electronics

MDPI

Article

TLECNN: A Transfer Learning-Enhanced CNN for Malware-Family Classification in IoT-Relevant Security Environment

Tuhin Shukla * and Nishchol Mishra

School of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal 462033, India; nishchol@rgpv.ac.in

* Correspondence: shukla.tuhin87@gmail.com

Abstract

The rapid growth of Internet of Things (IoT) devices has greatly expanded the attack surface for malware. It has created a need for accurate malware classification techniques suitable for resource-aware IoT security analysis in such environments. In this paper, a transfer learning-enhanced convolutional neural network (TLECNN) for image-based malware classification is proposed. This proposed framework is based on a ResNet50 backbone initialized with ImageNet pretrained weights and fine-tuned to classify malware samples from the MalImg dataset, containing 25 malware families. The framework systematically evaluates frozen, full, and selective fine-tuning strategies, including class-weighted learning, and retains selective fine-tuning without class weighting based on class-sensitive performance. The proposed model is evaluated using five optimization algorithms: Adam, NAdam, Adagrad, Adadelta, and Adamax. The experimental results show that the proposed TLECNN achieves an accuracy of 99.49%. In the reported optimizer evaluation, NAdam yielded the highest observed performance, with macro-precision of 98.87%, macro-recall of 98.73%, macro-F1-score of 98.79%, and MCC of 0.9940. The high macro-recall and F1-score are particularly relevant for malware family classification, where poor recognition of less frequent malware families may be concealed by high overall accuracy. The proposed framework demonstrates competitive performance in multiclass malware classification for centralized IoT-oriented security analysis. Computational characteristics were quantified, and the selected framework was further evaluated on an external multi-architecture ARM/MIPS and x86-64 ELF IoT malware dataset. These findings support the use of TLECNN for centralized or gateway-oriented IoT security analysis and provide a basis for extending transfer learning-based malware analysis to IoT-oriented cybersecurity applications.

Keywords: malware classification; transfer learning; convolutional neural network; ResNet50; Internet of Things; cybersecurity; deep learning

Check for updates

Academic Editor: Hung-Yu Chien

Received: 17 August 2026
Revised: 5 September 2026
Accepted: 8 September 2026
Published: 14 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The rapid expansion of Internet of Things (IoT) technologies has transformed industrial systems, healthcare, transportation, surveillance, smart homes, and other cyber-physical environments. These applications rely on interconnected devices that continuously collect, process, and exchange data through public and private networks [1]. However, the increasing number of connected devices has also expanded the attack surface available to malicious actors. Many IoT devices operate with limited computing resources, weak

authentication mechanisms, outdated firmware, and inadequate security controls, making them attractive targets for malware attacks [2].

Malware can be classified into several types based on its features and functions, including worms, viruses, adware, ransomware, Trojans, backdoors, and spyware [3]. This classification comprises numerous variants with comparable characteristics. Malware has become more infectious owing to advancements in obfuscation, code insertion, and fileless capabilities that circumvent firewalls [4].

Several image-based malware classification approaches have been proposed using CNNs, transfer learning models, attention mechanisms, and class-balancing strategies [5–15]. These methods have demonstrated that deep learning can achieve high classification performance on benchmark datasets such as MalImg [16]. Nevertheless, models trained entirely from scratch may require substantial computational resources, large training datasets, and careful hyperparameter selection. Their performance may also vary considerably across minority malware families because benchmark datasets are frequently imbalanced.

Transfer learning offers a practical mechanism for addressing some of these challenges. Instead of learning all visual representations from randomly initialized parameters, transfer learning reuses knowledge acquired from a large source dataset and adapts it to a related target task. Pretrained convolutional networks can provide general-purpose representations of edges, textures, shapes, and spatial patterns that may also be useful for identifying structural characteristics in malware images. This can improve convergence stability and reduce the computational cost of training a deep network entirely from scratch [6,10,12].

Accuracy alone may be insufficient for imbalanced malware datasets because strong performance on majority classes can conceal weaker classification of less frequent malware families. Despite the high accuracies reported in the literature, evaluation practices vary across studies, particularly in how class imbalance is handled and in the use of class-sensitive performance measures [7,9,11,17,18]. Furthermore, the effect of optimizer selection on transfer learning-based malware classification has not always been systematically examined within a common experimental framework.

To address these issues, this study proposes a transfer learning-enhanced convolutional neural network, referred to as TLECNN, for image-based malware family classification. The framework uses knowledge derived from pretrained ImageNet representations and adapts it to the structural patterns contained in malware images. The proposed model is evaluated on the MalImg benchmark dataset, which contains 9339 grayscale images distributed across 25 malware families. Multiple optimizers, including Adam, NAdam, Adagrad, Adadelta, and Adamax, are investigated to determine their influence on classification performance.

Malware visualization provides an alternative representation in which binary files are converted into grayscale or colour images. Bytes from a malware binary are interpreted as pixel intensities, producing visual textures that often preserve structural similarities among samples belonging to the same malware family [16]. This representation allows malware classification to be treated as an image classification problem and enables convolutional neural networks (CNNs) to automatically learn discriminative spatial features without extensive manual feature engineering.

The primary contributions of this research are:

1. Selective transfer learning adaptation: A ResNet50-based TLECNN framework is developed for malware family classification, with frozen, full, selective, and class-weighted fine-tuning strategies evaluated to determine an effective adaptation policy.
2. Controlled optimizer sensitivity and robustness evaluation: Adam, NAdam, Adagrad, Adadelta, and Adamax are examined under a common experimental framework,

complemented by paired statistical analysis and group-aware five-fold evaluation to assess optimizer sensitivity and performance stability.

3. Leakage-safe and class-sensitive evaluation: Exact-duplicate auditing and group-aware partitioning are incorporated to reduce duplicate leakage, while macro-averaged and class-wise measures are used to evaluate performance on the highly imbalanced MalImg benchmark.
4. Family-level comparison with a matched ResNet50 baseline: Beyond aggregate performance, a controlled class-wise comparison is performed across all 25 malware families to identify where the TLECNN adaptation provides additional discriminative benefit and where baseline performance is retained.
5. Resource-aware and external IoT-oriented validation: The retained framework is characterized using computational and model-resource metrics and further evaluated on external ARM/MIPS and x86-64 ELF malware datasets to assess its applicability beyond the MalImg benchmark for centralized or gateway-oriented IoT security analysis.

The remainder of this paper is organized as follows. Section 2 presents a review of related work on malware visualization, deep learning, and transfer learning-based malware classification. In Section 3, we introduce the proposed TLECNN framework, dataset, preprocessing procedure, model architecture, and evaluation metrics. The experimental setup and results with different optimizers are presented in Section 4. Section 5 compares the proposed approach with the baseline model and the previous studies. Finally, Section 6 concludes the study and provides directions for future research.

2. Literature Review

Image-based malware classification considers malware binaries as visual representations to learn structural patterns via image classification techniques. Nataraj et al. [16] demonstrated the utility of malware images for family classification, and subsequent work has utilized convolutional neural networks (CNNs) to automatically learn discriminative features. Recently, CNN-based and transfer learning methods have shown high classification performance on benchmark datasets such as MalImg [6,8,10].

Transfer learning has been particularly useful, as it allows pretrained networks to be adapted to malware images without training a deep model from scratch. Vasan et al. [6] proposed a fine-tuned CNN architecture for image-based malware classification. Kumar et al. [10] investigated fine-tuned CNNs and transfer learning for IoT-related malware classification. Go et al. [8] used ResNeXt for malware–image classification and demonstrated the effectiveness of deep-pretrained feature representations.

Recent work has also looked at attention mechanisms and class imbalance. Awan et al. [9] proposed a VGG19-based spatial attention framework in Electronics to classify 25 malware families and studied the performance with and without class balancing.

AlNajim et al. [13] used a pretrained MobileNet model with spatial and channel attention for malware classification in Electronics. More recent studies have explored EfficientNet-based feature fusion [19], dual-CNN architectures [20], and knowledge transfer strategies [21] for malware classification. Other studies have examined SMOTE-based balancing [11], transfer learning architectures [12], and attention-based CNNs [14]. These approaches demonstrate that achieving high overall accuracy is possible, but model complexity and unequal class distributions remain issues of concern.

The MalImg dataset is highly imbalanced, with some malware families containing several thousand samples and some with fewer than a hundred samples. Thus, overall accuracy may not be the best measure of classification performance. Recall and F1-score are especially helpful for studying the recognition of rare classes. Thus, recent studies have

considered class balancing, class-sensitive evaluation, and model efficiency measures in addition to overall accuracy [7,9,17,18].

However, the current work tends to consider each of these aspects—transfer learning, class imbalance, optimizer selection, and computational efficiency—separately. Relatively few studies have examined the effect of different optimizers on the same ResNet50-based transfer learning framework in terms of class-sensitive performance. To this end, the proposed TLECNN fills the gap by adapting an ImageNet-pretrained ResNet50 model to the 25-class malware family classification task and systematically comparing Adam, NAdam, Adagrad, Adadelta, and Adamax under a common experimental setting. The obtained model is evaluated in terms of accuracy, precision, recall, F1-score, and cross-validation and compared with recent approaches for malware classification.

3. Proposed TLECNN Framework

This section presents the proposed transfer learning-enhanced convolutional neural network (TLECNN) for image-based malware family classification. The framework uses an ImageNet-pretrained ResNet50 [22] backbone and a malware-specific classification head. It also incorporates controlled fine-tuning, class-weighted learning, optimizer evaluation, and resource-aware model selection.

The proposed framework is organized into four stages: architecture construction, fine-tuning and class imbalance analysis, optimizer evaluation, and stratified cross-validation. Figure 1 presents the architecture of the proposed TLECNN, while Algorithm 1 describes the complete experimental procedure.

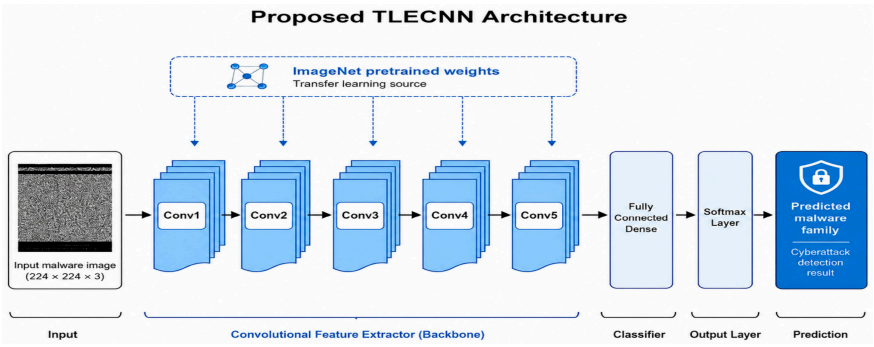


Figure 1. The architecture of the proposed TLECNN framework.

3.1. TLECNN Architecture and Fine-Tuning Strategy

The proposed TLECNN uses ResNet50 [22] as its feature extraction backbone. ResNet50 was selected because its residual connections allow a deep network to learn complex image features while reducing the difficulty of training very deep architectures.

The model was initialized using weights learned from the ImageNet dataset. Although ImageNet contains natural images rather than malware images, the pretrained model provides useful low-level representations such as edges, textures, shapes, and spatial patterns. These representations can be adapted to the visual patterns present in malware images.

The original classification layer of ResNet50 was removed because it was designed to classify 1000 ImageNet classes. A new malware-specific classification head was added to classify the 25 malware families in the MalImg dataset [16].

The architecture of the proposed model consists of the following components:

1. An input layer accepting images of size $224 \times 224 \times 3$;
2. An ImageNet-pretrained ResNet50 feature extraction backbone;
3. A global average pooling layer;
4. A fully connected layer containing 1024 units with ReLU activation;

5. A final fully connected layer containing 25 units with Softmax activation.

Algorithm 1: Proposed Imbalance-Aware and Resource-Aware TLECNN

Input:
MalImg dataset D
ImageNet-pretrained ResNet50
Candidate Fine-tuning configurations F
Optimizers O
Number of folds K = 5

1: Load, resize, and preprocess the MalImg images.
2: Create leakage-safe, group-aware training and evaluation partitions.
3: Construct TLECNN using the ImageNet-pretrained ResNet50 backbone and malware-specific classification head.
4: Evaluate the candidate fine-tuning and class-weighting configurations under identical training conditions.
5: Select the retained fine-tuning configuration f* using macro-F1-score as the primary criterion, supported by macro-recall, MCC, accuracy, and trainable parameter count.
6: With f* fixed as selective fine-tuning, evaluate each optimizer o ∈ O under identical training conditions.
7: Compare optimizer performance and quantify the difference between the strongest configurations using paired statistical analysis.
8: Retain optimizer o* from the reported optimizer evaluation and define M* = TLECNN(f*,o*).
9: Evaluate the robustness of M* using group-aware five-fold stratified folds.
10: Compare M* with ResNet50 baseline using identical experimental conditions.
11: Characterize M* using computational and resource metrics.
12: Return M* and the corresponding evaluation results.

Output:
Final TLECNN model M*
Predictive and computational results

Fine-Tuning Configurations

The study evaluates different ways of adapting the pretrained ResNet50 model to the MalImg dataset. In the frozen-backbone configuration, all ResNet50 layers remain frozen, and only the new classification head is trained. In the full fine-tuning configuration, all ResNet50 layers and the classification head are trained on the MalImg dataset. In the selective fine-tuning configuration, the final ResNet50 residual stage (conv5_x) and the malware-specific classification head are trainable, while the preceding backbone stages remain frozen. The purpose of this configuration is to retain general visual features learned from ImageNet while adapting the higher-level features to malware family classification. The fine-tuning configurations were evaluated as candidate adaptation strategies under identical experimental conditions. Macro-F1-score was used as the primary selection criterion due to the class imbalance in MalImg, with macro-recall, MCC, accuracy, and trainable parameter count used as supporting measures. The configuration selected through this controlled comparison was retained for the subsequent optimizer evaluation.

The experiments also evaluate selective fine-tuning with class-weighted learning. This configuration gives greater importance to malware families containing fewer training samples. The effectiveness of class weighting was experimentally evaluated against the corresponding unweighted selective fine-tuning configuration using identical training and

evaluation settings. Class weighting can be used to account for the unequal number of samples across the malware families, as shown in Table 1.

Table 1. Description of the MalImg dataset.

Class_ID	Label	Category	No. of Samples
0	Adialer. C	Dialer	122
1	Agent. FYI	Backdoor	116
2	Allaple. A	Worm	2949
3	Allaple. L	Worm	1591
4	Alueron.gen!J	Trojan	198
5	Autorun. K	Worm:AutoIT	106
6	C2LOP.gen!g	Trojan	200
7	C2LOP.P	Trojan	146
8	Dialplatform. B	Dialer	177
9	Dontovo. A	TrojanDownloader	162
10	Fakerean	Rogue	381
11	Instantaccess	Dialer	431
12	Lolyda. AA1	PWS	213
13	Lolyda. AA2	PWS	184
14	Lolyda. AA3	PWS	123
15	Lolyda. AT	PWS	159
16	Malex.gen!J	Trojan	136
17	Obfuscator. AD	TrojanDownloader	142
18	Rbot!gen	Backdoor	158
19	Skintrim. N	Trojan	80
20	Swizzor.gen!E	TrojanDownloader	128
21	Swizzor.gen!I	TrojanDownloader	132
22	VB.AT	Worm	408
23	Wintrim. BX	TrojanDownloader	97
24	Yuner. A	Worm	800

3.2. Proposed Imbalance-Aware and Resource-Aware Algorithm

Algorithm 1 presents the complete procedure used to select and evaluate the proposed TLECNN configuration. The procedure was designed to separate architecture selection from optimizer selection so that all optimizers were evaluated using an identical model structure and fine-tuning policy. In Algorithm 1, imbalance-aware refers to the explicit evaluation of class-weighted and unweighted configurations during model selection. Resource-aware refers to the inclusion of the trainable parameter count and computational/resource characterization in the assessment of the retained configuration. The term does not imply that the complete ResNet50-based model is deployed directly on a constrained IoT endpoint; the intended deployment context is a centralized malware analysis component within a broader IoT security framework.

3.3. Dataset Description

The Vision Research Lab released the MalImg dataset [16] to study signal and image processing algorithms for malware analysis. This dataset is publicly available. The MalImg dataset contains 9339 malware images distributed across 25 malware families. The dataset contains various types of malicious software, including worms, dialers, password stealers, and Trojan horses, among other categories of malware. The samples are provided as grayscale malware images and include families belonging to categories such as worms, dialers, backdoors, password stealers, and Trojan downloaders.

The dataset has an extremely imbalanced class distribution. The largest family, Allaple.A, contains 2949 samples, whereas the smallest, Skintrim.N, contains 80. This imbalance motivates the use of class-sensitive performance measures and the evaluation of class-weighted learning. Table 1 presents the class labels, malware categories, and number of samples in each family.

Representative samples from the MalImg dataset are shown in Figure 2. The grayscale images were resized to 224 × 224 pixels and represented using three channels to satisfy the input requirements of the ImageNet-pretrained ResNet50 model.

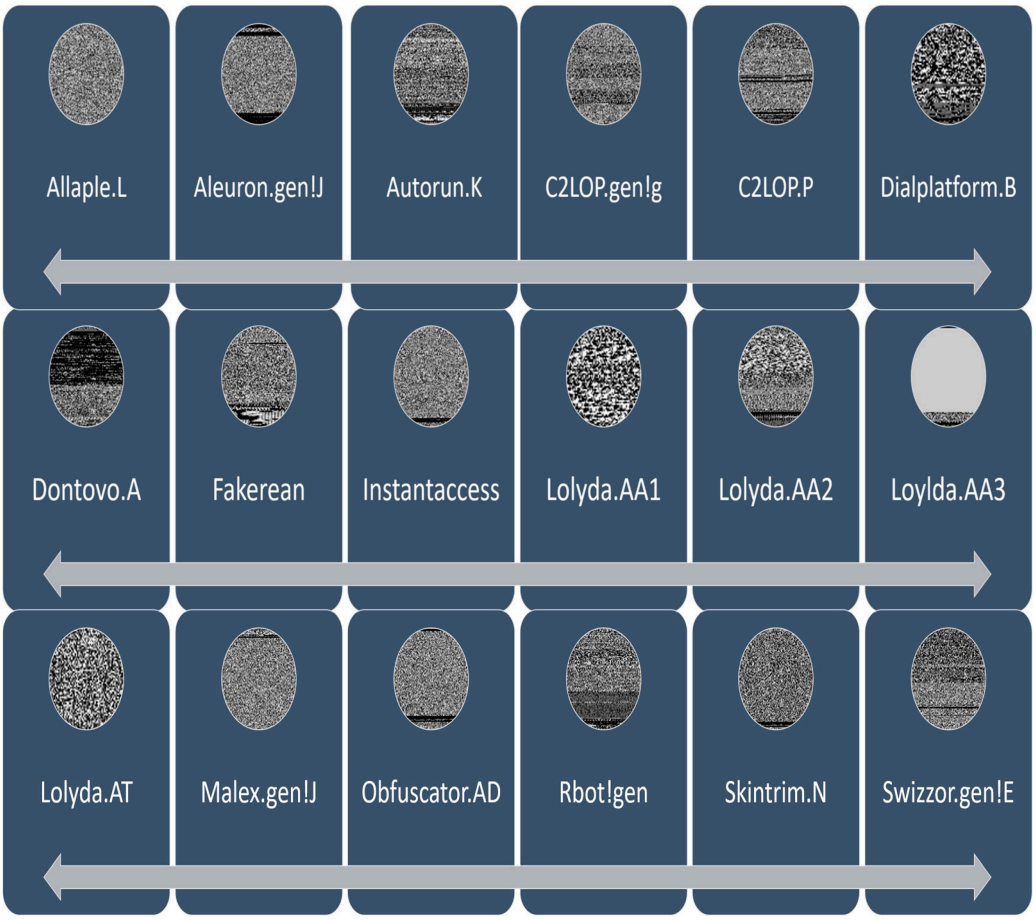


Figure 2. Representative samples of the MalImg dataset.

Exact-Duplicate Audit and Leakage-Safe Partitioning—Before constructing the revision-controlled experimental partitions, all 9339 MalImg files were examined for exact duplicates using SHA-256 hashing. The audit identified 8444 unique hashes, 895 duplicate extra files, and 260 duplicate groups. No exact-duplicate group contained samples belonging to different malware families. To prevent duplicate instances from contributing to both model fitting and evaluation, samples belonging to the same exact-duplicate group were kept within the same partition through group-aware splitting. The result-

ing fixed controlled-revision partition contained 7039 training samples and 2300 held-out evaluation samples.

3.4. Evaluation Metrics

The proposed model is evaluated using accuracy, precision, recall, and F1-score [23]. These measures provide complementary information about multiclass classification performance, particularly for an imbalanced dataset such as MalImg.

Accuracy is the proportion of malware samples correctly classified.

$$Accuracy = \frac{No\ of\ correctly\ classified\ samples}{Total\ no\ of\ samples} \quad (1)$$

Ref. [23] also specifies the accuracy equation for multiclass classification, where it gives the average per-class effectiveness of a classifier:

$$Average\ Accuracy = \frac{1}{l}\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i} \quad (2)$$

where for many classes C_i, tp_i are true positives for C_i, fp_i are false positives, fn_i are false negatives, and tn_i are true negative counts.

Precision describes the proportion of accurately predicted instances that ultimately prove positive. For every malware family c,

$$Precision = \frac{TPc}{TPc + FPc} \quad (3)$$

where TP_c is the number of correctly classified samples belonging to class c, and FP_c is the number of samples incorrectly assigned to that class.

Recall (sensitivity) describes the number of positive cases that our model can anticipate correctly. This statistic is sometimes useful in cases such as ours, in which minimizing the possibility of a false negative is more essential than minimizing the possibility of a false positive. This is necessary in a cyberattack, especially in an IoT environment where a false alert is less of a concern than letting malicious samples slip through the cracks.

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

The F1-score combines precision and recall to provide a comprehensive measure of accuracy.

$$\text{F1-score} = 2 \times \frac{\{\text{Precision} \times \text{Recall}\}}{\{\text{Precision} + \text{Recall}\}} \quad (5)$$

Confusion metrics help evaluate the success of a classification problem in which the output may have two or more classes. Because the MalImg dataset has a highly imbalanced class distribution, macro-averaged precision, recall, and F1-scores were additionally calculated by assigning equal weight to each of the 25 malware families. Balanced accuracy was used to represent the mean recall across classes, while the multiclass Matthews correlation coefficient (MCC) was employed as a complementary measure of overall agreement between the predicted and actual class labels. These measures provide a more class-sensitive assessment than overall accuracy alone and reduce the influence of dominant malware families in model comparisons.

4. Experimental Evaluation and Results

The proposed TLECNN framework was implemented using Keras with TensorFlow on Google Colab Pro. The experiments were performed using an NVIDIA P100 GPU with

32 GB of RAM. Images were resized to 224 × 224 × 3 and divided into 75% training and 25% validation sets during the initial experimental setup used for model development. For the revised evaluation protocol, this partitioning was converted into a leakage-safe group-aware split containing 7039 training samples and 2300 held-out evaluation samples. The same revised partition was used for controlled comparison experiments. The initial optimizer experiments were trained for 100 epochs using a batch size of 16 and a learning rate of 0.0001 under identical experimental conditions. Scikit-learn 1.9.1 was used to calculate classification metrics and confusion matrices, while Matplotlib 3.11.0 was used to visualize training and validation performance.

The experiment evaluation was conducted in two stages. The evaluation protocol separates controlled configuration selection from robustness assessment. For the revision-controlled fine-tuning, class-weighting, paired Adam–NAdam verification, and baseline comparisons, the MalImg dataset was divided into a fixed, leakage-safe, group-aware partition containing 7039 training samples and 2300 held-out evaluation samples. The same partition was used across all controlled comparisons to ensure consistency.

Firstly, ResNet50 was tested as a transfer learning baseline with Adam and NAdam optimizers. Second, the suggested TLECNN was evaluated with Adam, NAdam, Adagrad, Adadelta, and Adamax under the same experimental settings.

After configuration selection, the retained TLECNN model was evaluated separately using five-fold stratified group-aware cross-validation over all 9339 samples. In each outer fold, the held-out fold was used only for evaluation, while an inner group-aware training/validation partition was generated from the remaining samples for early stopping. Thus, the fixed 7039/2300 partition was used for controlled model comparison, whereas the five-fold procedure was used independently to assess robustness across data partitions. To investigate the stability of the selected configuration across different data partitions, stratified five-fold cross-validation was also used.

4.1. ResNet50 Baseline Evaluation

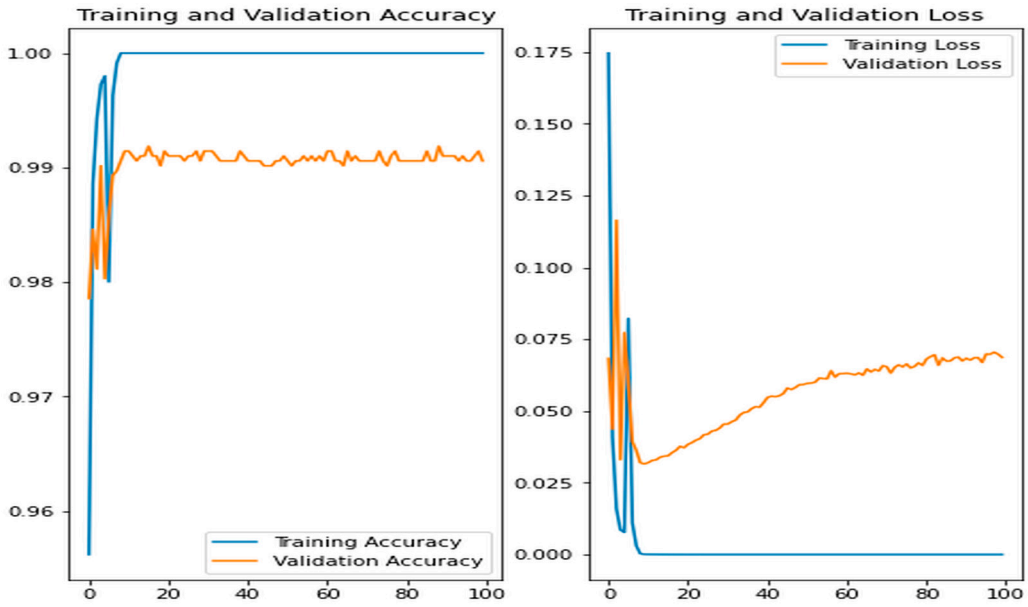
An ImageNet-pretrained ResNet50 [22] was first evaluated as a baseline for malware family classification. The original ImageNet classification layer was replaced with a global average pooling layer and a 25-class softmax output layer. Adam and NAdam were evaluated to establish the baseline performance before the proposed TLECNN classification head was introduced. The baseline results show similar overall accuracy for Adam and NAdam. NAdam provides slightly higher recall and F1-score, whereas Adam gives the higher reported precision (Table 2). The corresponding training and validation curves are shown in Figure 3.

Table 2. Performance of the ResNet50 baseline in the initial 75/25 experimental setup with the MalImg dataset.

No.	Model	Accuracy (%)	*p* (%)	R (%)	F1 (%)
1	ResNet50 with Adam	99.05	99.7	97.32	97.12
2	ResNet50 with NAdam	99.06	99.06	97.88	98.08

4.2. Fine-Tuning and Class-Weighting Ablation

To determine the appropriate adaptation strategy, frozen, full, selective, and selective class-weighted fine-tuning were evaluated under identical experimental conditions. Macro-F1-score was used as the primary selection criterion due to the class imbalance in MalImg, with macro-recall, MCC, accuracy, and trainable parameter count used as supporting measures. The results are presented in Table 3.



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

Figure 3. Comparison of ResNet50 on the MalImg dataset; (**a**) Graph of accuracy; (**b**) Graph of loss during training versus validation.

Table 3. Controlled fine-tuning and class-weighting ablation.

Configuration	Accuracy (%)	Macro-Precision	Macro-Recall	Macro-F1	MCC	Trainable Parameters
Frozen, unweighted	98.7826	0.968104	0.969946	0.968902	0.985696	2123,801
Full fine-tuning, unweighted	99.2609	0.982380	0.980401	0.979671	0.991337	25,658,393
Selective fine-tuning, unweighted	**99.6087**	**0.990787**	**0.990285**	**0.990450**	**0.995403**	17,099,801
Selective fine-tuning, class-weighted	99.3913	0.984079	0.985837	0.984898	0.992848	17,099,801

Note: Bold values indicate the best performance among the compared configurations.

The best overall and class-sensitive performance was obtained by selective fine-tuning without class weighting. Compared with full fine-tuning, it improved the macro-F1-score from 0.979671 to 0.990450 while using fewer trainable parameters. The performance did not improve with class weighting, so the unweighted selective configuration was used in the subsequent optimizer evaluation.

4.3. TLECNN Optimiser Evaluation

The proposed TLECNN was subsequently evaluated using Adam, NAdam, Adagrad, Adadelta, and Adamax. The architecture and experimental settings were kept identical so that the effect of optimizer selection could be assessed under a common setting. Following the fine-tuning and class imbalance analysis described in Section 3, the selected TLECNN configuration was used for the reported optimizer comparison.

Table 4 shows the held-out validation performance used for direct comparison of the five optimizers. The stratified five-fold validation described in Algorithm 1 was performed as a separate robustness stage after model and optimizer selection; Table 4 reports the initial 75/25 held-out validation results used for direct comparison of the five optimizers.

Table 4. Evaluation of proposed model.

No.	Model	Accuracy (%)	Macro-Precision (%)	Macro-Recall (%)	Macro-F1 (%)	MCC
1	Adam	99.19	98.18	97.98	98.02	0.9904
2	NAdam	99.49	98.87	98.73	98.79	0.9940
3	AdaDelta	96.23	87.88	90.03	88.33	0.9559
4	AdaGrad	98.20	95.99	95.62	95.76	0.9788
5	Adamax	98.50	96.51	96.32	96.33	0.9824

The confusion matrices were used further to calculate macro-averaged precision, recall, and F1-score because the MalImg dataset is very imbalanced. In the single-label multiclass setting, balanced accuracy is the same as macro-recall. MCC was also calculated to provide a complementary measure of classification agreement across the 25 malware families.

Within the reported optimizer evaluation, NAdam yielded the highest observed values for accuracy, macro-precision, macro-recall, macro-F1-score, and MCC, while Adam produced closely comparable performance. The remaining optimizers showed lower class-sensitive performance, with Adadelta producing the lowest values in this comparison. These results identify NAdam as the retained configuration from the reported optimizer evaluation, without implying general superiority over Adam.

Because the observed difference between Adam and NAdam was small, an additional paired analysis was performed using the common 2300-sample revision-controlled evaluation set. Adam and NAdam achieved macro-F1-scores of 0.990450 and 0.989050, respectively. McNemar's exact test yielded $p = 1.000$, and a paired bootstrap analysis with 5000 resamples produced a 95% confidence interval of -0.003759 to 0.007517 for the Adam–NAdam macro-F1 difference. As the confidence interval includes zero, the controlled analysis does not establish statistical superiority of either optimizer.

4.3.1. Adam Optimizer

Adam was evaluated using a learning rate of 0.0001 under the common experimental setting described above. Figure 4 presents the training and validation curves, while Figure 5 shows the corresponding confusion matrix.

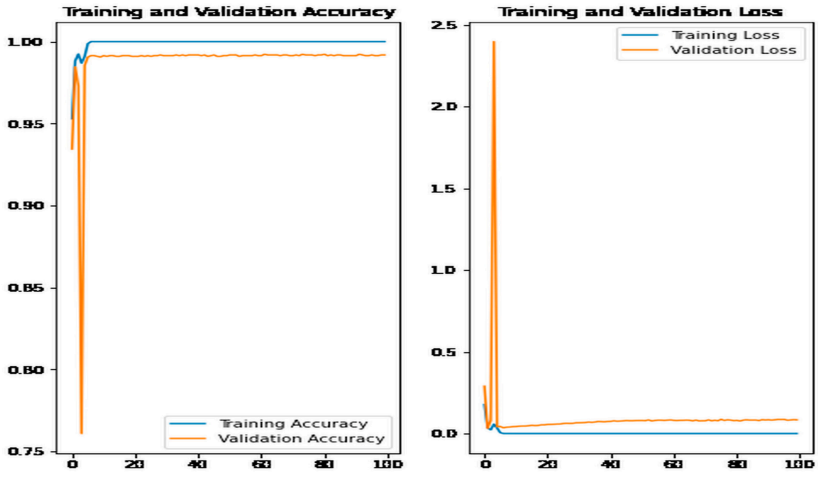


Figure 4. Performance of the model using the Adam optimizer; (**a**) Graph of accuracy; (**b**) Graph of loss during training versus validation.

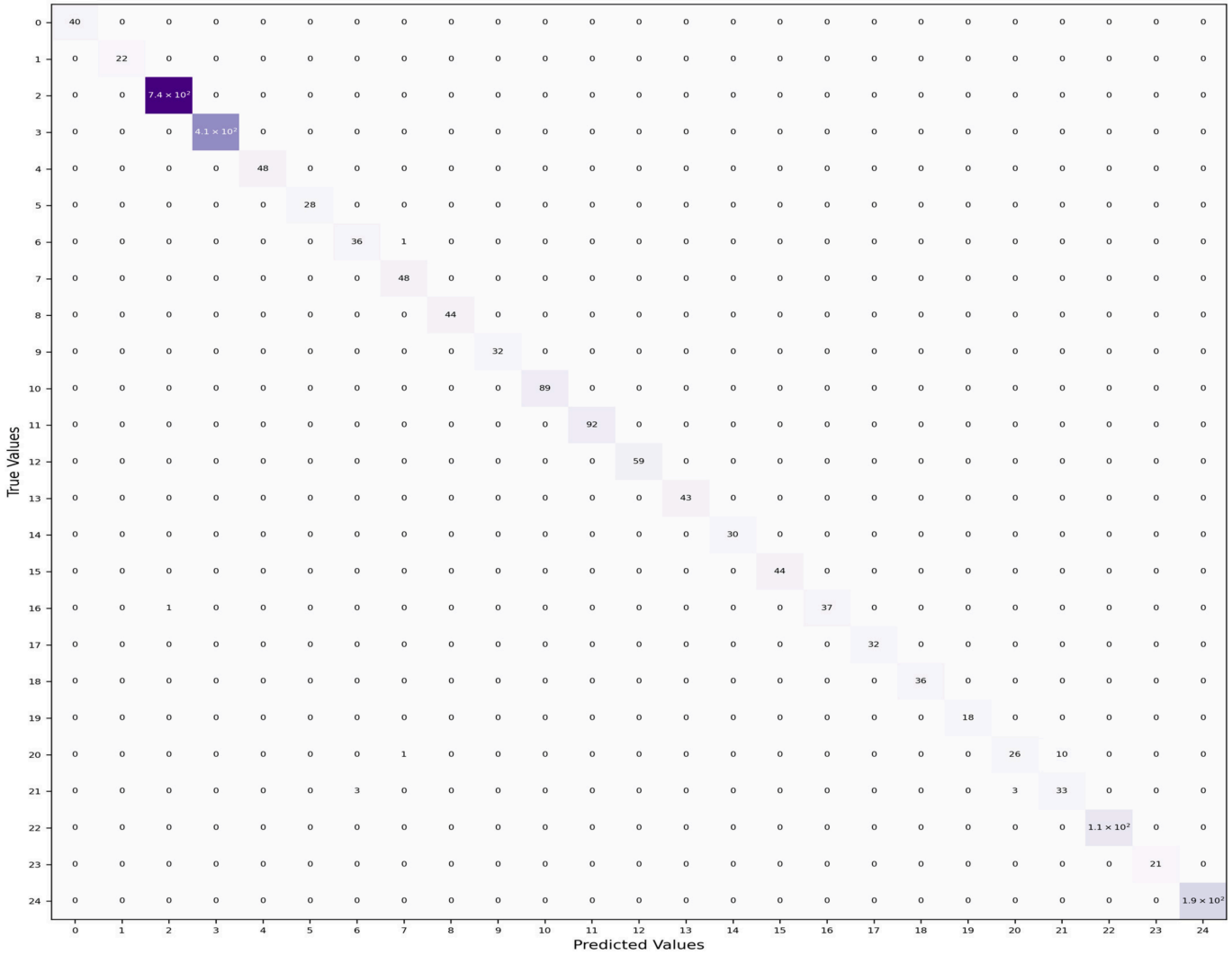


Figure 5. Confusion matrix when the Adam optimizer is used in the model.

4.3.2. NAdam Optimizer

NAdam was evaluated using a learning rate of 0.0001 under the common experimental setting described above. Figure 6 presents the training and validation curves, while Figure 7 shows the corresponding confusion matrix.

4.3.3. Adagrad Optimizer

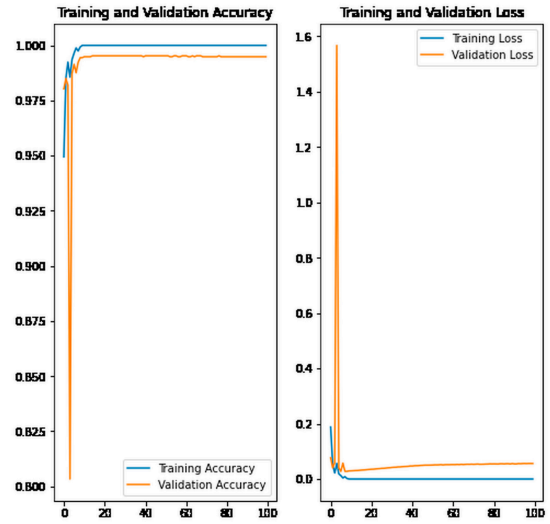
Adagrad was evaluated using a learning rate of 0.0001 under the common experimental setting described above. Figure 8 presents the training and validation curves, while Figure 9 shows the corresponding confusion matrix.

4.3.4. Adadelta Optimizer

Adadelta was evaluated using a learning rate of 0.0001 under the common experimental setting described above. Figure 10 presents the training and validation curves, while Figure 11 shows the corresponding confusion matrix.

4.3.5. Adamax Optimizer

Adamax was evaluated using a learning rate of 0.0001 under the common experimental setting described above. Figure 12 presents the training and validation curves, while Figure 13 shows the corresponding confusion matrix.



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

Figure 6. Performance of the model with the NAdam optimizer; (**a**) Graph of accuracy; (**b**) Graph of loss during training versus validation.

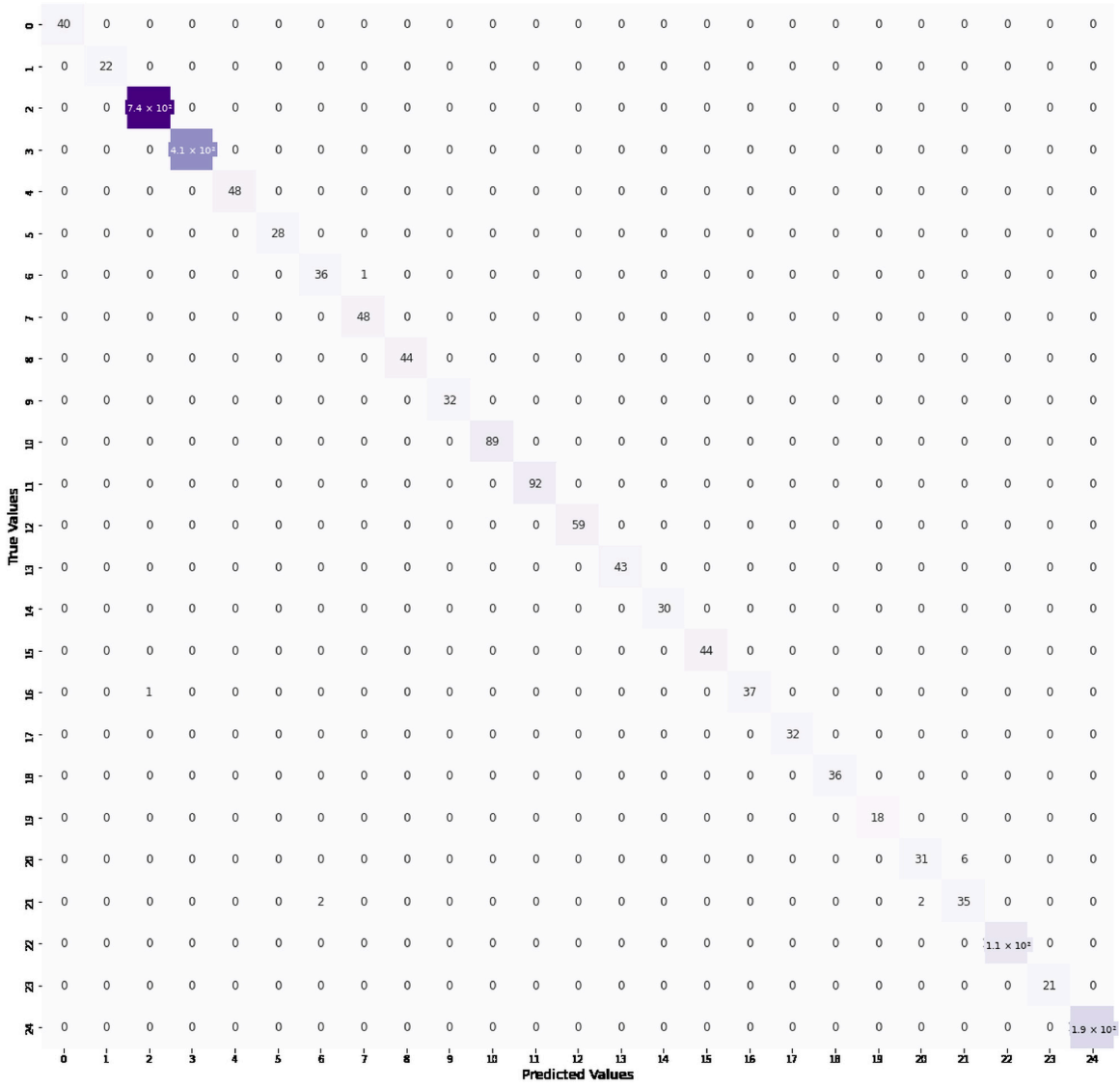
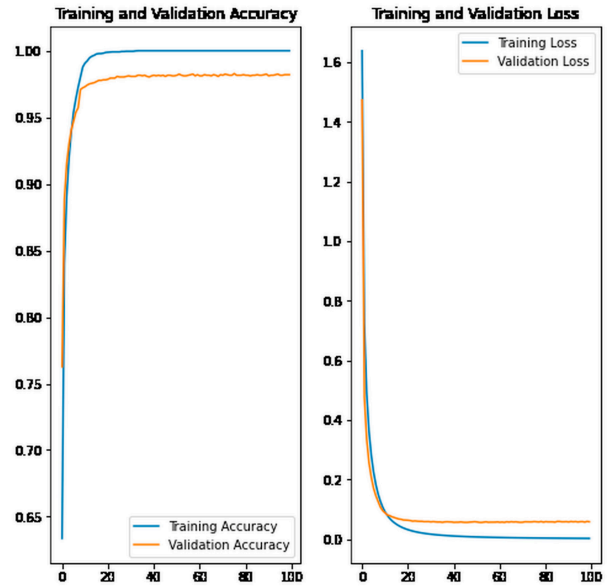


Figure 7. Confusion matrix when the NAdam optimizer is used in the model.



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

Figure 8. Performance of the model with Adagrad optimizer; (**a**) Accuracy; (**b**) Loss during training versus validation.

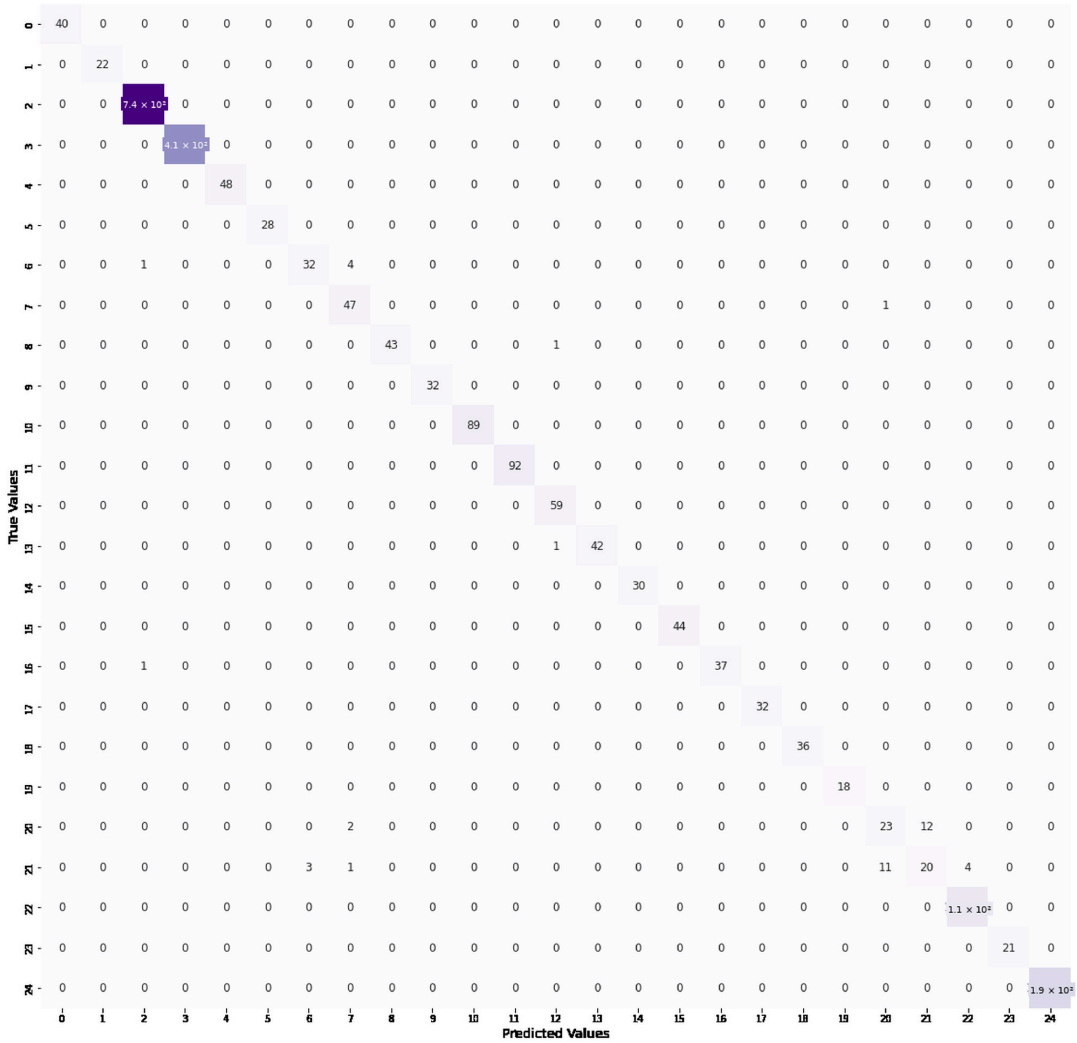
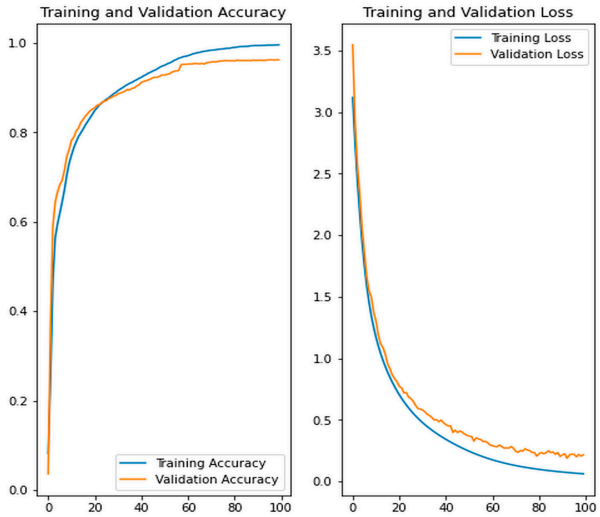


Figure 9. Confusion matrix when the Adagrad optimizer is used in the model.



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

Figure 10. Performance of the model with the AdaDelta optimizer; (**a**) Graph of accuracy; (**b**) Graph of loss during training versus validation.

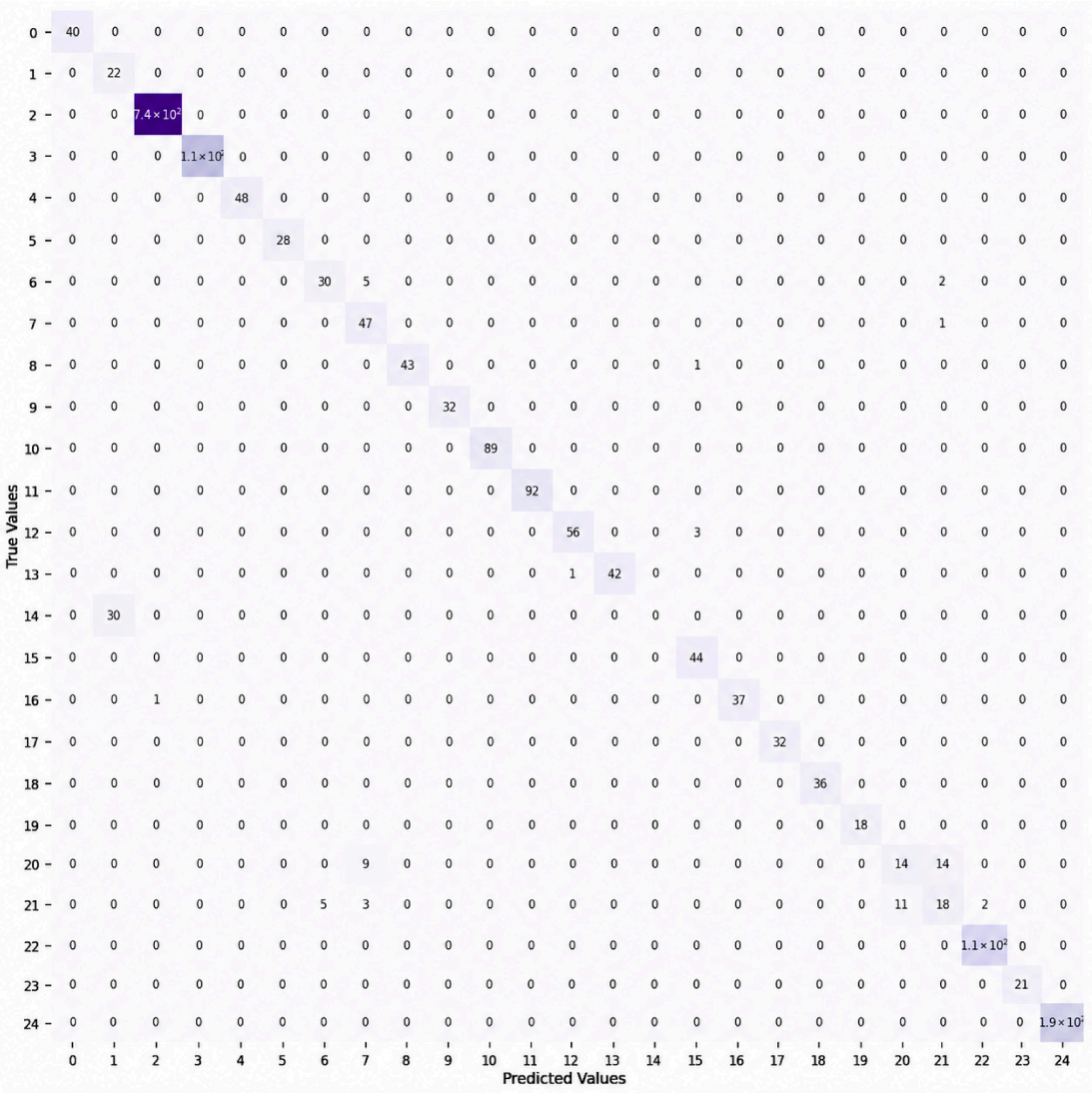
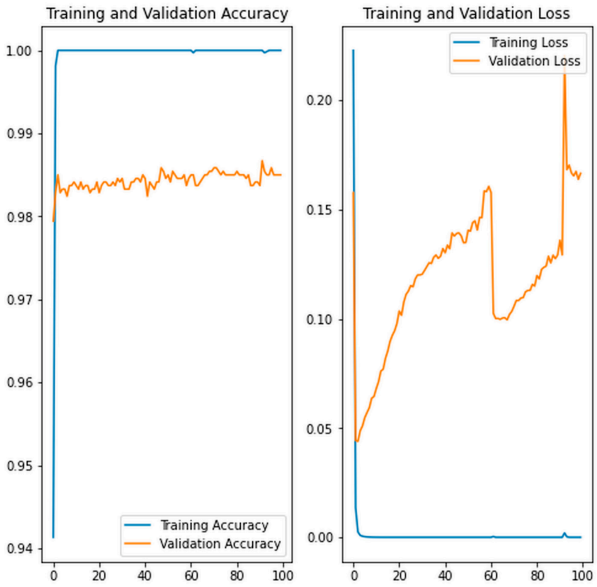


Figure 11. Confusion matrix using the AdaDelta optimizer on the model.



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

Figure 12. Performance of the model with Adamax optimizer; (**a**) Graph of accuracy; (**b**) Graph of loss during training versus validation.

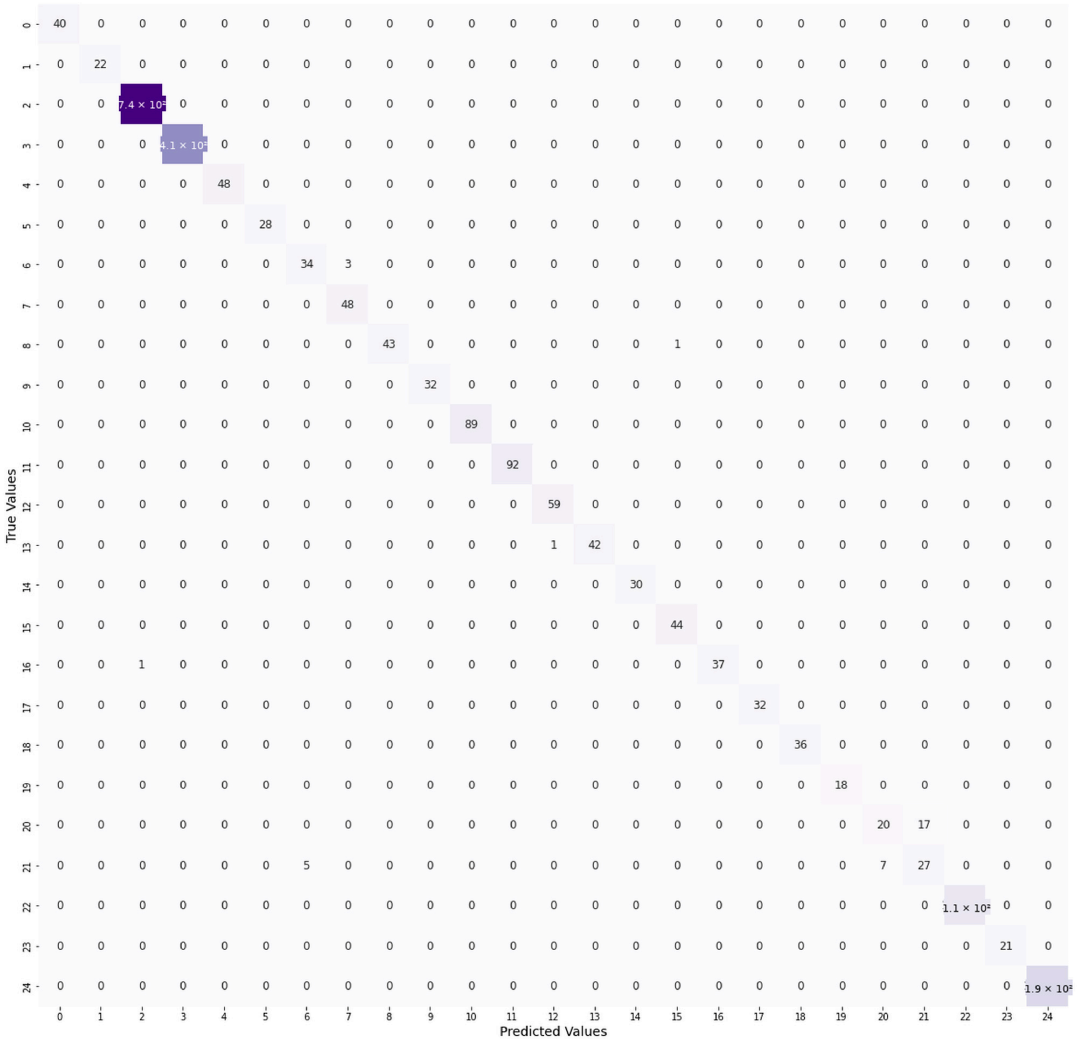


Figure 13. Confusion matrix optimized using Adamax.

4.4. Five-Fold Robustness Evaluation

To assess the robustness of the retained TLECNN configuration across different data partitions, a standalone five-fold stratified group-aware evaluation was performed on the full MalImg dataset. The fold-level results are summarized in Table 5 with the mean ± standard deviation.

Table 5. Five-fold robustness evaluation of the retained TLECNN configuration.

Fold	Accuracy (%)	Macro-Precision (%)	Macro-Recall (%)	Macro-F1 (%)	MCC
1	98.8554	97.3728	97.3219	97.2375	0.986774
2	99.0690	97.3641	97.2625	97.2377	0.989046
3	99.0751	97.7867	97.8153	97.5483	0.989124
4	98.9311	97.1185	97.0180	97.0259	0.987455
5	99.0967	97.6061	97.5614	97.5300	0.989390
Mean ± SD	99.0054 ± 0.1064	97.4497 ± 0.2554	97.3958 ± 0.3039	97.3159 ± 0.2215	0.988358 ± 0.001167

The five-fold evaluation achieved an average accuracy of 99.0054% ± 0.1064% and a macro-F1-score of 97.3159% ± 0.2215%. The small variation across folds suggests the retained configuration performs stably across the evaluated partitions.

4.5. Class-Sensitive and Computational Analysis

The confusion matrix analysis indicates that most malware families are classified with very high accuracy, while the principal errors occur among a small number of visually similar minority families, particularly the Swizzor variants. The macro-averaged results in Table 4 therefore provide a more informative comparison than overall accuracy alone. All five optimizer experiments use the same TLECNN architecture. The retained TLECNN contains 25,711,513 total parameters, of which 17,099,801 are trainable under selective fine-tuning, reducing the number of trainable adaptation parameters by 33.36% compared with full fine-tuning. It requires 7.7554 GFLOPs per image and approximately 102.85 MB FP32 storage, with mean batch-1 inference latency of 10.129 ms on an NVIDIA GPU. Float16 TFLite conversion reduced model size to 51.26 MB and dynamic-range quantization to 26.05 MB.

These results characterize computational requirements; direct deployment and energy consumption on physical IoT devices were not evaluated.

Consequently, optimizer selection does not change the size of the resulting inference model, and the observed performance differences arise from the optimization behaviour rather than differences in model capacity.

4.6. External Multi-Architecture ELF Validation

Using the publicly accessible multi-architecture IoT malware dataset published by Xia et al. [24], an external validation experiment was carried out to assess the suitability of the chosen TLECNN framework beyond the Windows-based MalImg benchmark. The source dataset includes ELF executables from Yokohama National University and VirusShare, and covers multiple processor architectures and IoT malware families. The validation shown here was performed for ARM, MIPS, and an independently verified subset of x86-64.

For the ARM/MIPS experiment, 7771 verified ELF samples were preserved that belong to five classes, namely, benign, Gafgyt, Mirai, Other, and Tsunami. The dataset was partitioned into approximately 70% for training, 10% for validation, and 20% for held-out testing. SHA-256 verification confirmed that no exact duplicates crossed the partitions. The

selected TLECNN configuration was retained, with the output layer adapted to the five external classes.

An additional independent experiment was conducted on the x86-64 ELF subset from the same source collection. In particular, after architecture verification and duplicate auditing, 4571 unique samples from three classes (benign, Gafgyt and Mirai) were retained and split into 70% training, 10% validation, and 20% testing. There was no SHA-256 overlap across partitions. The TLECNN framework was trained independently using the same selected configuration, but by adapting the output layer to the three x86-64 classes.

These external evaluations go beyond MalImg and extend the evaluation to ELF malware from multiple processor architectures evaluated independently. The results on ARM, MIPS, and x86-64 samples, with IoT-relevant families such as Mirai and Gafgyt in particular, further support the conclusion that the chosen TLECNN framework maintains effective classification performance beyond the MalImg dataset. As shown in Figure 14, the ARM/MIPS held-out test evaluation achieved 93.38% accuracy and 89.20% macro-F1-score, while the independently evaluated x86-64 subset achieved 96.94% accuracy and 95.18% macro-F1-score.

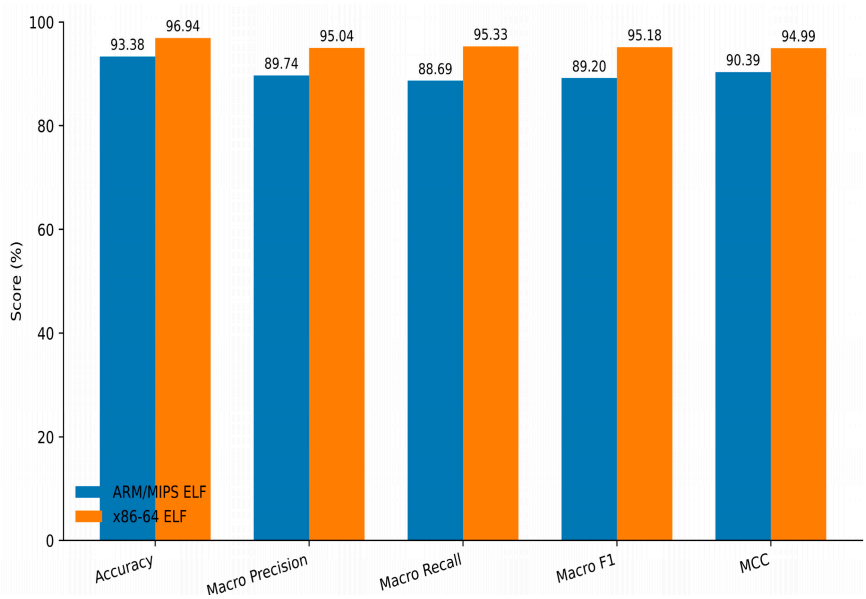


Figure 14. External validation performance of TLECNN across ARM/MIPS and x86-64 ELF malware. MCC is expressed as a percentage for visual comparison.

5. Comparison with Recent Malware Classification Approaches

5.1. Comparison with the ResNet50 Baseline

To ensure a controlled comparison, the ResNet50 baseline and TLECNN were evaluated using identical data partitions, preprocessing, selective fine-tuning policy, random seed, and training conditions. Both models were evaluated on the same 2300 samples. Table 6 therefore presents a matched NAdam comparison of TLECNN and the ResNet50 baseline on the same 2300 held-out samples, with common support across all malware families.

The class-wise analysis further demonstrates the consistency of the proposed approach across the 25 malware families. TLECNN and ResNet50 obtained identical F1-scores for 21 of the 25 families, while higher TLECNN F1-scores were observed for Fakerean, Swizzor.gen!E, Swizzor.gen!I, and Wintrim. BX. In particular, the F1-score increases from 0.9953 to 1 for Fakerean, from 0.8889 to 0.9375 for Swizzor.gen!E, from 0.8857 to 0.9275 for Swizzor.gen!I, and from 0.9744 to 1 for Wintrim. BX. Notably, three of these four families are TrojanDownloader classes, including both Swizzor variants, indicating that

the observed improvement is concentrated in a small subset of malware families rather than uniformly distributed across classes. Overall, the class-wise findings indicate that the proposed approach improves discrimination for some relatively challenging malware families while maintaining consistently high performance across the remaining classes.

Table 6. Class-wise comparison of TLECNN and ResNet50 under the controlled NAdam setting.

Malware Family	Support	TLECNN (NAdam)			ResNet50 (NAdam)		
		Precision	Recall	F1	Precision	Recall	F1
Adialer. C	26	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Agent. FYI	23	0.9583	1.0000	0.9787	0.9583	1.0000	0.9787
Allaple. A	737	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Allaple. L	403	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Alueron.gen!J	65	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Autorun. K	31	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
C2LOP.P	40	0.9286	0.9750	0.9512	0.9286	0.9750	0.9512
C2LOP.gen!g	51	0.9800	0.9608	0.9703	0.9800	0.9608	0.9703
Dialplatform. B	55	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Dontovo. A	35	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Fakerean	106	1.0000	1.0000	1.0000	1.0000	0.9906	0.9953
Instantaccess	99	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Lolyda. AA1	54	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Lolyda. AA2	44	0.9778	1.0000	0.9888	0.9778	1.0000	0.9888
Lolyda. AA3	23	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Lolyda. AT	37	1.0000	0.9730	0.9863	1.0000	0.9730	0.9863
Malex.gen!J	36	1.0000	0.9722	0.9859	1.0000	0.9722	0.9859
Obfuscator. AD	32	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Rbot!gen	51	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Skintrim. N	19	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Swizzor.gen!E	32	0.9375	0.9375	0.9375	0.9032	0.875	0.8889
Swizzor.gen!I	35	0.9412	0.9143	0.9275	0.8857	0.8857	0.8857
VB.AT	93	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Wintrim. BX	19	1.0000	1.0000	1.0000	0.95	1.0000	0.97436
Yuner. A	154	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

5.2. Comparison with Recent Studies

A comparison of models proposed in earlier studies that use machine and deep learning techniques on the same dataset is presented in Table 7 to place the proposed model in the context of recent malware classification research. The comparison focuses primarily on studies that use the MalImg dataset or closely related image-based malware classification approaches.

The results show that TLECNN achieves competitive performance compared with recent malware classification approaches. The accuracy of 99.49% is better than that of some transfer learning- and CNN-based techniques reported from 2020 to 2023, although a few recent architectures achieve slightly higher accuracy. The proposed framework also

maintains high macro-averaged precision, recall, and F1-score on the highly imbalanced MalImg dataset. The studies use different data partitions, balancing strategies, architectures, and evaluation protocols, so direct numerical comparisons should be taken with caution. The contribution of TLECNN therefore does not lie solely in overall accuracy but in combining transfer learning, class-imbalance-aware training, optimizer evaluation, and class-sensitive performance analysis within a common framework.

Table 7. Performance comparison with recent malware classification studies.

No.	Authors/Ref/Year	Model	Accuracy (%)	Precision (%)	RECALL (%)	F1 (%)
1	Vasan et al. [6]/2020	IMCFN	98.82	98.85	98.81	98.75
2	Awan et al. [9]/2021	VGG19	97.62	97.68	97.50	97.20
3	Kumar et al. [10]/2021	MCFT-CNN	99.19	97.72	97.76	97.68
4	Paardekooper et al. [18]/2022	GA CNN	98.50	-	-	95.5
5	Chong et al. [19]/2022	EfficientNet + 1D-CNN Fusion	98.55	97.95	97.98	97.97
6	Dao et al. [17]/2023	MLP	99.34	98.38	98.26	98.29
7	Panda et al. [25]/2023	SE-AGM	99.43	-	-	-
8	Al-Masri et al. [20]/2024	DCMN with Balanced Training	99.89	99.71	99.84	99.77
9	Jeong et al. [21]/2025	Self-MCKD + VGG8	99.57	99.29	99.25	99.25
10	Proposed TLECNN/2026	ResNet50-based TLECNN with NAdam	99.49	98.87	98.73	98.79

Note: Metrics are reported as provided in the respective studies; averaging strategies and experimental protocols may differ.

6. Conclusions

The study outlined a TLECNN, a transfer learning-enhanced convolutional neural network for malware family classification based on images. The proposed framework utilizes previously learned visual representations for malware images and investigates the effect of multiple optimizers on the classification performance. The model was evaluated on the MalImg dataset, which consists of 9339 greyscale malware images from 25 families, with five optimizers compared under a common experimental setting.

Among the evaluated optimizers, NAdam yielded the highest observed performance in the reported comparison, with 99.49% accuracy, 98.87% macro-precision, 98.73% macro-recall, 98.79% macro-F1-score, and an MCC of 0.9940. The class-wise analysis also showed improved recognition of several malware families compared with the ResNet50 baseline. Therefore, the configuration using NAdam achieved high class-sensitive performance, particularly for cybersecurity applications where it is important to correctly identify malicious samples and minimize false negatives. The high macro-recall and F1-score obtained with NAdam are particularly relevant for an imbalanced malware dataset, where reliable recognition of less frequent malware families is important. These results demonstrate the effectiveness of combining transfer learning, class-imbalance-aware analysis, and controlled optimizer selection for multiclass malware classification.

The current evaluation uses the MalImg benchmark and an IoT-specific malware dataset. The external ELF evaluation and resource characterization strengthen the relevance of TLECNN to centralized IoT-oriented malware analysis, while direct deployment on

constrained devices remains outside the scope of this study. Malware evasion techniques, including repackaging, were not evaluated in the present TLECNN experiments and remain an important direction for future IoT-oriented malware analysis [26]. Research can be conducted on model compression, edge deployment, adversarial robustness, and federated learning to improve the practicality and robustness of malware detection systems. Additionally, new AI paradigms, such as large-model-based agents [27], could enable the development of more adaptive and intelligent cybersecurity solutions.

Author Contributions: Conceptualization, T.S. and N.M.; methodology, T.S.; software, T.S.; validation, T.S. and N.M.; formal analysis, T.S.; investigation, T.S.; data curation, T.S.; writing—original draft preparation, T.S.; writing—review and editing, T.S. and N.M.; visualization, T.S.; supervision, N.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The MalImg dataset [16] was published by an independent research lab to evaluate signal and image processing methods in malware analysis. This information can be accessed at https://vision.ece.ucsb.edu/research/signal-processing-malware-analysis, accessed on 16 August 2026. The external multi-architecture IoT malware dataset used for validation is publicly available through the dataset repository reported by Xia et al. [24].

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

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
DL	Deep Learning
IoT	Internet of Things
MCC	Matthews Correlation Coefficient
ReLU	Rectified Linear Unit
RGB	Red, Green, and Blue
TLECNN	Transfer Learning-Enhanced Convolutional Neural Network

References

1. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
2. Albshaier, L.; Budokhi, A.; Aljughaiman, A. A review of security issues when integrating IoT with cloud computing and blockchain. *IEEE Access* **2024**, *12*, 109560–109595. [CrossRef]
3. Caviglione, L.; Choraś, M.; Corona, I.; Janicki, A.; Mazurczyk, W.; Pawlicki, M.; Wasielewska, K. Tight arms race: Overview of current malware threats and trends in their detection. *IEEE Access* **2020**, *9*, 5371–5396. [CrossRef]
4. Ferdous, J.; Islam, R.; Mahboubi, A.; Islam, M.Z. A state-of-the-art review of malware attack trends and defense mechanisms. *IEEE Access* **2023**, *11*, 121118–121141. [CrossRef]
5. Yuan, B.; Wang, J.; Liu, D.; Guo, W.; Wu, P.; Bao, X. Byte-level malware classification based on Markov images and deep learning. *Comput. Secur.* **2020**, *92*, 101740. [CrossRef]
6. 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]
7. Çayır, A.; Ünal, U.; Dağ, H. Random CapsNet forest model for imbalanced malware type classification task. *Comput. Secur.* **2021**, *102*, 102133. [CrossRef]
8. Go, J.H.; Jan, T.; Mohanty, M.; Patel, O.P.; Puthal, D.; Prasad, M. Visualization approach for malware classification with ResNeXt. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–7. [CrossRef]
9. 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]
10. Sudhakar; Kumar, S. MCFT-CNN: Malware classification with fine-tuned convolutional neural networks using traditional and transfer learning in Internet of Things. *Future Gener. Comput. Syst.* **2021**, *125*, 334–351. [CrossRef]

11. Bouchaib, P.; Bouhorma, M. Transfer learning and SMOTE algorithm for image-based malware classification. In Proceedings of the 4th International Conference on Networking, Information Systems & Security, Kenitra, Morocco, 1–2 April 2021; pp. 1–6. [CrossRef]
12. AlGarni, M.D.; AlRoobaea, R.; Almotiri, J.; Ullah, S.S.; Hussain, S.; Umar, F. An efficient convolutional neural network with transfer learning for malware classification. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 4841741. [CrossRef]
13. Alnajim, A.M.; Habib, S.; Islam, M.; Albelaihi, R.; Alabdulatif, A. Mitigating the risks of malware attacks with deep learning techniques. *Electronics* **2023**, *12*, 3166. [CrossRef]
14. Ravi, V.; Alazab, M. Attention-based convolutional neural network deep learning approach for robust malware classification. *Comput. Intell.* **2023**, *39*, 145–168. [CrossRef]
15. Naeem, H.; Ullah, F.; Naeem, M.R.; Khalid, S.; Vasan, D.; Jabbar, S.; Saeed, S. Malware detection in the industrial Internet of Things based on hybrid image visualization and deep learning model. *Ad Hoc Netw.* **2020**, *105*, 102154. [CrossRef]
16. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 July 2011. [CrossRef]
17. Dao, T.V.; Sato, H.; Kubo, M. MLP-Mixer-Autoencoder: A lightweight ensemble architecture for malware classification. *Information* **2023**, *14*, 167. [CrossRef]
18. Paardekooper, C.; Noman, N.; Chiong, R.; Varadharajan, V. Designing deep convolutional neural networks using a genetic algorithm for image-based malware classification. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022; pp. 1–8.
19. Chong, X.; Gao, Y.; Zhang, R.; Liu, J.; Huang, X.; Zhao, J. Classification of Malware Families Based on EfficientNet and 1D-CNN Fusion. *Electronics* **2022**, *11*, 3064. [CrossRef]
20. Al-Masri, B.; Bakir, N.; El-Zaart, A.; Samrouth, K. Dual Convolutional Malware Network (DCMN): An Image-Based Malware Classification Using Dual Convolutional Neural Networks. *Electronics* **2024**, *13*, 3607. [CrossRef]
21. Jeong, H.-J.; Lee, H.-J.; Kim, G.-N.; Choi, S.-H. Self-MCKD: Enhancing the Effectiveness and Efficiency of Knowledge Transfer in Malware Classification. *Electronics* **2025**, *14*, 1077. [CrossRef]
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
23. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [CrossRef]
24. Xia, L.; Chen, Y.; Han, L. A deep learning-based IoT malware detection approach for electric vehicle charging stations. *Sci. Rep.* **2026**, *16*, 10607. [CrossRef] [PubMed]
25. Panda, P.; Om Kumar, C.U.; Marappan, S.; Ma, S.; Manimurugan, S.; Nandi, D.V. Transfer Learning for Image-Based Malware Detection for IoT. *Sensors* **2023**, *23*, 3253. [CrossRef] [PubMed]
26. Chen, X.; Li, C.; Wang, D.; Wen, S.; Zhang, J.; Nepal, S.; Xiang, Y.; Ren, K. Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 987–1001. [CrossRef]
27. Wang, Y.; Pan, Y.; Su, Z.; Deng, Y.; Zhao, Q.; Du, L.; Luan, T.H.; Kang, J.; Niyato, D. Large Model-Based Agents: State-of-the-Art, Cooperation Paradigms, Security and Privacy, and Future Trends. *IEEE Commun. Surv. Tutor.* **2026**, *28*, 1906–1949. [CrossRef]

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