Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.086988

ARTICLE

When β Learns to Adapt: Stabilizing Variational Ensembles for Visual Malware Detection

Muhammet Tan[1], **Halit Bakır**[1], **Eid Albalawi**[2] and **A. H. Abdul Hafez**[2,*]

[1]Department of Computer Engineering, Sivas University of Science and Technology, Sivas, Turkey
[2]Department of Computer Science, College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa, Saudi Arabia

*Corresponding Author: A. H. Abdul Hafez. Email: aabdulhafiz@kfu.edu.sa

Received: 08 June 2026; Accepted: 03 September 2026; Published: 22 October 2026

ABSTRACT: Malware detection based on code visualization has emerged as a promising alternative to signature- and heuristic-based methods, particularly for identifying obfuscated and previously unseen threats. However, as multiple pre-trained convolutional neural networks (CNNs) are fused to enrich feature representations, the resulting high-dimensional feature spaces introduce redundancy, overfitting risk, and unstable optimization. This study presents a comprehensive benchmark for visual malware detection that systematically evaluates five architectural designs, ranging from single CNN classifiers and ensemble feature fusion to autoencoder bottlenecks, fixed-β VAEs, and an annealing-based β-VAE framework. Five pre-trained CNN backbones (VGG16, VGG19, Xception, ResNet50, and ConvNeXtLarge) were evaluated across three malware visualization datasets, namely Dumpware, MaleVis, and Malimg. A total of 345 experimental runs were conducted to analyze the interaction between ensemble size, latent dimensionality, and variational regularization. The results reveal a critical limitation of fixed-β designs: as the fused feature dimensionality increased from 1536 to 6656 features, fixed-β VAEs frequently exhibited posterior-collapse-consistent behaviour, severe loss instability, numerical divergence, and substantial degradation in validation performance. In contrast, the proposed annealing-based β-VAE stabilization framework, which combines gradual β scheduling with label smoothing, gradient clipping, and adaptive learning-rate scheduling, substantially improved optimization stability and generalization. On the Dumpware dataset, the proposed approach achieved validation accuracies up to 96.65%, while maintaining stable convergence across all ensemble configurations. Similar improvements were observed on MaleVis and Malimg, where the complete stabilization framework reduced the failure modes observed in fixed-β counterparts under high-dimensional feature fusion. These findings demonstrate that adaptive variational regularization is an effective mechanism for preserving discriminative information in large ensemble feature spaces and provide practical guidance for developing scalable, robust, and generalizable visual malware detection systems.

KEYWORDS: Malware detection; visual malware analysis; ensemble deep learning; beta-VAE; variational autoencoder; pre-trained CNN models

1 Introduction

The continuous evolution of malicious software poses persistent challenges to digital security across desktop, mobile, and networked environments. Traditional signature-based and heuristic methods [1–3] remain effective against known threats, but often fail to generalize to obfuscated, polymorphic, or previously unseen malware. In response, recent studies increasingly transform binary executables into image-like representations [4–7], enabling convolutional neural networks (CNNs) to learn discriminative visual

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

patterns that are difficult to engineer manually. However, as feature dimensionality increases, classification performance can suffer when the available sample capacity is limited, a behavior consistent with the peaking or Hughes phenomenon [8]. In visual malware detection, recent feature-fusion studies further show that combining heterogeneous or stacked feature sources can introduce scale mismatch, high dimensionality, and redundancy [9,10].

Single-backbone CNNs can perform well on image-transformed binaries, but their reliance on one feature extractor may limit representational diversity and robustness. Feature-level ensembles address this limitation by concatenating representations from multiple CNN backbones, thereby enriching the feature space. This benefit comes with an important cost: as more deep features are fused, the resulting vectors become larger and more redundant, increasing the risk of overfitting, optimization difficulty, and reduced generalization.

Autoencoder-based dimensionality reduction provides a natural way to compress such fused feature vectors while retaining discriminative information. Variational Autoencoders (VAEs), including β-Variational Autoencoders (β-VAEs), further regularize the latent space through a Kullback–Leibler (KL)-divergence penalty. Yet this regularization depends on the balance between reconstruction fidelity and latent constraint. In high-dimensional malware-detection pipelines, fixed β values may impose overly rigid pressure, leading to unstable optimization, degraded classification performance, and collapse-like latent behaviour. These risks increase with ensemble size and feature dimensionality, motivating adaptive regularization for stable learning.

This study therefore develops a code-visualization-based malware detection benchmark that integrates CNN backbones, ensemble feature fusion, and latent-space regularization. Five architectural designs are evaluated: individual CNN classifiers, ensemble feature-fusion models, deterministic autoencoder bottlenecks, fixed-β VAEs, and an annealing-based β-VAE variant. The goal is to compare not only model accuracy, but also how ensemble richness, latent dimensionality, and regularization strength jointly affect training stability and generalization in high-dimensional visual malware-detection pipelines.

1.1 Problem Definition and Research Gap

A persistent challenge in feature-fusion architectures is the curse of dimensionality. As multiple CNN backbones are concatenated, classification performance may initially improve but later deteriorate once dimensionality exceeds the effective sample capacity, a pattern known as the peaking or Hughes phenomenon [8]. Recent malware-focused studies likewise show that direct or stacked feature fusion can introduce heterogeneous feature scales, higher dimensionality, and redundant information, motivating attention-based alignment and feature-selection mechanisms [9,10]. Nevertheless, dimensionality-driven instability remains insufficiently investigated in code-visualization-based malware detection, where ensemble CNNs naturally produce very high-dimensional fused representations.

A related issue is the suitability of fixed β regularization in β-VAE models under high-dimensional or over-parameterized settings. Prior studies suggest that large or inflexible KL penalties can increase posterior-collapse risk, especially as latent dimensionality grows [11,12]. Empirical and theoretical analyses further indicate that excessive β values may over-smooth the latent space and reduce representational diversity [13], while Fil et al. [14] report reproducibility challenges in β-VAE training. Together, these findings suggest that fixed-β regularization may be unreliable for compressing large fused feature spaces in malware-detection pipelines.

Despite the growing literature on malware visualization, ensemble learning, and autoencoder-based modeling, most studies either improve classification through stronger CNN backbones or use autoencoders and VAEs for generic compression, latent extraction, or data generation. Comparatively little attention has

been paid to the interaction among ensemble feature dimensionality, latent bottleneck capacity, and β-regularization strength. In particular, it remains unclear whether fixed-β regularization stays reliable as fused feature spaces expand, or whether adaptive scheduling is needed for stable learning.

To address this gap, this study presents, to the best of our knowledge, one of the first systematic benchmarks in code-visualization-based malware detection to analyze the joint effects of ensemble size, latent-space dimensionality, and β-regularization across multiple datasets and model configurations. Across Dumpware, MaleVis, and Malimg, 345 training and evaluation runs were conducted across single CNNs, progressively expanded ensembles, deterministic autoencoder bottlenecks, fixed-β VAEs, and an annealing-based β-VAE framework. The analysis shows that moderate ensemble expansion can improve performance, larger ensembles become vulnerable to redundancy and instability, fixed-β VAEs become fragile as feature dimensionality increases, and a stabilized annealing-based configuration can alleviate these limitations by adaptively balancing latent regularization and reconstruction fidelity.

1.2 Main Contributions

The main contributions of this work are summarized as follows:

- This study presents a comprehensive benchmark for code-visualization-based malware detection, encompassing single CNN architectures, ensemble feature-fusion models, deterministic autoencoder bottlenecks, fixed-β variational autoencoders (VAEs), and annealing-based β-VAEs within a unified experimental framework.
- This work demonstrates that fixed-β variational regularization becomes increasingly unstable as the dimensionality of the ensemble feature space grows, frequently resulting in posterior-collapse-consistent behaviour, severe training-loss instability, and substantial degradation in classification performance.
- An annealing-based β-VAE configuration is evaluated together with label smoothing, gradient clipping, adaptive learning-rate scheduling, and stronger dropout as a complete stabilization framework for robust high-dimensional feature fusion.
- Extensive empirical evaluation across three benchmark malware-visualization datasets shows that the complete stabilization configuration provides an effective mechanism for balancing latent-space regularization and discriminative representation capacity in high-dimensional malware-detection pipelines.

The remainder of this article reviews related work, describes the datasets and model designs, presents the results, and discusses limitations and conclusions.

2 Related Works

Recent malware-detection studies increasingly use image-based binary representations and CNN classifiers, while ensemble CNNs are adopted to combine complementary features and improve robustness against obfuscation and family-level variability. Autoencoders and VAEs have also been explored for compression, latent representation learning, anomaly modeling, and synthetic data generation. However, the stability implications of applying latent regularization to high-dimensional ensemble feature spaces remain insufficiently studied.

Visualization-based malware detection has progressed from CNN-based image pipelines [15,16] to ensemble, multi-view, and multi-classifier strategies. Chaganti et al. [17] proposed a multi-view feature-fusion framework using static, dynamic, and image feature sets for malware classification; Çayır et al. [18] integrated Capsule Networks with bootstrap aggregation; and Tekerek and Yapici [15] used CycleGAN-based augmentation for byte-converted images. Later studies combined CNNs with Support Vector Machine (SVM), multilayer perceptron (MLP), AdaBoost, Random Forest, XGBoost, LightGBM, or soft/weighted

voting [19–21]. Weighted multi-CNN voting has also been optimized with Bayesian search [22]. These works confirm the value of ensemble learning, but mainly emphasize accuracy or decision-level fusion rather than instability in high-dimensional concatenated feature spaces.

Autoencoders and VAEs have also been used in malware analysis for feature compression, latent embedding, reconstruction-based classification, graph representation, and synthetic sequence generation [23–25]. Graph and convolutional VAE variants have also been explored for application programming interface (API)-call graphs, opcode generation, and assembly-file features [26–28]. Similarly, DroidEncoder [29] used artificial neural network (ANN)-, CNN-, and VGG19-based autoencoders to compress visualized Dalvik Executable (DEX) binaries before traditional classification. These studies establish the usefulness of latent feature extraction, but do not examine fixed-β regularization, ensemble feature fusion, or training instability under high-dimensional variational constraints.

Overall, prior work supports the value of visual CNNs, ensemble learning, and autoencoder/VAE-based modeling, but these components are usually studied separately. The open issue is how progressively larger fused CNN feature spaces interact with deterministic bottlenecks, fixed-β variational regularization, and annealing-based stabilization.

To make this distinction explicit, Supplementary Table S1 summarizes representative related studies and highlights the specific gap addressed by the present work.

The foregoing review shows that the combined effect of ensemble feature dimensionality, latent bottleneck design, and fixed-β regularization remains insufficiently understood in visual malware detection. In particular, it is unclear whether a β value that works for compact or single-backbone representations remains appropriate as the fused feature space scales to multiple CNN backbones. Addressing this risk requires a systematic multi-configuration benchmark and a practical alternative to fixed-β regularization, which motivate the present study.

3 Materials and Methods

3.1 Pretrained CNN Models

This study uses five ImageNet-pretrained CNN backbones: VGG16, VGG19, Xception, ResNet50, and ConvNeXtLarge. They were selected to provide architectural diversity, from classical convolutional designs to recent high-capacity models, and to serve both as individual baselines and as heterogeneous feature sources for ensemble fusion. The goal is therefore not only to compare standalone CNN performance, but also to examine how backbone diversity and feature dimensionality affect downstream feature-fusion, latent-compression, and β-regularization behavior. The main characteristics of the selected models are summarized in Table 1. All reported experiments use exactly this five-backbone set; no MobileNetV2-based configuration is included in the final benchmark.

Table 1: The pretrained CNN models adopted in this study.

Model	Size (MB)	ImageNet Accuracy	Parameters
VGG16 [30]	528	71.3%	138.4M
VGG19 [30]	549	71.3%	143.7M
Xception [31]	88	79.0%	22.9M
ResNet50 [32]	98	74.9%	25.6M
ConvNeXtLarge [33]	1310	86.7%	350.1M

3.2 Autoencoders

Autoencoders learn compact latent representations by compressing an input into a lower-dimensional bottleneck and reconstructing it [34]. In this study, deterministic autoencoder bottlenecks are introduced after feature fusion to compress large ensemble feature spaces before classification. This design tests whether a compact latent representation can preserve discriminative malware features while reducing redundancy introduced by concatenating multiple CNN backbones. Fig. 1 illustrates the autoencoder block adopted in this study.

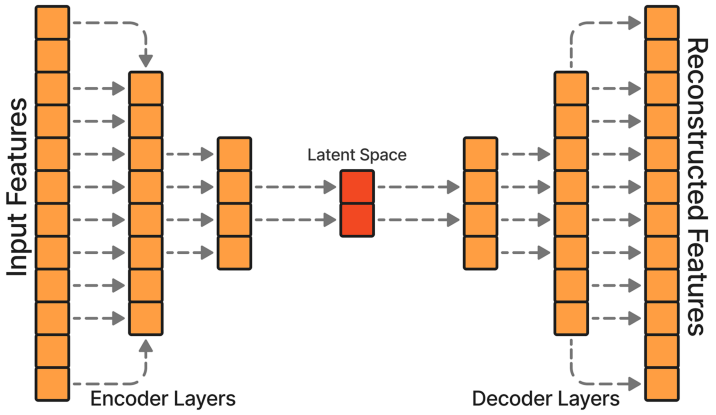


Figure 1: Visualization of the autoencoder block.

3.3 β-Variational Autoencoders

While deterministic autoencoders learn a single latent code, Variational Autoencoders (VAEs) learn a distribution over latent variables and balance reconstruction with latent-space regularization. For an input sample x and latent variable z, the standard VAE objective is defined through the evidence lower bound (ELBO) as:

$$L_{VAE}(\theta, \phi; x) = E_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - D_{KL}(q_\phi(z \mid x) \| p(z)), \tag{1}$$

where the first term encourages reconstruction and the second term is the Kullback–Leibler (KL) divergence that regularizes the approximate posterior $q_\phi(z \mid x)$ toward a prior $p(z)$, typically an isotropic Gaussian.

The β-Variational Autoencoder (β-VAE) extends this formulation by introducing a scaling factor $\beta > 0$ on the KL-divergence term, yielding the generalized objective:

$$L_{\beta\text{-}VAE}(\theta, \phi; x) = E_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - \beta D_{KL}(q_\phi(z \mid x) \| p(z)). \tag{2}$$

When $\beta = 1$, the objective reduces to the standard VAE; larger values strengthen latent regularization, whereas smaller values relax the KL penalty and allow more input information to pass through the bottleneck. This study focuses on sub-unitary values (0.05, 0.01, and 0.001) to examine how reduced KL pressure affects latent-space utilization, training stability, and classification performance in high-dimensional fused CNN feature spaces.

Both fixed-β and annealing-based β-VAE settings are evaluated. In the fixed setting, the same β value is used throughout training. In the annealing setting, β starts near zero and gradually reaches a predefined target value, allowing the model to prioritize reconstruction and classification early in training before strengthening latent regularization. This design directly tests whether β scheduling can improve stability as fused feature dimensionality increases.

3.4 Benchmark Datasets

The proposed framework was evaluated on three benchmark malware-visualization datasets with different class distributions, dataset scales, and family diversity. Dumpware [35] provides a recent and imbalanced 11-class setting, MaleVis [36] offers broader family-level variability across 26 malware classes, and Malimg [5] enables comparison with a widely used visual malware benchmark. Their main characteristics are summarized in Table 2.

Table 2: The datasets adopted in this study.

Dataset	Year	Classes	Samples	Characteristics	Purpose in Study
Dumpware	2021	11 (10 malware + 1 benign)	4294	Recent; imbalanced distribution across families	Evaluate performance under challenging multi-class, imbalanced conditions
MaleVis	2019	26 malware families	12,394	Diverse families with image-based representations	Provide broader variability and richer malicious behaviour patterns
Malimg	2011	25 malware families	9458	Earliest and most widely cited malware image dataset	Enable comparison with prior studies within the established literature

3.5 Input Representation and Preprocessing

The experiments use the image representations provided by the benchmark datasets rather than re-encoding raw binaries with a new visualization scheme, thereby preserving the dataset definitions used in prior work. Before being passed to the CNN backbones, all images were standardized to 224 × 224 pixels. Single-channel grayscale images were converted to three-channel format by channel replication, whereas existing Red–Green–Blue (RGB) inputs were retained. Pixel normalization followed the preprocessing convention of the corresponding ImageNet-pretrained backbone. This common input protocol ensured that performance differences arose from the feature extractors, fusion strategy, and latent regularization rather than from inconsistent input formatting.

3.6 Proposed Models and Evaluation Scenarios

Recent malware-detection research has explored several broad directions, including static analysis [37–39], dynamic analysis [40–42], hybrid approaches [43–46], and, more recently, visual malware classification based on image-transformed binaries [16,47,48]. Additional studies further show the relevance of visual representations for malware detection [49–51]. Building on the latter paradigm, the present study develops a multi-stage benchmark designed to analyze not only classification performance, but also the interaction

between feature-space dimensionality, latent compression, and variational regularization. To achieve this goal, five model families were constructed, each representing a distinct stage in the progression from standard visual classification to adaptive latent-space regularization. Fig. 2 summarizes the overall architecture of the proposed models.

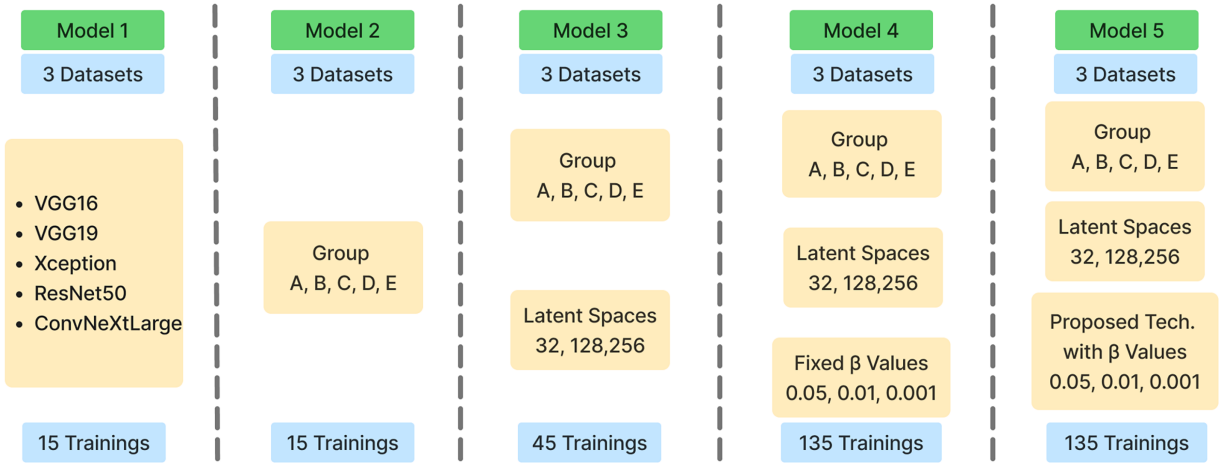


Figure 2: Architecture and experimental-run breakdown of the proposed models adopted in this study.

The benchmark was designed around two complementary axes of variation. The first axis concerns feature-space richness, implemented through progressively expanded ensembles of pre-trained CNN backbones. The second axis concerns latent-space control, implemented through deterministic autoencoder bottlenecks, fixed-β variational bottlenecks, and an annealing-based β-VAE. This design makes it possible to examine how feature fusion and latent regularization interact, and to identify the conditions under which increasing representational richness improves performance or instead leads to instability.

To systematically control ensemble complexity, five feature-fusion groups were defined, as summarized in Table 3. Feature fusion is implemented by concatenating the Global Average Pooling outputs of the selected frozen backbones along the feature dimension. No summation, voting, whitening, or Principal Component Analysis (PCA)-style projection is applied before the autoencoder or variational bottleneck. The resulting dimensions, which increase from 1536 to 6656 features, create a controlled progression from a single high-capacity representation to large, redundant multi-backbone feature spaces.

Table 3: Ensemble group configurations and fused feature dimensionality after global average pooling and feature concatenation.

Group	CNN Backbones	Backbone Feature Dimensions	Fused Dimension
A	ConvNeXtLarge	1536	1536
B	ConvNeXtLarge + VGG19	1536 + 512	2048
C	ConvNeXtLarge + VGG19 + VGG16	1536 + 512 + 512	2560
D	ConvNeXtLarge + VGG19 + VGG16 + Xception	1536 + 512 + 512 + 2048	4608
E	ConvNeXtLarge + VGG19 + VGG16 + Xception + ResNet50	1536 + 512 + 512 + 2048 + 2048	6656

3.6.1 Model 1: Single-Backbone CNN Baseline

Model 1 evaluates each frozen pre-trained CNN backbone independently: ConvNeXtLarge, VGG19, VGG16, Xception, or ResNet50. Feature maps are processed by Global Average Pooling and passed through fully connected layers with 512, 256, and 128 units, Rectified Linear Unit (ReLU) activations, dropout ($p = 0.2$), batch normalization, and a final softmax classifier adapted to the number of dataset classes. Training uses Adam with a learning rate of 1×10^{-5}, batch size 32, and early stopping based on validation loss. This model provides the baseline for the ensemble and latent-space architectures. Fig. 3 illustrates the architecture of Model 1.

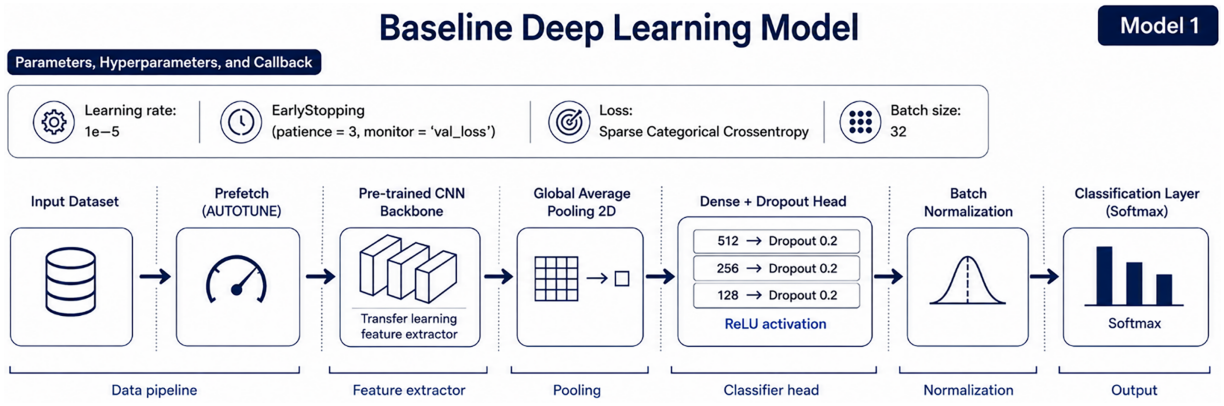


Figure 3: Architecture of the single-backbone CNN baseline (Model 1).

3.6.2 Model 2: Ensemble Feature-Fusion Model

Model 2 tests feature fusion without any latent bottleneck or variational regularization. Frozen CNN backbones extract features that are concatenated into a joint representation and classified through dense layers with 512, 256, 128, and 64 units, ReLU activations, dropout, batch normalization, and a final softmax layer. Training uses the same Adam learning rate, batch size, and early-stopping criterion as Model 1. Fig. 4 presents the architecture of Model 2.

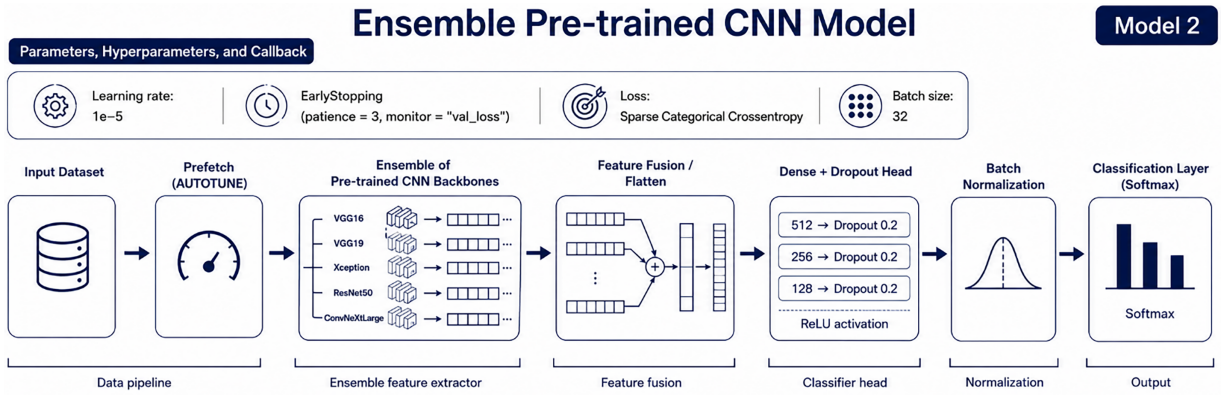


Figure 4: Architecture of the ensemble feature-fusion baseline (Model 2).

3.6.3 Model 3: Autoencoder-Enhanced Ensemble Model

Model 3 extends the ensemble setting by introducing a deterministic latent bottleneck after feature fusion to examine whether explicit compression can reduce redundancy while preserving discriminative

content. Fused features are passed through 512- and 256-unit dense layers with ReLU activation and dropout (p = 0.2), compressed to latent sizes of 32, 128, or 256, projected through an additional dense layer, and then classified by the softmax head. Across Groups A–E, this model tests the trade-off between information retention and compression strength. Fig. 5 shows the architecture of Model 3.

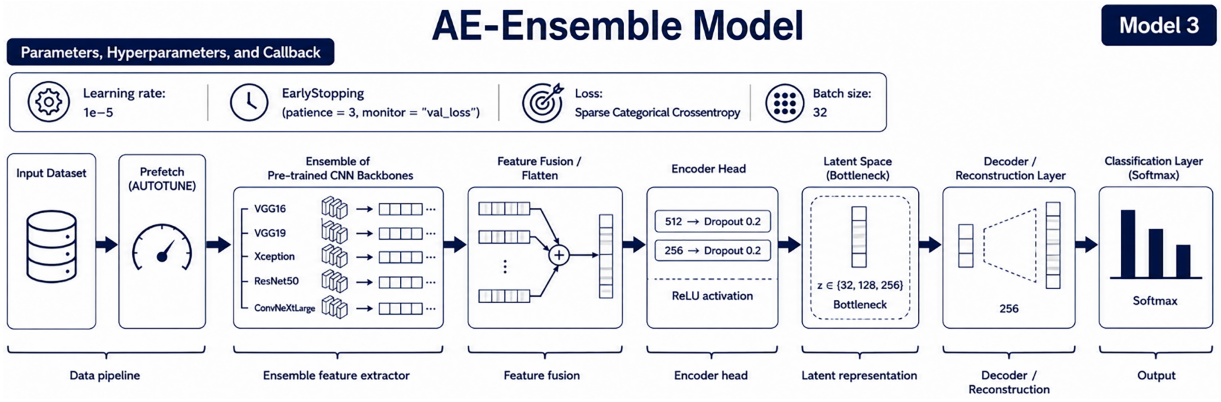


Figure 5: Architecture of the autoencoder-enhanced ensemble model (Model 3).

3.6.4 Model 4: Fixed-β Variational Ensemble Model

Model 4 replaces the deterministic autoencoder block with a β-VAE bottleneck to examine how fixed variational regularization affects classification and stability as ensemble size and latent dimensionality increase. Fused CNN features are mapped into a variational latent space with dimensions of 32, 128, or 256, while fixed β values of 0.05, 0.01, and 0.001 control the KL-divergence weight. Holding β constant provides a controlled setting for testing whether the same regularization strength remains suitable as feature dimensionality and redundancy grow. Fig. 6 illustrates the architecture of Model 4.

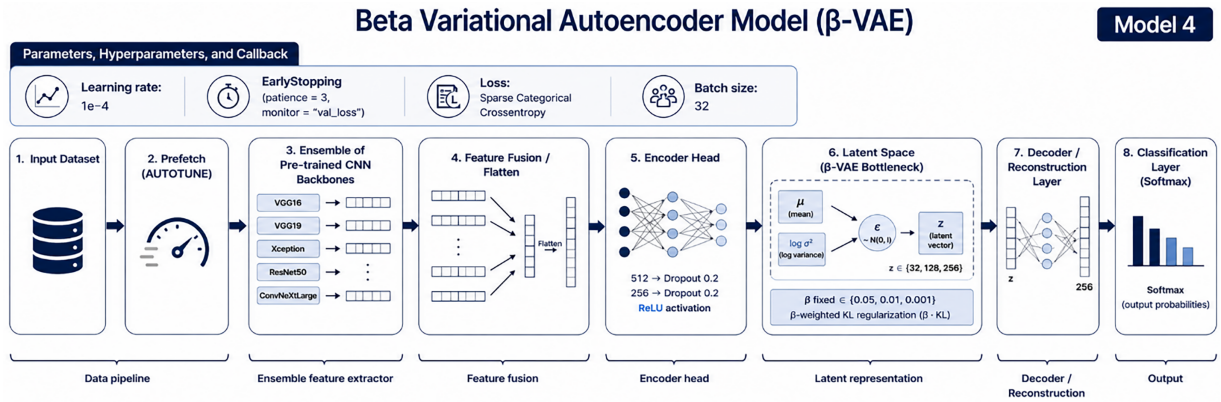


Figure 6: Architecture of the fixed-β variational ensemble model (Model 4).

3.6.5 Model 5: Annealing-Based β-VAE with Training Stabilization

Model 5 addresses the limitations observed in Model 4 as a complete stabilization configuration rather than as an ablation of a single component. Instead of applying a constant β, it initializes β near zero and gradually increases it during early training so that reconstruction and classification can stabilize before stronger latent regularization is imposed. Fig. 7 presents the architecture of Model 5.

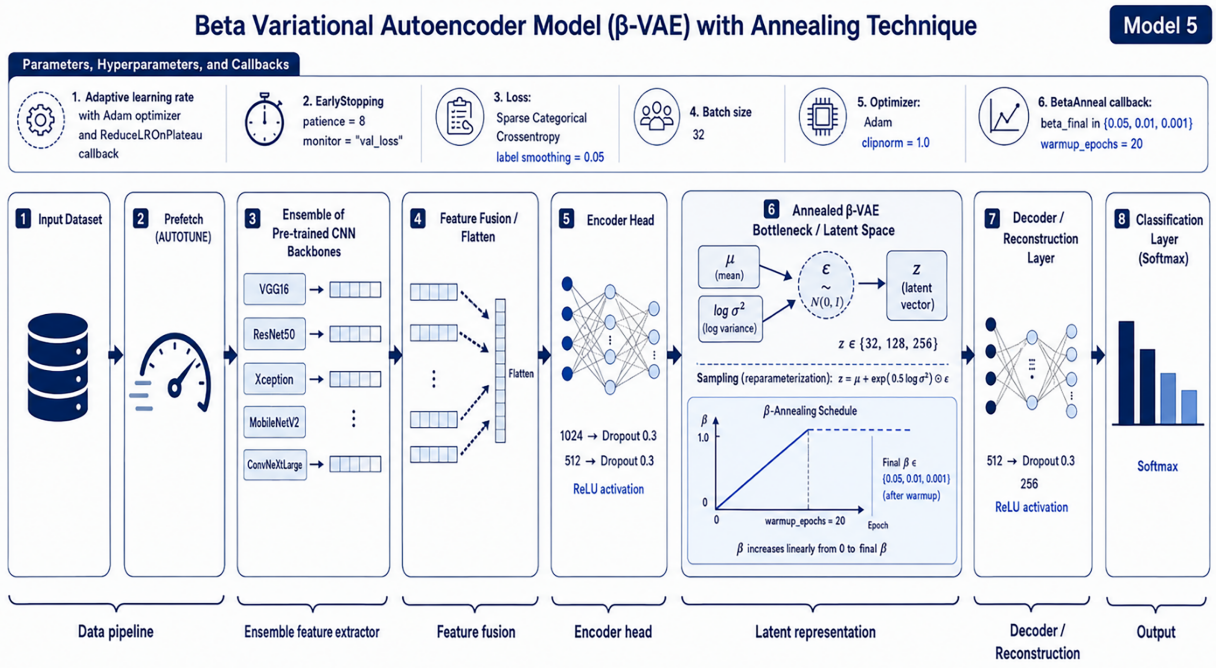


Figure 7: Architecture of the annealed β-VAE model with stabilization (Model 5).

In addition to β annealing, Model 5 incorporates several complementary stabilization strategies. Label smoothing with $\varepsilon = 0.05$ is applied in the classification head to reduce overconfidence and improve generalization. Gradient clipping with a clip norm of 1.0 is used to prevent unstable gradient updates, especially in deep or highly fused configurations. Adaptive learning-rate scheduling is implemented through ReduceLROnPlateau, allowing the optimizer to reduce its step size when validation loss ceases to improve. Furthermore, the dropout rate is increased from 0.2 in Model 4 to 0.3 in Model 5 in order to provide stronger regularization against overfitting. These refinements were selected to reduce optimizer instability and overconfidence while preserving comparability with the fixed-β setting. Therefore, the reported comparison should be interpreted as fixed-β VAE vs. a stabilized annealing-based configuration, not as proof that annealing alone explains the full performance difference. Together, these refinements transform the variational bottleneck from a potentially fragile component into a more robust mechanism for managing large fused feature spaces. Model 5 therefore serves as the proposed solution for stabilizing high-dimensional β-VAE-based malware-detection pipelines.

3.6.6 Training Objectives for Bottleneck Models

For clarity, the sparse categorical cross-entropy reported in the architecture summaries refers to the classification component of the objective. Models 3–5 include additional representation-learning terms because the bottleneck is trained to preserve the fused CNN feature vector, not to reconstruct the original malware image. The reconstruction branch is an auxiliary feature-preservation path; the final prediction is produced by a dense classification head from the learned latent/encoded representation, not by applying softmax directly to $\hat{h}$. Let h denote the concatenated backbone feature vector, z the latent representation, $\hat{h}$ the reconstructed feature vector, y the class label, and $\hat{y}$ the softmax prediction. Models 1 and 2 optimize only the classification loss:

$$\mathcal{L}_{1,2} = \mathcal{L}_{cls}(y, \hat{y}). \tag{3}$$

Model 3 uses a deterministic autoencoder bottleneck and optimizes a joint classification–reconstruction objective:

$$\mathcal{L}_3 = \mathcal{L}_{cls}(y, \hat{y}) + \lambda_{rec}\mathcal{L}_{rec}(h, \hat{h}). \tag{4}$$

Models 4 and 5 use a variational bottleneck and add a KL-divergence term:

$$\mathcal{L}_{4,5} = \mathcal{L}_{cls}(y, \hat{y}) + \lambda_{rec}\mathcal{L}_{rec}(h, \hat{h}) + \beta_t D_{KL}\big(q_\phi(z \mid h) \| p(z)\big). \tag{5}$$

Here, $\mathcal{L}_{cls}$ is sparse categorical cross-entropy for Models 1–4 and label-smoothed categorical cross-entropy for Model 5, $\mathcal{L}_{rec}$ is mean-squared reconstruction error over the fused feature vector, and λ_{rec} is set to 1 in the reported experiments. For Model 4, β_t is constant and equal to the selected target value (0.05, 0.01, or 0.001). For Model 5, β_t is annealed from a near-zero value to the same target value during the warmup phase. The schedule is defined as:

$$\beta_t = \min\left(\beta_{\max}, \beta_{\max}\frac{t}{T_w}\right), \tag{6}$$

where t is the current epoch, $T_w = 20$ epochs is the warmup length, and $\beta_{\max} \in \{0.05, 0.01, 0.001\}$ is the target regularization strength. Thus, validation loss in Models 3–5 is a composite objective rather than a pure classification loss, which explains why high validation accuracy can coexist with larger loss values in the variational models.

4 Experimental Results and Discussions

4.1 Experimental Setup

The experiments were conducted on a workstation with an Intel i7 12th-generation processor, 32 GB of RAM, and an Nvidia RTX 4080 Graphics Processing Unit (GPU). All deep learning experiments were implemented in TensorFlow. For each dataset, samples were partitioned into training and validation subsets using an 80/20 class-stratified split, preserving the relative class distribution in both subsets as far as possible. No sample-level augmentation was used after the split, and the same split protocol was applied across model families so that performance differences reflected architectural and optimization effects rather than changes in data partitioning. Each model was trained for a maximum of 150 epochs with early stopping based on validation loss and a patience of 5 epochs. Training used a common batch size of 32 and the same preprocessing protocol described in Section 3.5. These choices provide a reproducible basis for comparing absolute performance and training stability across latent dimensions and β values.

The three β values (0.05, 0.01, 0.001) were selected to span a representative range from relatively strong (0.05) to moderate (0.01) to weak (0.001) KL regularization, enabling a systematic investigation of how varying degrees of variational pressure interact with different levels of feature-space dimensionality. The three latent dimensions (32, 128, 256) were chosen to cover a spectrum from aggressive compression to near-lossless representation, providing a structured evaluation of the information-retention and regularization trade-off inherent in variational bottleneck models.

The full benchmark consists of 345 confirmed runs: 15 runs for Model 1 (5 backbones × 3 datasets), 15 runs for Model 2 (5 groups × 3 datasets), 45 runs for Model 3 (5 groups × 3 latent sizes × 3 datasets), 135 runs for Model 4 (5 groups × 3 latent sizes × 3 β values × 3 datasets), and 135 runs for Model 5 under the same grid as Model 4.

4.2 Evaluation Metrics

Model performance was evaluated using Accuracy, Precision, Recall, and F1-Score (F1-S), with class-balanced averaging used for the multi-class datasets so that minority families contributed to the aggregate scores. Training dynamics were monitored through Training Accuracy (T.A.), Training Loss (T.L.), Validation Accuracy (V.A.), Validation Loss (V.L.), and Stopped Epoch (S.E.). V.A. was treated as the primary generalization metric, whereas V.L. was used for early stopping. The abbreviation T.L. denotes training loss rather than test loss; for Models 3–5, both T.L. and V.L. correspond to composite objectives that include the auxiliary reconstruction term and, for variational models, the KL-divergence term. The Supplementary Tables emphasize V.A. and loss values to expose stability behavior across the full configuration grid, while F1-S is retained in the main baseline summary as an imbalance-aware performance reference.

4.3 Obtained Results

The results obtained from over 300 training runs, conducted across three benchmark datasets, clearly demonstrate the significant influence of feature-space dimensionality on the behaviours of β-Variational Autoencoders. The available confirmed per-run training statistics (S.E., T.A., T.L., V.A., V.L.) for Models 1–3 are provided in Tables S2–S4, and the configuration-wise Model 4 vs. Model 5 comparison across all groups, latent-space dimensions, and β values is provided in Tables S5–S7 of the Supplementary Materials. The following discussion focuses on the principal patterns and key findings.

4.3.1 Single-Backbone and Ensemble Baselines

On the Dumpware dataset, all individual CNN backbones achieved high performance, with validation accuracies ranging from 0.93 to 0.95. Among them, ResNet50 and VGG16 provided slightly stronger balance across accuracy and F1-scores. The ensemble model outperformed all single architectures, reaching the highest validation accuracy (0.9642) and F1-score (0.9672), confirming the benefit of feature-level fusion. On the MaleVis dataset, the individual CNN architectures demonstrated consistent yet moderate generalization performance, with validation accuracies around 86%–87%. When their feature representations were fused through feature-space concatenation, the overall validation accuracy improved slightly to 87%, accompanied by balanced precision (0.93) and recall (0.86), indicating that the feature-fusion strategy effectively enhanced discriminative capacity without overfitting. On the Malimg dataset, all individual CNN architectures achieved near-perfect classification performance, with validation accuracies ranging from 0.9829 to 0.9898 and highly consistent precision, recall, and F1-scores around 0.98–0.99. When the models were combined through feature fusion (Model 2), the performance remained comparably high (VA = 0.9893, F1 = 0.9935), confirming that the ensemble effectively integrated complementary features. Table 4 summarizes the validation accuracy and F1-score for all single-backbone configurations and the best ensemble baseline (Model 2, Group E) across all three datasets. Detailed training statistics including S.E., T.A., T.L., and V.L. are provided in Tables S2–S4.

4.3.2 Autoencoder Bottleneck (Model 3) vs. Ensemble Baseline

Model 3 shows that deterministic compression can be useful, but its effect is dataset- and group-dependent. In Group A, adding the autoencoder bottleneck reduced validation accuracy on all datasets, indicating information loss under aggressive latent compression. In Groups B–E, Model 3 often remained close to the ensemble baseline and occasionally surpassed it, but it did not provide a consistent accuracy advantage over the strongest non-variational model. Table 5 presents the best validation accuracy achieved by Model 2 (or Model 1 for Group A) and the best Model 3 configuration (highest V.A. across latent-space

dimensions 32, 128, and 256) per group and dataset. Available per-latent-dimension results are provided in Tables S2–S4.

Table 4: Validation accuracy (V.A.) and F1-score (F1-S) for single-backbone (Model 1) and full-ensemble baseline (Model 2, Group E) across all three datasets. Detailed training statistics are provided in Tables S2–S4.

Model	Dumpware		MaleVis		Malimg	
	V.A.	F1-S	V.A.	F1-S	V.A.	F1-S
VGG16	0.9419	0.9483	0.8677	0.8702	0.9893	0.9892
VGG19	0.9486	0.9467	0.8585	0.8664	0.9898	0.9876
Xception	0.9263	0.9312	0.8115	0.8185	0.9829	0.9846
ResNet50	0.9441	0.9534	0.8650	0.8788	0.9898	0.9912
ConvNeXtLarge	0.9441	0.9405	0.8669	0.8734	0.9882	0.9876
Feature Fusion (M2, Gr.E)	0.9642	0.9672	0.8700	0.8772	0.9893	0.9935

Table 5: Best validation accuracy (V.A.) of the ensemble baseline (Model 2, or Model 1 for Group A) and the best autoencoder-bottleneck configuration (Model 3, best V.A. among the reported latent-dimension runs) per group and dataset. Available per-latent-dimension results are provided in Tables S2–S4.

Group	Model	Dumpware		MaleVis		Malimg	
		M1/M2	M3 Best	M1/M2	M3 Best	M1/M2	M3 Best
A	M1/M3	0.9441	0.8772	0.8669	0.8238	0.9882	0.9734
B	M2/M3	0.9554	0.9598	0.8751	0.8724	0.9888	0.8774
C	M2/M3	0.9643	0.9576	0.8761	0.8765	0.9894	0.9931
D	M2/M3	0.9665	0.9576	0.8619	0.8591	0.9899	0.9915
E	M2/M3	0.9642	0.9665	0.8700	0.8726	0.9893	0.9904

Because the objective of Model 5 is stabilization rather than merely maximizing peak validation accuracy, Table 6 compares the best non-variational result from Models 1–3 with the best Model 5 result for each dataset. The comparison shows that Model 5 is competitive with the strongest discriminative baselines while offering the additional benefit of avoiding the fixed-β failure modes analyzed below.

Table 6: Dataset-level best validation accuracy (V.A.) comparison between the best non-variational configuration from Models 1–3 and the best stabilized Model 5 configuration.

Dataset	Best M1–M3 V.A.	Best M5 V.A.	Interpretation
Dumpware	0.9665	0.9665	Comparable peak accuracy
MaleVis	0.8765	0.8950	Higher peak accuracy with M5
Malimg	0.9931	0.9931	Comparable peak accuracy

4.3.3 Fixed-β vs. Annealing-β Stability Analysis

The comparison between Model 4 (fixed-β VAE) and Model 5 (annealing-based β-VAE with additional stabilization) shows a consistent stability pattern: fixed regularization becomes fragile as the fused feature

space grows, whereas the complete stabilization framework preserves usable optimization dynamics. On Dumpware, this effect is most visible in Groups C–E, where Model 4 often combines sharp validation accuracy drops with inflated training losses, especially at $\beta = 0.05$, while Model 5 maintains validation accuracy around 0.94–0.97. On MaleVis, fixed β values produce exploding or undefined losses in larger groups and reduce several configurations to near-random validation accuracy, whereas Model 5 remains around 0.85–0.87. Because direct latent-space diagnostics such as active-unit counts were not recorded, these failures are interpreted conservatively as posterior-collapse-consistent behaviour rather than direct evidence of posterior collapse. On Malimg, where baseline accuracy is already high, Model 4 still degrades sharply at $\beta = 0.05$ and diverges to NaN in Groups D and E, whereas Model 5 remains in the ≈0.985–0.993 range. The main benefit is therefore not simply a higher peak score, but the prevention of catastrophic optimization failures in large ensemble representations.

Model 5's composite loss values in Table 7 are not directly comparable with the purely discriminative losses of Models 1 and 2 because the β-VAE objective jointly optimizes classification, reconstruction, and KL-divergence terms. A moderate composite loss alongside high validation accuracy is therefore expected when the model balances discrimination, reconstruction, and latent regularization.

Table 7: Representative stability comparison of Model 4 (fixed-β VAE) and Model 5 (annealing-β-VAE with additional stabilization) at $\beta = 0.05$ and latent size LS = 128, across Groups A–E and all three datasets. T.L. denotes the composite training loss/objective, not test loss. NaN denotes numerical divergence. Divergence failures are treated as posterior-collapse-consistent behaviour rather than direct latent-space measurements. Configuration-wise results, including validation loss (V.L.), for all β values and latent dimensions are provided in Tables S5–S7.

Dataset	Group	Model 4 (fixed-β)		Model 5 (annealing)	
		V.A.	T.L.	V.A.	T.L.
Dumpware	A	0.8884	0.5571	0.9286	0.4354
	B	0.5491	10.0184	0.9420	0.4702
	C	0.5558	1879.48	0.9442	0.4531
	D	0.1875	9184.00	0.9531	0.4080
	E	0.4062	61.87	0.9531	0.4197
MaleVis	A	0.8233	0.3935	0.8519	0.4719
	B	0.5647	22.4867	0.8631	0.4791
	C	0.1205	NaN	0.8619	0.4700
	D	0.1205	NaN	0.8637	0.5393
	E	0.1205	NaN	0.8646	0.4646
Malimg	A	0.3106	2.5589	0.9745	0.4634
	B	0.2941	2.6435	0.9888	0.4299
	C	0.3133	2.5953	0.9910	0.4256
	D	0.0851	NaN	0.9915	0.4236
	E	0.0851	NaN	0.9899	0.4281

Table 7 provides a representative comparison for $\beta = 0.05$ and LS = 128; configuration-wise results for all β values and latent dimensions are provided in Tables S5–S7.

Taken together, these results show that fixed-β VAEs are fragile under large feature-fusion configurations, whereas the complete stabilization framework substantially reduces loss explosions and sustains robust generalization across the ensemble sizes, latent dimensions, and β settings evaluated in this study.

5 Limitations

Although the benchmark systematically evaluates five model families across three datasets, three latent dimensions, three β values, and five ensemble groups, several limitations remain. First, the experimental protocol was intentionally standardized to make the model comparisons controlled and reproducible. All runs used a 224×224 input size, batch size of 32, an 80/20 class-stratified validation split, and common optimizer and loss settings. Consequently, the present results do not measure sensitivity to alternative image resolutions, augmentation policies, class-rebalancing strategies, repeated random seeds, cross-validation folds, or wider learning-rate and regularization grids. These factors may influence both peak accuracy and the stability patterns reported here.

Second, Model 5 should be interpreted as a complete stabilization configuration rather than as an isolated test of β annealing alone. It combines gradual β scheduling with label smoothing, gradient clipping, adaptive learning-rate scheduling, and a higher dropout rate. This design was selected to test whether a practical stabilized variational bottleneck can remain trainable under high-dimensional feature fusion, but it does not quantify the separate contribution of each stabilizer. A full ablation study should therefore compare annealing alone, fixed-β training with matched auxiliary stabilizers, and leave-one-component-out variants for label smoothing, gradient clipping, learning-rate scheduling, and dropout.

Third, the study focuses on observable optimization and classification behaviour rather than direct latent-space diagnostics. The term posterior-collapse-consistent behaviour is therefore used conservatively: it is inferred from loss inflation, NaN divergence, and validation-accuracy collapse, not from direct measurements of latent usage. Future work should record epoch-wise KL curves, reconstruction and classification loss components, posterior variance, mutual information, active-unit counts, and disentanglement-oriented measures to distinguish posterior collapse from other forms of optimization failure more precisely.

Finally, this benchmark does not evaluate deployment-oriented and security-oriented properties. Computational cost, GPU memory use, training time, inference latency, model size, and robustness against adversarial or obfuscated malware variants were outside the present scope. Alternative autoencoder designs, supervised autoencoders, PCA/PLS/LDA compression baselines, attention-based fusion, and transformer-based fusion also remain important directions for extending the comparison.

6 Conclusions

This work presents a systematic benchmark for visual malware classification under high-dimensional feature fusion. Across Dumpware, MaleVis, and Malimg, five model families were evaluated: individual CNN backbones, feature-fusion ensembles, deterministic autoencoder bottlenecks, fixed-β VAEs, and a stabilized annealing-based β-VAE configuration. The results show that feature fusion can improve discriminative capacity, but also creates large and redundant feature spaces that expose the fragility of fixed-β variational regularization. Model 3 served as a useful compression baseline but did not consistently improve over the strongest discriminative models. Model 4, in contrast, frequently exhibited loss inflation, NaN divergence, and posterior-collapse-consistent accuracy degradation as fused dimensionality increased. The stabilized Model 5 configuration mitigated these failures by gradually increasing β after early representation learning and by using label smoothing, gradient clipping, adaptive learning-rate scheduling, and stronger dropout. Because these controls were evaluated together, the findings support the complete stabilization framework

rather than isolating a single causal component. Overall, the benchmark shows that adaptive stabilization is a practical route for maintaining robust optimization in scalable visual malware-detection pipelines.

Acknowledgement: None.

Funding Statement: This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [KFU264086].

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Muhammet Tan, Halit Bakır and A. H. Abdul Hafez; methodology, Muhammet Tan, Halit Bakır and A. H. Abdul Hafez; software, Muhammet Tan; validation, Muhammet Tan, Halit Bakır, Eid Albalawi and A. H. Abdul Hafez; formal analysis, Muhammet Tan and Halit Bakır; investigation, Muhammet Tan and Eid Albalawi; data curation, Muhammet Tan and Eid Albalawi; writing—original draft preparation, Muhammet Tan; writing—review and editing, Halit Bakır, Eid Albalawi and A. H. Abdul Hafez; visualization, Muhammet Tan; supervision, Halit Bakır and A. H. Abdul Hafez; project administration, A. H. Abdul Hafez; funding acquisition, A. H. Abdul Hafez. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The implementation of all five model configurations (Model 1–Model 5) used in this study is publicly available at https://github.com/muhammettan28/WhenBetaLearnstoAdapt. The 80/20 class-stratified train–validation partitioning procedure is described in the manuscript and implemented in the released code; the derived per-run and configuration-wise statistics generated for this study are reported in Supplementary Tables S2–S7. The benchmark datasets (Dumpware, MaleVis, Malimg) are publicly available from their respective original publications [5,35,36].

Ethics Approval: Not applicable.

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

Supplementary Materials: The supplementary material is available online at https://www.techscience.com/doi/10.32604/cmc.2026.086988/s1. Table S1: Representative related studies on visual malware detection, ensemble learning, and autoencoder/VAE-based modeling. Table S2: Confirmed per-run training statistics (S.E., T.A., T.L., V.A., V.L.) for Models 1, 2, and 3 on the Dumpware dataset across all groups and latent dimensions. Table S3: Confirmed per-run training statistics for Models 1, 2, and 3 on the MaleVis dataset. Table S4: Confirmed per-run training statistics for Models 1, 2, and 3 on the Malimg dataset. Table S5: Configuration-wise Model 4 vs. Model 5 comparison on the Dumpware dataset for all groups (A–E), β values (0.05, 0.01, 0.001), and latent dimensions (32, 128, 256). Table S6: Configuration-wise Model 4 vs. Model 5 comparison on the MaleVis dataset. Table S7: Configuration-wise Model 4 vs. Model 5 comparison on the Malimg dataset.

References

1. Yunmar RA, Kusumawardani SS, Widyawan, Mohsen F. Hybrid android malware detection: a review of heuristic-based approach. IEEE Access. 2024;12(8):41255–86. doi:10.1109/access.2024.3377658.
2. Awwal P, Naval DS. Development of heuristic adapted serial-based deep learning for efficient adversarial malware detection framework in windows. Knowl Based Syst. 2025;326(6):114032. doi:10.1016/j.knosys.2025.114032.
3. Kozachok AV, Bochkov MV, Kochetkov EV. Heuristic malware detection mechanism based on executable files static analysis. In: Proceedings of the 3rd International Conference on Information Technology and Nanotechnology; 2017 Apr 25–27; Samara, Russia. p. 132–9.
4. Bakour K, Ünver HM. VisDroid: android malware classification based on local and global image features, bag of visual words and machine learning techniques. Neural Comput Appl. 2021;33(8):3133–53. doi:10.1007/s00521-020-05195-w.

5. Nataraj L, Karthikeyan S, Jacob G, Manjunath BS. Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security; 2011 Jul 20; Pittsburgh, PA, USA. New York, NY, USA: ACM; 2011. p. 1–7. doi:10.1145/2016904.2016908.
6. Efe A, Hussin SHS. Malware visualization techniques. Int J Appl Math Electron Comput. 2020;8(1):7–20. doi:10.18100/ijamec.526813.
7. Khormali A, Abusnaina A, Chen S, Nyang D, Mohaisen A. COPYCAT: practical adversarial attacks on visualization-based malware detection. arXiv:1909.09735. 2019. doi:10.48550/arXiv.1909.09735.
8. Chao S, Dougherty ER. The peaking phenomenon in the presence of feature-selection. Pattern Recognit Lett. 2008;29(11):1667–74. doi:10.1016/j.patrec.2008.04.010.
9. Wu X, Wan L, Ren J, Zhang B. WAFDect: a malware detection model based on multi-source feature fusion. Comput Mater Contin. 2026;88:49. doi:10.32604/cmc.2026.077928.
10. Kumar S, Kumar A. MalSFF: Multi-architecture malware detection using multi-static feature fusion based on image visualization and learning methods. Comput Electr Eng. 2026;129(4):110834. doi:10.1016/j.compeleceng.2025.110834.
11. Ichikawa Y, Hukushima K. Learning dynamics in linear VAE: posterior collapse threshold, superfluous latent space pitfalls, and speedup with KL annealing. arXiv:2310.15440. 2023. doi:10.48550/arXiv.2310.15440.
12. Dai B, Wang Z, Wipf D. The usual suspects? Reassessing blame for VAE posterior collapse. arXiv:1912.10702. 2019. doi:10.48550/arXiv.1912.10702.
13. Rivera M. How to train your VAE. arXiv:2309.13160. 2024. doi:10.48550/arxiv.2309.13160.
14. Fil M, Mesinovic M, Morris M, Wildberger J. Beta-VAE reproducibility: challenges and extensions. arXiv:2112.14278. 2021. doi:10.48550/arXiv.2112.14278.
15. Tekerek A, Yapici MM. A novel malware classification and augmentation model based on convolutional neural network. Comput Secur. 2022;112(3):102515. doi:10.1016/j.cose.2021.102515.
16. Wang Z, Wang W, Yang Y, Han Z, Xu D, Su C. CNN- and GAN-based classification of malicious code families: a code visualization approach. Int J Intell Syst. 2022;37(12):12472–89. doi:10.1002/int.23094.
17. Chaganti R, Ravi V, Pham TD. A multi-view feature fusion approach for effective malware classification using deep learning. J Inf Secur Appl. 2023;72(21):103402. doi:10.1016/j.jisa.2022.103402.
18. Çayır A, Ünal U, Dağ H. Random CapsNet forest model for imbalanced malware type classification task. Comput Secur. 2021;102(2):102133. doi:10.1016/j.cose.2020.102133.
19. Khan SH, Alahmadi TJ, Ullah W, Iqbal J, Rahim A, Alkahtani HK, et al. A new deep boosted CNN and ensemble learning based IoT malware detection. Comput Secur. 2023;133:103385. doi:10.1016/j.cose.2023.103385.
20. Nethala S, Chopra P, Kamaluddin K, Alam S, Alharbi S, Alsaffar M. A deep learning-based ensemble framework for robust android malware detection. IEEE Access. 2025;13:46673–96. doi:10.1109/access.2025.3551152.
21. Kanumalli SS, Dalavayi N, Veernapu V, Kota P, Katari RP. Enhanced malware detection using convolutional neural network with robust ensemble algorithm—"LIGHTGBM". In: Proceedings of the 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE); 2024 Jan 24–25; Bangalore, India. New York, NY, USA: IEEE; 2024. p. 1–7. doi:10.1109/iitcee59897.2024.10467722.
22. Chaymae EY, Khalid C. Android malware detection through CNN ensemble learning on grayscale images. Int J Adv Comput Sci Appl. 2025;16(1):1208–17. doi:10.14569/ijacsa.2025.01601116.
23. Naway A, Li Y. Android malware detection using autoencoder. arXiv:1901.07315. 2019.
24. Xing X, Jin X, Elahi H, Jiang H, Wang G. A malware detection approach using autoencoder in deep learning. IEEE Access. 2022;10:25696–706. doi:10.1109/access.2022.3155695.
25. Ajayi B, Barakat B, McGarry K. Leveraging VAE-derived latent spaces for enhanced malware detection with machine learning classifiers. arXiv:2503.20803. 2025. doi:10.48550/arXiv.2503.20803.
26. Gunduz H. Malware detection framework based on graph variational autoencoder extracted embeddings from API-call graphs. PeerJ Comput Sci. 2022;8(2):e988. doi:10.7717/peerj-cs.988.
27. Choi A, Giang A, Jumani S, Luong D, Di Troia F. Synthetic malware using deep variational autoencoders and generative adversarial networks. EAI Endorsed Trans Internet Things. 2024;10:1–11. doi:10.4108/eetiot.6566.

28. Taylor T, Eleyan A. Using variational autoencoders to increase the performance of malware classification. In: Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC); 2021 Oct 31–Nov 2; Dubai, United Arab Emirates. New York, NY, USA: IEEE; 2021. p. 1–6. doi:10.1109/isncc52172.2021.9615643.
29. Bakır H, Bakır R. DroidEncoder: malware detection using auto-encoder based feature extractor and machine learning algorithms. Comput Electr Eng. 2023;110(2–4):108804. doi:10.1016/j.compeleceng.2023.108804.
30. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings; 2015 May 7–9; San Diego, CA, USA.
31. Chollet F. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017 Jul 21–26; Honolulu, HI, USA. New York, NY, USA: IEEE; 2017. p. 1800–7. doi:10.1109/cvpr.2017.195.
32. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016 Jun 27–30; Las Vegas, NV, USA. New York, NY, USA: IEEE; 2016. p. 770–8. doi:10.1109/cvpr.2016.90.
33. Liu Z, Mao H, Wu CY, Feichtenhofer C, Darrell T, Xie S. A ConvNet for the 2020s. In: Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022 Jun 18–24; New Orleans, LA, USA. New York, NY, USA: IEEE; 2022. p. 11966–76. doi:10.1109/cvpr52688.2022.01167.
34. Michelucci U. An introduction to autoencoders. arXiv:2201.03898. 2022. doi:10.48550/arXiv.2201.03898.
35. Bozkir AS, Tahillioglu E, Aydos M, Kara I. Catch them alive: a malware detection approach through memory forensics, manifold learning and computer vision. Comput Secur. 2021;103(1):102166. doi:10.1016/j.cose.2020.102166.
36. Bozkir AS, Cankaya AO, Aydos M. Utilization and comparision of convolutional neural networks in malware recognition. In: Proceedings of the 2019 27th Signal Processing and Communications Applications Conference (SIU); 2019 Apr 24–26; Sivas, Turkey. New York, NY, USA: IEEE; 2019. p. 1–4. doi:10.1109/siu.2019.8806511.
37. Singh HK, Singh JP, Tewari AS. Static malware analysis using machine and deep learning. In: Proceedings of International Conference on Computing and Communication Networks; 2021 Nov 19–20; Manchester, UK. Singapore: Springer Nature; 2022. p. 437–46. doi:10.1007/978-981-19-0604-6_41.
38. Guven M. Leveraging deep learning and image conversion of executable files for effective malware detection: a static malware analysis approach. AIMS Math. 2024;9(6):15223–45. doi:10.3934/math.2024739.
39. del Aguila RB, Contreras Pérez CD, Silva-Trujillo AG, Cuevas-Tello JC, Nunez-Varela J. Static malware analysis using low-parameter machine learning models. Computers. 2024;13(3):59. doi:10.3390/computers13030059.
40. Or-Meir O, Nissim N, Elovici Y, Rokach L. Dynamic malware analysis in the modern era—a state of the art survey. ACM Comput Surv. 2020;52(5):1–48. doi:10.1145/3329786.
41. Li C, Cheng Z, Zhu H, Wang L, Lv Q, Wang Y, et al. DMalNet: dynamic malware analysis based on API feature engineering and graph learning. Comput Secur. 2022;122(8):102872. doi:10.1016/j.cose.2022.102872.
42. Trizna D, Demetrio L, Biggio B, Roli F. Nebula: self-attention for dynamic malware analysis. IEEE Trans Inf Forensics Secur. 2024;19:6155–67. doi:10.1109/tifs.2024.3409083.
43. Alhashmi AA, Darem AA, Alashjaee AM, Alanazi SM, Alkhaldi TM, Ebad SA, et al. Similarity-based hybrid malware detection model using API calls. Mathematics. 2023;11(13):2944. doi:10.3390/math11132944.
44. Nisa M, Shah JH, Kanwal S, Raza M, Khan MA, Damaševičius R, et al. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. Appl Sci. 2020;10(14):4966. doi:10.3390/app10144966.
45. Bashir S, Maqbool F, Khan FH, Abid AS. Hybrid machine learning model for malware analysis in android apps. Pervasive Mob Comput. 2024;97(1):101859. doi:10.1016/j.pmcj.2023.101859.
46. Roy KS, Ahmed T, Udas PB, Karim ME, Majumdar S. A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. Intell Syst Appl. 2023;20(2):200283. doi:10.1016/j.iswa.2023.200283.

47. Tariq MA, Khan MI, Arif A, Iftikhar MA, Khan ARA. Malware images visualization and classification with parameter tunned deep learning model. Metall Mater Eng. 2025;31(2):68–73. doi:10.63278/1336.
48. Vashista NSR, Abhimanyu Kumar Patro K. Enhancing malware analysis using data visualization through shared code and attribute analysis. IEEE Access. 2025;13:107482–98. doi:10.1109/access.2025.3582164.
49. Moussas V, Andreatos A. Malware detection based on code visualization and two-level classification. Information. 2021;12(3):118. doi:10.3390/info12030118.
50. Qi X, Liu W, Lou R, Li Q, Jiang L, Tang Y. MC-ISA: a multi-channel code visualization method for malware detection. Electronics. 2023;12(10):2272. doi:10.3390/electronics12102272.
51. Bakour K, Ünver HM, Ghanem R. The android malware detection systems between hope and reality. SN Appl Sci. 2019;1(9):1120. doi:10.1007/s42452-019-1124-x.