Sistemasi: Jurnal Sistem Informasi
Volume 15, Nomor 10, 2026: 4194-4207
ISSN:2302-8149
e-ISSN:2540-9719

Image-based Malware and Benign Software Classification

[1]Maher F. Ismael*
[1]Department of Cybersecurity, College of Computing and Informatics, University of Al-Hamdaniya, Mosul, Iraq
*e-mail: maher.f@uohamdaniya.edu.iq

(***received*:** 27 September 2026, ***revised*:** 11 October 2026, ***accepted*:** 18 October 2026)

Abstract

The growing volume and diversity of malware challenge the scalability of manual analysis and the effectiveness of signature-based detection. An alternative approach is image-based learning, which has been the subject of numerous studies, but tend to focus on the classification of malware families instead of malicious-benign discrimination, and are based on large models where little attention is paid to the degree of data leakage and how it impacts on performance variability. This paper presents a simple convolutional approach to binary classification of executable-file images, without disassembly or dynamic execution – I-Base+. With only 18,881 trainable parameters, it is a simple framework which processes resized grayscale images with two convolutional blocks, global average pooling, and a binary output layer. Evaluation is based on 3,000 Microsoft-signed benign images and 8,444 pre-existing MalImg images from 25 malware families from which 895 images were removed that were exact duplicates. The evaluation protocol consists of training/validation and test partitions that are stratified, five random seeds, controlled ablations and a family-disjoint evaluation in which selected families of malware are not included in the training set. While providing a mean accuracy of 98.72% and a recall rate of 99.51% with a false-positive rate of 3.51%, the number of parameters required for I-Base+ is significantly fewer than the evaluated transfer-learning baselines, and CPU inference speed is significantly faster than the evaluated transfer-learning baselines. Withheld families' recall is 81.64%, indicating that generalization to unseen malware is still a major challenge. The results show a practical accuracy–efficiency trade-off for image-based malware classification, and indicate that it is necessary to consider beyond random sample splits.

Keywords: *Malware Detection, Binary Classification, Malware Visualization, Convolutional Neural Network, Global Average Pooling*

1 Introduction

Malware continues to mutate and repackage at an ever increasing pace, resulting in significant economic, operational and privacy losses around the globe, and the ability to reliably identify malicious software from harmless software continues to be one of the more important challenges in information security. Traditional methods like manual analysis, signature-based scanners and heuristics are not scalable as the number of unique malware samples keeps increasing by millions each year. They are also expensive in processing time and can be avoided by using obfuscation methods like packing [1], encryption and polymorphism. There is therefore a need for automated robust and intelligent techniques that are able to catch up with emerging variants [2]. One such technique is image based malware analysis, a combination of cyber security and computer vision [3]. The primary idea behind it is that the content of an executable program can be encoded into a two dimensional grayscale image [4] and that the structure and function of the code is visually represented as different textures [5]. The majority of benign software are organized and repetitive while malware categories (such as worms and Trojans) have characteristic visual signatures [6]. The patterns can be classified with high accuracy without the overhead of disassembling or dynamically executing them using Convolutional Neural Networks (CNNs) [7]. can classify these patterns with high accuracy without the overhead of disassembly or dynamic execution [8], [9].

Problem statement and research gap. The idea of converting binaries to gray-level images is not new [1] nor is the idea of the CNNs for the classification of these images [7, 8, 10] new. The gap that this

 ISSN:2302-8149
e-ISSN:2540-9719

addresses is elsewhere. First, most image-based studies focus on multi-class family attribution on MalImg: each test sample is already labelled as malicious [1], [7], [10]; it is a question that is much less studied whether an unknown executable file is malicious or benign, for which the false-positive rate is much more relevant as it is a determinant of user trust in the operationally more fundamental context [11]. Second, high accuracies can be achieved with very large models (e.g., VGG16 with over 138 million parameters [10]) and in several of the studies included here (Table 10) with no reported variability, ablation of the design choices or control of leakage between closely related variants of malware. These are known challenges of using machine learning in security [12]. This paper thus examines if a very small CNN with a parameter-free GAP layer can reliably distinguish malware and benign PE files, if evaluated using a protocol that is designed to expose such pitfalls.

For this purpose, the I-Base+ framework is presented. It transforms every executable into a normalised grayscale image, and then classifies it using a lightweight GAP-based DCNN, as shown in Fig. 2. It does not need to be disassembled, or executed, unlike the traditional static or dynamic analysis approaches.

This paper mainly presents the following contributions:

i. **A lightweight GAP-based DCNN for binary malware detection** :a fully specified two-block architecture with 18,881 trainable parameters (Table 3), which is approximately 445 times smaller than a conventional fully connected head, and is about 780 and 1,250 times smaller than the VGG16 and ResNet50 models used as baseline models.

ii. **A Binary Evaluation Protocol that is leakage aware:** removal of 895 exact duplicate samples in the independent stratified test set, 5 repeated runs reported with mean ± std. dev. with 95 % c. i., confusion matrices, ROC-AUC and PR-AUC, and a family-aware split in which the families used for testing are not seen during training.

iii. **Acontrolled evaluation of the design choices:** an ablation study of the classification head (GAP vs Flatten + fully connected layers) and network depth and input resolution, where GAP achieves about 1 point of accuracy for a factor of 445 fewer parameters compared to the other heads, along with texture-based (GIST-like + k-NN) and transfer-learning (VGG16, ResNet50) baselines, which are trained and tested on the same splits.

2 Literature Review

Malware detection has moved from static signature-based methods to behaviour-based and, more recently, to learning-based techniques. This section reviews the main trends in malware-image analysis from 2011 to 2025, during which research shifted from handcrafted features to deep learning.

Feature engineering and visualisation (2011–2017): Nataraj et al. [1] used GIST texture descriptors with a k-nearest-neighbour (k-NN) classifier to show that malware families have distinct texture properties, without disassembling the binaries. Han et al. [2] later used entropy graphs to improve the efficiency of texture extraction. However, such handcrafted descriptors remain sensitive to noise and obfuscation, which motivated learned feature extraction.

Deep learning and transfer learning (2018–2021): Cui et al. [7] and Kalash et al. [8] used CNNs to exploit the spatial structure of malware images automatically, eliminating manual feature design. To cope with the limited number of labelled training samples in cybersecurity, Vasan et al. [10] and Rezende et al. [13] applied transfer learning, reusing VGG16 and ResNet50 models pre-trained on ImageNet; later studies fine-tuned several pre-trained networks on MalImg [14] and added spatial attention to a VGG19 backbone [15]. Bhodia et al. [11] applied a ResNet34 transfer-learning model to a binary malware-versus-benign task that combined the 9,339 MalImg samples with 3,304 benign Windows files and reported an accuracy of 98.39 %, which makes it the closest prior work to the present study. These models perform very well, but they usually require many parameters, which increases memory and storage requirements and can limit deployment on resource-constrained devices.

 ISSN:2302-8149
e-ISSN:2540-9719

New architectures and current challenges (2022–2025): Recent work has explored more efficient backbones such as EfficientNet [3], lightweight networks such as IMCLNet [9], low-dimensional input representations [4], transformer-based CNNs [16] semantic image representations [17] and generative data augmentation that balances under-represented families [18]. Nevertheless, most of these methods target multi-class classification of malware families and rarely address the more fundamental binary separation of benign and malicious applications [9], [16]; exceptions include a channel-boosted CNN that separates benign from malicious IoT executables in image form [19] and a pipeline that combines deep features of malware images with classical classifiers for binary detection [20]. This gap matters because a protection system must first separate benign from malicious software with high accuracy and a low false-positive rate.

Other directions include hybrid pipelines that combine several image-based descriptors [6], DenseNet architectures whose dense connectivity encourages feature reuse [21] and semi-supervised learning for Android malware categories [22], as well as methods that improve the robustness of deep learning-based detectors against adversarial attacks [23]. Like most of the works above, they target multi-class categorisation rather than binary benign-versus-malware detection.

Unlike previous works, which mostly address multi-class family categorisation, often with large models, this study investigates a simple, lightweight CNN tailored to binary classification (benign vs. malware) and evaluates it under a leakage-aware protocol. Table 1 summarises representative previous studies and the limitations that motivate this choice.

Table 1 Comparative Analysis of Related Work and Corresponding Research Gaps

Reference	Year	Methodology	Focus	Critical Limitation / Research Gap
Nataraj et al. **[1]**	2011	GIST + k-NN	Texture Analysis	Reliance on handcrafted features; sensitive to noise.
Cui et al. **[7]**	2018	Custom CNN	Deep Learning	Requires extensive training data; prone to overfitting.
Bhodia et al. [11]	2019	Transfer Learning (ResNet34)	Benign vs. Malware	Large pre-trained model; no variability reported; zero-day setting simulated, not a family-wise split.
Vasan et al. **[10]**	2020	Transfer Learning (VGG16)	Accuracy Optimization	Large model (> 138 M parameters).
Mahdavifar et al. **[22]**	2020	Semi-Supervised DL	Android Malware Categories	Relies on unlabeled-data assumptions; not binary-focused.
Proposed Work	**Current Study**	**Lightweight GAP-based DCNN (18,881 parameters)**	**Benign vs. Malware**	**Independent test set, repeated runs, ablation and family-aware evaluation.**

3 Research Method

The proposed framework consists of four steps: (1) data collection and preprocessing, (2) image representation of the binary files, (3) classification using lightweight DCNN and (4) evaluation based on an experimental protocol that allows leakage.

 ISSN:2302-8149
e-ISSN:2540-9719

3.1 Dataset Description

The malware class comes from the well-known public MalImg image dataset [1], which contains 9,339 grayscale images belonging to 25 families. A byte-level (MD5) comparison revealed that 895 of these images are complete copies of another image of the same family (Yuner.A 667, Fakerean 182, Adialer.C 29, Autorun.K 15 and Lolyda.AT 2). Since identical copies falling into both the training and the test sets would inflate the measured performance, they were removed, which resulted in 8,444 unique malware images. The distributed copy stores the Rbot!gen family under two folder names, which were merged. The benign class consists of 3,000 Windows PE files (403 .exe and 2,597 .dll; mean 402 KB, min 4 KB, max 9.9 MB) extracted from the System32 and SysWOW64 directories of a Windows 10 Pro installation (version 22H2, build 19045, 64-bit). To ensure that only genuine operating-system files were included, only files with a valid Microsoft Authenticode signature, a valid MZ header and a size between 4 KB and 10 MB were retained; they were de-duplicated by SHA-256 hash and sampled at random with a fixed seed (42). The list of their SHA-256 hashes is provided as supplementary material.

MalImg is released as grayscale images created with the width-allocation scheme of [1], and the benign PE files were converted using the same scheme (Table 2). The final dataset consists of 11,444 samples (class ratio ≈ 2.8 : 1, malware : benign); because of this class imbalance, the following performance metrics are reported in addition to accuracy: precision, false-positive rate, Matthews correlation coefficient (MCC) and PR-AUC.
The data were divided by stratified random sampling into a training set (70 %, 8,010 samples: 5,910 malware / 2,100 benign), a validation set (15 %, 1,717 samples: 1,267 / 450) and an independent test set (15 %, 1,717 samples: 1,267 / 450). The validation set is used only for early stopping and hyper-parameter tuning, and the test set was never used for training, model selection or threshold tuning. Figure 1 shows a schematic representation of the binary-to-image transformation for a benign and a malicious executable.

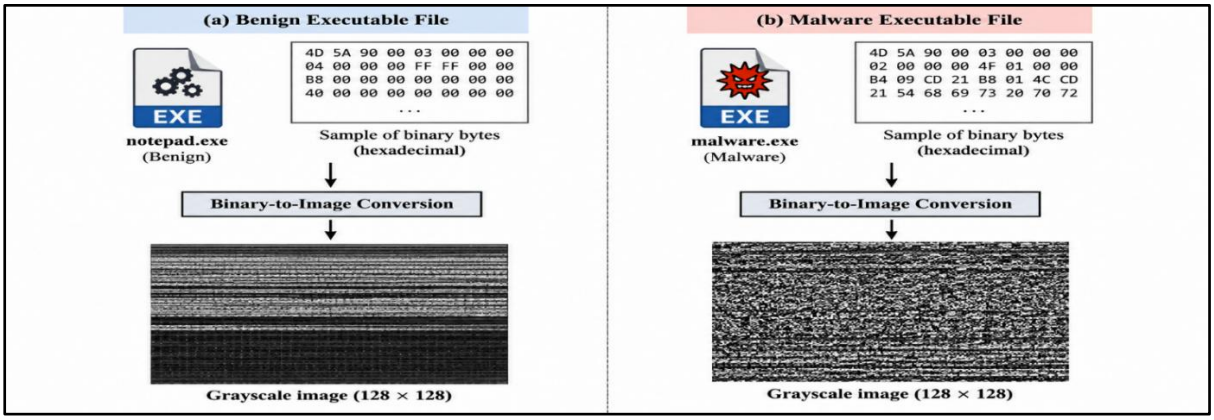


Figure 1 Schematic illustration of the binary-to-image conversion for (a) a benign executable and (b) a malicious executable.

3.2 Binary-to-Image Transformation

Each executable is read as a stream of 8-bit unsigned integers, and every byte is mapped to one grayscale pixel intensity (0 = black, 255 = white) [1], [17]. To preserve the spatial continuity of code and data sections (e.g., .text and .data), the image width is fixed according to the file size using the scheme of Nataraj et al. [1] shown in Table 2, and the height is computed as ⌊file size / width⌋ [1]. In this way, files of any length are transformed into two-dimensional textures in which consecutive bytes remain adjacent.

 ISSN:2302-8149
e-ISSN:2540-9719

Table 2 Image Width Allocation Based on File Size (adapted from Nataraj et al. [1])

File Size Range	Image Width (Fixed)
Less than 10 KB	32 pixels
10 KB – 30 KB	64 pixels
30 KB – 60 KB	128 pixels
60 KB – 100 KB	256 pixels
100 KB – 200 KB	384 pixels
200 KB – 500 KB	512 pixels
500 KB – 1,000 KB	768 pixels
Greater than 1,000 KB	1,024 pixels

As shown in Table 2, the image width grows with the file size, from 32 pixels for files smaller than 10 KB to 1,024 pixels for files larger than 1,000 KB; this yields a two-dimensional representation while preserving the sequential order of the bytes. Because the CNN requires a fixed input size, every image is then resized to 128 × 128 pixels with bilinear interpolation and normalised to [0, 1].

Downscaling and information loss. Resizing to 128 × 128 pixels discards information. A 200 KB executable corresponds to about 2×10^5 pixels, whereas the network input contains 16,384 pixels, so for large files each input pixel summarises on average more than ten bytes, and fine-grained byte-level details (e.g., individual instructions or strings) are lost; the aspect ratio of tall images is also distorted. What is preserved is the global texture and the layout of the sections, for example low-entropy padding, high-entropy packed or encrypted regions and the relative position of code and data, which are the cues the classifier is expected to rely on. The effect of this choice is quantified in the ablation study, where input resolutions of 64 × 64, 128 × 128 and 224 × 224 pixels are compared (Table 7).

Fig. 2 shows the whole transformation and classification process. Each executable file is processed as a sequence of 8-bit byte values, which are mapped to grayscale pixel intensities and arranged in a two-dimensional image. The resulting image is resized to 128 × 128 pixels and normalised to [0, 1] before being input to the proposed DCNN, which classifies it as benign or malware.

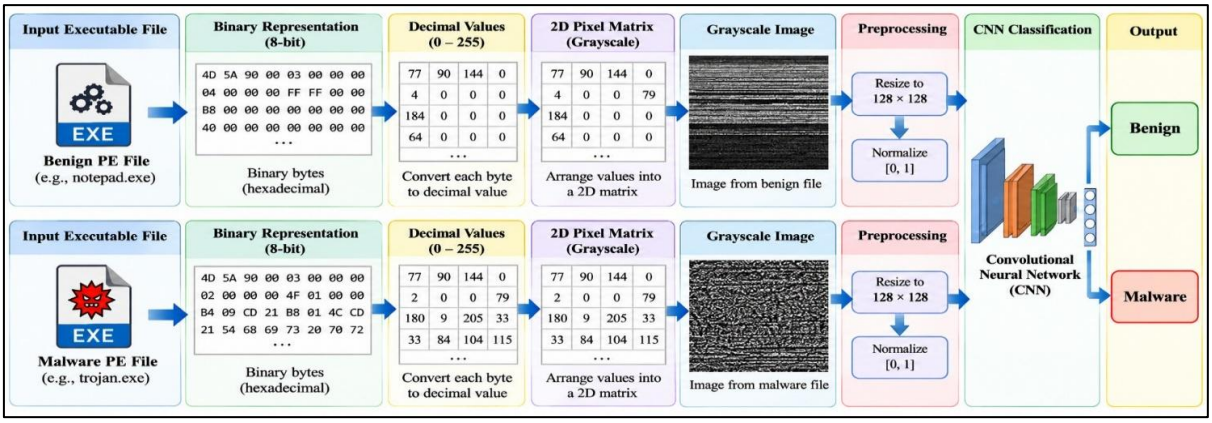


Figure 2 Block diagram of the proposed binary-to-image transformation and DCNN-based malware classification framework.

3.3 DCNN Model Architecture

To classify the generated images, a compact DCNN tailored to the binary task was designed (Fig. 3, Table 3). It learns hierarchical features directly from the input images without handcrafted feature engineering and consists of two convolutional blocks, each of which contains:

- **Convolutional layer:** 3 × 3 kernels, stride 1 and 'same' padding, with 32 filters in the first block and 64 in the second; it detects local features such as edges and texture patterns.
- **Rectified Linear Unit (ReLU):** introduces non-linearity into the model.

 ISSN:2302-8149
e-ISSN:2540-9719

- **Max-pooling layer:** 2 × 2 window with stride 2; it halves the spatial dimensions and reduces computation while retaining the most salient responses.

A One of the main architectural elements in this design is the Global Average Pooling (GAP) layer [24] immediately preceding the output layer. Rather than flattening the feature maps into a fully connected layer, GAP creates a summary of each of the 64 feature maps using the average value across their spatial dimensions, eliminating the weight matrix of a fully connected layer and greatly decreasing the number of trainable parameters (from 8,407,681 to 18,881 in this case). The 64-dimensional GAP output then goes to a dense unit with sigmoid activation, which gives the probability that the sample is malicious; the sample is classified as malware if this probability is at least 0.5. Reducing the number of parameters was expected to increase the accuracy of the model by reducing overfitting, but based on the results of the ablation experiment (Table 7), a fully connected head is more accurate on this dataset, so GAP is kept as a way to increase efficiency, not accuracy.

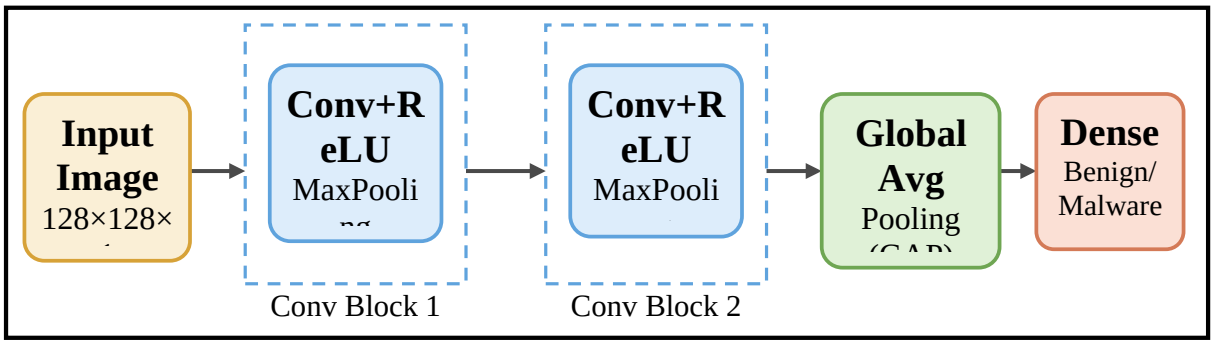


Figure 3 Proposed DCNN architecture: two convolutional blocks (32 and 64 filters), Global Average Pooling and a sigmoid output layer.

Table 3 Layer-by-Layer Configuration of the Proposed DCNN

Layer	Output shape	Kernel / stride	Filters	Activation	Parameters
Input	128 × 128 × 1	–	–	–	0
Conv2D-1 (same padding)	128 × 128 × 32	3 × 3 / 1	32	ReLU	320
MaxPooling-1	64 × 64 × 32	2 × 2 / 2	–	–	0
Conv2D-2 (same padding)	64 × 64 × 64	3 × 3 / 1	64	ReLU	18,496
MaxPooling-2	32 × 32 × 64	2 × 2 / 2	–	–	0
Global Average Pooling	64	–	–	–	0
Dense (output)	1	–	1 unit	Sigmoid	65
Total trainable parameters					**18,881**

3.4 Training Configuration

The model was trained in a supervised fashion using the Adam optimiser ($\beta_1 = 0.9$, $\beta_2 = 0.999$) in order to minimise the binary cross-entropy between the predicted probability and the actual label (benign = 0, malware = 1). The learning rate (10^{-3}, 3×10^{-4} or 10^{-4}) and the use of balanced class weights were chosen based on the validation MCC on the first split only, while the test set was excluded; the chosen configuration (learning rate 10^{-3}, no class weighting; validation MCC = 0.923, versus 0.800 for 10^{-4}) was fixed for all the experiments. The hyper-parameter tuning was carried out within 50 epochs; since all configurations were still improving at that time, the number of epochs was increased to 100. Training was done with mini-batches of 32 samples; early stopping monitored the validation loss with a patience of 10 epochs and restored the weights of the best epoch. The weights were initialised using the Glorot-uniform method, data augmentation was not performed and the decision threshold was set to 0.5. In the main experiment, the validation loss never stagnated for 10 consecutive epochs, so all runs trained for 100 epochs and the weights of the epoch with the minimum validation loss were restored. The experiments were implemented using TensorFlow 2.21.0 (Keras

 ISSN:2302-8149
e-ISSN:2540-9719

API) on an NVIDIA Tesla T4 GPU (Google Colaboratory). The experiment code and the exact data splits are available from the corresponding author.

3.5 Experimental Protocol and Evaluation Metrics

Repeated runs. The main experiment, the family-aware evaluation and the multi-class experiment were repeated five times (five random seeds or five folds); the ablation study and the baseline comparison were repeated with three seeds. Each seed changes both the stratified split and the weight initialisation. Results are reported as mean ± standard deviation (SD), together with the 95 % confidence interval $\bar{x} \pm t_{0.975,n-1} \cdot s / \sqrt{n}$ ($t_{0.975,4} = 2.776$ for $n = 5$).

Metrics: Malware is treated as the positive class. From the confusion matrix (true positives TP, true negatives TN, false positives FP and false negatives FN), the accuracy, precision, recall, F1-score, false-positive rate (FPR) and Matthews correlation coefficient (MCC) are computed with Equations (1)–(6).

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (3)$$

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}} \quad (4)$$

$$FPR = \frac{FP}{FP+TN} \quad (5)$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (6)$$

Threshold-independent performance is measured by the area under the ROC curve (ROC-AUC) and by the area under the precision–recall curve (PR-AUC, computed as average precision), which is more informative than ROC-AUC under class imbalance [25].

Family-aware evaluation. MalImg contains closely related variants (e.g., Allaple.A and Allaple.L, or Lolyda.AA1–AA3 and Lolyda.AT), and images of the same variant are often near-duplicates. With a random split, near-identical samples can fall into both the training and the test sets, which inflates the measured performance [12]. A family-aware five-fold group-wise cross-validation was therefore also performed: the 25 MalImg classes were merged into 19 parent families (e.g., all Allaple.* samples form one group), and every family was assigned to exactly one test fold, so that the malware families evaluated in a fold are never seen during training. Benign files, which carry no family label, were split by ordinary five-fold partitioning. The validation subset used for early stopping was drawn at random (stratified, 15 %) from the training portion of each fold, so that model selection never sees the held-out families.

Ablation study and controlled baselines. Using the same splits and seeds as the main experiment, the following ablation variants were trained: (A1) GAP replaced by Flatten + fully connected layer (128 units, ReLU) + dropout (0.5); (A2) a single convolutional block; (A3) three convolutional blocks (32–64–128 filters); and (A4) input resolutions of 64 × 64 and 224 × 224 pixels. The following baselines were trained and tested on the same splits: (B1) a GIST-like Gabor texture descriptor with a k-NN classifier (k = 3) following [1]; and (B2, B3) VGG16 and ResNet50 pre-trained on ImageNet, used as frozen feature extractors with a GAP + sigmoid head (a lighter setting than the fine-tuning of [10]); the grayscale input was replicated to three channels and preprocessed with the normalisation of each network. Because the backbones are frozen, their GAP features were computed once, and the sigmoid head was trained on them with the same optimiser, learning rate, batch size, epoch budget and early-stopping rule as I-Base+. The inference latency (batch size 1, from input image to output probability) of I-Base+, VGG16 and ResNet50 was measured on the same CPU (Google Colab CPU runtime). The paired difference between I-Base+ and variant A1 was tested with the exact McNemar

 ISSN:2302-8149
e-ISSN:2540-9719

test. Finally, to allow a task-matched comparison with the literature, the same DCNN with a 25-unit softmax output was trained for family classification on the MalImg samples only, using the same 70/15/15 protocol.

4 Results and Analysis

Table 4 reports the performance of I-Base+ on the independent test set (1,717 samples: 1,267 malware and 450 benign), averaged over five runs, and Table 5 gives the corresponding confusion matrices. Fig. 4 shows the ROC and precision–recall curves of a representative run.

Table 4 Performance of I-Base+ on the Independent Test Set (mean ± SD over five runs)

Metric	Mean ± SD	95 % confidence interval
Accuracy (%)	98.72 ± 0.36	98.28 – 99.16
Precision (%)	98.77 ± 0.56	98.07 – 99.47
Recall / detection rate (%)	99.51 ± 0.26	99.19 – 99.83
F1-score (%)	99.14 ± 0.24	98.84 – 99.43
False-positive rate (%)	3.51 ± 1.62	1.50 – 5.52
MCC (%)	97 ± 1	96 – 98
ROC-AUC (%)	99.52 ± 0.18	99.30 – 99.75
PR-AUC (%)	99.59 ± 0.24	99.29 – 99.89

Table 5 Confusion Matrices on the Independent Test Set

	Predicted benign	Predicted malware	Total
Representative run (seed 0)			
True benign	436	14	450
True malware	2	1,265	1,267
Summed over five runs			
True benign	2,171	79	2,250
True malware	31	6,304	6,335

Totals in the lower part of Table 5 are summed over the five runs; each run evaluates a different random test set of 1,267 malware and 450 benign samples, so a sample may appear in the test set of more than one run.

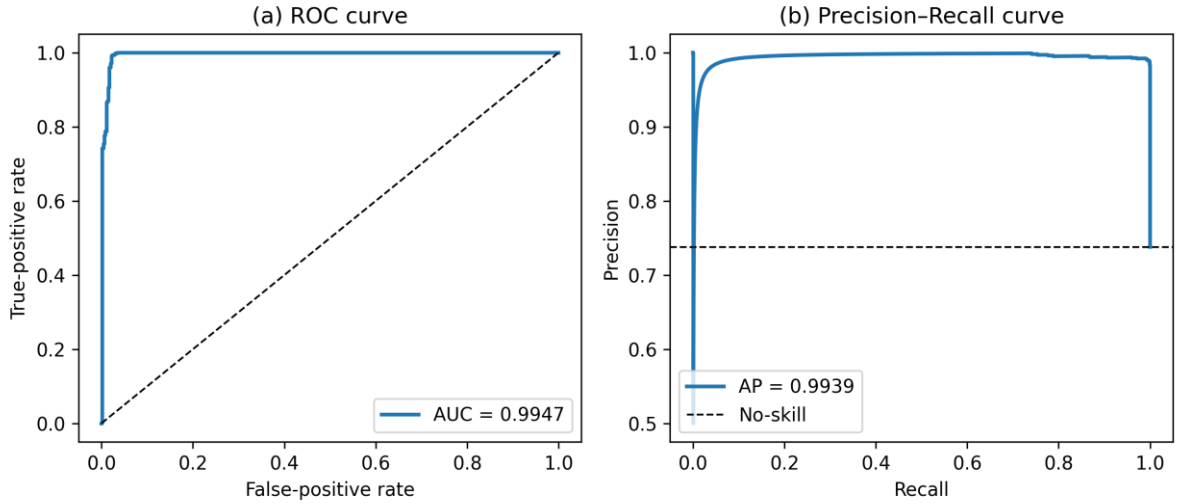


Figure 4 (a) ROC curve and (b) precision–recall curve of I-Base+ on the independent test set (seed 0; ROC-AUC = 99.47 %, average precision = 99.39 %).

Over the five runs, 6,304 of the 6,335 malware test samples were detected (recall 99.51 %), while 79 of the 2,250 benign test files were wrongly flagged as malware (false-positive rate 3.51 %). The model therefore errs on the side of flagging: missed malware is very rare, and most errors are false

 ISSN:2302-8149
e-ISSN:2540-9719

alarms on legitimate system files. Because the decision threshold was fixed at 0.5, these figures correspond to a single operating point; the ROC curve (Fig. 4a) suggests that a stricter threshold, chosen on validation data, could reduce the false-positive rate at some cost in recall. This matters in practice, because frequent false alarms on legitimate software erode user trust in an anti-virus system. Accuracy, recall and F1 vary little across the five runs (Table 4), whereas the false-positive rate varies more (95 % CI 1.50–5.52 %) because, with 450 benign test files, each false alarm changes it by 0.22 percentage points. The short dip of the precision–recall curve at very low recall (Fig. 4b) arises because a few benign files receive output scores as high as the most confident malware samples.

4.1 Family-Aware Evaluation

Table 6 compares the random stratified split with the family-aware split, in which the malware families of each test fold are never seen during training. Two effects are visible. On the benign side, the false-positive rate (4.20 ± 3.10 %) and the precision (97.41 ± 1.86 %) remain close to the random-split values, as expected, since benign files are partitioned at random in both settings. On the malware side, recall drops from 99.51 % to 81.64 ± 22.39 %, and accuracy from 98.72 % to 87.19 ± 13.44 % (a difference of 11.53 percentage points). The drop depends strongly on which families are held out: accuracy was 99.65 % when the Allaple variants were unseen, 95.86 % for Lolyda, Obfuscator, Skintrim and Wintrim, 88.97 % for Adialer, Agent, Alueron, Instantaccess and Rbot, 86.35 % for C2LOP, Dialplatform, Swizzor and Yuner, and only 65.10 % (recall 44.88 %) for Autorun, Dontovo, Fakerean, Malex and VB. In other words, the random-split figures describe how well the model recognises new samples of known families; they overestimate its ability to detect families whose byte-level texture differs from everything seen in training. Because the folds differ strongly in the number of malware samples (the Allaple fold alone holds more than half of them), per-fold accuracy also reflects the class ratio of each fold and should be compared with caution. The ROC-AUC of 95.72% shows that much of the ranking information is preserved, so a lower threshold might recover part of the lost recall, at the cost of a higher false-positive rate.

Table 6 Random Split versus Family-Aware (Unseen-Family) Evaluation (mean ± SD)

Evaluation setting	Accuracy (%)	Recall (%)	FPR (%)	F1 (%)	ROC-AUC(%)
Random stratified split (5 seeds)	98.72 ± 0.36	99.51 ± 0.26	3.51 ± 1.62	99.14 ± 0.24	99.52 ± 0.18
Family-aware split (5 folds)	87.19 ± 13.44	81.64 ± 22.39	4.20 ± 3.10	87.26 ± 15.14	95.72 ± 6.16

4.2 Ablation Study

Table 7 isolates the contribution of each design choice. The ablation does not support the expectation that GAP improves accuracy. Replacing GAP with a fully connected head (A1) raised accuracy from 98.62 ± 0.39 % to 99.57 ± 0.09 % and lowered the false-positive rate from 3.85 % to 0.96 %; the difference is significant in each of the three runs (exact McNemar test, p = 0.0009, 0.0001 and 0.0074). The advantage of GAP lies in model size: the fully connected head needs 8,407,681 parameters, about 445 times more than I-Base+. The training behaviour also differs: A1 met the early-stopping criterion after 27 epochs on average, whereas the GAP model was still improving at the 100-epoch limit. The small GAP network therefore shows no sign of overfitting, although part of its gap to A1 may reflect the fixed training budget rather than capacity alone. The depth results are consistent with a capacity limitation: with a single convolutional block (A2) accuracy fell to 92.12 % and the false-positive rate rose to 14.07 %, whereas a third block (A3) reached 99.44 ± 0.22 % with a false-positive rate of 1.70 %, close to A1 but with 92,801 parameters, i.e. about 90 times fewer. Input resolution mattered little: 64 × 64 inputs gave accuracy within one standard deviation of 128 × 128 (98.76 % vs. 98.62 %), and 224 × 224 gave no benefit (97.98 %), so the global texture needed for the binary decision survives strong downscaling.

Inference time on the T4 GPU (batch size 1) was between 2.44 and 2.77 ms per image for all variants, so at this scale the parameter count does not translate into measurable latency. The benefit of GAP is therefore memory rather than speed: the weights of I-Base+ occupy about 74 KB in 32-bit floating point, against about 32 MB for A1, whereas the number of multiply–accumulate operations, dominated by the convolutional layers, differs by only about 10 % (about 80 M versus 89 M).

 ISSN:2302-8149
e-ISSN:2540-9719

The two-block configuration is retained as the reference model because it was fixed before the ablation and, together with its 64 × 64 variant, is the smallest configuration above 98.5 % accuracy; where false alarms are costly, the three-block variant is preferable, and its family-aware evaluation is left for future work. Since 64 × 64 inputs perform equally well, the input resolution could also be reduced further without loss.

Table 7 Ablation Study on the Independent Test Set (mean ± SD over three runs)

Variant	Head	Conv. blocks	Input	Parameters	Accuracy (%)	F1 (%)	FPR (%)
I-Base+ (full)	GAP	2	128^2	18,881	98.62 ± 0.39	99.07 ± 0.26	3.85 ± 2.05
A1: Flatten + FC	Flatten + FC(128)	2	128^2	8,407,681	99.57 ± 0.09	99.71 ± 0.06	0.96 ± 0.13
A2: one block	GAP	1	128^2	353	92.12 ± 2.91	94.63 ± 2.01	14.07 ± 4.63
A3: three blocks	GAP	3	128^2	92,801	99.44 ± 0.22	99.62 ± 0.15	1.70 ± 1.03
A4a: low resolution	GAP	2	64^2	18,881	98.76 ± 0.41	99.16 ± 0.28	4.07 ± 1.58
A4b: high resolution	GAP	2	224^2	18,881	97.98 ± 0.22	98.64 ± 0.14	5.48 ± 1.64

The I-Base+ values in Table 7 differ slightly from Table 4 because the ablation uses only the first three of the five seeds.

4.3 Comparison with Baselines under an Identical Protocol

Table 8 compares I-Base+ with texture-based and transfer-learning baselines trained and evaluated on exactly the same data, splits and seeds, so that the differences can be attributed to the models rather than to the experimental setting. The texture-based Gabor + k-NN baseline is clearly weaker (96.60 ± 0.44 % accuracy, false-positive rate 9.33 %), which shows that the learned convolutional features carry more information than a fixed GIST-like descriptor. Both transfer-learning baselines are more accurate than I-Base+ on the same three seeds: VGG16 reaches 99.40 ± 0.12 % and ResNet50 99.67 ± 0.19 %, against 98.62 ± 0.39 %, and both have lower false-positive rates (1.33 % and 0.96 % against 3.85 %). Their ROC-AUC values (99.96 % and 99.86%) are also above that of I-Base+ (99.52% over five runs). These baselines, however, rely on ImageNet pre-training and are about 780 and 1,250 times larger, and on the same CPU they need 180.68 and 90.20 ms per image against 7.22 ms for I-Base+. The three-block GAP variant (A3, Table 7), trained from scratch, closes most of the gap: its accuracy (99.44 ± 0.22 %) is comparable to that of VGG16 with 159 times fewer parameters, and it is 0.23 points below ResNet50 with 254 times fewer. The comparison therefore describes a trade-off rather than a single winner: where memory and latency are constrained, I-Base+ or its three-block variant gives accuracy within about one point of models three orders of magnitude larger, whereas a pre-trained ResNet50 is preferable when resources are not a constraint.

The VGG16 and ResNet50 latencies include the forward pass through the frozen backbone, which dominates their cost; the Gabor + k-NN baseline was not timed because its cost depends mainly on the size of the reference set.

Table 8 Controlled Comparison on the Same Splits (binary task, mean ± SD over three runs)

Method	ImageNet pre-training	Parameters	Accuracy (%)	F1 (%)	FPR (%)	CPU inference (ms/image)
B1: GIST-like Gabor + k-NN [1]	No	– (non-parametric)	96.60 ± 0.44	97.72 ± 0.29	9.33 ± 2.44	–
B2: VGG16 transfer learning [10]	Yes	14,715,201	99.40 ± 0.12	99.59 ± 0.08	1.33 ± 0.00	180.68
B3: ResNet50 transfer learning [13]	Yes	23,589,761	99.67 ± 0.19	99.78 ± 0.13	0.96 ± 0.68	90.20
I-Base+ (proposed)	**No**	**18,881**	**98.62 ± 0.39**	**99.07 ± 0.26**	**3.85 ± 2.05**	**7.22**
I-Base+ three blocks (A3, Table 7)	No	92,801	99.44 ± 0.22	99.62 ± 0.15	1.70 ± 1.03	–

 ISSN:2302-8149
e-ISSN:2540-9719

4.4 Positioning with Respect to Published Results

Published MalImg results are reported for 25-class family classification, which is a more challenging task than binary detection, and with different protocols; a direct ranking of the published performances would therefore be misleading. In Table 9, the results are grouped according to the performed task. For the multi-class problem, the same DCNN used above, trained with a 25-unit softmax layer, obtained an accuracy of 97.05 ± 0.57 % (95 % CI 96.34–97.76 %; macro-F1 90.83 ± 1.33 %). This is comparable to the 97.18 % of Nataraj et al. [1], better than the 94.50 % of Cui et al. [7] and 1.77 points below the 98.82 % of the fine-tuned VGG16 of Vasan et al. [10], while using more than three orders of magnitude fewer parameters (0.02 M vs. > 138 M) and a version of the data from which exact duplicates have been removed, so that no copies can increase the score (near-duplicates might still remain). The lower macro-F1 indicates that the errors are concentrated in the smaller families, whose errors contribute to the macro average to a higher degree than to the overall accuracy (e.g., the attention maps of CNNs for Swizzor.gen!E and Swizzor.gen!I have been reported to be almost the same [26]). For binary classification, the closest published result is the 98.39 % accuracy of the ResNet34 model of Bhodia et al. [11], obtained on similar data (MalImg combined with benign Windows files) but with a different benign set and protocol; I-Base+ obtains 98.72 % with about one thousandth of the parameters of ResNet34. It should be noted that the 95 % confidence interval of the multi-class performance (96.34–97.76 %) contains the 97.18 % of [1], although the two figures were obtained on different versions of the data and with different protocols

Table 9 Published Results on MalImg Grouped by Task (for context only; protocols differ)

Reference	Method	Trainable parameters	Evaluation protocol	Accuracy (%)
Multi-class (25 MalImg families)				
Nataraj et al. (2011) [1]	GIST + k-NN	– (non-parametric)	10-fold cross-validation	97.18
Cui et al. (2018) [7]	CNN with bat-algorithm data balancing	not reported	as reported in [7]	94.50
Vasan et al. (2020) [10]	IMCFN (fine-tuned VGG16)	> 138 M	as reported in [10]	98.82
This work	I-Base+ DCNN (softmax head)	0.02 M	de-duplicated, 70/15/15, 5 seeds	97.05 ± 0.57
Binary (benign vs. malware)				
Bhodia et al. (2019) [11]	ResNet34 transfer learning	≈ 21.8 M	MalImg + 3,304 benign files	98.39
This work	I-Base+ DCNN (sigmoid head)	0.02 M	de-duplicated, 70/15/15, 5 seeds	98.72 ± 0.36

The mere accuracy does not reflect the differences between the studies. Therefore, Table 10 contrasts the studies in aspects which have an influence on the validity of the comparison. In other words, the table points out the differences between the studies in terms of the task, benign software use, the procedure of duplicates removal, evaluation methodology, variability description, generalisability to new families, and the model size. If the original paper did not provide information about a certain aspect of the study, then "not reported" is noted rather than implying its absence. Recently it was shown that published results on MalImg dataset are usually difficult to reproduce due to the lack of hyper-parameters and data split description [26], that is why both are described here. As a result, the most significant difference between the I-Base+ and other methods is not higher accuracy, but (i) accuracy in the range of models that are three to four orders of magnitude larger, (ii) evaluation on a de-duplicated data set and the independent test set with confidence intervals, and (iii) the performance estimate on unseen malware families.

 ISSN:2302-8149
e-ISSN:2540-9719

Table 10 Methodological Comparison with the Studies in Table 9

Study	Task	Benign files	Duplicates removed	Test protocol	Variability reported	Unseen families	Parameters
Nataraj et al. [1]	25-class	No	Not reported	10-fold CV	Not reported	n/a	Non-parametric
Cui et al. [7]	25-class	No	Not reported	As in [7]	Not reported	n/a	Not reported
Vasan et al. [10]	25-class	No	Not reported	As in [10]	Not reported	n/a	> 138 M
Bhodia et al. [11]	Binary	Yes (3,304)	Not reported	As in [11]	Not reported	Simulated zero-day	≈ 21.8 M
I-Base+ (this work)	Binary + 25-class	Yes (3,000, signed)	Yes (895)	Independent 15 % test set	Mean ± SD, 95 % CI	Yes (5-fold, all families)	0.02 M

4.5 Discussion

Overall, we find that a CNN only two blocks deep, with fewer than 19,000 parameters, is capable of distinguishing benign PE samples from malicious ones with 98.72 ± 0.36 % accuracy and 99.51 % recall on a separate test set. False positives are not much higher when entire malware families are excluded from the training process, while recall is family-dependent (as low as 44.88 % for the most difficult fold), which suggests that the model generalizes well to new variants of known families and underperforms on families with unusual textures. While there can be many factors contributing to such discriminatory power, one possible (though unproven) reason is the difference between the visual characteristics of the two classes:

Benign software: regular and repetitive textures reflecting structured code libraries, resources and compiler layouts.

Malware: random, high-entropy, noise-like textures resulting from the compression, encryption and obfuscation methods that allow attackers to conceal malicious code. Our hypothesis about the importance of textures in discriminating malware from legitimate applications is consistent with studies on the explainability of CNN models trained on MalImg, which showed that such models pay attention to localized regions, including the PE header, resources and high-entropy regions [26]; a corresponding analysis for the binary task is left for future work.The ablation study (Table 7) clarifies the role of GAP. Because GAP averages each feature map over the whole image, the decision depends on the global distribution of textures rather than on the position of specific byte patterns, and the classifier needs only 65 weights. This makes the model extremely small, but it does not make it more accurate: a fully connected head is about one point more accurate at 445 times the size, and a third convolutional block closes most of that gap. GAP should therefore be read as a size–accuracy trade-off, and the appropriate configuration depends on the memory budget of the target device; the family-aware results (Table 6) remain the more relevant indicator of performance on previously unseen malware.

5 Conclusion

This paper introduce I-Base+, a new lightweight image-based binary classification system that classifies Windows executables as malicious or benign based on their image representations. The executables are depicted as standard grayscale images, and classified by a two-block DCNN consisting of a Global Average Pooling (GAP) layer and 18,881 trainable parameters. On an independent test set, averaged from 5 runs, the classifier achieves an accuracy of 98.72 ± 0.36 %, a recall of 99.51 ± 0.26 %, an F1-score of 99.14 ± 0.24 % and a false-positive rate of 3.51 ± 1.62 %. When the malware families are unseen during training, the false-positive rate remains low (4.20 %), but the average recall falls to 81.64 %, meaning that the main open problem for such a detector is the detection of completely new families. The GAP layer reduces the number of parameters by about 445× compared to a fully connected head, at the cost of about one percentage point of accuracy, and a three-block version (92,801 parameters) recovers most of this loss (99.44 %). Under the same

 ISSN:2302-8149
e-ISSN:2540-9719

protocol, frozen ImageNet-pretrained VGG16 and ResNet50 baselines are 0.8 and 1.1 percentage points more accurate than I-Base+, but 780 and 1,250 times larger and 12–25 times slower on a CPU. The three-block version is as accurate as the VGG16 baseline without pre-training, and a Gabor texture-based baseline is clearly weaker (96.60 %). These results are not suitable for comparison with multi-class family-classification accuracies; however, when trained for 25-class classification on the de-duplicated MalImg dataset, the same network achieves an accuracy of 97.05 ± 0.57 %, which is in the range of much larger published networks. Lastly, 895 MalImg images have been identified as exact duplicates and should be removed in future evaluations of this dataset. Finally, 895 MalImg images were found to be exact replicas of other images in the dataset and ought to be excluded from future evaluations on this dataset. Future work involves testing the three-block variant under the family-aware protocol, evaluating the framework on recent malware corpora and temporal splits, and examining its robustness to adversarial byte-level manipulation.

References

[1] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "*Malware Images: Visualization and Automatic Classification*," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security (VizSec)*, 2011, pp. 1–7, DOI: 10.1145/2016904.2016908.

[2] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "*Malware Analysis using Visualized Images and Entropy Graphs*," *International Journal of Information Security*, Vol. 14, No. 1, pp. 1–14, 2015, DOI: 10.1007/s10207-014-0242-0.

[3] R. Chaganti, V. Ravi, and T. D. Pham, "*Image-based Malware Representation Approach with EfficientNet Convolutional Neural Networks for Effective Malware Classification*," *Journal of Information Security and Applications*, Vol. 69, Art. no. 103306, 2022, DOI: 10.1016/j.jisa.2022.103306.

[4] T. T. Son, C. Lee, H. Le-Minh, N. Aslam, and V. C. Dat, "*An Enhancement for Image-based Malware Classification using Machine Learning with Low Dimension Normalized Input Images*," *Journal of Information Security and Applications*, Vol. 69, Art. No. 103308, 2022, DOI: 10.1016/j.jisa.2022.103308.

[5] A. Copiaco, L. El Neel, T. Nazzal, H. Mukhtar, and W. Obaid, "*A Neural Network Approach to a Grayscale Image-based Multi-File Type Malware Detection System*," *Applied Sciences*, Vol. 13, No. 23, Art. No. 12888, 2023, DOI: 10.3390/app132312888.

[6] S. Venkatraman, M. Alazab, and R. Vinayakumar, "*A Hybrid Deep Learning Image-based Analysis for Effective Malware Detection*," *Journal of Information Security and Applications*, Vol. 47, pp. 377–389, 2019, DOI: 10.1016/j.jisa.2019.06.006.

[7] Z. Cui, F. Xue, X. Cai, Y. Cao, and G. G. Wang, "*Detection of Malicious Code Variants based on Deep Learning*," *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 7, pp. 3187–3196, 2018, DOI: 10.1109/TII.2018.2822680.

[8] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "*Malware Classification with Deep Convolutional Neural Networks*," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018, pp. 1–5, DOI: 10.1109/NTMS.2018.8328749.

[9] B. Zou, C. Cao, F. Tao, and L. Wang, "*IMCLNet: A Lightweight Deep Neural Network for Image-based Malware Classification*," *Journal of Information Security and Applications*, Vol. 70, Art. No. 103313, 2022, DOI: 10.1016/j.jisa.2022.103313.

[10] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "*IMCFN: Image-based Malware Classification using Fine-Tuned Convolutional Neural Network Architecture*," *Computer Networks*, Vol. 171, Art. No. 107138, 2020, DOI: 10.1016/j.comnet.2020.107138.

[11] N. Bhodia, P. Prajapati, F. Di Troia, and M. Stamp, "*Transfer Learning for Image-based Malware Classification*," in *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP)*, 2019, pp. 719–726, DOI: 10.5220/0007701407190726.

[12] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "*Dos and Don'ts of Machine Learning in Computer Security*," in *Proceedings of the 31st USENIX Security Symposium*, 2022, pp. 3971–3988.

[13] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. de Geus, "*Malicious Software Classification using Transfer Learning of ResNet-50 Deep Neural Network*," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017, pp. 1011–1014, DOI: 10.1109/ICMLA.2017.00-19.

[14] W. El-Shafai, I. Almomani, and A. AlKhayer, "*Visualized Malware Multi-Classification Framework using Fine-Tuned CNN-based Transfer Learning Models*," *Applied Sciences*, Vol. 11, No. 14, Art. No. 6446, 2021, DOI: 10.3390/app11146446.

[15] M. J. Awan, O. A. Masood, M. A. Mohammed, A. Yasin, A. M. Zain, R. Damaševičius, and K. H. Abdulkareem, "*Image-based Malware Classification using VGG19 Network and Spatial Convolutional Attention*," *Electronics*, Vol. 10, No. 19, Art. no. 2444, 2021, DOI: 10.3390/electronics10192444.

[16] M. Ashawa, N. Owoh, S. Hosseinzadeh, and J. Osamor, "*Enhanced Image-based Malware Classification using Transformer-based Convolutional Neural Networks (CNNs)*," *Electronics*, Vol. 13, No. 20, Art. No. 4081, 2024, DOI: 10.3390/electronics13204081.

[17] Y. Yu, B. Cai, K. Aziz, X. Wang, J. Luo, M. S. Iqbal, P. Chakrabarti, and T. Chakrabarti, "*Semantic Lossless Encoded Image Representation for Malware Classification*," *Scientific Reports*, Vol. 15, Art. No. 7997, 2025, DOI: 10.1038/s41598-025-88130-0.

[18] A. Tekerek and M. M. Yapici, "*A Novel Malware Classification and Augmentation Model based on Convolutional Neural Network*," *Computers & Security*, Vol. 112, Art. No. 102515, 2022, DOI: 10.1016/j.cose.2021.102515.

[19] M. Asam, S. H. Khan, A. Akbar, S. Bibi, T. Jamal, A. Khan, U. Ghafoor, and M. R. Bhutta, "*IoT Malware Detection Architecture using a Novel Channel Boosted and Squeezed CNN*," *Scientific Reports*, Vol. 12, Art. No. 15498, 2022, DOI: 10.1038/s41598-022-18936-9.

[20] K. Shaukat, S. Luo, and V. Varadharajan, "*A Novel Deep Learning-based Approach for Malware Detection*," *Engineering Applications of Artificial Intelligence*, Vol. 122, Art. No. 106030, 2023, DOI: 10.1016/j.engappai.2023.106030.

[21] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "*An Efficient DenseNet-based Deep Learning Model for Malware Detection*," *Entropy*, Vol. 23, No. 3, Art. no. 344, 2021, DOI: 10.3390/e23030344.

[22] S. Mahdavifar, A. F. A. Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "*Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning*," in *2020 IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, 2020, pp. 515–522, DOI: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00094.

[23] K. Shaukat, S. Luo, and V. Varadharajan, "*A Novel Method for Improving the Robustness of Deep Learning-based Malware Detectors Against Adversarial Attacks*," *Engineering Applications of Artificial Intelligence*, Vol. 116, Art. No. 105461, 2022, DOI: 10.1016/j.engappai.2022.105461.

[24] M. Lin, Q. Chen, and S. Yan, "*Network in Network*," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014, arXiv:1312.4400.

[25] T. Saito and M. Rehmsmeier, "*The Precision-Recall Plot is More Informative than the ROC Plot when Evaluating Binary Classifiers on Imbalanced Datasets*," *PLoS ONE*, Vol. 10, No. 3, Art. No. e0118432, 2015, DOI: 10.1371/journal.pone.0118432.

[26] M. Brosolo, P. Vinod, and M. Conti, "*Through the Static: Demystifying Malware Visualization via Explainability*," arXiv preprint arXiv:2503.02441, 2025.