. Kesswani, DIDS: A Deep Neural Network based real-time Intrusion detection system for IoT. Decision Analytics Journal, 2022. **5**: p. 100142.
12. Narayanan, B.N., O. Djaneye-Boundjou, and T.M. Kebede. Performance analysis of machine learning and pattern recognition algorithms for malware classification. in 2016 IEEE national aerospace and electronics conference (NAECON) and ohio innovation summit (OIS). 2016. IEEE.
13. Zhang, H., et al., Classification of ransomware families with machine learning based onN-gram of opcodes. Future Generation Computer Systems, 2019. **90**: p. 211-221.
14. Ijaz, A., et al., Innovative Machine Learning Techniques for Malware Detection. Journal of Computing & Biomedical Informatics, 2024. **7**(01): p. 403-424.
15. Schultz, M.G., et al. Data mining methods for detection of new malicious executables. in Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001. 2000. IEEE.
16. Deng, H., et al., MCTVD: A malware classification method based on three-channel visualization and deep learning. Computers & Security, 2023. **126**: p. 103084.
17. D'Angelo, G., M. Ficco, and F. Palmieri, Association rule-based malware classification using common subsequences of API calls. Applied Soft Computing, 2021. **105**: p. 107234.
18. Jain, M., W. Andreopoulos, and M. Stamp, Convolutional neural networks and extreme learning machines for malware classification. Journal of Computer Virology and Hacking Techniques, 2020. **16**: p. 229-244.
19. Kasongo, S.M. and Y. Sun, A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Computers & Security, 2020. **92**: p. 101752.
20. Naeem, H., B. Guo, and M.R. Naeem. A light-weight malware static visual analysis for IoT infrastructure. in 2018 International conference on artificial intelligence and big data (ICAIBD). 2018. IEEE.
21. Vasan, D., et al., IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. Computer Networks, 2020. **171**: p. 107138.
22. Vasan, D., et al., Image-Based malware classification using ensemble of CNN architectures (IMCEC). Computers & Security, 2020. **92**: p. 101748.
23. Kumar, S., B. Janet, and S. Neelakantan, IMCNN: Intelligent Malware Classification using Deep Convolution Neural Networks as Transfer learning and ensemble learning in honeypot enabled organizational network. Computer Communications, 2024. **216**: p. 16-33.
24. Szegedy, C., et al. Rethinking the inception architecture for computer vision. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
25. Sandler, M., et al. Mobilenetv2: Inverted residuals and linear bottlenecks. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

26. Hu, J., L. Shen, and G. Sun. Squeeze-and-excitation networks. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
27. Nataraj, L., et al. Malware images: visualization and automatic classification. in Proceedings of the 8th international symposium on visualization for cyber security. 2011.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.