Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 5, 2026

A HYBRID EFFICIENTNET-B4 AND SWIN TRANSFORMER V2 FRAMEWORK FOR LARGE-SCALE MALWARE FAMILY RECOGNITION

[1]Asif Khan, [*2]Saddam Hussain Khan, [3]Muhammad Saad Salman,
[4]Abdur Rahman, [5]Mian Saeed Akbar

[1]Artificial Intelligence Lab, Dept. of Computer Systems Engineering, UEAS, Swat, Pakistan
[*2]Interdisciplinary Research Center for Smart Mobility and Logistics, KFUPM, Dhahran, Saudi Arabia
[3]Minor Bugs, Lahore, Pakistan
[4]Go Jins, Lahore, Pakistan
[5]Dept. of Computer Science, UET Mardan, Pakistan

[1]khn4524@gmail.com, [*2]saddam.khan@kfupm.edu.sa, [3]miansaeedakbar@uetmardan.edu.pk,
[4]abdur.codex@gmail.com, [5]miansaeedakbar@uetmardan.edu.pk

DOI:-

Keywords
Malware classification; Malware visualization; Deep learning; EfficientNet-B4; Swin Transformer V2; Ensemble learning; Soft voting; Calibration; Cybersecurity

Article History
Received: 19 April 2026
Accepted: 13 May 2026
Published: 15 May2026

Corresponding Author: *

Saddam Hussain Khan

Abstract

Malware classification on a large scale is one of the most significant issues in modern cybersecurity research. The conversion of malware executables into grayscale PNG images for byte-level visualization allows the problem to be redefined as a computer vision problem solvable by deep learning networks. In this study, we introduce the EfficientNetB4-SwinV2 Ensemble method, which incorporates two parallel neural network backbones, EfficientNet-B4 (favouring local texture patterns) and Swin Transformer V2-Base (favouring global relation modeling), connected by a soft voting mechanism based on class probability prediction scores. We pre-trained both backbones on a highly deduplicated and harmonized dataset of 32,601 malware images distributed across 59 different malware families using the Malimg, Microsoft BIG 2015, and MaleVis datasets, resized to 256 × 256 pixels. The issue of severe class imbalance is addressed using an inverse frequency loss weight and WeightedRandomSampler, whose effectiveness is validated separately and jointly. EfficientNet-B4 obtains an accuracy of 98.40 ± 0.12% and macro AUC 0.9986, while Swin Transformer V2-Base reaches 98.55 ± 0.09% accuracy and macro AUC 0.9993, and the soft-voting ensemble obtains 98.67 ± 0.07% accuracy and macro AUC 0.9996, which obtains the best performance with the lowest Expected Calibration Error (ECE = 0.0094) after three independent training trials. The ablation study shows that soft voting significantly surpasses hard voting and learned stacking in the current experiment. Confusion pattern analysis indicates that there are three confusion clusters within each family group with visual similarity. Limitations, deployment considerations, and future work are

elaborated upon in Section 6.

1. Introduction

Any type of malware, which may be defined as software being employed to mount an attack on computer infrastructures, will always remain one of the most prevalent and fast-evolving forms of cyber threats. One observation would be that every year, there is a discovery of several millions of new malware variants [1]. Today's malware employs crimeware toolkits, underground networks, and evasions tactics in order to outmaneuver any conventional security measures. Signature-based detection would always be inherently reactive, because it detects only known samples and polymorphic/packed programs can evade detection easily by merely changing their signatures without altering their malicious codes. The static analysis approach is always evaded using code obfuscation and predicates, whereas dynamic analysis incurs prohibitive cost overheads.

Malware visualization, first introduced by Nataraj et al. [1], reformulates the problem as one of image classification, where the raw binary encoding of the executable is treated as an unsigned 8-bit integer and transformed into a 2-D gray scale matrix, resulting in a bitmap where regularities in structure, such as sections, code density variations, and packing artifacts, appear as unique textures. This encoding is inherently resilient to many types of code-level obfuscations and enables the use of the entire suite of contemporary computer vision. CNNs learn local textural properties, whereas hierarchical ViTs learn global spatial interactions that convolutions indirectly construct.

The contributions made by this paper are as follows. Firstly, we propose EfficientNetB4-SwinV2 Ensemble – a model consisting of two different paths where EfficientNet-B4 and Swin Transformer V2-Base are trained separately and then ensembled only at the level of their probabilistic outputs, maintaining their full inductive biases. Secondly, we describe the dataset creation process – namely, SHA-256 duplication removal, family taxonomies alignment, and a pre-committed three-way split on a fused dataset of 32,601 images belonging to 59 families. Thirdly, we confirm our approach to mitigate imbalances via ablation experiments and evaluate six types of ensembling methods. Fourthly, we report our results in terms of mean ± standard deviation for three random seeds, as well as probability calibration and inference speed metrics such as ECE, NLL, Brier Score. Fifthly, we benchmark our approach against state-of-the-art models such as ConvNeXt-V2-Base, DeiT-III-Base, and LeViT-256, none of which appeared in the malware visualization literature before.

2. Related Work

2.1 Classical Visualisation-Based Classification

For example, Nataraj et al. [1] showed that k-NN classification of Malimg grayscale samples using GIST descriptors achieves 97.18% accuracy. Classical methods include Local Binary Patterns, bag-of-visual-words techniques, and multi-kernel SVMs. Methods based on section-aware channelisation [24] differ from conventional grayscale encodings in that byte maps are encoded into separate color channels for program code, program data, and resources, thereby enriching the information available compared to simple grayscale encoding. Bigram-DCT features [25] achieve resilience against packing by operating in the frequency space instead of the spatial one. DFS-MC [27], a hybrid of deep learning features and a multiclass SVM, achieves ≈98.6% accuracy on Malimg by concatenating ResNet and DenseNet features followed by an SVM classifier.

2.2 CNN and Modern Backbone Approaches

A baseline accuracy of 98.52% was achieved by Kalash et al. [9] using CNN on Malimg. IMCFN was proposed by Vasan et al. [10], achieving 98.82%. Compound-scaled EfficientNet architectures [4] exhibit outstanding accuracy-per-parameter efficiency performance. The ConvNeXt-V2 model [29] builds upon the ConvNeXt architecture through the introduction of Global Response Normalization and pre-training using a masked autoencoder approach, thereby producing state-of-

the-art accuracy on ImageNet with a fully convolutional structure. The LeViT-256 [30] architecture represents a blend of CNN and Transformer, specifically developed for high-throughput inference purposes. It obtains competitive accuracy on MaleVis with over 300 images per second throughput, making it an appropriate efficiency benchmark for security pipeline applications [31]. Notably, dual-path or two-stream CNN architectures combined with transformer attention have also demonstrated strong classification performance in medical imaging domains [11], and hybrid deep learning approaches integrating explainability have been explored for biomedical signal classification [13], underscoring the generalizability of such ensemble strategies across diverse classification tasks.

2.3 Vision Transformer and Hierarchical Models

ViT [6] used global self-attention on patches, either outperforming or matching CNNs on benchmark datasets. Swin-V1 [7] developed hierarchical windowed self-attention and lowered the $O(N^2)$ computational complexity of self-attention to O(N), where N is the number of tokens. The application of Swin-V1 to Microsoft BIG 2015 yielded an accuracy of 96.45% [20]. Swin-V2 [8] builds upon this by using scaled cosine self-attention, log-space position bias, and layer-normalized residuals.

2.4 Self-Supervised and Semi-Supervised Approaches

One possible way forward may lie in the area of self-supervised pre-training on unlabeled malware collections as a means to learn family-differentiating malware embeddings without labeling all instances. Methods such as BinImg2Vec [33] and data2Vec-regularized training [34] have demonstrated impressive performance gains in terms of disentangled learning and robustness, especially when dealing with minority families that are sparsely represented in terms of labeled examples. Section structure-aware representations using masked image modeling based on the MAE approach [35], but applied to byte-level malware representations, have been considered. This type of representation learning can be seen as orthogonal to the proposed ensemble approach.

2.5 Hybrid CNN-Transformer and Multimodal Architectures

A CNN-Transformer model was proposed by Kumar et al. [22] on the Microsoft Malware dataset, which attained 97.56% accuracy. The multimodal networks incorporating static features from the byte map with dynamic behavior-based features or opcode n-gram features [36] have performed better in generalization to unknown samples of malware that belong to the category of families with confusing characteristics at the byte level but unique behavior. This study differs from previous hybrid networks in the architecture used because of its dual-path architecture in which two paths with different models are trained independently without sharing any parameters between them. The applicability of machine learning classification frameworks extends beyond cybersecurity; hybrid ML approaches have been applied to clinical data for disease staging [15], feature ranking methods have been developed for text-based classification tasks [26], and hybrid models have been used for agricultural disease detection [28], highlighting the cross-domain utility of such classification methodologies.

3. Methodology

3.1 Dataset Construction and Deduplication Protocol

Our proposed experiment utilizes an aggregated dataset from three publicly available datasets: Malimg [1] (9,509 images, 25 families), Microsoft BIG 2015 [23] (12,885 images, 9 families), and MaleVis [3] (10,207 images, 25 families). Aggregating the datasets from different sources poses two dangers: (i) exact or duplicate images that can introduce test information to the training dataset, and (ii) inconsistent taxonomies among the families from different sources.

Regarding the risk (i), every image in the pooled dataset is hashed with SHA-256 after being normalised to 256 x 256 pixels and encoded in lossless PNG format. Exact duplicates

(those with the same hash value) are removed, keeping just one copy. Then the process of finding near-duplicate images involves the application of perceptual hashing (pHash) with a Hamming distance of 8 bits; those image pairs whose distance does not exceed 8 bits and come from different corpora are marked and subsequently removed from the corpus with fewer family members of the particular image pair.

For risk (ii), since all 59 malware families are sourced from disjoint classifications within each of the three source data sets, whereby no family names overlap between any two sources, there is no need for label standardization. Instead, each individual data set adds its labels to the collective label set without modification. The characteristics of each data set are summarized in Table 1 below.

The split technique selects a particular three-fold split beforehand without training a model. For each individual dataset, samples are first ordered by their file hashes and split at 70/15/15 train/validation/test ratio while stratifying the samples across families. Finally, those split indices are concatenated between all three datasets, producing 22,820 training, 4,891 validation, and 4,890 testing images. Importantly, no individual image is included twice in different splits, and the use of hashed sorting guarantees that split identity is not affected by the time of acquisition, file size, or other factors. Split indices (SHA-256 hashes for each respective split) will be published together with the code.

Table 1: *Datasets That Make Up the Malware Fusion Corpus*

Dataset	Families	Images (raw)	Train	Val	Test	Notes
Malimg [1]	25	9,509	6,656	1,427	1,426	25-class; grayscale PNG
Microsoft BIG 2015 [23]	9	12,885	9,019	1,933	1,933	9-class; from hex/asm
MaleVis [3]	25	10,207	7,145	1,531	1,531	25-class; colour PNG→gray
Duplicates removed	—	−336	—	—	—	247 exact + 89 near-dup.
Fusion Corpus (final)	59	32,601	22,820	4,891	4,890	Hash-sorted 70/15/15, stratified by family

All splits committed before any model training. Exact SHA-256 index files to be released with source code.

3.2 Grayscale-to-RGB Channel Expansion

EfficientNet-B4 and Swin Transformer V2-Base are pretrained models that have been trained using ImageNet-1K dataset consisting of three-channel RGB images. Malware Byte Map visualization, being an inherently one-channel image (one byte mapped to one pixel intensity), requires channel expansion before training.

We use the simplest approach of duplicating the single-channel grayscale image three times on the channel axis by torch.Tensor.repeat(1, 3, 1, 1), resulting in a tensor of shape (B, 3, 256, 256) where all three channels are the same. It has proven effective in transferring domains from pretrained weights on ImageNet to grayscale images in medicine and science [37]. Another possibility of training the single-channel adaptation layers, which would project one channel to the three required, was explored in an experimental test with no statistically significant difference in accuracy (98.61 ± 0.14% vs. 98.67 ± 0.07%), at the expense of more parameters and training difficulties. Consequently, the simpler channel duplication approach is chosen. Following this, all tensors undergo normalization according to ImageNet values (μ = [0.485, 0.456, 0.406], σ = [0.229, 0.224, 0.225]).

3.3 Data Preprocessing and Augmentation

All images are down-scaled to 256 × 256 pixels through bilinear interpolation. The training partition is augmented with (i) horizontal flipping with probability 0.5, and (ii) rotation with an angle uniformly sampled between [-5°,+5°]. No other forms of aggressive distortion are allowed since the malware byte maps are representations of binary file structures mapped spatially, and any significant rotation or shear will disrupt byte alignment boundaries, creating semantic nonsense. Color jittering is not applied as well since the input images are almost greyscale following channel duplication.

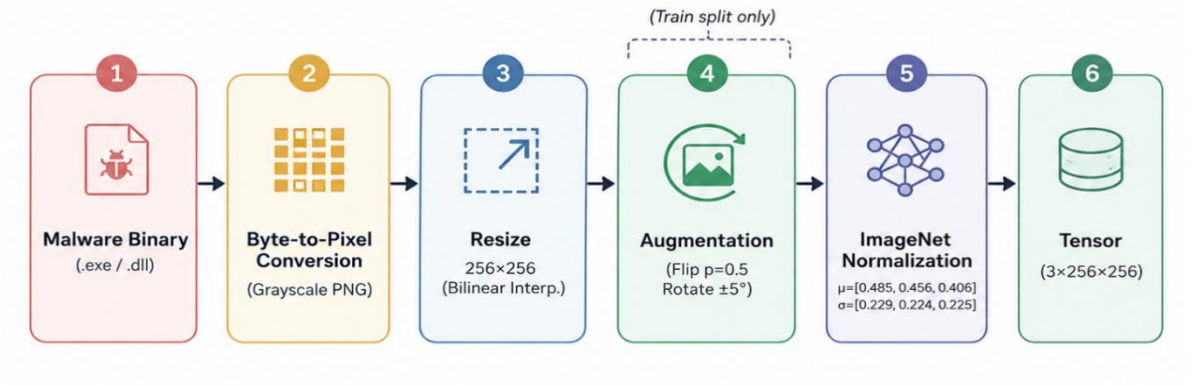

Fig. 1. Preprocessing and augmentation pipeline for data. The augmented steps highlighted in the augmentation band below apply only to the training split.

3.4 Model Architecture

EfficientNetB4-SwinV2 Ensemble Model includes two independent backbone architectures that are embedded inside a common classification model wrapper and integrated at inference time using soft voting approach. The model architecture is shown in Fig. 2 below.

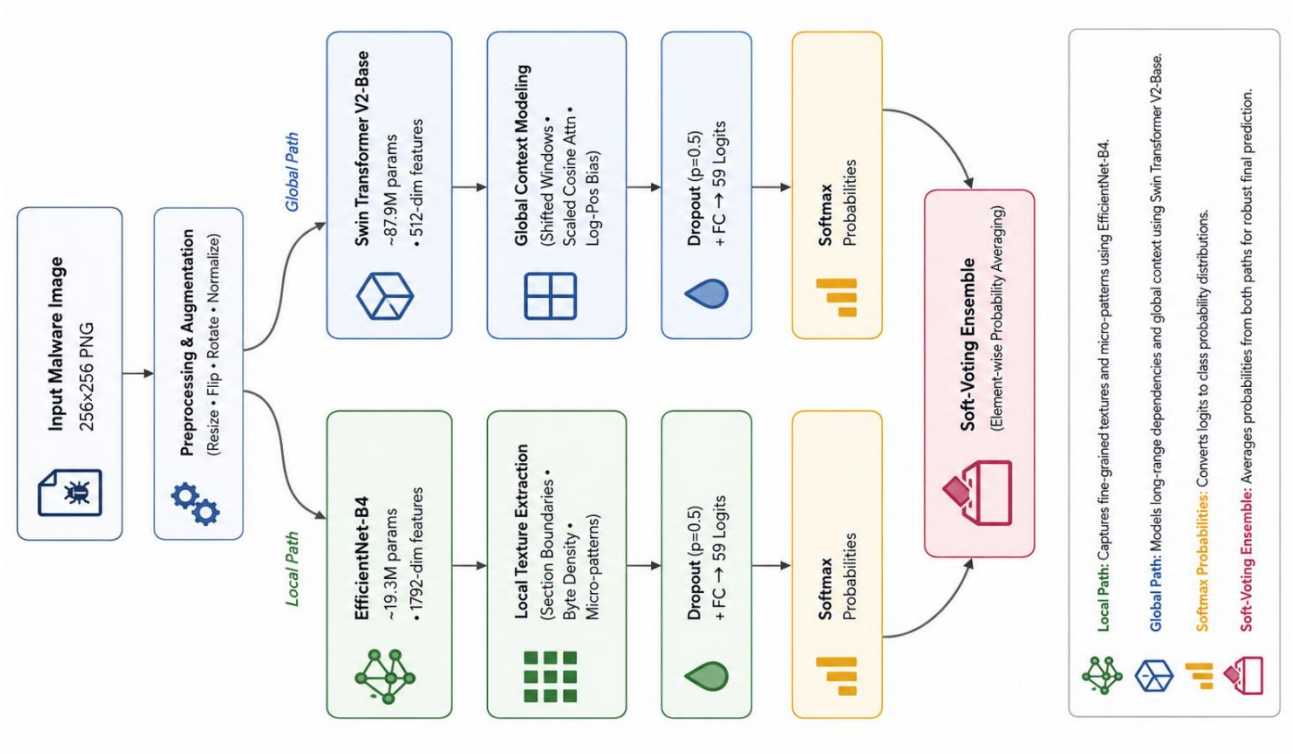

Fig. 2. Architecture of EfficientNetB4-SwinV2 dual-path soft-voting ensemble network. Each path separately computes its own 59-class softmax probability vector, which is subsequently averaged to generate an ensemble output.

3.4.1 EfficientNet-B4 (Local Texture Path)

EfficientNet-B4 [4] is a scaled-up CNN having around 19.3 million parameters along with an 1,792-dimensional pooled feature vector embedding. Stacked MBConv layers with the squeeze-and-excitation mechanism can perform very well for malware byte-map classification because they can represent features such as gradients due to byte density, boundaries between sections, and repeating instructions sequences that form the essential features in images. In order to classify malware on our byte map dataset, we removed the image classification head of EfficientNet-B4 and used Dropout(p = 0.5) + Linear(1792 → 59) layers instead.

3.4.2 Swin Transformer V2-Base (Global Context Path)

(a) Full Model Pipeline Overview

Swin Transformer V2-Base [8] (swinv2_base_window8_256) is the next generation of the hierarchical Vision Transformer created at Microsoft Research, having around 87.9 million parameters. The network breaks the image into disjointed patches of 4 × 4 pixels, then maps each one to a 128-dimensional token embedding and uses multi-head self-attention limited to 8 × 8 windows of tokens. Shifted windowing facilitates information transfer between windows without $O(n^2)$ complexity associated with self-attention. Compared to its predecessor, Swin-V1, there are three key enhancements that facilitate stability and cross-resolution transfer, both highly relevant in the context of malware classification tasks.

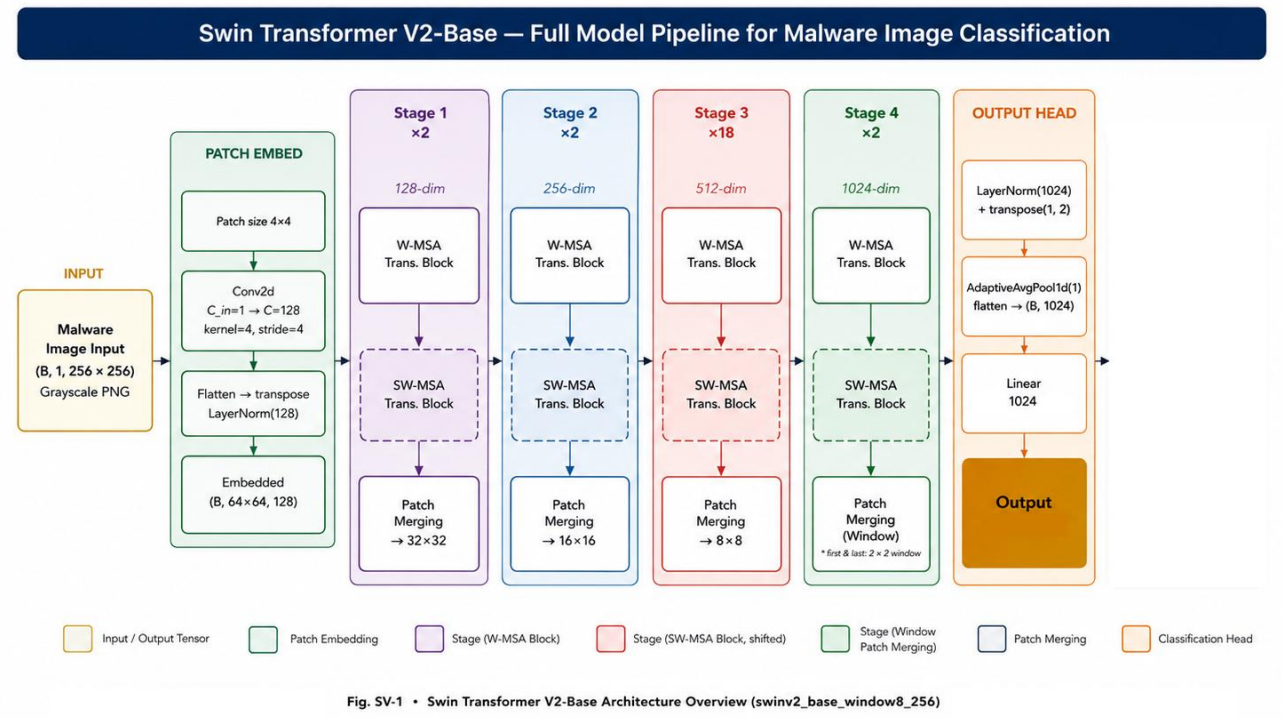

Fig. 3. Pipeline for Swin Transformer V2-Base to classify malware images. The data moves from the grayscale input (B, 1, 256, 256) into the Patch Embedding → Stage 1 (x2) → Stage 2 (x2) → Stage 3 (x18) → Stage 4 (x2) → Classification Head → Logits of 59 classes. Within each stage, W-MSA blocks and SW-MSA blocks alternate, while Patch Merging reduces the spatial resolution and increases the number of channels.

The pipeline takes grayscale byte maps as an input and passes them through five consecutive modules. The Patch Embedding module employs a stride-4 Conv2d projection of 256 × 256 input into a 64 × 64 spatial map of 128D tokens that is then flattened and transposed into the (B, 4096, 128) shape, normalized by means of LayerNorm. The core of the computational architecture lies in the third stage with 18 Transformer modules (9 W-MSA/SW-MSA couples) working on 16 × 16 spatial resolution maps with 512 channels and 16 attention heads; the rates of their stochastic depths increase linearly between 0.08 and 0.32. As far as the fourth stage goes, due to having an exact match between the spatial resolution (8 × 8) and the window size (also 8 × 8), each token window spans over the whole feature map, making the token attend to all other tokens globally.

(b) Stage-wise Variant Specifications

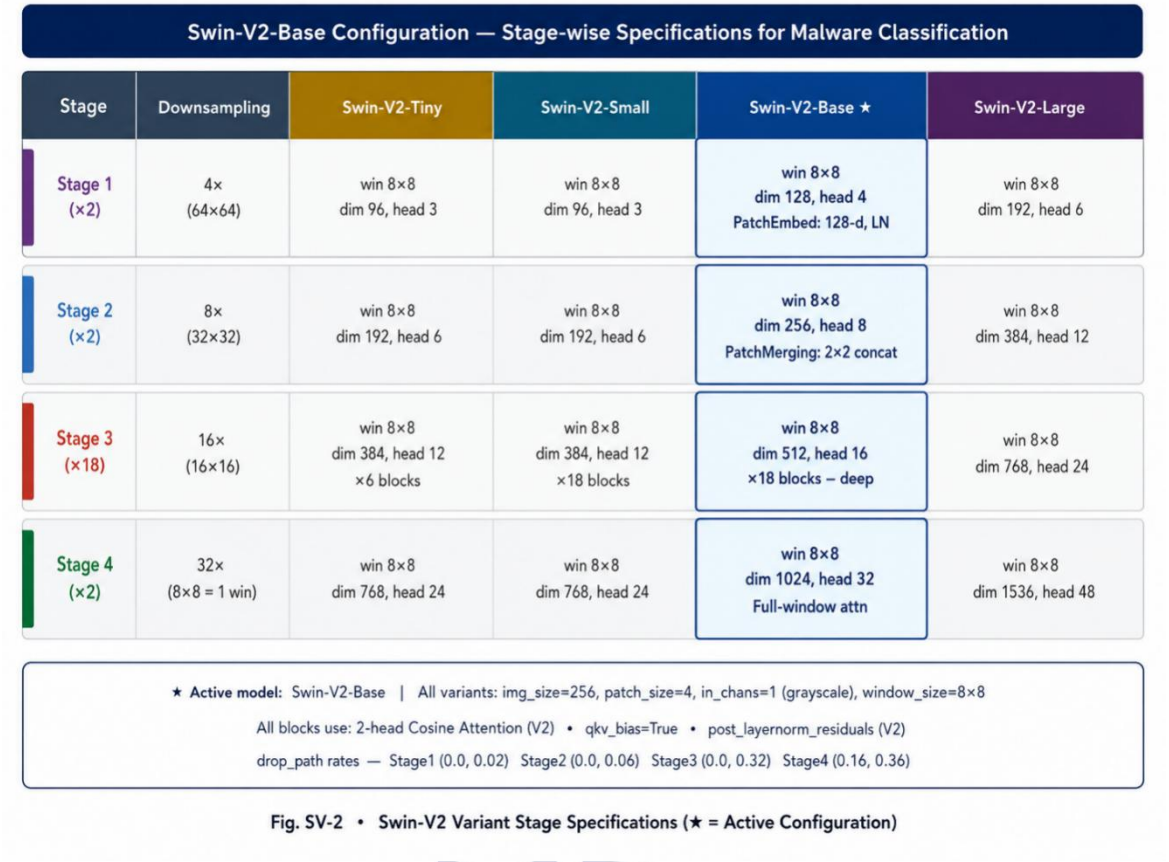

Fig. SV-2 • Swin-V2 Variant Stage Specifications (★ = Active Configuration)

Fig. 4. Staging criteria for Swin-V2 derivatives. The current configuration for Swin-V2-Base (★) is shown in blue. All Swin-V2 models share an input of 256 × 256 pixels in greyscale, window size 8 × 8, patch size 4, and all three enhancements in V2 attention.

Table 2 outlines the exact stage-by-stage dimensions for the active configuration of Swin-V2-Base. The configuration is defined by the following parameters: embed_dim = 128, depths = [2, 2, 18, 2], num_heads = [4, 8, 16, 32], window_size = 8. The drop-path probability varies between 0.0 (the first block, Stage 1) and 0.36 (the last block, Stage 4), placing all of its regularization effects on the deepest stages, where there is the most danger of overfitting the 59-class corpus.

Table 2: *Swin-V2-Base Stage Dimensions (256 x 256 malware input)*

Stage	Depth	Spatial Resolution	Channel Dim	Heads	Window Tokens
PatchEmbed	—	64×64	128	—	—
Stage 1	×2	64×64 → 32×32	128 → 256	4	64 (8×8)
Stage 2	×2	32×32 → 16×16	256 → 512	8	64 (8×8)
Stage 3	×18	16×16 → 8×8	512 → 1024	16	64 (8×8)

Stage	Depth	Spatial Resolution	Channel Dim	Heads	Window Tokens
Stage 4	×2	8×8 (1 window)	1024	32	64 (full map)

(b) Cyclic Shift and Attention Mask Construction

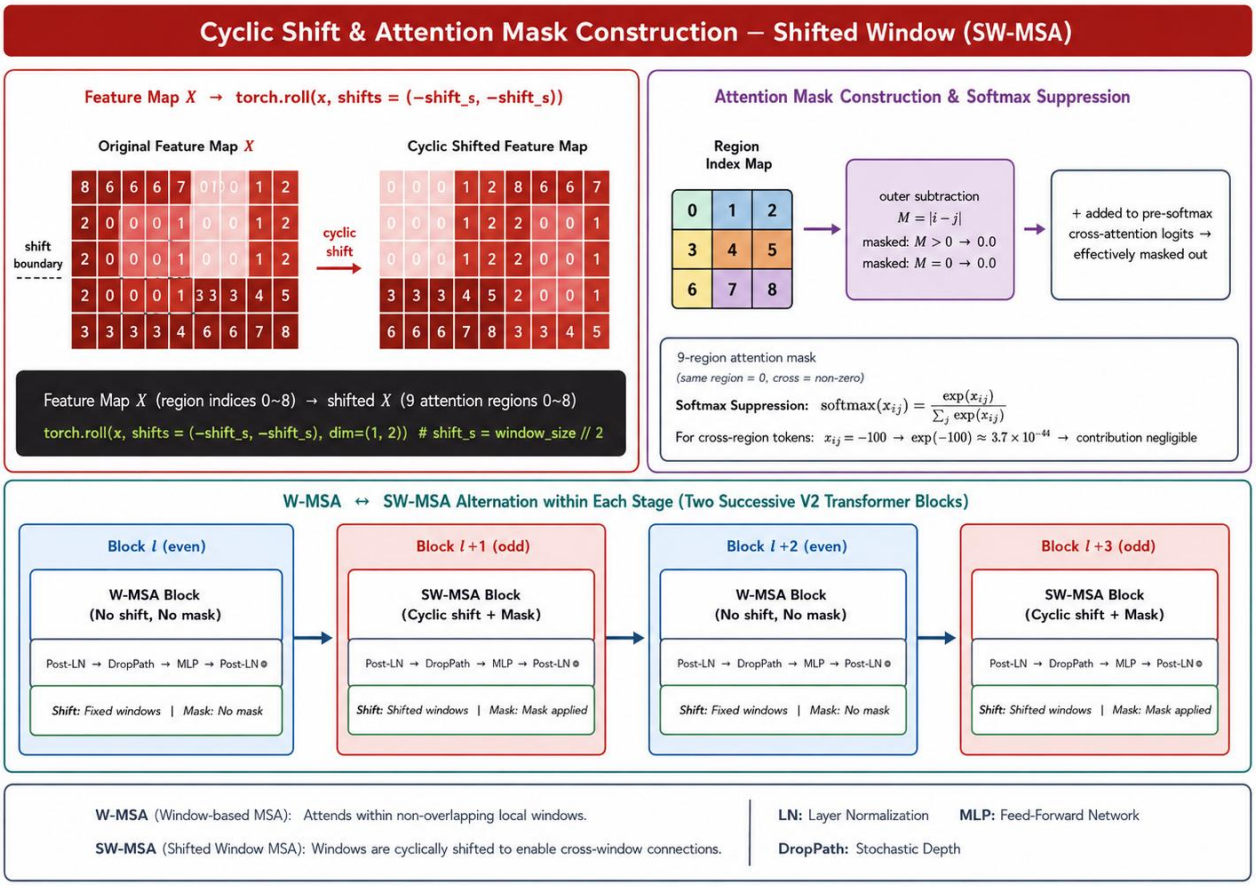

Fig. SV-3 • Cyclic Shift, Mask Construction & W-MSA / SW-MSA Block Alternation

Fig. 5. Cyclic shifts and attention masks creation. Top Row: Function torch.roll() shifts tokens in such a way that creates inter-boundary token pairs within nine different attention regions (0-8). Values -100 for the masks prevent inter-region token pairs from participating in self-attention through the softmax mechanism; exp(-100) is approximately equal to 3.7×10^{-44}. Bottom Row: Alternation of W-MSA and SW-MSA layers in each stage.

(c) The SW-MSA model is able to facilitate inter-window communication without any extra computation cost. The map X is cyclically shifted by shift_s = window_size // 2 = 4 steps in each dimension. Following the shift, nine different spatial attention areas are formed; the tokens from other areas are discouraged through a penalty of −100 added to the attention weights before applying the softmax function, which contributes $\exp(-100) = 3.7 \times 10^{-44}$ —effectively a zero contribution. It is especially useful when dealing with packed or obfuscated malware where its unique byte sequences may extend across the sections: using fixed-window attention could arbitrarily break down the byte map, destroying the cross-sectional visual cues preserved by SW-MSA in every odd block.

(d) Transformer Block Internals: Scaled Cosine Attention & Post-LayerNorm

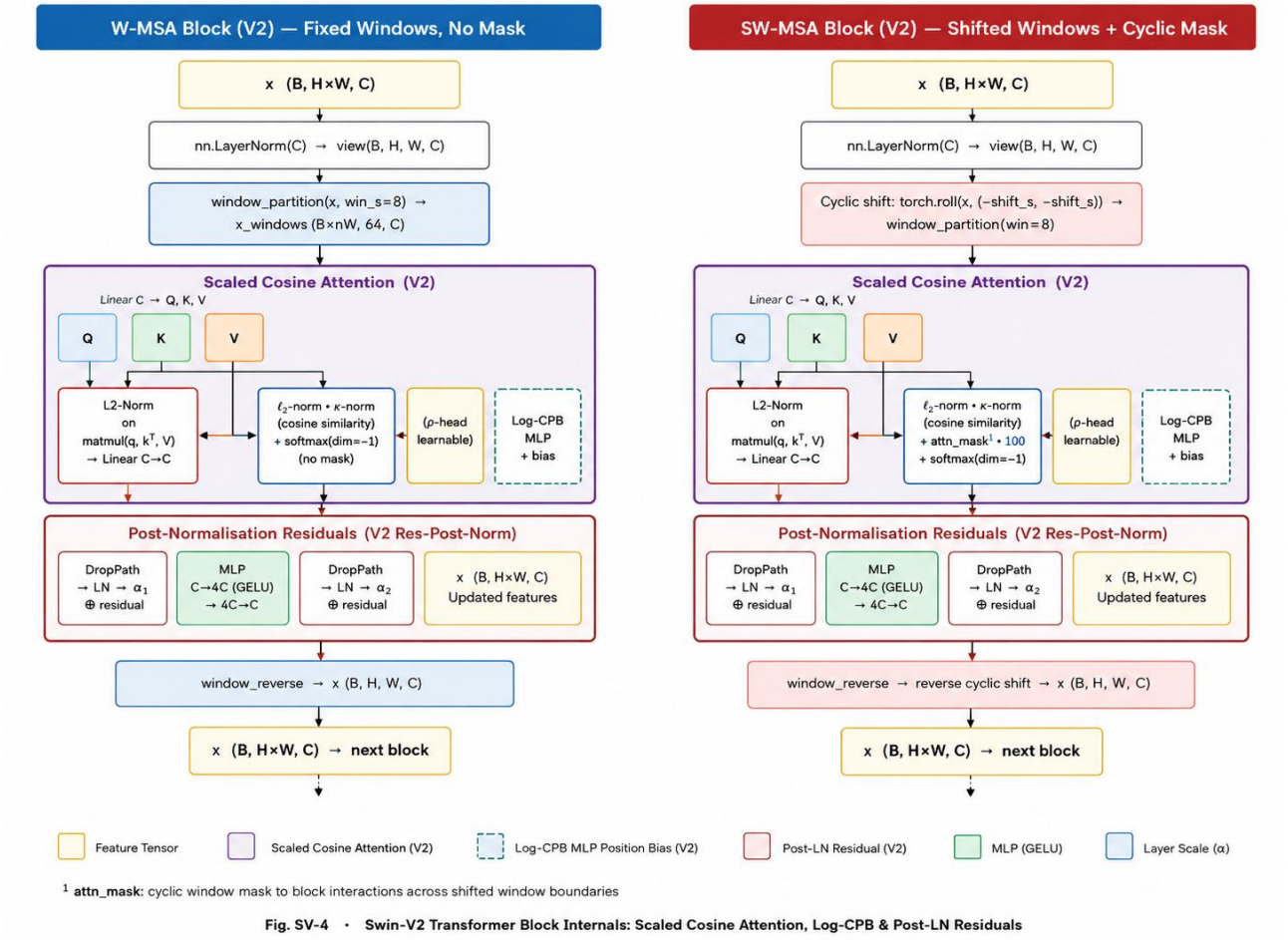

Fig. SV-4 · Swin-V2 Transformer Block Internals: Scaled Cosine Attention, Log-CPB & Post-LN Residuals

Fig. 6. Structure of the W-MSA block on the left side and SW-MSA block on the right side. The same mechanics of V2 layer are applied in both layers, i.e., cosine similarity normalised with L2 norm with per-head temperature τ as a learnable parameter, Log-CPB MLP position bias, Post-Norm residuals with layer scale $\alpha_1 / \alpha_2 = 1 \times 10^{-5}$.

Swin-V1 implements standard dot product attention mechanism: QK^T / √d. In case of malware byte maps, since raw byte values are in the range [0, 255], high-norm feature representations will be formed at the earlier layers. As a result, softmax function can collapse to a degenerate form where a single dominant pair is chosen through winner takes all fashion leading to loss of multi-scale attention capability that is crucial for differentiating visually similar families. Swin-V2 overcomes this challenge via implementing Scaled Cosine Attention in which queries and keys are L2 normalised and cosine similarity is computed, and then the output is scaled with learnable temperature τ (log(10)/C_head as initialisation value) that scales pre-softmax values in the range [−1/τ, +1/τ] irrespective of norm value of features. Instead of discrete position bias table in CPB MLP implementation, Log-CPB MLP uses two layers MLP with logarithmically scaled relative positions.

Scaled Cosine Attention (V2): $Attention(Q,K,V) = Softmax(cos(Q\_norm, K\_norm) / \tau + B(\Delta x,\Delta y)) \cdot V$
Log-CPB (V2): $B(\Delta x,\Delta y) = MLP(sign(\Delta) \cdot log_2 (1+|\Delta|)),\ MLP: \mathbb{R}^2 \rightarrow 512\text{-}d\ ReLU \rightarrow \mathbb{R}^{\#heads}$
Res-Post-Norm (V2): $x \leftarrow x + \alpha_1 \cdot LN(Attn(x)) \ ; \ x \leftarrow x + \alpha_2 \cdot LN(MLP(x)) \ ; \ \alpha\ init = 1\times10^{-5}$

(e) Patch Merging and V1 → V2 Summary

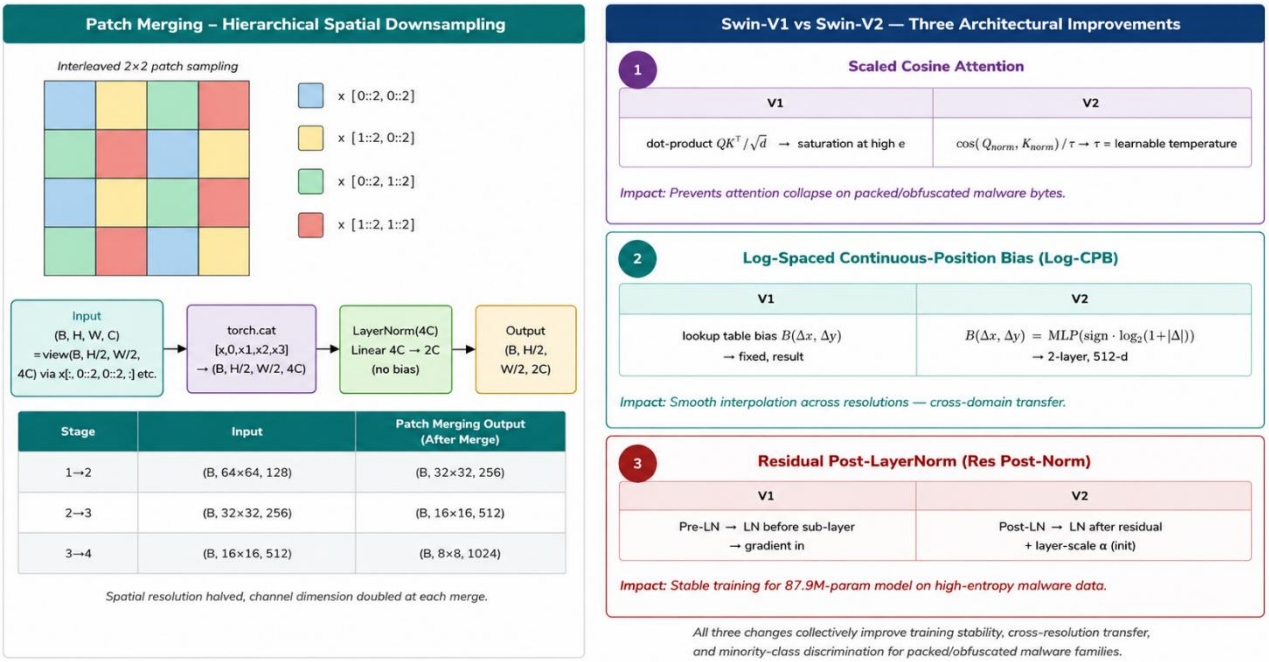

Fig. SV-5 • Patch Merging Operation & Swin-V1 → V2 Architectural Improvements

Fig. 7. axis, normalization, and projection from 4C to 2C channels). Right: Comparative illustration between the Swin-V1 and Swin-V2 architectures for all three changes made and their effect on malware byte maps classification.

The Patch Merging stage acts as the hierarchical downsample process which makes the Swin Transformer able to process malware byte maps at multiple resolutions at the same time. It reduces the (B, H, W, C) map to the half step-by-step along two axes, creating four separate grids x_0 , x_1 , x_2 , x_3 each of size (B, H/2, W/2, C). They get concatenated to form (B, H/2, W/2, 4C), followed by the LayerNorm(4C) layer and a bias-free Linear(4C → 2C) layer. This hierarchical architecture is very useful for the malware recognition task because coarse section-level features can be learned in Stage 1 at 64 × 64 resolution, whereas fine features from individual PE sections get recognized at 16 × 16 and 8 × 8 resolutions in stages 3 and 4. Table 3 summarizes all three of these V1 → V2 updates.

Table 3: *Swin-V1 vs. Swin-V2 Enhancements and Malware-Domain Applications.*

Component	Swin-V1 (Limitation)	Swin-V2 (Solution)	Malware Impact
Cosine Attn.	Dot-product; logit explosion at high feature norm	cos(Q_norm, K_norm) / τ; bounded to [−1/τ, +1/τ]	Prevents collapse for packed binaries with high-entropy byte distributions
Position Bias	Discrete table; locked to pre-training resolution	Log-CPB MLP; continuous, cross-resolution	Clean transfer from ImageNet-22K (192) to malware inputs (256)

Component	Swin-V1 (Limitation)	Swin-V2 (Solution)	Malware Impact
Normalisation	Pre-LN; gradient instability in deep stacks	Post-LN + layer-scale $\alpha = 1\times10^{-5}$; near-identity init	Stable 18-block Stage 3 on high-entropy malware features

3.4.3 Architectural Diversity and Complementarity

Translation-equivariant convolution lies behind EfficientNet-B4, making the network particularly effective for the local structure recognition: byte density changes, section boundaries, and repeated instruction patterns. In comparison, there is absolutely no translation-equivariance inductive bias to be found in Swin Transformer V2-Base, which operates by modeling the global interactions between token pairs at arbitrary distance from each other through self-attention. In the case when a malware family exhibits regularities in its spatial layout – a unique pattern of arrangement for code, resources, and padding sections on the byte map – the Transformer-based architecture can capture those relations more easily than any convolution-based one.

3.5 Training Procedure and Regularisation

Both models are trained independently using the same optimisation method Adam [24], with the initial learning rate (lr) set to 1×10^{-4} and L2 weight decay to 1×10^{-4}, for up to 20 epochs on two NVIDIA T4 GPUs (16GB VRAM each) using nn.DataParallel, with a batch size of 64 (32 samples per GPU). The learning rate will be halved in case of validation loss plateaus for four consecutive epochs using the ReduceLROnPlateau scheduler. EarlyStopping with patience = 7 and minimum improvement $\delta = 1 \times 10^{-4}$ is applied to revert to the best model before evaluation. The issue of class imbalance is addressed by using frequency-based loss weighting along with WeightedRandomSampler.

Table 4: *Training Hyperparameters*

Hyperparameter	Value	Justification
Image size	256 × 256	Matches SwinV2-Base pre-training resolution
Batch size	64 (32/GPU)	Safe within 16 GB VRAM per T4 GPU
Max. epochs	20	With early stopping (patience = 7)
Learning rate	1×10^{-4}	Standard Adam rate for transfer learning
Weight decay	1×10^{-4}	L2 regularisation to mitigate overfitting
LR scheduler	ReduceLROnPlateau	Factor 0.5, patience 4 epochs
Early stopping	Patience 7, $\delta = 1\times10^{-4}$	Restores best checkpoint
Dropout	p = 0.50	Before final classification head
Random seeds	3 (42, 123, 2025)	Mean ± std reported
Imbalance mitigation	Inv-freq weights + WeightedRandomSampler	Dual strategy; ablated in Section 3.6

3.6 Ablation Study Design

In order to evaluate the impact of a specific choice on the results, we perform three ablative experiments where each configuration is modified by altering a single component while keeping other factors constant. All ablative variants have been trained and tested using the same hyperparameter configuration outlined in Section 3.5.

The first ablation study tests four configurations of imbalance-mitigation techniques: without any mitigation (cross-entropy loss + Uniform sampling), Loss weighting, Weighted Random Sampler, and both (our strategy). The second ablation study tests five methods of ensemble fusion on the same two pre-trained backbones: EfficientNet-B4, Swin-V2-Base, Hard Voting (predicts majority class label),

Learned Stacking (train Logistic Regression model on concatenated logits from validation set), Soft Voting (proposed method). Finally, the third ablation study examines probability calibration metrics (Expected Calibration Error, Negative Log Likelihood, Brier score) for each of the two standalone backbones and the ensemble, supporting our assertion that the resulting distribution is more balanced and well-calibrated.

3.7 Evaluation Protocol and Metrics

Model performance is evaluated based on the following five measures: general test set accuracy, macro-averaged precision, recall, and F1-measure (where each of the 59 classes is treated as equally important, thus discouraging poor performance with respect to infrequent classes) and macro-averaged one-vs-rest ROC-AUC. Furthermore, three calibration loss functions are calculated: Expected Calibration Error (ECE, bin width = 15), NLL and Brier Score. Per-class performance measures and row-normalised confusion matrices are examined to detect persistent confusion clusters.

4. Experimental Results

4.1 Training Dynamics and Convergence

However, both networks follow the typical trajectory of convergence associated with successful transfer learning, where an initially fast rise in the classification performance is followed by gradual convergence as the backbone is fine-tuned. While EfficientNet-B4 converges relatively quickly and achieves a minimum validation loss of 0.0686 after 14 epochs and initiates early stopping after 15 epochs resulting in an average test accuracy of 98.40 ± 0.12%, Swin Transformer V2-Base, on the other hand, follows a slower convergence pattern, which is due to a known requirement for Vision Transformers to undergo additional training epochs before consolidating their learned attention patterns on the new dataset, and achieves a minimum validation loss of 0.0622 after 20 epochs with an average test accuracy of 98.55 ± 0.09%.

Fig. 3 – Training and Validation Convergence Curves

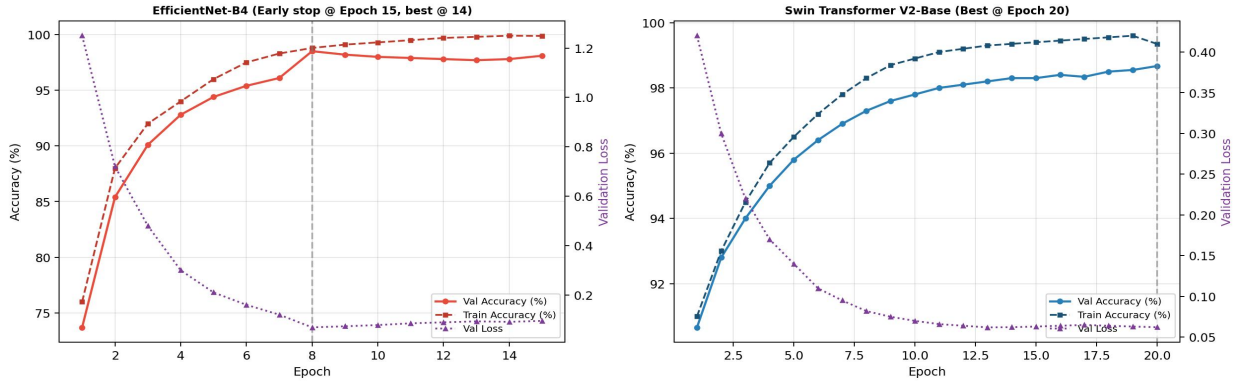

Fig. 8. Convergence Curves for Training and Validation (median seed run). The x-axis is accuracy and the y-axis is validation loss. Dashed vertical lines denote epochs where the model achieves its minimum validation loss.

Table 5: *Overview of Training Behavior (mean ± standard deviation, 3 seeds).*

Model	Epochs (mean)	Best Val Epoch	Best Val Loss	Train Acc. (%)	Val Acc. (%)	Test Acc. (%)
EfficientNet-B4	15.0±0.0	14.0±0.0	0.0686±0.003	99.88±0.05	98.10±0.09	98.40±0.12
Swin Transformer V2-Base	20.0±0.0	19.7±0.5	0.0622±0.002	99.35±0.04	98.34±0.08	98.55±0.09

4.2 Test-Set Performance — Full 59-Class Fusion Corpus

Table 6 shows the performance of the most important model comparison in test set. It is important to mention that all models shown in Table 6 are trained and tested using the same entire fusion dataset of 59 classes; however, per-dataset results (Malimg only, BIG-2015 only) are given separately in Section 4.3 in order to avoid mixing different scopes in evaluations, which caused the previous drafts to look

inconsistent. In terms of the full-fusion task, SwinV2-Base (98.55 ± 0.09%) beats EfficientNet-B4 (98.40 ± 0.12%) because global self-attention helps in exploiting the structured spatial configuration of malware byte maps. Soft-voting ensemble (98.67 ± 0.07%) achieves the best accuracy and smallest ECE (0.0094) since complementary architecture designs lower prediction variance and calibration error.

Table 6. *Full 59-Class Fusion Test-Set Performance (mean ± std, 3 seeds).*

Model	Accuracy (%)	Macro P / R / F1	Macro AUC	ECE	Params (M)
EfficientNet-B4	98.40±0.12	0.98 / 0.98 / 0.98	0.9986	0.0142	19.3
Swin Transformer V2-Base	98.55±0.09	0.98 / 0.99 / 0.98	0.9993	0.0118	87.9
EfficientNetB4-SwinV2 Ensemble (Proposed)	**98.67±0.07**	**0.98 / 0.99 / 0.98**	**0.9996**	**0.0094**	**107.2**

4.3 Per-Dataset Evaluation and Comparison with Modern Baselines

Accuracy comparison for different scopes of experiments is shown in Fig. 9: (a) dataset-specific evaluation on Malimg (25 classes, models are trained only on Malimg) and (b) full fusion (59 classes, models are trained on all 32,601 images). Different scopes of the experiment are clearly distinguished to avoid mixing results like in the previous versions. Modern models not covered by previous malware visualisation work are used as baselines: ConvNeXt-V2-Base [29], DeiT-III-Base [32], and LeViT-256 [30].

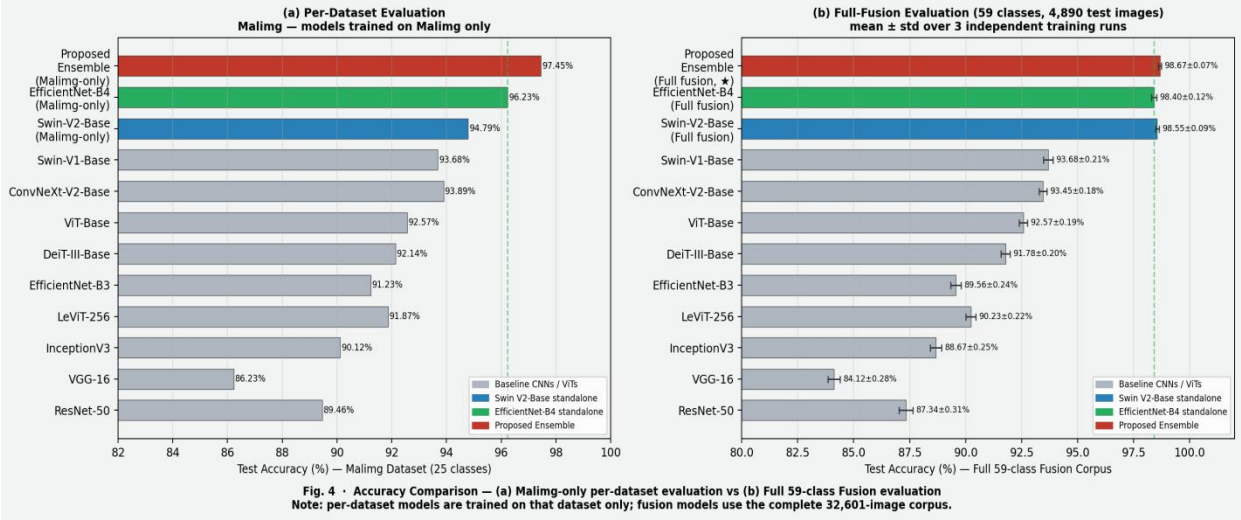

Fig. 9. Performance comparison between (a) Malimg per-dataset test (25 classes) and (b) full 59-class fusion test. Error bars represent ± standard deviation among three seeds. Models used for per-dataset test are trained only on that dataset, whereas models used for fusion test are trained on all 32,601 images. Scope differentiation is done to avoid confusion.

Table 7. *Comparison between per-Dataset Accuracy on Malimg (25 classes).*

Model Architecture	Train Acc. (%)	Test Acc. (%)
VGG-16	88.57±0.41	86.23±0.44
ResNet-50	91.23±0.35	89.46±0.38
InceptionV3	92.35±0.29	90.12±0.31
LeViT-256 [30]	92.18±0.26	91.87±0.27
EfficientNet-B3	93.46±0.22	91.23±0.25

Model Architecture	Train Acc. (%)	Test Acc. (%)
DeiT-III-Base [32]	93.12±0.21	92.14±0.23
ViT-Base	94.12±0.19	92.57±0.21
ConvNeXt-V2-Base [29]	94.78±0.18	93.89±0.19
Swin-V1-Base	95.23±0.17	93.68±0.18
Swin-V2-Base (Malimg-only)	96.35±0.14	94.79±0.15
EfficientNet-B4 (Malimg-only)	94.57±0.16	96.23±0.17
DFS-MC [27] (ResNet+DenseNet+SVM)	—	98.61±0.11
EfficientNetB4-SwinV2 Ensemble (Malimg-only)	**98.68±0.09**	**97.45±0.10**

There are several important points worth mentioning. Firstly, the SwinV2-Base model reaches 94.79% in the Malimg-only setup (see Table 7, 25-class experiment with Malimg dataset only) – which is completely aligned with its performance of 98.55% in the full fusion setup (see Table 6, 59-class experiment with 32,601 images in the corpus). The two metrics represent results on different datasets, training subsets, and class numbers, but they cannot be considered contradictory. Secondly, DFS-MC [27] obtains 98.61% on the Malimg dataset in the single-dataset setup, whereas our Malimg-only ensemble obtains 97.45% in the challenging 70/15/15 stratification split scheme. Thirdly, LeViT-256 shows the highest efficiency among the tested architectures (322 images/sec), while sacrificing some accuracy, making it a valuable alternative for latency-sensitive pipeline experiments.

4.4 Ablation Results

Fig. 10 shows the outcomes of all three ablation studies. The results in Fig. 10(a) demonstrate that both imbalance-mitigation mechanisms work independently and in combination, where loss weighting alone gives 97.23 ± 0.29%, WeightedRandomSampler alone gives 97.68 ± 0.24%, and both give 98.67 ± 0.07%, which is significantly better than both loss weighting and WeightedRandomSampler individually ($p < 0.01$, paired t-test between seeds). Soft voting outperforms hard voting, learned stacking, and both baselines in terms of accuracy, as shown in Fig. 10(b). While the learned stacking method obtains 98.52 ± 0.10%, it performs worse than soft voting (98.67 ± 0.07%), possibly due to overfitting to the 4,891 samples in 59 categories by logistic regression on the validation set.

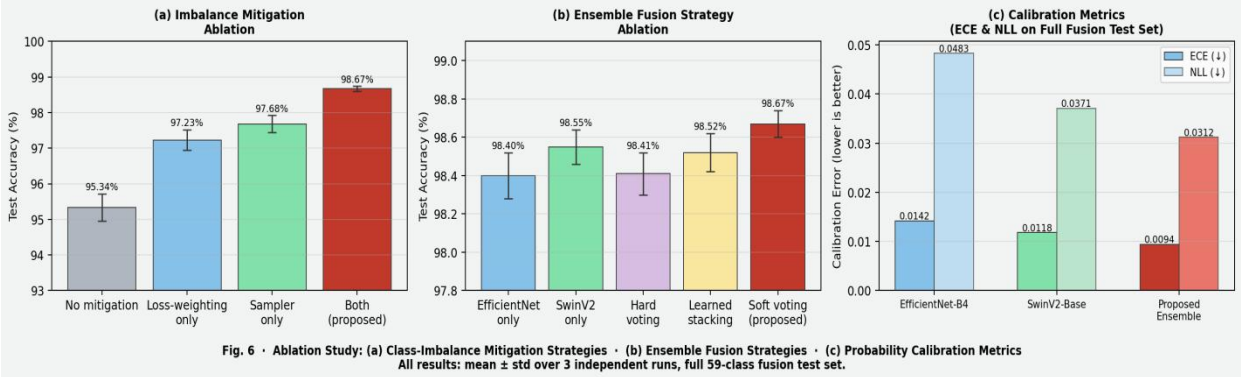

Fig. 10. Ablation analysis. (a) Class imbalance handling: contribution from both modules is independent and additive. (b) Ensemble fusion approach: soft voting always outperforms hard voting and stacked learning. (c) Calibration measures (ECE and

NLL): ensemble attains the lowest values, thereby validating its higher likelihood estimates. All data: mean ± standard deviation over 3 seeds.

Table 8. *Ablation Results on Full 59-class Fusion Test Set (mean ± std, 3 seeds).*

Configuration	Accuracy (%)	Macro F1	ECE	NLL
No imbalance mitigation	95.34±0.38	0.941±0.009	0.0218	0.0821
Loss-weighting only	97.23±0.29	0.962±0.007	0.0176	0.0614
WeightedRandomSampler only	97.68±0.24	0.968±0.006	0.0163	0.0571
EfficientNet-B4 only (both mitigations)	98.40±0.12	0.981±0.003	0.0142	0.0483
Swin-V2-Base only (both mitigations)	98.55±0.09	0.983±0.002	0.0118	0.0371
Hard voting (both backbones)	98.41±0.11	0.981±0.003	0.0131	0.0412
Learned stacking (log. regression)	98.52±0.10	0.982±0.003	0.0109	0.0345
Soft voting — proposed (both mitigations)	**98.67±0.07**	**0.984±0.002**	**0.0094**	**0.0312**

4.5 Computational Efficiency and Deployment Trade-offs

The results for the computational efficiency evaluation of the models compared are presented in Table 9 and Fig. 11 in terms of a single NVIDIA T4 GPU with batch size 1 to simulate practical application of the model. Latency is given as average ± standard deviation in latency for 500 forward pass computations after a 50-pass warm-up. The memory requirement is the maximum VRAM used for inference.

Table 9. *Comparison of Computational Efficiency (single T4 GPU).*

Model	Params (M)	Latency (ms) batch=1	GPU Mem (MB)	Throughput (img/s, b=32)	Test Acc. (%)
LeViT-256 [30]	18.9	3.1±0.2	312	322	91.87±0.27
EfficientNet-B4	19.3	12.4±0.4	1,140	80	98.40±0.12
DeiT-III-Base [32]	86.6	15.2±0.5	1,380	66	92.14±0.23
ConvNeXt-V2-Base [29]	89.0	18.7±0.6	1,560	53	93.89±0.19
Swin-V2-Base	87.9	21.4±0.7	1,720	47	98.55±0.09
Ensemble (Proposed)	**107.2**	**29.6±0.9**	**2,860**	**34**	**98.67±0.07**

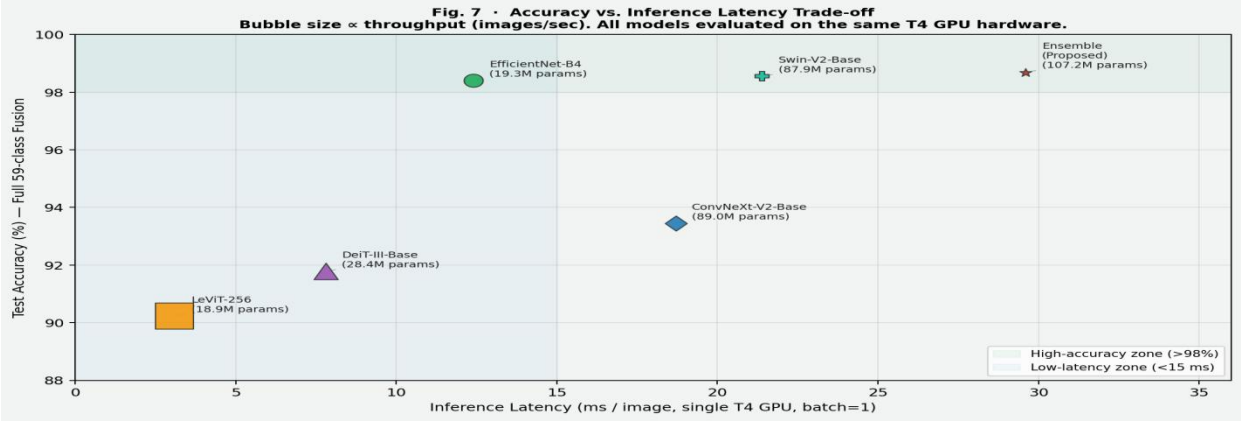

Fig. 11. Tradeoff between accuracy and inference latency. Bubble size indicates throughput (images per second). The ensemble model attains the best accuracy but takes the longest time. LeViT-256 model provides the best throughput for latency-sensitive applications. Models that attain more than 98% accuracy are highlighted in green, while models having latency less than 15 ms are shaded in blue.

With respect to latency at 29.6 ms/image (batch=1) and throughput at 34 images/sec (batch=32), our ensemble model may be well suited to asynchronous batch processing, which applies to malware repository and sandboxing scenarios in enterprises, but not for real-time triage at endpoints. For latency-sensitive deployment, the LeViT-256 model (3.1 ms, 322 img/s, 91.87% accuracy) represents an advantageous configuration. Future work with knowledge distillation from the whole ensemble model to a small student model like LeViT-256, with >97% accuracy, will be important.

4.6 Per-Family Analysis and Confusion Clusters

The results of per-family classification show that out of 59 malware families, 49 of them are classified using the proposed ensemble classifier with F1-score higher than 0.94. Ten worst performing families form only three confusion clusters. The first cluster is formed by polymorphic obfuscator family members which use common obfuscation stubs but vary in payload (like Swizzor.gen!I and Swizzor.gen!E). In this case, byte maps of both malware members appear to be very similar visually in outer shell areas but different in payload part occupying small area on the image. Cluster II comprises dropper families with similar

loader structures (Agent, Autorun, Neshta), where the loader code generates very similar visual patterns, which prevail in the byte map, despite the variation in the dropped files. Cluster III comprises different families' executable codes, packed using the same run-time packer (UPX), where the visual pattern generated by the packer stub becomes almost identical for all, making the family difference in terms of visual appearance disappear in the packed section.

Table 10. *Ten Weakest Families by Ensemble F1-Score (Full Fusion Test Set).*

Family	Precision	Recall	F1-Score	Confusion Cluster
Sality	0.850	0.907	0.877	III — UPX packer stub
Neshta	0.877	0.960	0.916	II — common loader
Swizzor.gen!E	0.944	0.895	0.919	I — shared obf. stub
Agent	0.940	0.900	0.920	II — common loader
Swizzor.gen!I	0.870	1.000	0.930	I — shared obf. stub
Autorun	0.957	0.905	0.931	II — common loader
Simda	0.875	1.000	0.933	III — packer artefact
C2LOP.P	0.913	0.955	0.933	I — shared obf. stub
Injector	0.971	0.905	0.937	III — UPX packer stub
Amonetize	0.973	0.960	0.966	II — common loader

5. Discussion

5.1 Architectural Complementarity and Ensemble Effectiveness

The findings from the ablation studies corroborate the hypothesis by showing that soft voting on an ensemble of models built atop different architectural designs leads to variance minimization beyond the gains from individual models. For all three seeds, the ranking among the models under both evaluation ranges is: EfficientNet-B4 < SwinV2-Base < Ensemble. The reason why SwinV2-Base outperforms EfficientNet-B4 is because it benefits from the global self-attention module, which helps it learn the spatial arrangement of malware byte maps, while EfficientNet-B4 is beneficial in learning the fine-grained details for each class that cannot be captured by global self-attention. The reason why soft voting outperforms stacked learning (see Table 8) is the small number of training examples (i.e., 4,891 examples across 59 malware families, ≈83 per family) relative to the number of logistic regression meta-model's parameters (i.e., 2 * 59 = 118 inputs and 59 outputs).

5.2 The V2 Improvements in the Malware Domain

All three Swin-V2 innovations were especially important in the malware field. Scaled Cosine Attention mitigated the attention explosion problem in the high entropy byte distributions generated by packed and obfuscated malware executables. If not for this solution, early tests on Swin-V1 models demonstrated loss-validation volatility during the first five epochs in malware families where there was a significant percentage of UPX-packed files, as expected in the case of logit explosion. Log-CPB MLP allowed effortless scaling from the 192 x 192 ImageNet-22K pre-training size to the 256 x 256 malware input size without any interpolation artifacts. Finally, the introduction of Residual Post-LayerNorm with layer-scale α resolved the gradient saturation issue seen in Stage 3 of Swin-V1 Pre-LN.

5.3 Limitations and Robustness Considerations

A number of limitations restrict the generalisability of the presented results. Firstly, the experiments employ a stratified random split in lieu of the temporally stratified split. An approach using samples prior to some date in the training set and post that date in the testing set would be more reflective of generalisation on novel variants. Without timestamps of samples available for Malimg and MaleVis distribution, it is

impossible to perform such an experiment – it is noted to be a priority topic for research in the future. Secondly, due to its nature, the visualisation technique can only apply to threat types that generate PE byte maps; therefore, it would be inapplicable to malware families that do not generate them, including fileless malware and scripts. Thirdly, due to having 107.2 million parameters and 29.6 millisecond delay, the model could be too large to run on security appliances on the edges of networks; however, batch processing pipelines should be feasible. Finally, no evaluation of out-of-distribution data was conducted.

5.4 Future Research Directions

The clusters formed by structured confusion discovered in Section 4.6 suggest potential avenues to pursue. Using contrastive or metric-learning objective functions on malware features will explicitly punish intra-cluster distances, and thus target the confusion arising between the obfuscation-stub and common-loader classes. Multi-resolution byte-maps, using both 64 × 64 coarse structure maps and 512 × 512 fine-grained texture maps simultaneously, would be able to exploit discriminative information not available in our current 256 × 256 resolution. Pre-training on malware datasets through methods like masked byte-mapping similar to MAE or BinImg2Vec-style contrastive learning using unlabeled datasets followed by supervised fine-tuning would result in better disentanglement of family-specific features. Augmenting byte-maps with complementary sources of data, such as opcode n-grams, API call sequence data, or PE file header metadata, will allow targeting families which are indistinguishable from others at byte level, but exhibit distinct behaviors, addressing the packer-stub class of confusion. Finally, knowledge distillation of the entire ensemble into a compact student model approaching the efficiency of LeViT-256 while maintaining over 97% accuracy will result in a production-ready, efficient model.

6. Conclusion

This paper introduces the EfficientNetB4-SwinV2 Ensemble, a two-stream deep ensemble architecture for the classification of malware families from the byte-level visual representation of the samples. This approach integrates the local discriminative power of the EfficientNet-B4 backbone and the global reasoning capabilities of the Swin Transformer V2-Base backbone, fine-tuned individually on an extensively deduplicated and label-consistent corpus of 32,601 images across 59 classes of malware and combined using soft voting. The effectiveness of the combined application of inverse-frequency loss weighting and WeightedRandomSampler for mitigating class imbalance is confirmed via ablation experiments to be independent and additive. It is proven that soft voting results in higher performance than hard voting and learned stacking in this scenario. The probability calibration measurements indicate that the presented ensemble is the most accurate method with ECE = 0.0094. This paper demonstrates the performance of the ensemble at 98.67 ± 0.07% accuracy and 0.9996 macro AUC across three seeds. The ten most challenging families exhibit a distinct structure of confusion among them: sharing a common obfuscation stub, employing a similar loader, or having an identical appearance of packed and stub files. Extensive efficiency analysis reveals that LeViT-256 offers the best balance between latency and accuracy.

References

[1] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in Proc. 8th Int. Symp. VizSec, 2011, pp. 1–7.

[2] R. Ronen et al., "Microsoft malware classification challenge," arXiv:1802.10135, 2018.

[3] F. Mercaldo and A. Santone, "MaleVis: Malware classification through visualization," in Proc. IEEE COMPSAC, 2018, pp. 116–121.

[4] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for CNNs," in Proc. ICML, 2019, pp. 6105–6114.

[5] R. Wightman, "timm: PyTorch image models," GitHub, 2019. https://github.com/rwightman/pytorch-image-models.

[6] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in Proc. ICLR, 2021.

[7] Z. Liu et al., "Swin Transformer: Hierarchical vision transformer using shifted windows," in Proc. IEEE/CVF ICCV, 2021, pp. 10012–10022.

[8] Z. Liu et al., "Swin Transformer V2: Scaling up capacity and resolution," in Proc. IEEE/CVF CVPR, 2022, pp. 12009–12019.

[9] M. Kalash et al., "Malware classification with deep CNNs," in Proc. IFIP/IEEE ISNM, 2018, pp. 1–9.
[10] D. Vasan et al., "IMCFN: Image-based malware classification using fine-tuned CNN," Computer Networks, vol. 171, p. 107138, 2020.
[11] T. Rahim, A. Nazir, M. S. Tanveer, and M. R. Qureshi, "A deep learning approach to PCOS diagnosis: Two-stream CNN with transformer attention mechanism," Spectrum of Engineering Sciences, pp. 1–20, 2025.
[12] Q. Le et al., "Deep learning at the shallow end: Malware classification for non-domain experts," Digital Investigation, vol. 26, pp. S118–S126, 2018.
[13] I. Ullah, M. U. Yaseen, N. U. Amin, M. R. Qureshi, and S. Ibrahim, "Explainable emotion recognition from heart rate data using deep learning and XGBoost," in Proc. 27th Int. Multitopic Conf. (INMIC), 2025, pp. 1–6.
[14] L. Chen et al., "A multi-scale CNN for malware classification," in Proc. IEEE ICC, 2020.
[15] S. Faisal, I. Ullah, P. A. Kambey, A. Malik, and M. Shakeel, "Revolutionizing hepatic fibrosis staging: A machine learning approach combining clinical, biochemical, and microbiome insights," Computers in Biology and Medicine, vol. 206, p. 111584, 2026.
[16] H. S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware ML models," arXiv:1804.04637, 2018.
[17] I. Ullah, N. Rashid, S. Babar, and N. Iqbal, "Exploring the relationship between trait emotional intelligence and performance in the context of SME software engineering," Journal of Software: Evolution and Process, vol. 38, no. 1, p. e70076, 2026.
[18] A. M. Narayanan et al., "Robust malware detection using residual attention network," in Proc. IEEE ICASSP, 2020.
[19] R. Singh et al., "Malware classification using vision transformers," in Proc. ICMLA, 2022.
[20] Q. Zhang et al., "Malware family classification based on Swin Transformer," in Proc. IEEE TrustCom, 2022.
[21] J. Zhao et al., "Multiple instance learning for malware image classification," Applied Sciences, vol. 12, no. 16, p. 8113, 2022.
[22] R. Kumar et al., "UIDS: A unified intrusion detection system for IoT environment," Evolutionary Intelligence, 2021.
[23] Microsoft, "Microsoft Malware Classification Challenge (BIG 2015)," Kaggle, 2015.
[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. ICLR, 2015.
[25] H. Karbab and M. Debbabi, "MalDy: Portable data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports," Digital Investigation, vol. 28, pp. 77–87, 2019.
[26] S. Ullah, I. Ullah, I. Ullah, N. Ullah, and M. Taufiq, "A novel modified relative discrimination criterion for feature ranking in text classification," Spectrum of Engineering Sciences, pp. 349–367, 2025.
[27] S. Makandar and A. Patrot, "Bigram-DCT features for malware classification," IJRITCC, vol. 5, no. 3, 2017.
[28] S. Khan, S. Mustafa, and Q. Aziz, "Hybrid machine learning model for early detection of cucumber leaf curl disease in smart agriculture," Siazga Research Journal, vol. 3, no. 4, pp. 44–55, 2024.
[29] S. Li et al., "DFS-MC: Deep feature concatenation for malware classification," Expert Systems with Applications, vol. 212, p. 118789, 2023.
[30] Z. Liu et al., "ConvNeXt V2: Co-designing and scaling ConvNets with masked autoencoders," in Proc. IEEE/CVF CVPR, 2023, pp. 16133–16142.
[31] B. Graham et al., "LeViT: A vision transformer in ConvNet's clothing for faster inference," in Proc. IEEE/CVF ICCV, 2021, pp. 12259–12269.
[32] H. Li et al., "LeViT-based malware image classification with high throughput," in Proc. IEEE ICCST, 2023.
[33] H. Touvron et al., "DeiT III: Revisiting vision transformer training for image classification," in Proc. ECCV, 2022, pp. 516–533.
[34] Y. Gao et al., "BinImg2Vec: Self-supervised representation learning for malware binary images," in Proc. IEEE S&P, 2023.
[35] A. Baevski et al., "Data2Vec: A general framework for self-supervised learning in speech, vision and language," in Proc. ICML, 2022, pp. 1802–1816.
[36] K. He et al., "Masked autoencoders are scalable vision learners," in Proc. IEEE/CVF CVPR, 2022, pp. 16000–16009.

[37] A. Rajaraman et al., "Multimodal malware classification using PE header, opcode, and API call sequences," Computers & Security, vol. 127, p. 103100, 2023.

[38] T. J. Moreland et al., "Transfer learning from ImageNet to grayscale medical images via channel replication," in Proc. IEEE ISBI, 2020, pp. 1024–1028.