electronics

MDPI

Article

Enhanced Image-Based Malware Classification Using Transformer-Based Convolutional Neural Networks (CNNs)

Moses Ashawa [1,*], Nsikak Owoh [1], Salaheddin Hosseinzadeh [1] and Jude Osamor [2]

[1] Department of Cybersecurity and Networks, Glasgow Caledonian University, Glasgow G4 0BA, UK; salaheddin.hosseinzadeh@gcu.ac.uk (S.H.)
[2] School of Computer Science & Engineering, University of Westminster, London W1B 2HW, UK; j.osamor@westminster.ac.uk
* Correspondence: moses.ashawa@gcu.ac.uk

Abstract: As malware samples grow in complexity and employ advanced evasion techniques, traditional detection methods are insufficient for accurately classifying large volumes of sophisticated malware variants. To address this issue, image-based malware classification techniques leveraging machine learning algorithms have been developed as a more optimal solution to this challenge. However, accurately classifying content distribution-based features with unique pixel intensities from grayscale images remains a challenge. This paper proposes an enhanced image-based malware classification system using convolutional neural networks (CNNs) using ResNet-152 and vision transformer (ViT). The two architectures are then compared to determine their classification abilities. A total of 6137 benign files and 9861 malicious executables are converted from text files to unsigned integers and then to images. The ViT examined unsigned integers as pixel values, while ResNet-152 converted the pixel values into floating points for classification. The result of the experiments demonstrates a high-performance accuracy of 99.62% with effective hyperparameters of 10-fold cross-validation. The findings indicate that the proposed model is capable of being implemented in dynamic and complex malware environments, achieving a practical computational efficiency of 47.2 s for the identification and classification of new malware samples.

Keywords: malware classification; pixel intensity; gray images; neural networks; image visualization

check for updates

Citation: Ashawa, M.; Owoh, N.; Hosseinzadeh, S.; Osamor, J. Enhanced Image-Based Malware Classification Using Transformer-Based Convolutional Neural Networks (CNNs). *Electronics* **2024**, *13*, 4081. https://doi.org/10.3390/electronics13204081

Academic Editor: Zbigniew Kotulski

Received: 12 September 2024
Revised: 3 October 2024
Accepted: 15 October 2024
Published: 17 October 2024

1. Introduction

Malicious software poses a significant risk to individuals, businesses, and corporate entities. Known collectively as malware, these applications comprise a wide range of harmful programs designed to exploit human and system vulnerabilities. Malware includes diverse types such as viruses, worms, ransomware, and spyware, each contributing to an increasingly dynamic and complex threat environment. The prevalence and complexity of malware attacks have risen sharply in recent years, highlighting the growing prevalence, complexity, and severity of these cyber threats [1]. Techniques for enhancing malware sophistication involve a range of advanced strategies aimed at evading detection, exploiting system vulnerabilities, and increasing the overall impact of attacks. Among these techniques, obfuscation methods such as polymorphism and metamorphism are commonly employed to obscure the malicious code and hinder efforts to identify and neutralize the threat [2,3].

There are traditional techniques designed to detect malware. However, traditional methods heavily rely on malware signature [4] and other conditions such as system configuration, user activity, environment checks, and other time-based conditions. While antivirus tools such as virustotal are effective in detecting signature and conditional code malware, variants with polymorphic or obfuscation attributes can easily evade detection. Hence, the introduction of machine learning techniques. The integration of machine learning and artificial intelligence has significantly improved decision-making processes and strengthened security measures, such as fraud detection, threat identification, and other anomaly detections, including malware. Machine learning algorithms detect more sophisticated malware variants [5].

However, hackers are also leveraging machine learning techniques to their advantage, enhancing the capabilities of malicious software to bypass detection mechanisms and evade security detections and controls [6–10]. Also, many of the existing machine learning techniques are faced with diverse challenges when classifying malware based on images, such as the loss of critical structural and behavioral information when converting binaries into images [11]. This challenge can cause malware data and visual misrepresentation, which can lead to decreased accuracy in distinguishing between different malware families and benign software.

In image-based malware classification, visual representations of malware patterns are used to identify malicious software. However, when malware authors manipulate the binary values within the malware's converted text file, this modification alters the corresponding image representation without impacting the malware's functionality. Such changes can deceive existing image-based classifiers, leading to the erroneous classification of the malware as benign [12,13].

Several researchers have enhanced detection accuracy by addressing the issue of class imbalance in malware datasets through the application of techniques such as the bat algorithm [14] and data augmentation techniques [15] to analyze large amounts of data to identify patterns and similarities among different types of malware [16]. However, the existing malware classification techniques have not yet addressed the challenge of differentiating between floating-point numbers and pixel intensity within flat regions of images [17,18]. Inaccurate differentiation between the decimal equivalents of unsigned integer byte values in malware binaries and floating-point numbers or pixel intensity values can significantly impair image classification accuracy. To mitigate this issue, we propose a Convolutional Neural Network (CNN) approach that effectively distinguishes between benign and malicious applications by accurately differentiating floating-point numbers from pixel intensity values in unsigned malware integers. This technique not only improves classification performance but also provides a visual representation of the malware byte stream. The main scientific question in this paper is as follows: How effective are image-based malware classification techniques, specifically convolutional neural networks (CNNs) using ResNet-152 and vision transformers (ViT), in accurately classifying malware variants based on content distribution features extracted from grayscale images?

The main contributions of this paper are as follows:

1. We propose a novel, enhanced image-based malware classification technique that combines convolutional neural networks (CNNs) with image-based processing for malware detection. By converting malware samples into grayscale images, the framework captures intricate attributes and behaviors that are often missed by existing methods.
2. The proposed method integrates structured features with content-based features, processed through principal component analysis (PCA) for normalization and dimensionality reduction. Integrating these features enhances the model's efficiency by reducing computational complexity while retaining essential information for accurate malware classification. The combination of these techniques ensures the model's scalability.
3. We present a comprehensive, step-by-step methodology for converting malware samples from executable binaries to images. This approach includes distinguishing between floating-point numbers and pixel intensity values, which is critical for accurate malware classification.

The remainder of this paper is structured as follows: Section 2 reviews the related work. Section 3 covers the proposed model, including data collection, feature extraction, image representation, and feature normalization. Section 4 presents and analyzes the research results. Finally, Section 5 concludes the paper and outlines directions for future work.

2. Related Work

The related work considered in this study is focused on machine learning algorithms for malware detection and classification, looking at their strengths and weaknesses based on the results obtained in those previous studies. We also looked at previous works on domain knowledge and information gain for malware classification. Some commonly used machine learning algorithms for malware classification with their strengths and limitations in this related work include decision trees, random forests, support vector machines, and neural networks. Decision trees are simple and easy to interpret, but they may suffer from overfitting and lack generalization ability, as discussed by Lan et al. [19]. Random forests can handle large malware datasets and are less prone to overfitting, but they may be computationally expensive [20]. As Kharoubi et al. state, support vector machines are effective for binary classification tasks and can handle high-dimensional data, but they may struggle with large datasets [21]. In studying the detection of complex virus particles, Ito et al. [22] stated that neural networks are highly flexible and can learn complex patterns. Still, they require a large amount of training data and may be susceptible to overfitting.

In recent years, researchers have been exploring ways to overcome the limitations of individual machine learning algorithms by combining them into ensemble models. Ensemble models, such as gradient boosting and stacking, aim to leverage the strengths of multiple algorithms while mitigating their weaknesses. By combining the predictions of multiple models, ensemble methods can often achieve higher accuracy and better generalization performance than any individual algorithm alone. A detailed survey on ensemble learning in [23] stated that overfitting avoidance, representation, and computational advantage of the algorithm are effective for solving class imbalance and concept drift, which occur during feature distribution and label change. Additionally, because other models in the ensemble can correct errors made by one model, ensemble models can offer improved robustness and stability. Ensemble models work by training multiple individual models and then combining their predictions to make a final prediction.

Gradient boosting is a well-liked ensemble method that sequentially trains weak models and combines their predictions in a way that focuses on the samples that the earlier models failed to predict well, as highlighted in [24]. This iterative process helps to improve the overall accuracy of the ensemble model during an early stage of the malware attack. Another commonly used ensemble method is stacking [25], which involves training multiple models and then training a meta-model on the predictions of these models. The meta-model learns to combine the predictions of the individual models in a way that optimizes the final prediction. This stacking technique can further improve the overall performance of the models. One advantage of stacking is that it can capture different aspects of the data by using diverse base models [26]. Each base model may have its strengths and weaknesses, and by combining their predictions, the ensemble can leverage the strengths of each model while mitigating their weaknesses. Combining the strengths of the models can lead to a more robust and accurate final prediction. Additionally, stacking allows for the incorporation of various types of models, such as linear models, tree-based models, or neural networks, which can capture different types of relationships in the data.

By combining these diverse models, the ensemble can exploit the complementary information they provide, resulting in improved performance and more accurate predictions. Furthermore, stacking can also help reduce overfitting as it combines multiple models that have been trained on different subsets of the data. This diversity in training data helps to generalize the predictions and prevent the ensemble from relying too heavily on any one model. Moreover, the process of stacking involves a meta-model that learns how to combine the predictions of the base models effectively. This meta-model acts as a final layer of learning, refining the predictions of the individual models and potentially uncovering patterns or relationships that were not evident in the original data.

Neural networks' machine learning algorithm for malware analysis is a powerful tool in detecting and classifying malicious software. By training the neural network with a large dataset of known malware samples, it can learn to identify patterns and characteristics that are common among malware, allowing it to detect and classify new, previously unseen malware samples accurately. Using a deep neural network, Ding and Zhu [27] detected opcode sequences and feature vectors of malware executables to determine the intensity of the malware threat. They used a combination of static and dynamic analysis techniques to collect the necessary data for their study. By analyzing the opcode sequences, which represent the low-level instructions executed by a program, they were able to identify patterns and characteristics specific to malware. Additionally, the feature vectors provided a higher-level representation of the executable files, capturing important attributes such as file size, entropy, and imported functions. The comprehensive approach allowed them to accurately assess the level of threat posed by each malware sample during classification.

One approach to malware classification is to utilize domain knowledge and information gain-based techniques in conjunction with neural networks. By incorporating domain knowledge, such as understanding the characteristics and behaviors of different types of malware, the classification model can make more informed decisions. Wagner et al. [28] used an arc diagram to visualize malware samples using domain knowledge to externalize the rules for an easy analysis process. The externalized knowledge aids in understanding the correlation between the rule table and its details in the classification function of the feature attributes. This research is similar to the one performed by Ostaheli in [29], which adopted an information gain approach to classify malware. The related work is summarized below:

3. Method

This section outlines the methodology employed in the development and implementation of the proposed framework. The proposed framework consists of six primary methodological components: data collection, feature extraction and conversion, image representation, normalization, image classification, and edge detection. Each of these components is integral to the overall functionality and performance of the system. Each of the methodological components are discussed in the subsequent sections. The proposed system architecture integrates various feature sets, including pixel intensity, texture patterns, grayscale, color, sequence, opcode frequencies, entropy, etc. The model incorporates both structured and content-based malicious features to enhance its robustness and performance, particularly for identifying sophisticated samples with encrypted or polymorphic attributes. This approach is informed by the understanding that analyzing specific content within malicious payloads can reveal critical functionalities, such as initiating denial-of-service attacks or facilitating backdoor installations.

3.1. Data

The malware sample used in this paper is obtained and filtered from the MalNet project [30]. A dataset comprising 17,094 malicious samples and 13,482 benign samples was collected and stored on a network attached storage system with two virtual machines (VM1 and VM2) (see Figure 1). The collected samples were subsequently refined to include 6137 benign and 9861 malicious instances, with redundant features eliminated using the technique described by Ahmed et al. [31]. VM1 is allocated for analytical tasks, whereas VM2 is assigned for monitoring purposes. A significant advantage of our proposed model is its seamless integration with virtual machine nodes and SQL servers, which optimizes log retrieval and analysis processes, thereby reducing the necessity for manual intervention in monitoring malware escalation during our experiment.

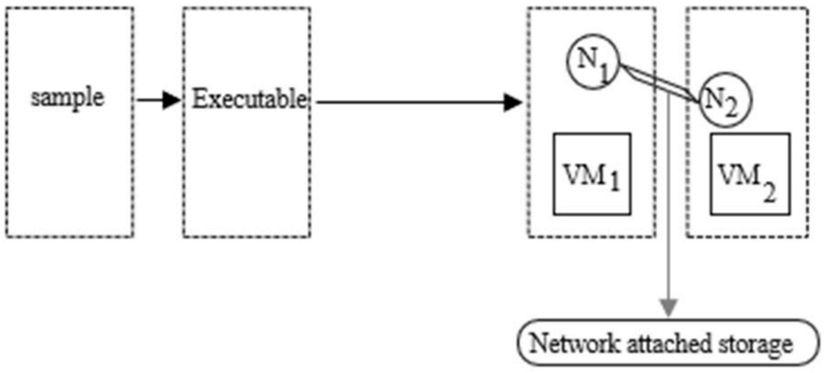


Figure 1. Showing how the virtual machines are configured to store the executable files.

3.2. CNN Architecture Selection

In the context of malware classification, choosing the appropriate convolutional neural network (CNN) architecture is crucial to balance accuracy, computational efficiency, and generalization. In this paper, we choose ResNet-152 architecture over existing options such as VGG, BN-Inception [32], DenseNet-201 [33], and MobileNet-v2 architecture [34] because of its several advantages compared to other CNN architectures. Two key advantages were considered when choosing the ResNet-152 architecture for the proposed model. First, ResNet-152 has residual learning abilities to mitigate degradation problem [35], which is common in other architecture such as VGG. ResNet-152 allows the proposed model to capture intricate sample patterns without gradient vanishing which is important in classifying large and complex malware datasets.

Second, ResNet-152 allows for the progressive extraction of both low-level and high-level features which are crucial in detecting subtle differences in pixel intensity. Malware can exhibit slight disparities from benign software, and the ability of a model to discern these subtle differences through deeper images is key. Compared to VGG, which has a fixed set of convolutional layers, ResNet's residual connections allow the network to learn more detailed and nuanced features. While VGG architectures have strong performance on small- to medium-sized datasets, they lack the residual capabilities of ResNet. Inception architectures were not a choice for the proposed model. In [36], Zhang et al. noted that inception is less effective in capturing deep hierarchical relationships needed for complex image malware classification tasks.

3.3. Feature Extraction and Conversation

Figure 2 illustrates the step-by-step process for converting an executable file into an image. Initially, the executable file stored on VM1 was converted into a text file containing features such as edges, pixel arrangements, color distribution, section names, and shape transitions.

Malware samples, when represented as images, can reveal intricate patterns based on the distribution of byte values. The pixel intensity feature captures the brightness of each pixel, which corresponds to the underlying byte or sequence of bytes in the malware file. This feature is essential for identifying subtle byte-level changes or regularities that may signify malicious behavior. Grayscale simplifies the data by reducing the dimensionality of the color space to a single channel. Grayscale feature is chosen to enable the proposed model focus on structural similarities and differences in the malware data without the distraction of color variations. Other features such as edges, patterns, and section names were chosen for the fact that malware samples may exhibit specific repeating or unique patterns that are best captured as textures within the image representation. Edges and section names enable the detection of spatial relationships between pixel intensities. This enables the proposed model to identify underlying structural features that may indicate obfuscation and compression methods used by the malware.

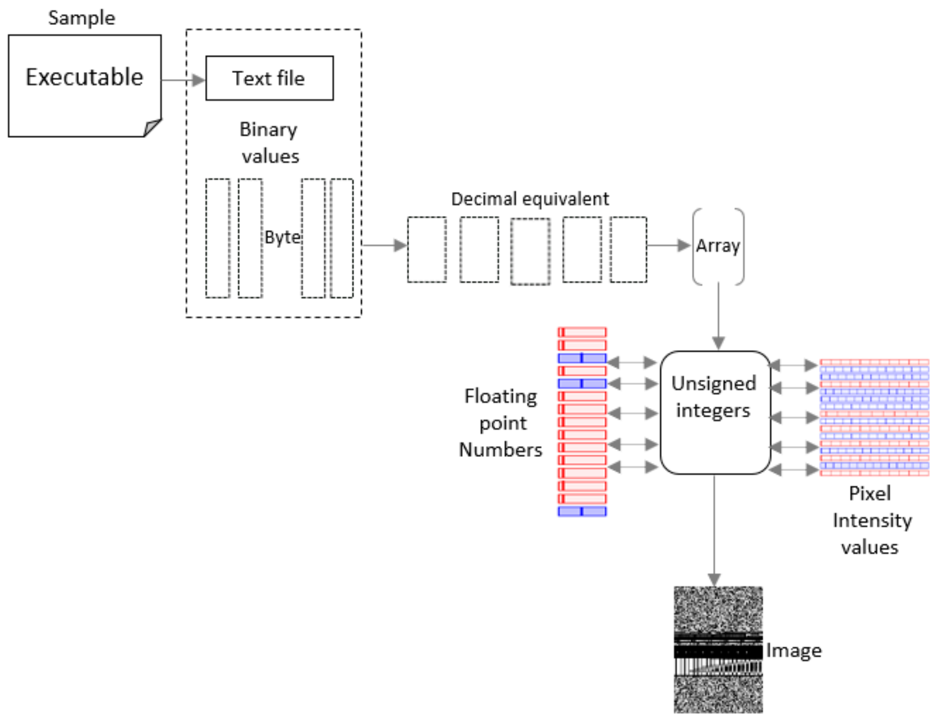


Figure 2. Feature extraction and conversion process.

This text file represents a byte stream encompassing the binary values of the executable. In malware analysis, it is crucial to transform raw byte data into a structured format suitable for machine learning algorithms. The byte stream was converted into its decimal equivalents. This method facilitates the normalization of the data, allowing the proposed model to more effectively represent sequences or frequencies of specific byte values and thus simplify the data representation for improved analysis. Following the techniques of Washington et al. [37], we stored the decimal equivalent into an array and converted it into unsigned integers accordingly. The unsigned integers are then represented as floating point numbers and pixel intensity values.

Representing malware samples as images and classifying them based on pixel intensity values and floating-point numbers to derive rich feature sets and exploit the detailed spatial information inherent in pixel values. This potentially uncovered color patterns that are not apparent through conventional analysis. In scenarios where sophisticated malware alters its code to evade detection, the proposed model analyzes and compares the floating-point representations and pixel intensity patterns to identify and uncover underlying similarities. The unsigned integers are then converted into a PNG image file. PNG image was chosen to avoid the limitation associated with GIF, which has only a 256-color palette.

The preprocessing strategy adopted in this paper addressed two key challenges. First raw byte streams from executable files are complex, unstructured, and not directly suitable for machine learning algorithms, as they do not inherently convey meaningful patterns or structures for analysis. The conversion of raw byte data into decimal equivalents facilitates structuring the data in a more accessible and interpretable format for the proposed model. Secondly, binary and byte-level representations can be overwhelming and may not reveal clear patterns that can be easily analyzed in machine learning models. Transforming the data into floating-point numbers and pixel intensity values creates a simplified representation, which aids in deriving rich feature sets and detecting hidden patterns more effectively. This also addressed the challenge of malware evasion. This analyzes subtle pixel intensity patterns and floating-point representations to uncover underlying similarities that traditional methods may miss.

3.4. Image Representation

This subsection provides a brief overview of the various methods for representing malware samples. These methods include graphical views, image views, high-level language representations, assembly code representations, network traffic analyses, and hexadecimal views. Each method offers a unique perspective on the malware, facilitating diverse approaches to analysis and detection [38]. The graphical view represents malware behavior or code flow using graphical diagrams such as state machines, call graphs, or flowcharts.

The state machine represents malware by describing its behavior in terms of a finite number of states, transitions between those states, and the actions associated with the transitions. When representing malware using state machines, each state represents a specific behavior of the malware, and transition represents how the malware moves from one state to another based on certain conditions or events. For example, as shown in Figure 3, the idle state S_0 is when a user downloads a file or application from a source with a suspicious extension such as .exe, .bat, etc. S_1 is when the threat actor spoofs the suspicious extensions for the malware to be injected into the target system when certain conditions are achieved. S_2 and S_3 represent the payload delivery and spreading state where the target system enables file encryptions and modifications during the infection state S_4. S_5 is the dynamic operation that involves transforming the contents of a vast array of user files modified in S_4 into encrypted data. Our approach uses an image representation approach. To represent malware as an image using structured and content-based features, we convert the malware sample into a raw binary stream and memory dump separately. Both features are then converted into an image by encoding it into pixel values of images. Malware, like any other digital data, is typically stored in binary format [39], consisting of a sequence of 0 s and 1 s. This sequence is represented by binary encoding for easy manipulation and processing into any form, such as digital images.

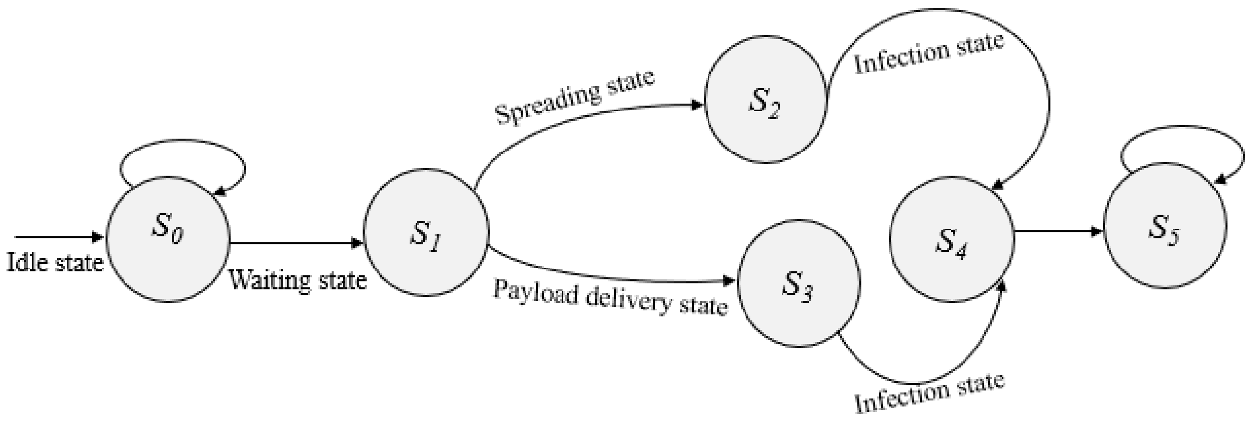


Figure 3. State machine malware representation.

In this paper, malware is represented as a digital image. Each pixel represents a single point in the image and is assigned a specific value that determines its color and intensity based on the feature attribute. To represent malware as an image, we divide the binary data of the collected malware sample into bytes. Each byte is mapped to a corresponding pixel value in the image by converting the binary values to grayscale intensity values, as shown in Figure 4.

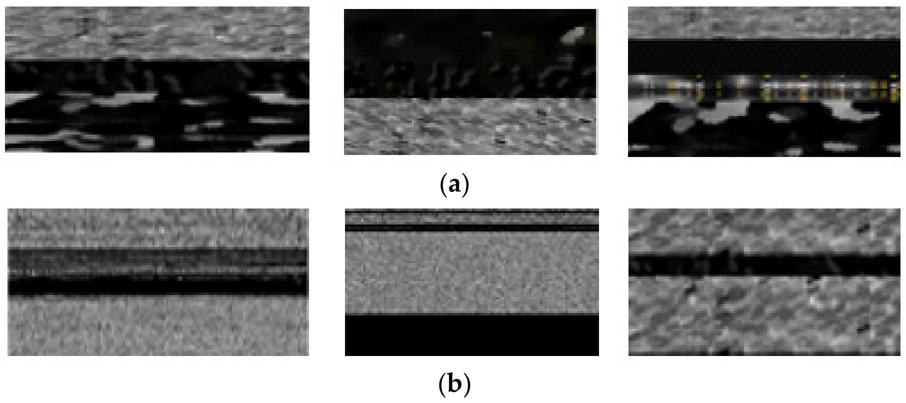


Figure 4. Rendering of malware sample image. (**a**) Pictures integrated in the malware sample and (**b**) Images of malware that share similarities across various malware categories.

3.5. Feature Normalization

A combination of structured and content-based features was extracted from the collected dataset, as discussed in the previous sections. Figure 5 shows a logical flow of the image magnitude as input into the model. Features with high dimensions were reduced using principal component analysis as follows:

$$PC_a = c_1X_1 + c_2X_2 + \cdots + c_dX_d \tag{1}$$

where PC is the principal component (a), which is the dependent variable; X_d are original independent features represented as X_j; and $c_1,\ c_2 \ldots, c_d$ are coefficients determining the impact of each independent variable. We normalized the features for effective model performance using the min–max normalization equation to scale the values of each variable $X_1, X_2, \ldots, X_d$ to a range between 0 and 1 as follows:

$$\text{Normalized}_{\text{Xi}} = \frac{X_i - Min_{x_i}}{Max_{X_i} - Min_{X_i}} \tag{2}$$

$$Normalized\_PCa = Normalized\_c1 \times Normalized\_X1 + Normalized\_c2 \times Normalized\_X2 + \ldots + Normalized\_cd \times Normalized\_Xd \tag{3}$$

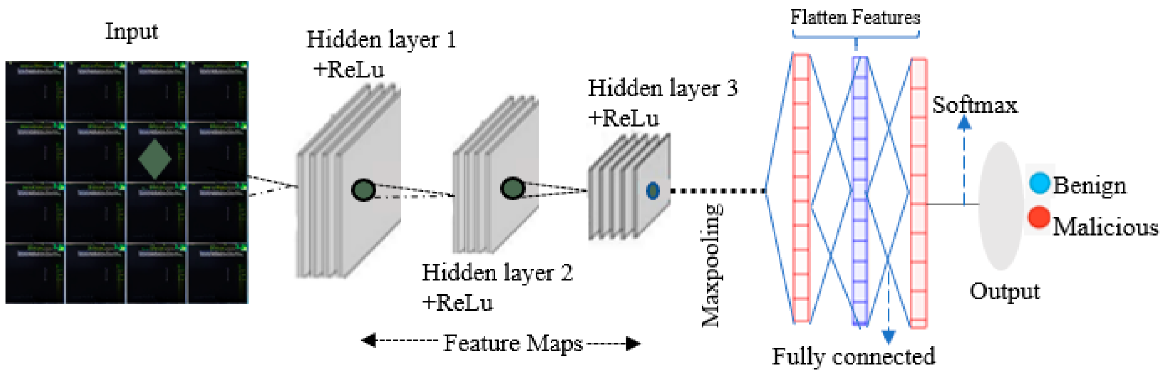


Figure 5. The summary of the architectures of the proposed enhanced image-based malware classification model.

Equation (2) scales the values of X_i to the range [0, 1] and transforms the values of X_i, which represent features, into a standardized range between 0 and 1. This scaling process is often employed in machine learning to bring all values into a comparable range. Equation (3), on the other hand, normalizes the coefficients associated with these features.

Using the stochastic method, we convert the normalized binary features into image byte streams as follows:

$$D_i = i \in \{\mu, \nu \ldots, \Psi - 1\} \tag{4}$$

where Ψ represents the binary length of malware, D_i refers to the ith byte in a sequence, and μ and ν are the range values starting from μ and ending at $\Psi - 1$, $D_i \in \{0, 1 \ldots, \Psi - 1\}$. To form a single binary stream for malware image, we concatenate all the D_i sequences together as follows:

$$\boldsymbol{S} = D_\mu D_\nu \ldots D_{\Psi-1} \tag{5}$$

$$\boldsymbol{S} = f\left(D_\mu D_\nu \ldots D_{\Psi-1}\right) \tag{6}$$

where $\boldsymbol{S}$ is the concatenated stream, f represents the function that combines all the individual binary sequences D_i into the final binary stream $\boldsymbol{S}$. f depends on how the sequences are to be combined. In this paper, we use simple concatenation instead of logical operations presented in Equation (6), which provides a general framework for representing the binary stream based on the sequences of the binary byte from the collected dataset used in this paper. To obtain an image representation that can be fed into our model, we convert the byte stream executable into a grayscale image where each pixel of the grayscale represents a feature value. We organize feature values and assign unique intensities for image pattern similarities. To process the concatenated byte stream$\boldsymbol{S}$ into an image magnitude for the model, we use the edge detection equation as follows:

$$ED = EdgeDet(\boldsymbol{S}) \tag{7}$$

$$ED = EdgeDet\left(f\left(D_\mu D_\nu \ldots D_{\Psi-1}\right)\right) \tag{8}$$

$$ED = EdgeDet\left(f\left(\frac{D_\mu - min(D_\mu)}{max(D_\mu) - min(D_\mu)}, \frac{D_v - min(D_v)}{max(D_v) - min(D_v)}, \ldots, \frac{D_{\Psi-1} - min(D_{\Psi-1})}{max(D_{\Psi-1}) - min(D_{\Psi-1})}\right)\right) \tag{9}$$

Equation (7) illustrates a direct application of edge detection to the byte stream $\boldsymbol{S}$. Equation (8) presents a more refined methodology, where $\boldsymbol{S}$ is first transformed into a set of feature vectors before edge detection is applied. This method forms the core of our machine learning pipeline, which aims to accentuate specific features or structures within the image by detecting edges after processing the raw data. By focusing on structural properties, our approach enhances the capability of the machine learning pipeline to classify patterns within the image more effectively than existing models. As illustrated in Figure 5, we utilize a convolutional neural network (CNN) to learn from the labeled dataset and develop a predictive model. During the training phase, the data are partitioned into training and validation subsets to facilitate both the model's learning process and the assessment of its performance. This process involves the iterative tuning of hyperparameters to optimize accuracy and generalization, with entropy loss serving as the criterion for training the network.

The architecture of the convolutional neural network (CNN) implemented in this paper comprises several key layers. The model begins with an input layer designed to accept grayscale images of malware binaries. Following this, the images are processed through three hidden layers, each equipped with 128 filters of size 5 × 5 and a stride of 1 to allow each filter to traverse the input image one pixel at a time. To mitigate the risk of overfitting and reduce dimensionality, a 2 × 2 pooling window is applied, facilitating the extraction of dominant features from the input data. To further combat overfitting, a dropout rate of 0.3 is integrated into the architecture. The final layer consists of two neurons using SoftMax activation function to enable the model to output probabilities for binary classification.

4. Results and Discussion

This section presents the results, including a comprehensive analysis of the model's performance and evaluation metrics. In this paper, different visualization tools are used to interpret how different layers of our model processed and classified the image input.

4.1. Image Classification

To evaluate the robustness and accuracy of our method, we applied a 10-fold cross-validation to the entire dataset. The classification process was based on features extracted from the image magnitude formats, as discussed in the previous section. Upon importing the image magnitude for training, only a grayscale representation was generated (see Figure 6). We then trained the model with the imported sample and performed 10 iterations at each fold. At 0 iteration, our CNN model processed feature maps as a holistic representation of the malware samples, identifying both benign and malicious characteristics within each sample. However, this iteration lacked the necessary visualization clarity to distinguish between benign and malicious attribute data points in the image magnitude. To address this, we increased the iteration count at intervals of 2000 counts (see Figure 7) while maintaining the Kruskal stress level at 0.5. This approach was aimed at enhancing the model's capability to differentiate between benign and malicious features within the malware dataset.

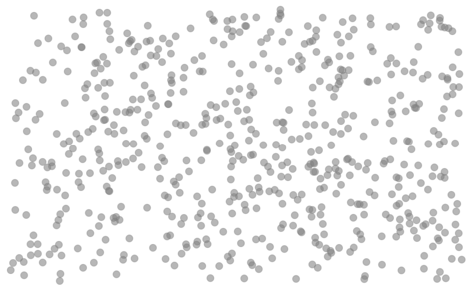


Figure 6. Training at zero iterations.

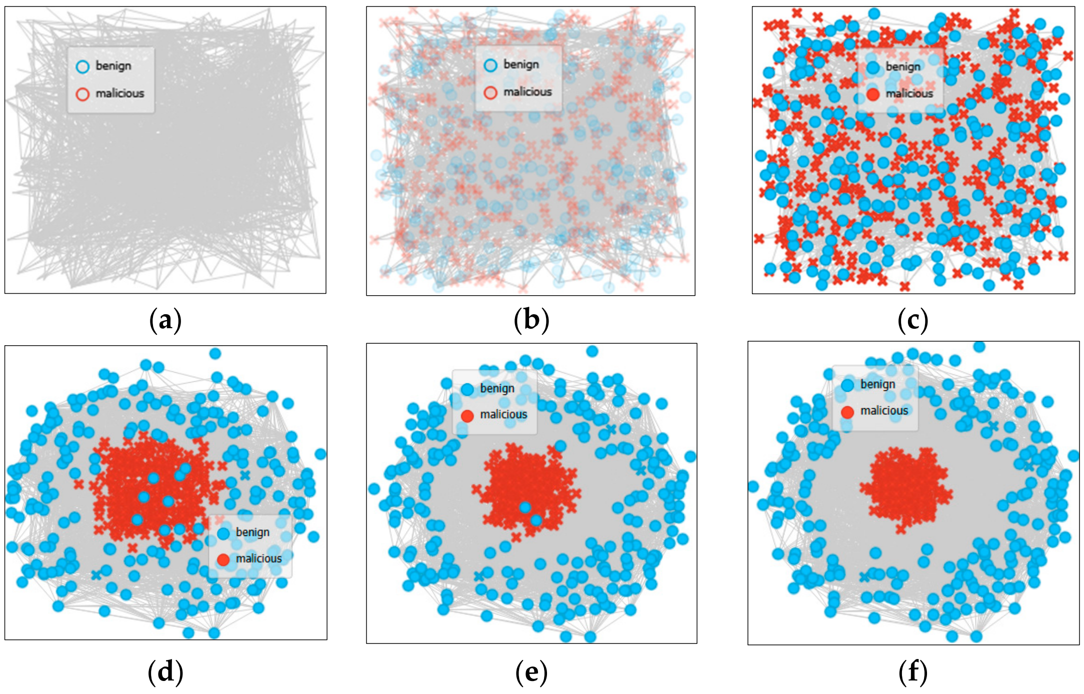


Figure 7. *Cont.*

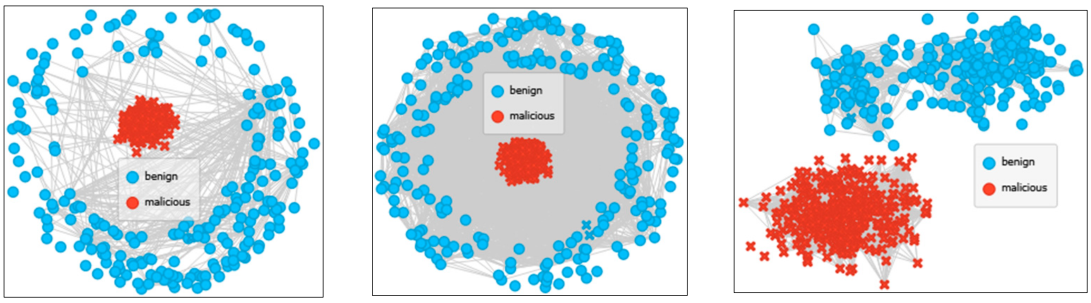


Figure 7. Image classification at different iterations. (**a**) 2000th count, (**b**) 4000th count (**c**) 6000th count, (**d**) 8000th count, (**e**) 10,000th count, (**f**) 12,000th count, (**g**) 14,000th count, (**h**) 16,000th count and (**i**) 18000th count.

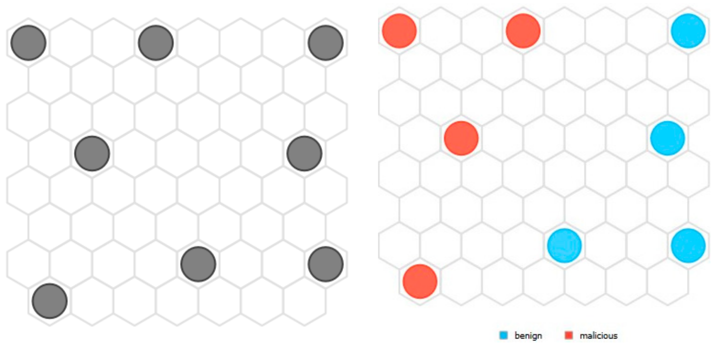
The consistent Kruskal stress level of 0.5 indicates that the edge detection technique employed in the model's multi-dimensional scaling was meticulously controlled, ensuring both the quality and interpretability of feature mapping in these subsequent iterations. This balance allowed our model to improve classification accuracy while avoiding overfitting and preserving the interpretability of the feature space.

The 10,000th and 14,000th iterations clustered similar features within the sample space across all concatenated image streams. At the 18,000th iteration, the features were distinctly classified by our model (see Figure 7a–i). Using multi-dimensional scaling [40], we were able to visualize the data and identify patterns and relationships between different malware samples at every iteration.

4.2. Edge Detection

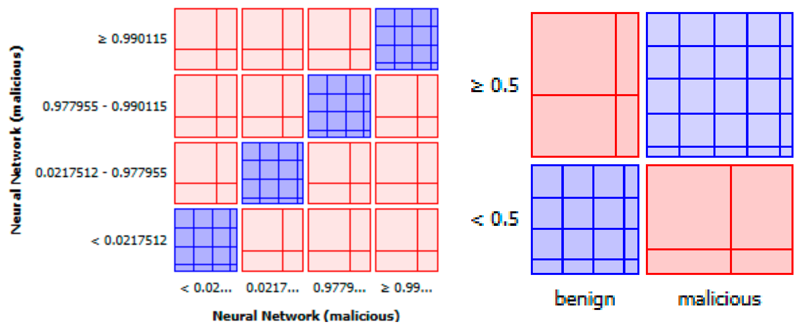
Self-organizing map (SOM) is an unsupervised neural network model that reduces dimensionality by mapping high-dimensional data into a lower-dimensional grid, typically 2D, while preserving the topological properties of the input space [41]. We applied the SOM technique to the flattened feature space obtained from CNNs after max-pooling. Flattening here refers to converting the multi-dimensional output of CNN layers into a one-dimensional vector [42]. By reducing dimensionality using SOM, our model was enabled to focus on the most salient features that are most indicative of whether a byte stream is malicious or benign. We set SOM initially to automatically determine its dimensions, which would influence the granularity of clustering. However, this approach did not lead to the successful classification of the byte streams, suggesting that the automatically determined grid size may not have been optimal, possibly because it either over-clustered or under-clustered the data. This resulted in non-separability between malicious and benign byte streams (see Figure 8a). From Equation (3), we used PCA to reinitialize the SOM with specific dimensions (8×8 grid) to balance the resolution of micro-cluster (see Figure 8b).

A sieve diagram in the context of convolutional neural networks (CNNs) is a visualization tool used to understand and interpret how a CNN processes an image, particularly how different layers of the network filter or "sieve" the input data [43]. Each layer of our CNNs model applies a set of filters to the input image, and different patterns are classified based on their edges, as stated in Equations (7)–(9). After the filters are applied, the resulting feature maps represent the areas of the image that activated or responded to these filters. The sieve diagram then visualized how these feature maps evolve as data pass through each layer of the model network. In the first layer, filters detected the edges (Figure 9a) by identifying the structural features from the malware images and then classified edges (Figure 9b) by segmenting the images into regions. Figure 10 shows the principal component analysis evolution to reduce the computational cost of the model.



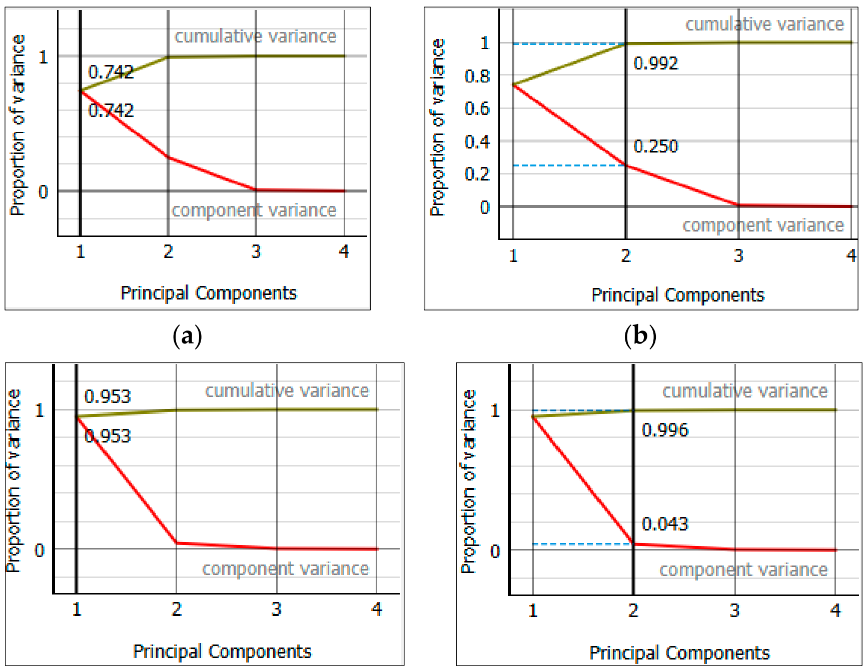
(**a**) (**b**)

Figure 8. Image SOM visualization. The gray color shows that there is no specific categorization of the image pixel intensity.



(**a**) (**b**)

Figure 9. Image sieve diagram visualization for the sample space showing benign and malicious classes.



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

Figure 10. PCA dimensionality reduction. (**a**) First component, (**b**) second component, (**c**) third component, and (**d**) fourth component.

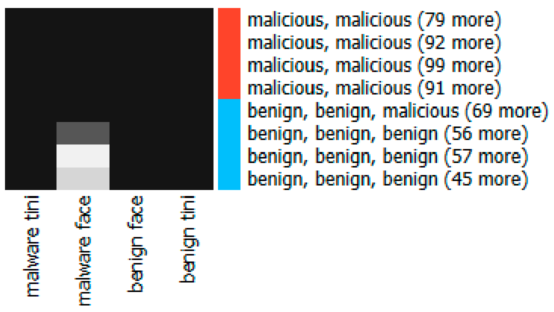
This component captured the most dominant features of the input data, such as the edge patterns of the overall texture in image input. The second component captures 99.2% of the variance. This exceptionally high value demonstrates that almost all remaining variability in the dataset after accounting for the first component is captured. This component might be focusing on slightly different, yet still crucial, aspects of the malware data that were not fully captured by the first component. The principal component indicates the proportion of the dataset's total variance captured by the components. The result indicates how much of the data's information was preserved when projecting onto those components. The first principal component captures 74.2% of the total variance in the data. Suggesting that a substantial portion of the malware dataset's variability is explained by this component, which is interpreted as the most significant direction in the data. With over 70% of variance, it suggests that the proposed model can already make reasonably good distinctions between the benign and malicious samples based on this component alone. This indicates that the proposed model can capture the distinguishing characteristics between benign and malicious samples. This likely allows for good classification performance in terms of accuracy, as most of the variance is captured in the first few components. The third component captures 95.3% of the variance.

This indicates that even the third principal component is highly significant, capturing most of the remaining variance after the first two components. This might correspond to subtler features in the data, potentially finer patterns or textures in images that further enhance the model's ability to distinguish between classes. The fourth component has an explained variance of 99.6%, which is very high. To summarize, 95.3% and 99.6% of the variance indicates that the proposed model is able to represent samples' complexity even with dimensionality reduction and still retains the dataset's important characteristics to allow the model to efficiently differentiate between benign and malicious samples.

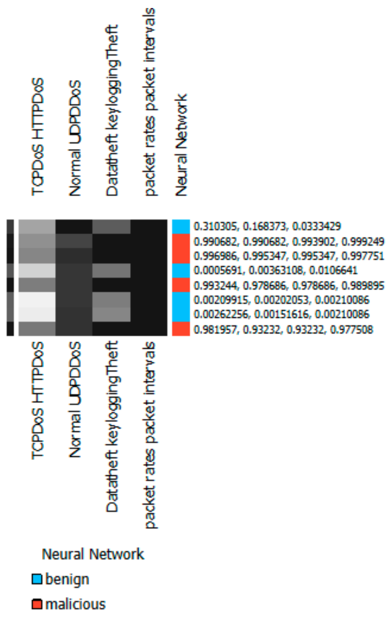
This suggests that even in the fourth dimension, the dataset retains significant variance, possibly due to complex, multi-faceted features that require multiple components to be fully represented. High explained variance in PCA indicates that our proposed model performs well because the principal components retain most of the essential information in the data. The PCA results show that the principal components of the model capture a significant amount of variance. This illustrates that the dimensionality reduction techniques used in our model are effective.

4.3. Information Gain

Figure 11 shows typical network features embedded in the image. Malware behaviors such as data theft, keylogging, and various types of DoS attacks manifest differently in their binary representations. Many existing models find it challenging to translate these behaviors into visual patterns that a model can effectively differentiate. Without accurate differentiation, it is difficult to highlight which image regions or corresponding feature maps are most indicative of these behaviors due to overlapping features. Different types of malware might share similar features, especially in their visual representations. For instance, both TCPDoS and HTTP DoS attacks may have overlapping patterns. Information gain helps in distinguishing these by focusing on subtle differences, but the overlapping nature of these features could also reduce the effectiveness of traditional information gain measures. The visualization of some malware samples as classified by our model is shown in Figure 12. Malware type is consistently highlighted in the heatmap; it indicates that this region contains features that are highly informative for our model to classify as either malicious or benign.

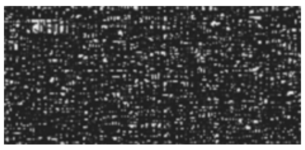


(**a**)

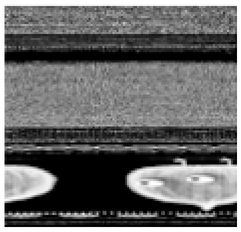


(**b**)

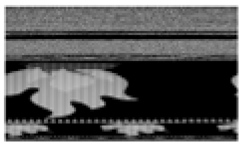
Figure 11. Heatmap showing regions of interest as classified by the model. (**a**) Image cluster showing malware names and their textual cluster classifications. (**b**) Image cluster showing malware activities and their score clusters.



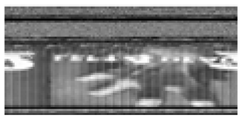
(**a**)



(**b**)



(**c**)



(**d**)

Figure 12. Visualized malware images based on their malware families. (**a**) QakBot, (**b**) Gamarue, (**c**) Sodinokibi, and (**d**) Ryuk.

To maintain the effectiveness of our model, we used confidence scores of the heatmap to determine the likelihood that a given image belongs to a particular malware category by highlighting areas of interest and corresponding to the parts of the image that the network considers important for distinguishing between different types of malware. For example, if a certain region of the image corresponds to a particular. The experimental results indicate that the proposed method achieved a higher accuracy of 99.62%, surpassing current state-of-the-art methods (see Table 1).

Table 1. Comparison between the proposed model and current state-of-the-art methods.

Year and Reference	Architecture	Accuracy (%)
2021 [44]	Spatial Attention CNN with VGG16	97.42
2021 [11]	CapsNet	96.58
2021 [45]	SSPNet1	96.86
2022 [46]	CCN	99.00
2022 [47]	CNN + VGG16	98.92
2022 [48]	CNN	99.60
2023 [49]	Transfer learning	99.60
2023 [50]	Deep Learning (DL) + (CNN)	97.00
2023 [51]	Visualization and Deep learning	99.44
2024 [52]	Generative Adversarial Network	99.50
Our model	CNN	99.62

The analyses of the proposed method relative to the existing approaches in this paper reveal not just the accuracy figures but also how different architectures behave under various data scenarios and against different malware types. Looking at the accuracy trend, from 2021 to 2024, there is a general improvement in accuracy, with most models achieving >97%. Our proposed model slightly outperforms the top accuracy of 99.60% [48,49] with a score of 99.62%. Models based on architectures like CapsNet [11] and SSPNet1 [45] demonstrate slightly lower performance (~96.5%), which could be indicative of how those architectures handle data complexities or malware variations. In terms of architecture, GANs [52] and Transfer Learning [49] may perform better in this scenario. GANs can generate synthetic data to augment the dataset, and transfer learning can leverage pre-trained models to mitigate the effects of a small dataset. CapsNet [11] may struggle with diverse data due to its sensitivity to complex relationships, whereas Visualization + Deep Learning [51] may excel since it can highlight distinguishing features through visualization. Our model achieves the highest accuracy (99.62%) among the methods in the table, which is attributed to optimized CNN layers and potentially better hyperparameters and training strategies. GANs [52] and Transfer Learning [49] may perform better in limited data scenarios. GANs can generate synthetic data to augment the dataset, and transfer learning can leverage pre-trained models to mitigate the effects of a small dataset.

The trend in accuracy improvements over time reflects the continuous advancements in machine learning models for classifying malware. Early models, such as the Spatial Attention CNN with VGG16 from 2021, achieved 97.42%, which was a significant result at the time. Over the years, newer models such as CNN (2022) and Transfer Learning (2023) show even higher accuracies of around 99.62%. This trend demonstrates how deep learning techniques have evolved with more sophisticated architectures, optimization techniques, and data handling. Many of the models compared in this paper have accuracy percentages very close to one another, especially in recent years (2022–2024). For example, CCN (99.00%), CNN + VGG16 (98.92%), and Generative Adversarial Network (99.50%) all hover around the upper 90s. The proximity in performance suggests that the models are converging towards a performance ceiling, where improvements are becoming more incremental rather than substantial. Our model's performance at 99.64% accuracy is certainly competitive and slightly better than previous models in the dataset.

5. Conclusions and Future Work

In this paper, we introduced an enhanced image-based malware classification model that integrates both structured and content-based features, normalized using advanced dimensionality reduction techniques such as principal component analysis (PCA) and edge detection algorithms. By converting malware binaries into grayscale images, the model leverages the unique patterns and structures inherent in these images to improve malware detection accuracy. Employing a convolutional neural network (CNN), the proposed method achieved an impressive classification accuracy of 99.62%, outperforming several state-of-the-art models. The robustness and reliability of the model were demonstrated through a comprehensive evaluation, including 10-fold cross-validation and comparisons with existing techniques.

The results show that the proposed model not only exceeds existing models in accuracy but also exhibits enhanced resilience to variations in data. The findings of this study hold significant implications for the field of malware detection and classification. The proposed model demonstrates the efficacy of leveraging image-based representations of malware binaries in conjunction with advanced machine learning techniques. By transforming malware binaries into grayscale images, the model captures distinctive structural patterns that are often overlooked by traditional binary-based approaches. The use of PCA and edge detection for dimensionality reduction and feature extraction played a critical role in enhancing the model's performance by reducing computational complexity while improving the discriminative power of the features provided to CNN. The robustness of the proposed model, validated through rigorous 10-fold cross-validation, underscores its reliability across diverse datasets and conditions. This highlights its suitability for real-world applications, particularly in environments where malware data variability and noise are prevalent.

Despite the promising results achieved in this study, some limitations are observed. The first limitation is that although the proposed system exhibits high accuracy and computational efficiency, the dataset used consists of only 16,000 samples. This relatively limited sample size may affect the generalizability of the model, particularly when applied to larger and more heterogeneous datasets. The last limitation of the comparison between ResNet-152 and the vision transformer (ViT) architectures is based on a specific set of hyperparameters, which may not be universally optimal, especially in IoT environment, thereby limiting the model's adaptability to varying conditions and datasets. Future research will focus on adapting the proposed model to address specific challenges in IoT malware, including developing lightweight versions that can operate efficiently under the resource constraints of IoT devices.

Author Contributions: M.A. conceived and designed the experiments and wrote the paper. N.O. worked on the figures, tables, and grammatical checks. S.H. and J.O. reviewed the draft of the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used are publicly available: MalNet [30].

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

References

1. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Venkatraman, S. Robust Intelligent Malware Detection Using Deep Learning. *IEEE Access* **2019**, *7*, 46717–46738. [CrossRef]
2. Campion, M.; Preda, M.D.; Giacobazzi, R. Learning metamorphic malware signatures from samples. *J. Comput. Virol. Hacking Tech.* **2021**, *17*, 167–183. [CrossRef]
3. Almomani, I.; Alkhayer, A.; El-Shafai, W. A Crypto-Steganography Approach for Hiding Ransomware within HEVC Streams in Android IoT Devices. *Sensors* **2022**, *22*, 2281. [CrossRef]
4. Aslan, O.; Samet, R. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
5. Majid, A.-A.M.; Alshaibi, A.J.; Kostyuchenko, E.; Shelupanov, A. A review of artificial intelligence based malware detection using deep learning. *Mater. Today Proc.* **2023**, *80*, 2678–2683. [CrossRef]
6. Anderson, H.S.; Kharkar, A.; Filar, B.; Roth, P. Evading Machine Learning Malware Detection. 2017. Available online: https://github.com/EndgameInc/gym-malware (accessed on 15 September 2024).
7. Bahador, M.B.; Abadi, M.; Tajoddin, A. HLMD: A signature-based approach to hardware-level behavioral malware detection and classification. *J. Supercomput.* **2019**, *75*, 5551–5582. [CrossRef]
8. Yunmar, R.A.; Kusumawardani, S.S.; Widyawan; Mohsen, F. Hybrid Android Malware Detection: A Review of Heuristic-based Approach. *IEEE Access* **2024**, *12*, 41255–41286. [CrossRef]
9. Arabo, A.; Dijoux, R.; Poulain, T.; Chevalier, G. Detecting Ransomware Using Process Behavior Analysis. *Procedia Comput. Sci.* **2020**, *168*, 289–296. [CrossRef]
10. Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A Review of Android Malware Detection Approaches Based on Machine Learning. *IEEE Access* **2020**, *8*, 124579–124607. [CrossRef]
11. Venkatraman, S.; Alazab, M.; Vinayakumar, R. A hybrid deep learning image-based analysis for effective malware detection. *J. Inf. Secur. Appl.* **2019**, *47*, 377–389. [CrossRef]
12. Zou, B.; Cao, C.; Tao, F.; Wang, L. IMCLNet: A lightweight deep neural network for Image-based Malware Classification. *J. Inf. Secur. Appl.* **2022**, *70*, 103313. [CrossRef]
13. Guo, H.; Li, Y.; Shang, J.; Gu, M.; Huang, Y.; Gong, B. Learning from class-imbalanced data: Review of methods and applications. *Expert Syst. Appl.* **2017**, *73*, 220–239. [CrossRef]
14. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.-G.; Chen, J. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
15. Vasan, D.; Alazab, M.; Wassan, S.; Naeem, H.; Safaei, B.; Zheng, Q. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Comput. Netw.* **2020**, *171*, 107138. [CrossRef]
16. Gibert, D.; Mateu, C.; Planes, J.; Vicens, R. Using convolutional neural networks for classification of malware represented as images. *J. Comput. Virol. Hacking Tech.* **2018**, *15*, 15–28. [CrossRef]
17. Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *J. Netw. Comput. Appl.* **2020**, *153*, 102526. [CrossRef]

18. Burnap, P.; French, R.; Turner, F.; Jones, K. Malware classification using self organising feature maps and machine activity data. *Comput. Secur.* **2018**, *73*, 399–410. [CrossRef]
19. Lan, T.; Hu, H.; Jiang, C.; Yang, G.; Zhao, Z. A comparative study of decision tree, random forest, and convolutional neural network for spread-F identification. *Adv. Space Res.* **2020**, *65*, 2052–2061. [CrossRef]
20. Yoo, S.; Kim, S.; Kim, S.; Kang, B.B. AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification. *Inf. Sci.* **2021**, *546*, 420–435. [CrossRef]
21. Kharoubi, R.; Oualkacha, K.; Mkhadri, A. The cluster correlation-network support vector machine for high-dimensional binary classification. *J. Stat. Comput. Simul.* **2019**, *89*, 1020–1043. [CrossRef]
22. Ito, E.; Sato, T.; Sano, D.; Utagawa, E.; Kato, T. Virus Particle Detection by Convolutional Neural Network in Transmission Electron Microscopy Images. *Food Environ. Virol.* **2018**, *10*, 201–208. [CrossRef]
23. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
24. Rhode, M.; Burnap, P.; Jones, K. Early-stage malware prediction using recurrent neural networks. *Comput. Secur.* **2018**, *77*, 578–594. [CrossRef]
25. Ali, M.; Haque, M.-U.; Durad, M.H.; Usman, A.; Mohsin, S.M.; Mujlid, H.; Maple, C. Effective network intrusion detection using stacking-based ensemble approach. *Int. J. Inf. Secur.* **2023**, *22*, 1781–1798. [CrossRef]
26. Chen, S.; Wang, C.; Chen, Z.; Wu, Y.; Liu, S.; Chen, Z.; Li, J.; Kanda, N.; Yoshioka, T.; Xiao, X.; et al. WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing. *IEEE J. Sel. Top. Signal Process.* **2022**, *16*, 1505–1518. [CrossRef]
27. Yuxin, D.; Siyi, Z. Malware detection based on deep learning algorithm. *Neural Comput. Appl.* **2019**, *31*, 461–472. [CrossRef]
28. Wagner, M.; Rind, A.; Thür, N.; Aigner, W. A knowledge-assisted visual malware analysis system: Design, validation, and reflection of KAMAS. *Comput. Secur.* **2017**, *67*, 1–15. [CrossRef]
29. Staheli, D. Visualization for cyber security. In Proceedings of the 2018 IEEE Symposium on Visualization for Cyber Security (VizSec), Institute of Electrical and Electronics Engineers (IEEE), Berlin, Germany, 22 October 2018.
30. Freitas, S.; Duggal, R.; Chau, D.H. MalNet: A Large-Scale Image Database of Malicious Software. Available online: https://www.mal-net.org/ (accessed on 11 September 2024).
31. Ahmed, Y.A.; Koçer, B.; Huda, S.; Al-Rimy, B.A.S.; Hassan, M.M. A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *J. Netw. Comput. Appl.* **2020**, *167*, 102753. [CrossRef]
32. Hasanah, S.A.; Pravitasari, A.A.; Abdullah, A.S.; Yulita, I.N.; Asnawi, M.H. A Deep Learning Review of ResNet Architecture for Lung Disease Identification in CXR Image. *Appl. Sci.* **2023**, *13*, 13111. [CrossRef]
33. Sanghvi, H.A.; Patel, R.H.; Agarwal, A.; Gupta, S.; Sawhney, V.; Pandya, A.S. A deep learning approach for classification of COVID and pneumonia using DenseNet-201. *Int. J. Imaging Syst. Technol.* **2023**, *33*, 18–38. [CrossRef]
34. Sunnetci, K.M.; Kaba, E.; Çeliker, F.B.; Alkan, A. Comparative parotid gland segmentation by using ResNet-18 and MobileNetV2 based DeepLab v3+ architectures from magnetic resonance images. *Concurr. Comput.* **2023**, *35*, e7405. [CrossRef]
35. Gwon, G.-H.; Lee, J.H.; Kim, I.-H.; Jung, H.-J. CNN-Based Image Quality Classification Considering Quality Degradation in Bridge Inspection Using an Unmanned Aerial Vehicle. *IEEE Access* **2023**, *11*, 22096–22113. [CrossRef]
36. Zhang, X.; Wu, K.; Chen, Z.; Zhang, C. MalCaps: A Capsule Network Based Model for the Malware Classification. *Processes* **2021**, *9*, 929. [CrossRef]
37. Christodorescu, M.; Jha, S. Static Analysis of Executables to Detect Malicious Patterns. In Proceedings of the 12th USENIX Security Symposium, Washington, DC, USA, 4–8 August 2003.
38. Ramesh, G.; Menen, A. Automated dynamic approach for detecting ransomware using finite-state machine. *Decis. Support Syst.* **2020**, *138*, 113400. [CrossRef]
39. Yücel, Ç.; Koltuksuz, A. Imaging and evaluating the memory access for malware. *Forensic Sci. Int. Digit. Investig.* **2020**, *32*, 200903. [CrossRef]
40. Yuan, B.; Wang, J.; Wu, P.; Qing, X. IoT Malware Classification Based on Lightweight Convolutional Neural Networks. *IEEE Internet Things J.* **2022**, *9*, 3770–3783. [CrossRef]
41. Wickramasinghe, C.S.; Amarasinghe, K.; Manic, M. Deep Self-Organizing Maps for Unsupervised Image Classification. *IEEE Trans. Ind. Inform.* **2019**, *15*, 5837–5845. [CrossRef]
42. Zhao, B.; Li, W.; Xia, L.; Li, S.; Yang, Z.; Huang, Y.; Zhou, M. A CNN-based FBG demodulation method adopting the GAF-assisted ascending dimension of complicated signal. *Opt. Commun.* **2021**, *499*, 127296. [CrossRef]
43. Yang, Z.; He, B.; Liu, Y.; Wang, D.; Zhu, G. Classification of rock fragments produced by tunnel boring machine using convolutional neural networks. *Autom. Constr.* **2021**, *125*, 103612. [CrossRef]
44. 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]
45. Ding, E.; Cheng, Y.; Xiao, C.; Liu, Z.; Yu, W. Efficient Attention Mechanism for Dynamic Convolution in Lightweight Neural Network. *Appl. Sci.* **2021**, *11*, 3111. [CrossRef]
46. Chaganti, R.; Ravi, V.; Pham, T.D. Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification. *J. Inf. Secur. Appl.* **2022**, *69*, 103306. [CrossRef]
47. Kumar, S.; Janet, B. DTMIC: Deep transfer learning for malware image classification. *J. Inf. Secur. Appl.* **2022**, *64*, 103063. [CrossRef]

48. Tekerek, A.; Yapici, M.M. A novel malware classification and augmentation model based on convolutional neural network. *Comput. Secur.* **2022**, *112*, 102515. [CrossRef]
49. Ahmed, M.; Afreen, N.; Ahmed, M.; Sameer, M.; Ahamed, J. An inception V3 approach for malware classification using machine learning and transfer learning. *Int. J. Intell. Netw.* **2023**, *4*, 11–18. [CrossRef]
50. Chaganti, R.; Ravi, V.; Pham, T.D. A multi-view feature fusion approach for effective malware classification using Deep Learning. *J. Inf. Secur. Appl.* **2023**, *72*, 103402. [CrossRef]
51. Deng, H.; Guo, C.; Shen, G.; Cui, Y.; Ping, Y. MCTVD: A malware classification method based on three-channel visualization and deep learning. *Comput. Secur.* **2023**, *126*, 103084. [CrossRef]
52. Sharma, O.; Sharma, A.; Kalia, A. MIGAN: GAN for facilitating malware image synthesis with improved malware classification on novel dataset. *Expert Syst. Appl.* **2024**, *241*, 122678. [CrossRef]