applied sciences

MDPI

Article

Securing Edge Devices: Malware Classification with Dual-Attention Deep Network

Gasim Alandjani

Computer Science and Engineering Department, Yanbu Industrial College, KSA, Yanbu Al Sinayiah 46452, Saudi Arabia; gasim@rcjy.edu.sa

Featured Application: The proposed method reveals the widespread real-world applicability of malware detection, particularly in securing IoT devices. Its faster inference speed and high fidelity score illustrate the practicability of on-device malware detection without incorporating desktop/server-class hardware via the internet for faster inference.

Abstract: Detecting malware is a crucial defense mechanism against potential cyber-attacks. However, current methods illustrate significant limitations in achieving high performance while maintaining faster inference on edge devices. This study proposes a novel deep network with dual-attention feature refinement on a two-branch deep network to learn real-time malware detection on edge platforms. The proposed method introduces lightweight spatial-asymmetric attention for refining the extracted features of its backbone and multi-head attention to correlate learned features from the network branches. The experimental results show that the proposed method can significantly outperform existing methods in quantitative evaluation. In addition, this study also illustrates the practicability of a lightweight deep network on edge devices by optimizing and deploying the model directly on the actual edge hardware. The proposed optimization strategy achieves a frame rate of over 545 per second on low-power edge devices.

Keywords: IoT malware classification; malware detection; deep learning; dual-attention; edge device

Citation: Alandjani, G. Securing Edge Devices: Malware Classification with Dual-Attention Deep Network. *Appl. Sci.* **2024**, *14*, 4645. https://doi.org/10.3390/app14114645

Academic Editor: Gianluigi Ferrari

Received: 13 April 2024
Revised: 18 May 2024
Accepted: 23 May 2024
Published: 28 May 2024

1. Introduction

Malware typically refers to malicious code that intends to deface confidential data, financial information, or any digital resources of a computer system. The evaluation of malicious software is a never-ending process [1,2]. In particular, the ease of communication, like the availability of the internet, digital assets, and online transactions, enables the evolution of malware faster than ever [3,4]. The casualties inflicted by malware have also increased in recent years. According to a statistic [5], the global inflicted damage yielded by malware is calculated at SAR 6 trillion in the year 2021. Also, this inflicted cost has been predicted to rise to SAR 10 trillion in 2025. The increasing threat of malware highlights the need for innovative and effective malware detection measures that effectively work with low-power edge computing infrastructures.

In the past, malware was identified using signature-based techniques. These methods compare suspicious files to pre-established malware signatures through static detection [6–9]. It is important to note that this type of malware defense requires a significant amount of handcrafted malware feature samples, such as text signs, regular expressions, filenames, and byte codes [2]. Despite the use of these features, these methods can only detect a few malware variants that match the predetermined features. Additionally, anti-analysis techniques like obfuscation, packing, and polymorphism can easily bypass these defenses with minor modifications. Furthermore, traditional approaches are computationally expensive, requiring a secure environment to analyze each suspicious file. These limitations highlight the need for more dynamic and practical methods of malware classification.

Recent research on detecting malware places significant emphasis on developing AI-powered solutions that leverage deep learning techniques to overcome the limitations of traditional approaches [1,2,10]. These approaches treat malware analysis as a classical image classification task, representing the malware binaries as images. Over the past decade, several innovative works with complex network architectures have been introduced [3,11–14], which have shown considerable improvements over traditional methods. To evaluate the practicality of these existing methods in developing robust malware detection, especially for edge platforms, we assessed the performance of various classification methods, including specialized malware classification methods, using a standard benchmark malware dataset as illustrated in Figure 1.

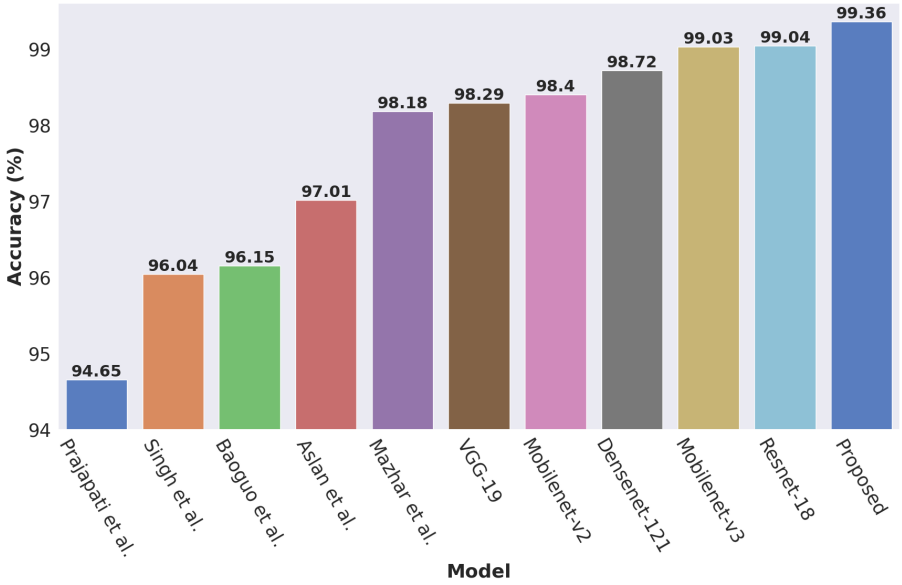


Figure 1. Comparison between different classification methods for malware classification. The proposed method outperforms existing methods by a notable [3,15–22].

As Figure 1 illustrates, the existing deep networks, explicitly, the malware classification methods, failed to achieve a high fidelity rate in the benchmark dataset. Our initial evaluation found that the existing malware classification methods fail to extract and utilize salient features from the malware image. In addition to that, the state-of-the-art (SOTA) methods are developed primarily for desktop or server-class hardware [23,24]. Due to computational and optimization limitations, the existing malware detection works assume that malware detection should be performed on high-performance computing devices like servers via API [25,26]. However, detecting malware on local low-power devices in real-world scenarios is more practical, as it can address the threat without any external dependency [27,28]. Deploying efficient and independent detection mechanisms can revolutionize future solutions, such as integrating them into the secure computing frameworks of 5G/6G infrastructures [29,30]. The current limitation of existing methods and countless applications of efficient malware detection motivated us to develop a robust classifier with an optimization strategy for real-world usage.

This study proposes a novel deep network to classify the malware classes. Our proposed architecture incorporates a two-branch deep network to leverage local–global attention on different image scales. Here, one of the two branches of the proposed network include an edge device-friendly Mobilenet-v2 backbone [20] with an efficient spatial asymmetric attention (SAM) [31] module to extract and refine malware features from high-resolution images. Our deep network's second branch (the auxiliary branch in the later sections) learns salient feature extraction on low-resolution malware images. Thus, it can handle the missing information of a malware input appearing by the compression

artifacts [32], attenuation, parsing errors, etc. We correlate the extracted features of both branches with multi-head attention (MHA) [33,34] to achieve a higher detection accuracy. We compared our proposed method with the existing malware classification methods and studied our method on numerous hardware. Our proposed method outperforms the existing deep malware detection and image classification methods by a notable margin. In addition to that, we illustrate an optimization strategy for deploying our deep model in low-power edge devices to reveal the practicability of deep, efficient malware detection for securing future solutions, including securing IoT and 5G/6G infrastructures [35,36]. To the best concern, this is the first work in the literature demonstrating the optimization and practicability of malware detection on real-edge hardware with sophisticated experiments. In a nutshell, the proposed network is hardware-independent and could be incorporated into any system based on the application requirement. The main contribution of this study is as follows:

- We propose a novel deep network with attention mechanisms and an auxiliary branch to learn salient features from malware images. The proposed network also incorporates a Faster SAM (FSAM) with 83% lower trainable parameters than the well-known SAM. Apart from FSAM, we also propose to leverage MHA to correlate our feature branches for efficient malware classification.
- We illustrate the practicability and optimization strategy for DAMN for real-world usage. We achieved a frame per second (FPS) of 545.29 on a real-edge device. Higher frame rates on low-power computing devices reveal countless possibilities for securing future IoT and network applications.
- We densely study the existing methods from malware and image classification domains to summarize the performance of deep networks for malware classification. We outperform the existing works for the benchmark dataset in multiple evaluation metrics by a large margin.

The rest of the paper is organized as follows. Section 2 reviews the related works, Section 3 details our proposed method, and Section 4 densely evaluates and summarizes the results. In addition to that, Section 5 illustrates the optimization and deployment scope of malware detection algorithm on edge devices. Finally, Section 6 concludes this work.

2. Related Works

Machine learning for malware analysis is relatively new in cybersecurity. These learning-based approaches can be divided into two subcategories based on feature extraction. This section briefly discusses both subcategories of the learning-based malware classification approach and provides an inside look at malware classification on edge platforms.

2.1. Traditional Machine Learning

Traditional learning-based approaches for malware analysis heavily rely on handcrafted feature extraction. Typically, these methods extracted the malware features manually and then utilized their handcrafted features to feed shallow classifiers like support vector machine (SVM), naive Bayes classifier, decision trees, k-nearest algorithms, etc. [37–41]. However, the effectiveness of these methods relies heavily on feature engineering. Additionally, these shallow classifiers are not known for their scalability [2], which limits their ability to handle large numbers of malware samples. This is why recent works on malware classification have shifted towards deep learning-based solutions, which have shown promising results in achieving state-of-the-art performance.

2.2. Deep Learning

Deep learning-based malware classifiers have illustrated a notable performance gain over traditional methods in the past decade. In a recent study, Gilbert et al. [13] proposed a LeNet-like [42] stacked convolutional neural network (CNN) to classify malware images. They reported a validation accuracy of 99.37% on the nine-class malware dataset.

Luo et al. [43] utilized a local binary pattern (LBP) to extract malware features and leverage similar network architecture as Gilbert et al. [13] to classify the Malimg dataset. In their work, they reported 93.17% classification accuracy on the validation data. Agarap et al. [17] also utilized a similar feature extraction architecture to their prior methods and revised the softmax classifier with an SVM classifier in their work to achieve an accuracy of 77.23% in the Malimg dataset. However, in their setup, GRU-SVM outperformed their CNN-SVM structures by a notable margin. Later, Ajay et al. [16] also proposed a CNN with four consecutive blocks, combining convolution and max pooling operation to achieve 96.10% on the Malimg dataset. Yeo et al. [44] also used a CNN with flow data to reach just over 85% classification on a nine-class dataset. Kalash et al. [14] also proposed a deep network called M-CNN and reported an accuracy of 98.52% in the validation phase while using the Malimg dataset. In another work, Yuan et al. [12] proposed a deep stacked CNN with 13 convolution layers to achieve an accuracy of 99.26% on a nine-class malware dataset with a 10-fold validation strategy. In addition, Prajapati et al. [15] compared different network architectures in their study with a 17-class malware dataset and reported an accuracy of 89.55% with a 2D CNN architecture. They found that with pre-trained based Resnet-152 [22], VGG-19 [18] architecture can outperform their 2D CNN [42] with a marginal score.

Although several novel works explore different CNN architectures in their respective works, it has been observed that training a CNN without pre-trained weight achieved unsatisfactory performance in malware classification [45]. It is worth noting the existing malware benchmark datasets lack data diversity. Therefore, many recent works utilized Imagenet [46] pre-trained on their respective methods [47]. For instance, Rezende et al. [48] proposed to use a VGG-16 [18] with Imagenet pre-trained weights to classify malware images. They achieved a validation accuracy of 90.77% accuracy with a 10-fold validation strategy on a 20-class dataset. Similarly, Khan et al. [49] also used transfer learning on Resnet-18 [22], 34, 50, 101, 152, and GoogleNet [50] to achieve 83%, 86.51%, 86.62%, 85.94%, 87.98%, and 84% validation accuracy. Also, Awan et al. [3] leveraged a VGG-19 architecture with frozen weights and fused it with simple spatial attention. They reported an accuracy of 97.38% with class balancing on the Malimg dataset. Similarly, Aslan et al. [11] combined two pre-trained weights (i.e., ALexNet [51] and Resnet0152 [22]) to classify the malware images. They reported 97.18% accuracy on the Malimg dataset.

Most of the IoT malware detection studies are based on static analysis [52,53]. Haddadpajouh et al. [54] used their OPCODE-based IoT dataset with Recurrent Neural Network (RNN) [55] and achieved an accuracy rate of 98,18%. Su et al. [56] used the IoTPoT dataset and converted the malware binary files into gray-scale images. They used the CNN-based classifier and achieved an accuracy rate of 94%. Alasmary et al. [57] created their dataset and compared the characteristics of IoT malware and Android malware, mainly based on Linux, using Control Flow Graphs. Also, they proposed a detection mechanism based on CNN, which achieved a very high accuracy rate of 99.66%. Dovom et al. [58] used the IoT dataset and achieved an average accuracy rate of 96.41% with the Fuzzy Pattern Tree algorithm. Vasan et al. [59] presented a Cross-Architecture IoT Malware Detection method called MTHAEL. In addition to these methods, a few recent methods studied malware detection on Android platforms to evaluate the practicability of malware detection on IoT devices [60,61]. It is worth noting that the existing process has evaluated its method on IoT malware benchmark datasets, mostly with x64 computational architecture or specific IoT hardware like Android power devices [52,62]. However, the IoT and edge devices typically leverage the numerous ARM architectures based on their specific applications [63].

2.3. Malware Detection on IoT and Edge Devices

In response to the escalating threat of attacks on IoT devices, the focus on malware classification on edge platforms like Android has intensified in recent years. Haddadpajouh et al. [54] utilized their OPCODE-based IoT Dataset alongside Recurrent Neural Network (RNN) techniques [55], achieving an impressive accuracy rate of 98.18%. Su et al. [56]

employed the IoTPoT dataset, transforming malware binary files into grayscale images and utilizing a CNN-based classifier to achieve a notable accuracy rate of 94%. Alasmary et al. [57] contributed by constructing a dataset comparing IoT and Android malware characteristics, predominantly Linux-based, through Control Flow Graphs, proposing a CNN-based detection mechanism that attained an exceptional accuracy rate of 99.66%. Dovom et al. [58] achieved an average accuracy rate of 96.41% using the Fuzzy Pattern Tree algorithm with the IoT Dataset. Furthermore, Vasan et al. [59] introduced MTHAEL, a Cross-Architecture IoT Malware Detection method, adding to the arsenal of approaches to combating IoT malware threats.

Rajesh et al. [27] introduced a Bayes classifier to identify malware present on Android devices, with DroidMat differentiating malware as benign through an analysis of intents, permissions, and API calls. Additionally, Varsha et al. [64] focused on extracting static features from Android application package (APK) files, including manifest and application executable data, as part of their malware detection approach.

2.4. Malware Detection on 5G/6G Infrastructure

Recent exploration into malware classification has extended beyond IoT devices to encompass 5G and emerging 6G infrastructures. Ning et al. [60] introduced a framework tailored explicitly for detecting malware on 5G-enabled smartphones, achieving a notable 95.69% accuracy by analyzing APK files. Ankita et al. [65] proposed a machine learning-based approach for detecting malware and ransomware attacks for future 6G infrastructure. Sousa et al. [66] presented a multistage botnet detection mechanism that operates without requiring software installation on edge hardware. They emphasized the importance of centralized administration in intrusion detection systems utilizing a software-defined 5G architecture to manage infected devices effectively. Researchers underscored the necessity of identifying cyber threats and employing machine learning (ML) methods to mitigate risks to the 5G network. Ishtiaque et al. [67] addressed unidentified and suspicious circumstances in 5G networks, employing various machine learning algorithms, with the Linear Regression algorithm yielding the best results, achieving 92.12% precision on test data and 92.13% on train data. Similarly, Manoj et al. [68] compared different CNN and LSTM models for malware detection on 5G/6G devices, with Ankita et al. [69] proposing a CNN-DMA network for application in 5G/6G infrastructures.

Notably, existing malware detection methods, especially for 5G/6G, have focused on integrating deep models into infrastructure rather than deploying them on specific hardware. However, deploying deep models on ARM-based low-power IoT devices presents considerable challenges. In contrast, our proposed method highlights the difficulties of deploying deep models on low-power edge devices through practical experiments. We deployed our model on an ARM-powered Jetson board to assess the performance and challenges of deploying deep networks on such hardware. This optimization and deployment strategy explores the challenges inherent in any 5G/6G infrastructure, including server-class hardware, rather than focusing on making the theoretical framework like previous studies.

3. Learning Malware Detection

Figure 2 shows the proposed methods' learning overview. This work takes malware binaries as input and converts them into 2D images. The image input is then scaled into two sizes and fed into the proposed network's branches. Our network learns salient features from the given malware image and classifies them based on the extracted and refined features with attention mechanisms.

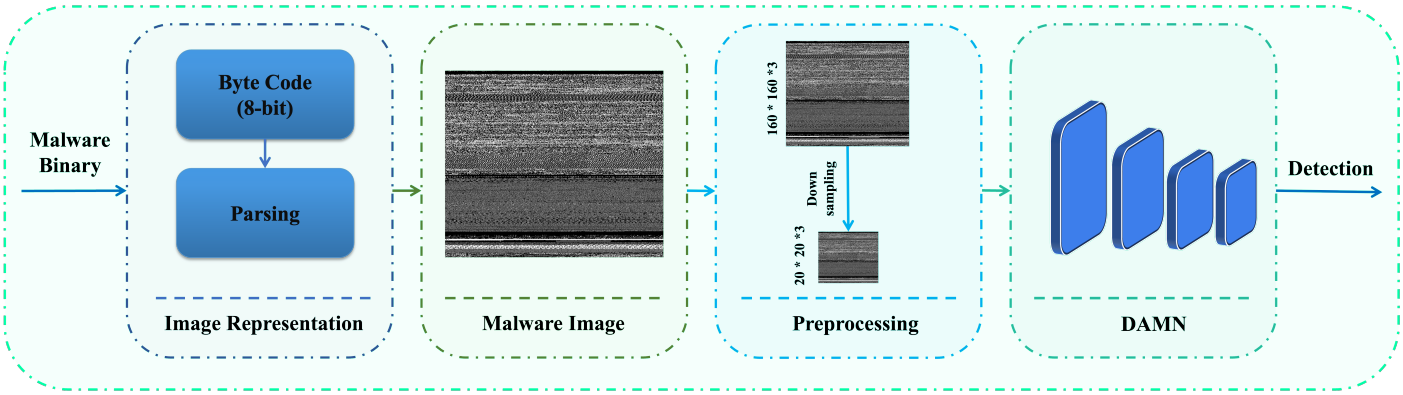


Figure 2. Overview of the proposed method. We parse the malware binary files into 2D images and pre-process them to learn malware classification. We classify the malware images using our novel deep network.

3.1. Model Architecture

The proposed dual-attention malware network (DAMN) comprises two distinct feature branches to learn salient features from different image scales. As Figure 3 illustrates, our main branch takes an image input $\mathbf{I_M} \in [0,1]^{H \times W \times 3}$. H and W represent the height and width of the input. We incorporate a mobile-friendly Mobilenet-v2 [19] architecture (without fully connected layers) [20] pre-trained block as the backbone to extract generic features. It is worth noting that Mobilenet-v2 is considered one of the most efficient backbones for low-power edge devices. We introduce the FSAM to refine the extracted features of Mobilenet-v2 by incorporating local–global attention. Apart from that, our proposed network also incorporates a novel auxiliary attention branch. It aims to learn the artifacts and missing information that may appear due to compression [32] and binary-to-image conversion. We concatenate the output of both branches and apply a MHA to refine the learned features. The final layer of the proposed DAMN is a fully connected layer that learns to calculate the probability of being a malware (class). In addition to that, we utilize the dropout layers in our network to reduce overfitting [32,70,71].

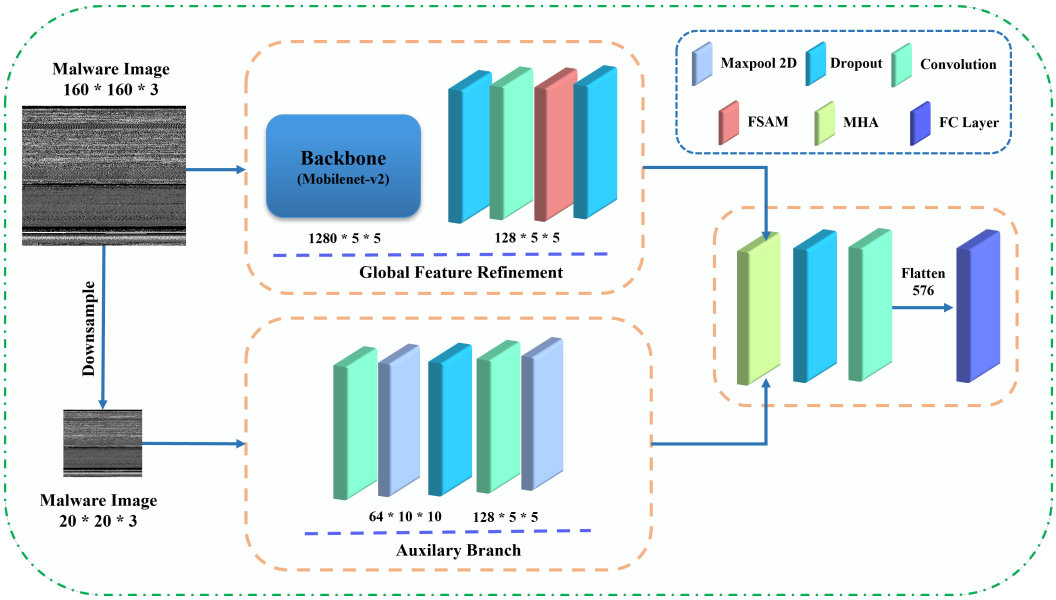


Figure 3. Overview of the proposed dual-attention malware network (DAMN). It comprises a dual-attention strategy with an auxiliary feature extraction branch. The auxiliary branch focuses on learning artifacts, FSAM refines backbone features, and MHA correlates features extracted by both branches.

3.1.1. Faster Spatial-Asymmetric Attention Module

SAM is known for its capability of refining features with global-local attention. It has significantly impacted reconstructing nona-Bayer images with real-world image noises. Despite showing significant performance gain, the SAM is computationally expensive. Explicitly, it leverages a 9×9 square convolution for pursuing global attention from a given input. On the downside, malware classification suffers substantially from data limitations, and our target platform is mostly edge devices. Consequently, utilizing such computationally expensive blocks for malware classification counters two-fold limitations: (i) a large number of trainable parameters leads to overfitting, and (ii) significantly slower computation time with high temperature. To address both limitations, we propose to replace the large kernel convolution with small kernel dilation convolution [72] as shown in Figure 4. Table 1 compares the proposed FSAM and SAM modules. It helps reduce the trainable parameters of the original SAM by 83% without affecting the performance. We pursue Faster SAM as follows:

$$\mathbf{F_V} = \tau(\mathbf{C_S}([\mathbf{Z_A}(\mathbf{A_V}(\mathbf{X})); \mathbf{Z_M}(\mathbf{A_V}(\mathbf{X}))])) \tag{1}$$

$$\mathbf{F_H} = \tau(\mathbf{C_S}([\mathbf{Z_A}(\mathbf{A_H}(\mathbf{X})); \mathbf{Z_M}(\mathbf{A_H}(\mathbf{X}))])) \tag{2}$$

In this context, the functions $\mathbf{A}(\cdot)$ and $\mathbf{C}(\cdot)$ denote the asymmetric convolution operation and square convolution, respectively, while τ represents the sigmoid activation function. Additionally, $\mathbf{Z_A}$ and $\mathbf{Z_M}$ refer to the average pooling and max pooling operations, which generate two 2D feature maps denoted as $\mathbf{X_A} \in \mathbb{R}^{1\times H\times W}$ and $\mathbf{X_M} \in \mathbb{R}^{1\times H\times W}$, respectively. These mapped features are then concatenated and presented as a 2D map.

Overall, the aggregated bi-directional attention over a given feature of malware is obtained as:

$$\mathbf{F_C} = \mathbf{F_V} + \mathbf{F_H} \tag{3}$$

A squeeze-extractor descriptor [73] has also been utilized to pursue a global descriptor as follows:

$$\mathbf{F_G} = \mathbf{M_F}(\mathbf{Z_G}(\mathbf{C_D}(\mathbf{X}))) \tag{4}$$

where $\mathbf{M_F}$ and $\mathbf{Z_G}$ represent consecutive fully connected layers and global pooling operations, respectively. Additionally, $\mathbf{C_D}$ in Equation (4) denotes the proposed small kernel dilated convolution operation. This operation plays a crucial role in reducing the trainable parameters of the proposed Faster SAM module by 83% compared to its base module.

Table 1. Comparison between proposed FSAM and well-known SAM module. Our FSAM is 83% lighter compared to the original SAM module.

Module	Input	Sqaure Kernel	Dilation	Param. (M)	Comp. (GFlops)
SAM	$128 \times 5 \times 5$	9×9	1	1.43	0.0713
FSAM	$128 \times 5 \times 5$	3×3	4	**0.25**	**0.0123**

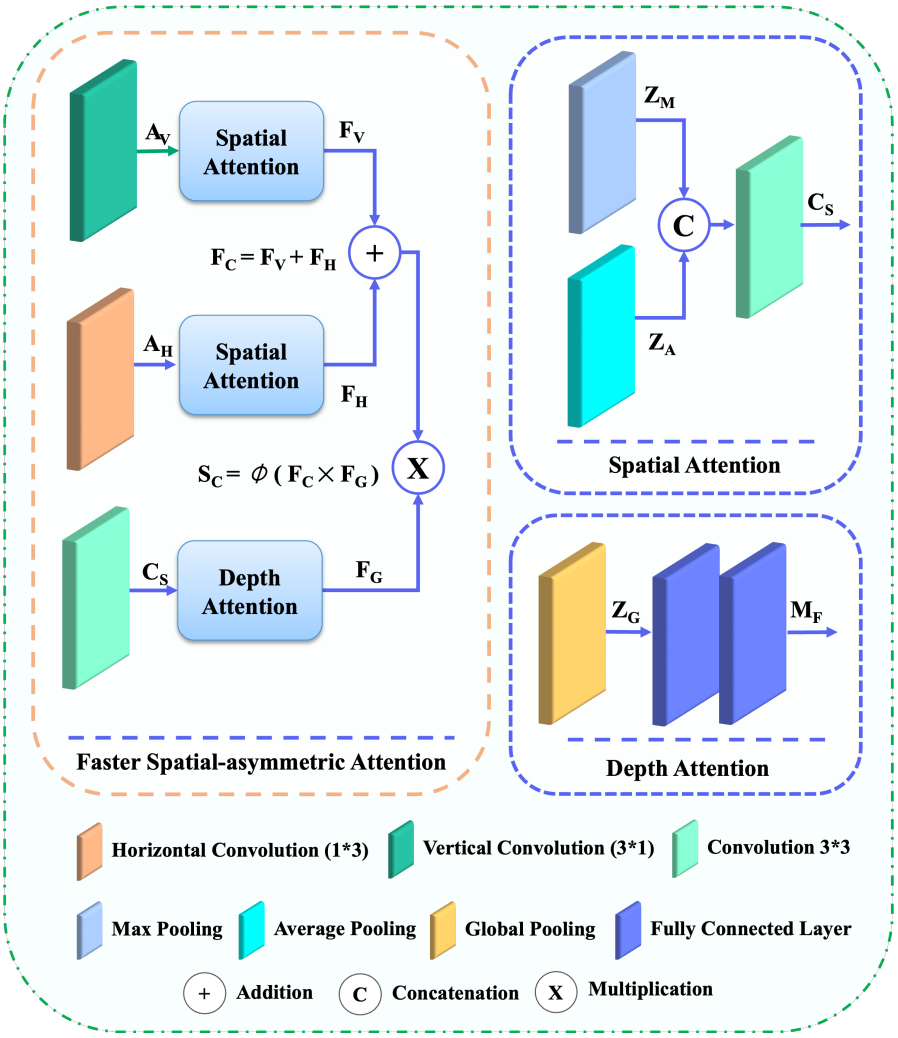


Figure 4. Overview of the proposed FSAM module. We replace the large kernel of the SAM module with dilated convolution. Overall, we reduced the complexity of the SAM module by 83%.

3.1.2. Auxiliary Branch

The auxiliary attention branch serves as a vital component of the proposed DAMN architecture. Its primary objective is to capture missing information introduced by factors such as compression artifacts, the conversion of malware binaries to images, and network distortions. In this branch, a low-sampled image denoted as $\mathbf{I_A} \in [0,1]^{H \times W \times 3}$ is taken as input, where H and W represent the height and width of the input image, respectively. This input undergoes two consecutive 3×3 convolutions followed by max-pooling operations before being fed into the MHA mechanism and its parallel branch for further refinement.

3.1.3. Multi-Head Attention (MHA)

Multi-head attention is a well-known attention mechanism in artificial intelligence, particularly in natural language processing and computer vision [33]. It allows models to focus on different parts of the input sequence simultaneously, capturing complex dependencies and relationships within the data. Figure 5 demonstrates an overview of the MHA block. This study uses the MHA to correlate our extracted features to achieve effective malware classification results. We perceive the MHA throughout this study as follows:

$$MHA(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = concat(head_1, \ldots, head_h)\mathbf{W}^O \quad (5)$$

$$\text{where} \quad \text{head}_i = \text{Attention}(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V) \tag{6}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{7}$$

Here, we utilize three matrices: **Q**, **K**, and **V**. They respectively represent the query, key, and value matrices. These matrices are essential for capturing different aspects of the input data. Each attention head, denoted as head_i, generates its output, allowing the model to focus on different input parts simultaneously. To obtain the final output, the outputs of all attention heads are concatenated along the feature dimension and multiplied by the output weight matrix $\mathbf{W}^O$. Additionally, each attention head is associated with its own set of learnable weight matrices, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$, enabling the model to learn distinct representations for different attention heads. The number of attention heads h and the dimensionality of the key vectors d_k are hyperparameters that influence the model's capacity to capture complex dependencies within the data. Finally, the softmax activation function is applied to compute the attention scores, facilitating the weighted aggregation of values based on their relevance to the queries.

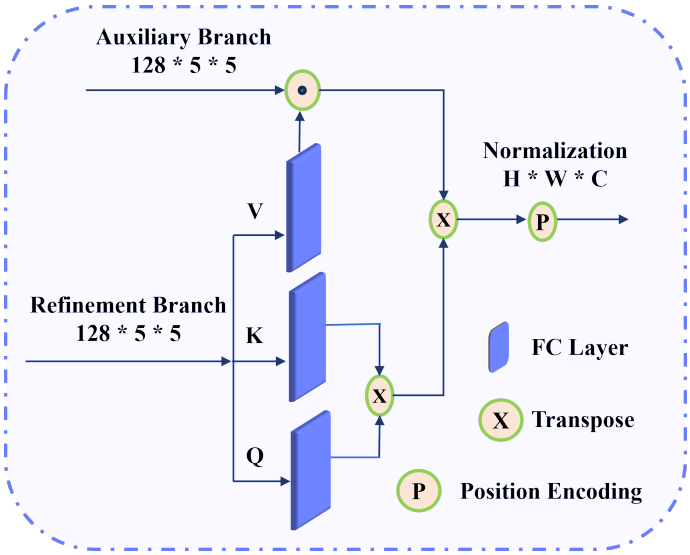
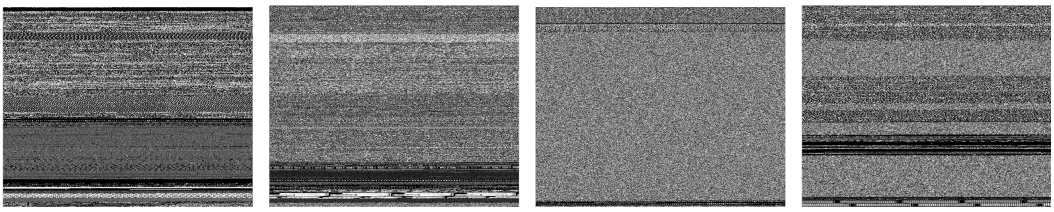


Figure 5. Overview of MHA block. We leverage MHA to correlate the extracted features of both network branches to achieve effective classification results.

3.2. Dataset Preparation

Throughout this study, we employ the widely used Malimg dataset as our benchmark for evaluating our proposed method [40]. Notably, Malimg is one of the most widespread malware classification datasets. Many previous studies leveraged this dataset as a standard benchmark dataset. We follow the footprint of the previous studies to conduct our experiments.

The Malimg dataset comprises 9339 malware samples categorized into 25 distinct classes, encompassing well-known malware families such as Yuner.A, VB.AT, Malex.gen!J, Autorun.K, Rbot!gen, Swizzor.gen!I, and C2Lop.p, among others. Notably, these malware images are derived from malware binaries, where the transformation from binaries to images involves converting the malware binaries into 8-bit vectors. Subsequently, these vectors are transformed into grayscale images by mapping the 8-bit vectors to pixel values, representing the intensity [3,40]. This dataset serves as a comprehensive benchmark for assessing the performance of our proposed method in malware image classification tasks. Figure 6 illustrates the pre-processed images from Malimg dataset.



(a) (b) (c) (d)

Figure 6. Malware images from Maling dataset. (**a**) Adialer.C (**b**) Autorun.K (**c**) Wintrim.BX (**d**) Swizzor.geniE.

3.3. Training Details

The proposed method is implemented using the PyTorch 2.0 framework [74]. During training, we employed a learning rate of 1×10^{-4}, which was adjusted every two epochs using a weight decay of 1×10^{-4}. We utilized the Adam optimizer with the objective function set to minimize the cross-entropy loss. Furthermore, we resized all training and testing images to dimensions of $160 \times 160 \times 3$ for the main branch and 20×20 for auxiliary branches. We applied various augmentation techniques to prevent overfitting [70,75], such as random flipping, blur, and rotation. Algorithm 1 demonstrates the training detail of the proposed DAMN.

Algorithm 1 Learning malware classification with DAMN.

Input: Training set D_{train}, validation set D_{val}
Output: Trained CNN model M
Initialize CNN model M with random weights
Initialize learning rate η_0, initial batch size B_0, number of epochs N_{epochs}, learning rate decay factor α
Initialize epoch counter $e = 1$
for $i = 1$ to N_{epochs} **do**
    if $i \mod 2 = 0$ **then**
        Update learning rate: $\eta_i = \alpha \cdot \eta_{i-1}$
    Sample mini-batches B_{train} from D_{train} with augmentation
    for each mini-batch B_{train} **do**
        Compute loss L using forward pass of M on B_{train}
        Update weights of M using backpropagation and gradient descent with learning rate η_i
    for each mini-batch $B_{val} \in D_{val}$ **do**
        Compute validation accuracy using forward pass of M on B_{val}
    Compute average validation accuracy for this epoch
    if average validation accuracy > best validation accuracy **then**
        Update best validation accuracy: $best\_val\_acc =$ average validation accuracy
        Save current weights of M as best weights: $best\_weights = M$.get_weights()
Set weights of M to best weights: M.set_weights($best\_weights$)

All models were trained for 50 epochs with a fixed batch size of 64. Our experiments were conducted on a machine equipped with an AMD Ryzen 3200G central processing unit (CPU) clocked at 3.6 GHz, 16 GB of random-access memory, and a Nvidia GeForce GTX 1060 (6 GB) graphical processing unit (GPU). These hardware specifications provided sufficient computational resources for effectively training and evaluating our models.

4. Results and Analysis

This section illustrates the results obtained through the proposed network, a comparison between different network architectures, and an analysis.

4.1. Comparison

The proposed method was evaluated against two categories of image classification methods: (i) malware and (ii) state-of-the-art (SOTA) image classification methods. All networks were trained for consistency using our pre-processed dataset. This uniform pre-processing step allowed us to fairly compare the performance of different models on the benchmark dataset.

To accommodate the input dimensions of our dataset, we modified the input layer of all comparison models accordingly. Subsequently, we conducted comprehensive evaluations by summarizing the performance of each deep model using standard evaluation metrics such as accuracy, F1 score, precision, and recall. These metrics provide insights into the overall effectiveness of the proposed method compared to existing approaches in both malware and general image classification tasks. This rigorous evaluation process ensures the reliability and validity of our comparative analysis.

4.1.1. Comparison with Malware Detection Methods

We compared 11 advanced malware classification models, utilizing deep learning to better understand their performance. Please note that the existing works on malware detection methods are not publicly available. Therefore, we implemented all of the existing methods for the comparison. We cross-checked the implementation by reproducing their claimed results on their reported dataset. Later, we trained each model with their recommended hyperparameters and trained them until they converged with the given dataset. After applying our pre-processing and augmentation techniques, we evaluated each model's performance in a benchmark dataset. Table 2 illustrates the comparison between proposed network and state-of-the-art malware classification models. Our proposed method outperformed the existing methods in all evaluation metrics, achieving a 1.18% gain in accuracy, 0.0214 gain in precision, 0.0155 gain in recall, and 0.0101 gain in F1-score. It is worth noting that we did not perform any weight balancing, which was included in previous studies, to enhance the scores. We evaluated all methods similarly to how they would be evaluated in real-world scenarios, and our proposed method still outperformed its counterparts by learning salient features through two distinct feature branches.

Table 2. Comparison with malware image classification methods. The proposed method outperforms the existing method in evaluation metrics with a straightforward learning strategy.

Model	Accuracy	Precision	Recall	F1-Score
Ajay et al. [16]	96.04	0.9096	0.9239	0.9595
Agarap et al. [17]	94.55	0.8777	0.8957	0.9469
Yeo et al. [44]	93.47	0.8682	0.8851	0.9351
Luo et al. [43]	94.44	0.8926	0.9048	0.9438
Kalash et al. [14]	96.04	0.9151	0.9253	0.9608
Prajapati et al. [15]	94.65	0.8990	0.9168	0.9459
Yuan et al. [12]	96.15	0.9256	0.9408	0.9618
Aslan et al. [11]	97.01	0.9341	0.9397	0.9708
Gibert et al. [13]	95.29	0.9075	0.9209	0.9528
Edmar et al. [48]	94.33	0.8719	0.8855	0.9448
Awan et al. [3]	98.18	0.9678	0.9724	0.9823
Ankita et al. [69]	98.10	0.9616	0.9633	0.9796
DAMN (Proposed)	**99.36**	**0.9892**	**0.9879**	**0.9924**

4.1.2. Comparison with Image Classification Methods

Over the past decade, numerous deep network architectures have emerged for image classification, significantly enhancing classification accuracy and fidelity for generic images. These architectures have not only advanced the field of image classification but have also been widely adopted as backbones or directly applied in various vision tasks to expedite their respective objectives. Recent works in malware classification, such as those

by Aslan et al. [11], Awan et al. [3], among others, have also begun leveraging SOTA image classification models. While these methods have individually investigated a few SOTA network architectures, a comprehensive evaluation of these image classification models is still necessary.

This study aims to thoroughly evaluate existing SOTA image classification models to assess their efficacy in malware classification tasks. To adapt these models for malware classification, we modified their final layer to output probabilities for the 25 malware classes present in our dataset. Additionally, we leveraged pre-trained weights from ImageNet to initialize the models, aiming to achieve optimal performance. Table 3 summarizes the performance of various SOTA image classification methods, providing insights into their effectiveness when applied to malware classification.

This comprehensive evaluation illuminates the performance of different SOTA models and facilitates a better understanding of their impact on malware classification tasks, ultimately contributing to advancements in cybersecurity research.

Table 3. Comparison with state-of-the-art image classification models. The proposed DAMN can outperform the SOTA image classification methods in malware image classification.

Model	Accuracy	Precision	Recall	F1-Score
VGG-16 [18]	97.86	0.9641	0.9614	0.9778
VGG-19 [18]	98.29	0.9713	0.9703	0.9833
AlexNet [51]	95.08	0.8943	0.9116	0.9501
Densenet121 [20]	98.72	0.9806	0.9838	0.9875
Efficientnet [76]	97.75	0.9528	0.9537	0.9775
GoogLenet [50]	97.97	0.9629	0.9666	0.9802
Mobilenet-v2 [19]	98.40	0.9673	0.9607	0.9837
Mobilenet-v3 [21]	99.03	0.9773	0.9790	0.9906
Resnet18 [22]	99.03	0.9835	0.9819	0.9906
Shufflenet-v2 [77]	98.08	0.9637	0.9710	0.9806
Squeezenet [78]	98.29	0.9721	0.9712	0.9820
Swin Transformer [79]	97.86	0.9594	0.9609	0.9778
VIT-B-16 [80]	96.26	0.9273	0.9322	0.9622
Wide ResNet [81]	95.94	0.9350	0.9259	0.9598
DAMN (Proposed)	**99.36**	**0.9892**	**0.9879**	**0.9924**

As Table 3 illustrates, the proposed method substantially outperforms the image classification methods in all evaluation metrics. Also, it is worth noting that several SOTA image classification methods, like VGG-19, Squeezenet, mobile net-v2, mobile net-v3, densenet121, etc., can outperform the existing malware classification methods with a marginal score.

4.2. Ablation Study

The impact of each novel block has been meticulously examined through sophisticated experiments in this study. Initially, we stripped our proposed learning strategies, including FSAM, auxiliary branch, MHA, etc., from the proposed network architecture. Subsequently, we systematically reintroduced each proposed module to assess its impact on the final output. The base model excludes all novel components, including FSAM, auxiliary branch, and MHA. The FSAM variant includes an FSAM block with a base backbone. The SAMAUX variant adds an auxiliary branch over the FSAM variant, and DAMN comprises all novel components. Table 4 provides a comprehensive overview of the ablation study results, clearly demonstrating the meaningful impact of these modules on the reported final results.

Moreover, the ablation results validate the feasibility and efficacy of these proposed modules in malware classification. By systematically analyzing the network's performance with and without each module, we gain valuable insights into the contributions of individual components to enhancing the overall classification accuracy. These findings underscore

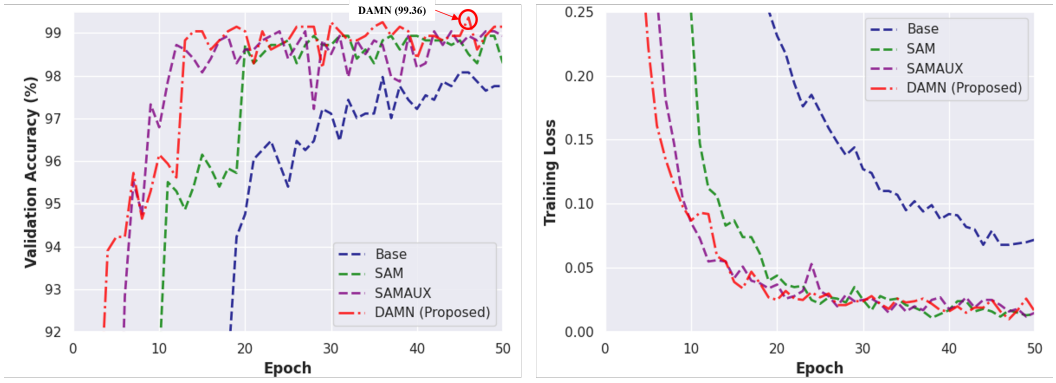
the importance of each novel block and provide helpful guidance for future research and development efforts in malware classification.

Our proposed components offer enhanced accuracy and lightweight design. Despite significantly enhancing classification accuracy, these components impose minimal computational overhead. In summary, our innovative additions improve classification accuracy by 1.29% over the base variant, requiring only an additional 0.2 million trainable parameters.

Table 4. Ablation study with different network variants of the proposed DAMN. Our novel components significantly improve the classification score by adding a small overhead.

Network Varient	Backbone	FSAM	MHA	Param. (M)	Comp. (GFlops)	Accuracy (%)	Precision	Recall	F1-Score
Base	X	X	X	3.76	0.6605	98.07	0.9660	0.9638	0.9799
FSAM	Y	X	X	4.01	0.6728	98.93	0.9732	0.9780	0.9889
SAMAUX	Y	Y	X	4.02	0.6748	99.04	0.9880	0.9865	0.9906
DAMN (Proposed)	Y	Y	Y	4.08	0.6782	**99.36**	**0.9892**	**0.9879**	**0.9924**

Apart from the objective scores, we visualize the training phase's validation accuracy and loss. As Figure 7 shows, our proposed DAMN is more stable than its other variants. Also, the dropout and our proposed auxiliary branch noticeably help our method to reduce overfitting and learn more valuable features among the experimented variants.



(a) (b)

Figure 7. Training procedures of proposed DAMN and its variants. (**a**) Validation accuracy vs. epoch. (**b**) Training loss vs. epoch.

4.3. Learning Analysis

Figure 8 illustrates the class-wise malware prediction of the proposed DAMN and its variants. It can be seen that malware from the same family (i.e., Swizzor.gen!I and the Swizzor.gen!E) substantially affects the performance of deep networks. The malware from homogeneous features is more complex to identify. Here, our proposed model can learn even such hard-to-distinguish features to understand the disparity between close malware.Our proposed auxiliary feature branches MHA and FSAM help our proposed network learn and refine salient information from malware images.

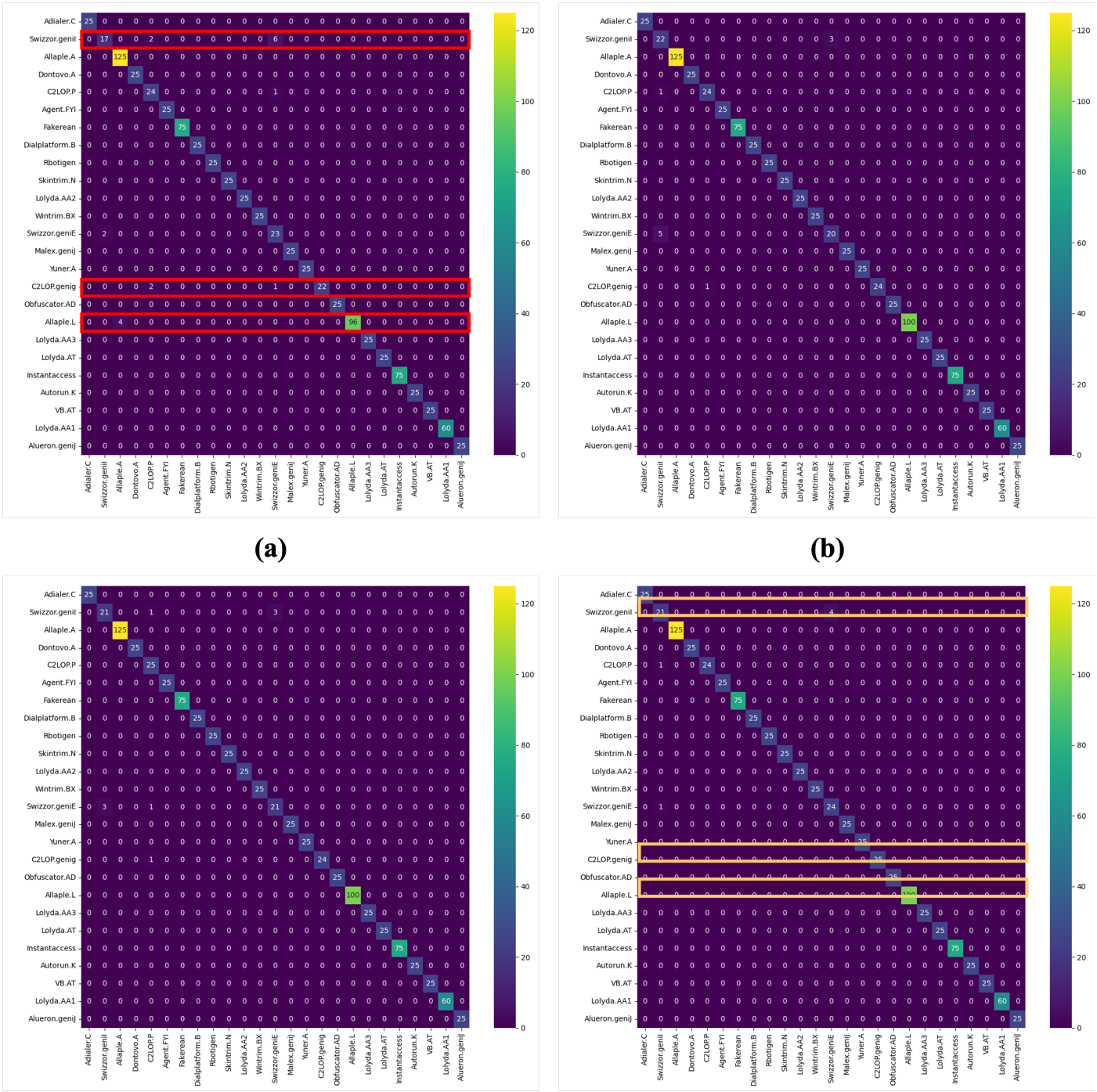


Figure 8. Confusion matrix of proposed DAMN and its variants. It is best viewed in zoom and color. Red and yellow boxes highlight performance on critical malware classes of network variants. (**a**) Base, (**b**) FSAM, (**c**) MHA, (**d**) DAMN (proposed).

5. Malware Detection on Edge Platform

Malware detection on edge platforms can be crucial in the security domain. In particular, efficient malware detection that can detect malware locally on edge devices can change the paradigm of IoT security. It is worth noting that low-power edge devices mainly power the IoT and 5G infrastructures. Therefore, for evaluating the proposed method on any specific hardware infrastructure, the proposed method has been optimized and

evaluated on a generic hardware platform like Jetson. It allowed the proposed method to identify and address the challenges of deploying it on any infrastructure without knowing its specific hardware or infrastructure requirements. Overall, we extensively evaluated the practicability of a lightweight, efficient malware detection algorithm by deploying it on low-power ARM-based hardware.

5.1. Optimization and Deployment

Deep learning optimization for edge devices is among the most challenging tasks [82]. The network architecture with attention mechanisms makes this difficult task even more complicated [83]. To address these limitations, we proposed a novel optimization strategy for optimizing malware detection on IoT devices. It is worth noting that Figure 9 illustrates the overview of our proposed optimization strategy.

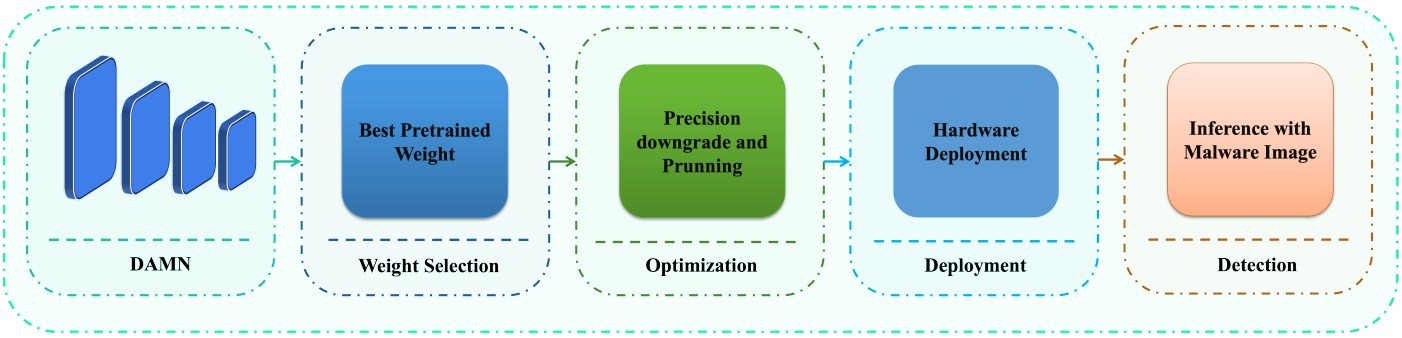


Figure 9. Overview of the proposed optimization and deployment strategy. We leverage quantization and pruning to make our model faster on real edge devices.

We selected our best pre-trained weight for optimization to deploy and infer efficiently on edge devices. To optimize the weight to achieve faster inference time, we downgraded the precision of our pre-trained weights from float32 to float16. We leveraged the TensorRT optimization library [84] to perform post-training quantization [83,85]. Also, the same library allowed us to select the best precision values for a respective layer (depending on the hardware) automatically without any human inputs. Also, we pruned the near-zero layers by leveraging the same optimization tools [86]. Unfortunately, we found that the weight normalization performed while designing the MHA is not executable on the edge devices. Even NVIDIA's most advanced edge hardware struggles to optimize the normalization layers for efficient acceleration. Therefore, we pruned this normalization layer from our optimized weights to achieve a smoother and faster inference time.

Apart from the optimization, we also deployed and tested our model on real edge devices. It is worth noting that IoT and 5G infrastructures are powered mainly by low-power edge devices. Therefore, for evaluating the proposed method on any specific hardware infrastructure, the proposed method has been optimized and evaluated on a generic hardware platform like Jetson. It allowed the proposed method to identify and address the challenges of deploying it on any infrastructure without knowing any specific hardware or infrastructure requirements. We selected an NVIDIA Jetson Orin board to deploy and test our proposed strategy for malware detection. Our Jetson Orin comprises 8 Ampere GPU cores, 12 ARM Cortex-A78 CPU cores, and 32 GB HBM2e memory with 64 MB L2 and 4 MB L3 caches. We leveraged Docker to enable the utilization of CUDA cores for GPU acceleration of the target edge board. We deployed the optimized weight on our Docker environment and operated the hardware in its efficient mode (30 watts) [87] to ensure the practicability of our proposed method on IoT platforms.

5.2. Inference Analysis

Table 5 illustrates the performance of the proposed method on numerous hardware platforms. It can be seen that our proposed method can achieve over 100 FPS inference

speed on a mid-level desktop GPU. Also, without optimizing or processing in the edge platform, our mode can maintain a frame rate of over 30 FPS. Our method can achieve real-time performance on such low-power devices without optimization. Our Mobilenet-v2 backbone and novel FSAM block helped us achieve higher accuracy while maintaining a real-time inference speed on edge devices.

To push our model limit further, we optimized our model for the target device. Optimizing our model can facilitate fast inference time on edge devices. It achieves 432.19 FPS with float32 optimization and 545.29 FPS with float16 optimization. It is worth noting that we performed post-training quantization techniques to perceive the maximum inference speed and optimization process. We did not employ fine-tuning techniques like quantization-aware training to improve the accuracy further. However, without fine-tuning, our proposed method can still achieve an industry standard 95 (%) accuracy for detecting malware on edge devices.

Table 5. Inference analysis of DAMN on different hardware platforms. We achieved 545.29 FPS by optimizing our method for edge devices. It confirms the practicability of our proposed method, even on low-power devices.

Acceleration	**Unoptimized**			**Optimized**	
Architecture	**CPU (X64)**	**GPU (GTX 3060)**	**Jetson (ARM64)**	**Jetson (ARM64)**	
Weight Precision		Float32		Float32	Float16
Accuracy		99.36		95.19	94.97
Precision		0.9892		0.95187	0.9497
Rrcall		0.9879		0.95187	0.9497
F1		0.9924		0.95187	0.9497
FPS	49.93	103.83	35.28	**432.19**	**545.29**

5.3. Discussion

The proposed method reveals several aspects of malware classification by incorporating sophisticated experiments. We illustrate how efficient feature learning with a straightforward training strategy can achieve state-of-the-art performance for malware classification. Despite there being a severe data imbalance in the benchmark dataset, the proposed method outperforms existing methods without exploiting any class information. Our FSAM, AB, and MHA help us learn salient features for malware classification. It is worth noting that attention guidance modules like MHA incorporate mobile-unfriendly normalization layers. Such layers can make the deployment of generic hardware impractical. We prune out such normalization layers to address the limitation while optimizing the edge devices.

Apart from outperforming the existing method, we also illustrate the practicability of malware detection on edge devices. The proposed method is generic and specially optimized for mobile-friendly edge devices, including IoT and 5G infrastructures. Therefore, the proposed method can be deployed on any ARM power edge platform with traditional server or desktop-class hardware. Despite achieving a higher FPS on edge devices, our method drops 4 (%) accuracy as a trade-off between speed and accuracy. Such accuracy drops after optimization are shared among the research community [85]. Nevertheless, quantization-aware training can help recover the dropped accuracy by fine-tuning the optimized weights [86,88]. We planned to study the quantization-aware training for malware detection in a future study. Apart from the dropped accuracy, our model enables inference with a speed of 545 FPS. Notably, such a higher inference speed confirms the practicability of an optimized malware detection algorithm in low-power hardware.

In addition, we evaluated our optimized weight on a single-edge board due to hardware limitations. It would be an interesting future scope for the malware detection research community to study the performance of deep learning-based malware detection algorithms on numerous edge hardware from different manufacturers.

6. Conclusions

This study proposes a two-branch dual-attention deep network to classify malware. The proposed DAMN comprises an FSAM module to refine extracted features of a pre-trained Mobilenet-v2. Additionally, our proposed model incorporates an auxiliary feature branch, which aims to learn the salient missing features from low-quality malware images. MHA has allowed us to correlate the extracted features to achieve a higher fidelity score. Our proposed network illustrates significant performance gain without partial performance-gaining techniques like weight balancing. We optimized our proposed method for edge devices. Our optimization scheme achieved an FPS of over 545 in the edge platform. In addition to that, we compared our method and extensively studied the performance of different deep learning-based classification methods. Our proposed method can outperform the existing classification method by a notable margin. The proposed study has been planned to be extended by incorporating quantization-aware training into future research.

Funding: This research received no funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article; further inquiries can be directed to the corresponding author.

Acknowledgments: This research is a self-motivated, independent work and did not receive any additional support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gibert, D.; Planes, J.; Mateu, C.; Le, Q. Fusing feature engineering and deep learning: A case study for malware classification. *Expert Syst. Appl.* **2022**, *207*, 117957. [CrossRef]
2. Abusitta, A.; Li, M.Q.; Fung, B.C. Malware classification and composition analysis: A survey of recent developments. *J. Inf. Secur. Appl.* **2021**, *59*, 102828. [CrossRef]
3. Awan, M.J.; Masood, O.A.; Mohammed, M.A.; Yasin, A.; Zain, A.M.; Damaševičius, R.; Abdulkareem, K.H. Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention. *Electronics* **2021**, *10*, 2444. [CrossRef]
4. Rouissat, M.; Belkheir, M.; Alsukayti, I.S.; Mokaddem, A. A lightweight mitigation approach against a new inundation attack in RPL-based IoT networks. *Appl. Sci.* **2023**, *13*, 10366. [CrossRef]
5. Pytorch. Cybercrime to Cost the World $10.5 Trillion Annually by 2025 Code. 2021. Available online: https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021 (accessed on 12 November 2022).
6. Schultz, M.G.; Eskin, E.; Zadok, F.; Stolfo, S.J. Data mining methods for detection of new malicious executables. In Proceedings of the 2001 IEEE Symposium on Security and Privacy. S&P 2001, Oakland, CA, USA, 14–16 May 2000; IEEE: New York, NY, USA, 2000; pp. 38–49.
7. Christodorescu, M.; Jha, S. Static analysis of executables to detect malicious patterns. In Proceedings of the 12th USENIX Security Symposium (USENIX Security 03), Washington, DC, USA, 4–8 August 2003.
8. Ahmadi, M.; Ulyanov, D.; Semenov, S.; Trofimov, M.; Giacinto, G. Novel feature extraction, selection and fusion for effective malware family classification. In Proceedings of the of the Sixth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 9–11 March 2016; pp. 183–194.
9. Zhang, Y.; Huang, Q.; Ma, X.; Yang, Z.; Jiang, J. Using multi-features and ensemble learning method for imbalanced malware classification. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; IEEE: New York, NY, USA, 2016; pp. 965–973.
10. Alzaidy, S.; Binsalleeh, H. Adversarial Attacks with Defense Mechanisms on Convolutional Neural Networks and Recurrent Neural Networks for Malware Classification. *Appl. Sci.* **2024**, *14*, 1673. [CrossRef]
11. Aslan, Ö.; Yilmaz, A.A. A new malware classification framework based on deep learning algorithms. *IEEE Access* **2021**, *9*, 87936–87951. [CrossRef]
12. Yuan, B.; Wang, J.; Liu, D.; Guo, W.; Wu, P.; Bao, X. Byte-level malware classification based on markov images and deep learning. *Comput. Secur.* **2020**, *92*, 101740. [CrossRef]
13. Gibert, D. *Convolutional Neural Networks for Malware Classification*; University Rovira i Virgili: Tarragona, Spain, 2016.
14. Kalash, M.; Rochan, M.; Mohammed, N.; Bruce, N.D.; Wang, Y.; Iqbal, F. Malware classification with deep convolutional neural networks. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; IEEE: New York, NY, USA, 2018; pp. 1–5.

15. Prajapati, P.; Stamp, M. An empirical analysis of image-based learning techniques for malware classification. In *Malware Analysis Using Artificial Intelligence and Deep Learning*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 411–435.
16. Singh, A.; Handa, A.; Kumar, N.; Shukla, S.K. Malware classification using image representation. In Proceedings of the Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Beer-Sheva, Israel, 27–28 June 2019; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 2019; pp. 75–92.
17. Agarap, A.F. Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification. *arXiv* **2017**, arXiv:1801.00318.
18. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
19. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
20. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
21. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 28 October–2 November 2019; pp. 1314–1324.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
23. Syeda, D.Z.; Asghar, M.N. Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning. *Appl. Sci.* **2024**, *14*, 1015. [CrossRef]
24. Gyamfi, N.K.; Goranin, N.; Ceponis, D.; Čenys, H.A. Automated system-level malware detection using machine learning: A comprehensive review. *Appl. Sci.* **2023**, *13*, 11908. [CrossRef]
25. Rey, V.; Sánchez, P.M.S.; Celdrán, A.H.; Bovet, G. Federated learning for malware detection in IoT devices. *Comput. Netw.* **2022**, *204*, 108693. [CrossRef]
26. Cheng, S.M.; Hong, B.K.; Hung, C.F. Attack detection and mitigation in MEC-enabled 5G networks for AIoT. *IEEE Internet Things Mag.* **2022**, *5*, 76–81. [CrossRef]
27. Kumar, R.; Zhang, X.; Wang, W.; Khan, R.U.; Kumar, J.; Sharif, A. A multimodal malware detection technique for Android IoT devices using various features. *IEEE Access* **2019**, *7*, 64411–64430. [CrossRef]
28. Mujtaba, G.; Ryu, E.S. Human character-oriented animated gif generation framework. In Proceedings of the 2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC), Karachi, Pakistan, 15–17 July 2021; IEEE: New York, NY, USA, 2021; pp. 1–6.
29. Zhao, D.; Ren, J.; Lin, R.; Xu, S.; Chang, V. On orchestrating service function chains in 5G mobile network. *IEEE Access* **2019**, *7*, 39402–39416. [CrossRef]
30. Rahman, A.u.; Mahmud, M.; Iqbal, T.; Saraireh, L.; Kholidy, H.; Gollapalli, M.; Musleh, D.; Alhaidari, F.; Almoqbil, D.; Ahmed, M.I.B. Network Anomaly Detection in 5G Networks. *Math. Model. Eng. Probl.* **2022**, *9*, 397. [CrossRef]
31. Sharif, S.; Naqvi, R.A.; Biswas, M. SAGAN: Adversarial Spatial-asymmetric Attention for Noisy Nona-Bayer Reconstruction. *arXiv* **2021**, arXiv:2110.08619.
32. Sharif, S.; Mahboob, M. Evil method: A deep CNN model for Bangla handwritten numeral classification. In Proceedings of the 2017 4th International Conference on Advances in Electrical Engineering (ICAEE), Dhaka, Bangladesh, 28–30 September 2017; IEEE: New York, NY, USA, 2017; pp. 217–222.
33. Wang, Z.; Cun, X.; Bao, J.; Zhou, W.; Liu, J.; Li, H. Uformer: A general u-shaped transformer for image restoration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 17683–17693.
34. Zamir, S.W.; Arora, A.; Khan, S.; Hayat, M.; Khan, F.S.; Yang, M.H. Restormer: Efficient transformer for high-resolution image restoration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 5728–5739.
35. Ahmed, I.; Anisetti, M.; Ahmad, A.; Jeon, G. A multilayer deep learning approach for malware classification in 5G-enabled IIoT. *IEEE Trans. Ind. Inform.* **2022**, *19*, 1495–1503. [CrossRef]
36. Hasan, M.K.; Ghazal, T.M.; Saeed, R.A.; Pandey, B.; Gohel, H.; Eshmawi, A.; Abdel-Khalek, S.; Alkhassawneh, H.M. A review on security threats, vulnerabilities, and counter measures of 5G enabled Internet-of-Medical-Things. *IET Commun.* **2022**, *16*, 421–432. [CrossRef]
37. Narayanan, B.N.; Djaneye-Boundjou, O.; Kebede, T.M. Performance analysis of machine learning and pattern recognition algorithms for malware classification. In Proceedings of the 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS), Dayton, OH, USA, 25–29 July 2016; IEEE: New York, NY, USA, 2016; pp. 338–342.
38. Kinable, J.; Kostakis, O. Malware classification based on call graph clustering. *J. Comput. Virol.* **2011**, *7*, 233–245. [CrossRef]
39. Anderson, B.; Storlie, C.; Lane, T. Improving malware classification: Bridging the static/dynamic gap. In Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence, Raleigh, CA, USA, 19 October 2012; pp. 3–14.
40. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 July 2011; pp. 1–7.

41. Mushtaq, S.; Alandjani, G.; Abbasi, S.F.; Abosaq, N.; Akram, A.; Pervez, S. Hybrid geo-location routing protocol for indoor and outdoor positioning applications. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 7. [CrossRef]
42. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
43. Luo, J.S.; Lo, D.C.T. Binary malware image classification using machine learning with local binary pattern. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; IEEE: New York, NY, USA, 2017; pp. 4664–4667.
44. Yeo, M.; Koo, Y.; Yoon, Y.; Hwang, T.; Ryu, J.; Song, J.; Park, C. Flow-based malware detection using convolutional neural network. In Proceedings of the 2018 International Conference on Information Networking (ICOIN), Chiang Mai, Thailand, 10–12 January 2018; pp. 910–913.
45. Alandjani, G.O. Blockchain Technology and Impacts on Potential Industries. In Proceedings of the 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC), Houston, TX, USA, 7–9 February 2023; IEEE: New York, NY, USA, 2023; pp. 1–4.
46. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; IEEE: New York, NY, USA, 2009; pp. 248–255.
47. Alandjani, G.O.; Bouk, A.H. Meme Generation Using Deep Neural Network to Engage Viewers on Social Media. *Yanbu J. Eng. Sci.* **2021**, *18*, 81–87. [CrossRef]
48. Rezende, E.; Ruppert, G.; Carvalho, T.; Theophilo, A.; Ramos, F.; Geus, P.d. Malicious software classification using VGG16 deep neural network's bottleneck features. In *Information Technology-New Generations*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 51–59.
49. Khan, R.U.; Zhang, X.; Kumar, R. Analysis of ResNet and GoogleNet models for malware detection. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 29–37. [CrossRef]
50. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
51. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 84–90. [CrossRef]
52. Gulatas, I.; Kilinc, H.H.; Zaim, A.H.; Aydin, M.A. Malware threat on edge/fog computing environments from Internet of things devices perspective. *IEEE Access* **2023**, *11*, 33584–33606. [CrossRef]
53. Alandjani, G. Leveraging vulnerabilities in sensor based IOT edge computing networks. *Int. J. Future Gener. Commun. Netw.* **2021**, *14*, 11–20.
54. HaddadPajouh, H.; Dehghantanha, A.; Khayami, R.; Choo, K.K.R. A deep recurrent neural network based approach for internet of things malware threat hunting. *Future Gener. Comput. Syst.* **2018**, *85*, 88–96. [CrossRef]
55. Medsker, L.R.; Jain, L. Recurrent neural networks. *Des. Appl.* **2001**, *5*, 2.
56. Su, J.; Vasconcellos, D.V.; Prasad, S.; Sgandurra, D.; Feng, Y.; Sakurai, K. Lightweight classification of IoT malware based on image recognition. In Proceedings of the 2018 IEEE 42Nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 23–27 July 2018; IEEE: New York, NY, USA, 2018; Volume 2, pp. 664–669.
57. Alasmary, H.; Khormali, A.; Anwar, A.; Park, J.; Choi, J.; Abusnaina, A.; Awad, A.; Nyang, D.; Mohaisen, A. Analyzing and detecting emerging Internet of Things malware: A graph-based approach. *IEEE Internet Things J.* **2019**, *6*, 8977–8988. [CrossRef]
58. Dovom, E.M.; Azmoodeh, A.; Dehghantanha, A.; Newton, D.E.; Parizi, R.M.; Karimipour, H. Fuzzy pattern tree for edge malware detection and categorization in IoT. *J. Syst. Archit.* **2019**, *97*, 1–7. [CrossRef]
59. Vasan, D.; Alazab, M.; Venkatraman, S.; Akram, J.; Qin, Z. MTHAEL: Cross-architecture IoT malware detection based on neural network advanced ensemble learning. *IEEE Trans. Comput.* **2020**, *69*, 1654–1667. [CrossRef]
60. Lu, N.; Li, D.; Shi, W.; Vijayakumar, P.; Piccialli, F.; Chang, V. An efficient combined deep neural network based malware detection framework in 5G environment. *Comput. Netw.* **2021**, *189*, 107932. [CrossRef]
61. Zhou, Y.; Jiang, X. Dissecting android malware: Characterization and evolution. In Proceedings of the 2012 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–23 May 2012; IEEE: New York, NY, USA, 2012; pp. 95–109.
62. Jeon, J.; Park, J.H.; Jeong, Y.S. Dynamic analysis for IoT malware detection with convolution neural network model. *IEEE Access* **2020**, *8*, 96899–96911. [CrossRef]
63. Sharif, S.; Mobin, I.; Mohammed, N. Augmented quick health. *Int. J. Comput. Appl.* **2016**, *134*, 1–6. [CrossRef]
64. Varsha, M.; Vinod, P.; Dhanya, K. Identification of malicious android app using manifest and opcode features. *J. Comput. Virol. Hacking Tech.* **2017**, *13*, 125–138. [CrossRef]
65. Ankita, A.; Rani, S. Machine learning and deep learning for malware and ransomware attacks in 6G network. In Proceedings of the 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, 3 July 2021; IEEE: New York, NY, USA, 2021; pp. 39–44.
66. Sousa, B.; Dias, D.; Antunes, N.; C'amara, J.; Wagner, R.; Schmerl, B.; Garlan, D.; Fidalgo, P. MONDEO-Tactics5G: Multistage botnet detection and tactics for 5G/6G networks. *Comput. Secur.* **2024**, *140*, 103768. [CrossRef]
67. Mahmood, I.; Alyas, T.; Abbas, S.; Shahzad, T.; Abbas, Q.; Ouahada, K. Intrusion Detection in 5G Cellular Network Using Machine Learning. *Comput. Syst. Sci. Eng.* **2023**, *47*. [CrossRef]

68. Basnet, M.; Poudyal, S.; Ali, M.H.; Dasgupta, D. Ransomware detection using deep learning in the SCADA system of electric vehicle charging station. In Proceedings of the 2021 IEEE PES Innovative Smart Grid Technologies Conference-Latin America (ISGT Latin America), Lima, Peru, 15–17 September 2021; IEEE: New York, NY, USA, 2021; pp. 1–5.
69. Anand, A.; Rani, S.; Anand, D.; Aljahdali, H.M.; Kerr, D. An efficient CNN-based deep learning model to detect malware attacks (CNN-DMA) in 5G-IoT healthcare applications. *Sensors* **2021**, *21*, 6346. [CrossRef]
70. Sharif, S.; Mahboob, M. Deep hog: A hybrid model to classify bangla isolated alpha-numerical symbols. *Neural Netw. World* **2019**, *29*, 111–133. [CrossRef]
71. Mujtaba, G.; Khowaja, S.A.; Jarwar, M.A.; Choi, J.; Ryu, E.-S. FRC-GIF: Frame Ranking-Based Personalized Artistic Media Generation Method for Resource Constrained Devices. *IEEE Trans. Big Data* **2023**, 1–14. [CrossRef]
72. Wei, Y.; Xiao, H.; Shi, H.; Jie, Z.; Feng, J.; Huang, T.S. Revisiting dilated convolution: A simple approach for weakly-and semi-supervised semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7268–7277.
73. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
74. Pytorch. PyTorch Framework Code. 2016. Available online: https://pytorch.org/ (accessed on 27 March 2024).
75. Sharif, S.; Mahboob, M. A comparison between hybrid models for classifying Bangla isolated basic characters. In Proceedings of the 2017 4th International Conference on Advances in Electrical Engineering (ICAEE), Dhaka, Bangladesh, 28–30 September 2017; IEEE: New York, NY, USA, 2017; pp. 211–216.
76. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
77. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
78. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
79. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 10012–10022.
80. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
81. Zagoruyko, S.; Komodakis, N. Wide residual networks. *arXiv* **2016**, arXiv:1605.07146.
82. Sun, R.Y. Optimization for deep learning: An overview. *J. Oper. Res. Soc. China* **2020**, *8*, 249–294. [CrossRef]
83. Liu, Z.; Wang, Y.; Han, K.; Zhang, W.; Ma, S.; Gao, W. Post-training quantization for vision transformer. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 28092–28103.
84. NVIDIA Corporation. TensorRT. Available online: https://developer.nvidia.com/tensorrt (accessed on 2 April 2024).
85. Hubara, I.; Nahshan, Y.; Hanani, Y.; Banner, R.; Soudry, D. Accurate post training quantization with small calibration sets. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 4466–4475.
86. Liang, T.; Glossner, J.; Wang, L.; Shi, S.; Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* **2021**, *461*, 370–403. [CrossRef]
87. NVIDIA Corporation. Jetson Orin NX Series and Jetson AGX Orin Series. Available online: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/ (accessed on 2 March 2024).
88. Nagel, M.; Fournarakis, M.; Bondarenko, Y.; Blankevoort, T. Overcoming oscillations in quantization-aware training. In Proceedings of the International Conference on Machine Learning, Baltimore, MA, USA, 17–23 July 2022; pp. 16318–16330.