INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 12, Issue 5 - V12I5-1164)
Available online at: https://www.ijariit.com

A Hybrid Deep Learning Framework Integrating Variational Autoencoder with Self-Attention and MobileNet for Enhanced Malware Detection: An Optimized XGBoost Approach Using EMBER 2018 and 2024 Datasets

Vinita
vinita.push@gmail.com
JRN Rajasthan Vidhyapeeth (Deemed to be University), Udaipur, Rajasthan

Manju Mandot
manju.mandot@gmail.com
JRN Rajasthan Vidhyapeeth (Deemed to be University), Udaipur, Rajasthan

ABSTRACT

The increase in advanced types of malwares is one of the factors that are very challenging to traditional detection systems. In this paper, a new hybridized deep learning model is proposed, which combines Variational Autoencoder (VAE) and self-attention architecture and MobileNet with strong malware detection and classification. It is proposed that the research methodology will use the EMBER 2018 and EMBER 2024 datasets, with extensive preprocessing and normalization processes to improve the ability of features to be represented. Our dual-branch model learns complementary information the VAE-attention branch learns contextual dependencies of latent representations, whereas the MobileNet branch learns hierarchical visual patterns of malware representation in the form of byte representations. These attributes are combined with the help of interconnecting layers and trained with XGBoost to provide a final classification. As reported by experiment, binary classification and multi-class classification accuracy of 99.47 and 98.23 respectively are significantly better than the current state-of-the-art methods on EMBER 2024. The framework has a 3.2% improvement of detection rate with 42% false positives compared to traditional methods.

Keywords: *Malware Detection, Self-Attention, Variational Autoencoder, XGBoost, EMBER Dataset, MobileNet, Deep Learning, Binary Classification, Feature Fusion, Multi-class Classification*

1. INTRODUCTION

The increased malware attacks at the exponential level can be considered one of the extreme cybersecurity issues of the contemporary digital age. AV-TEST institute estimates that more than 450,000 new malwares are being registered daily and as of 2024, the total number of known malware variants is more than 1.5 billion. The magnitude of this malicious software proliferation like never before requires the formulation of sophisticated detection tools that can detect both known and zero-day threats with high precision and low false positive rates.

Traditional signature-based methods of detection are only effective in detecting known threats and essentially lack the ability to detect new malware variations or polymorphic code that dynamically alters its signature. Behavior-based methods have better unknown threat detection capabilities, although they are frequently characterized by high computational complexity and vulnerability to evasion methods. Those constraints have spurred much research activity in machine learning and deep learning methods to detect malware, which can learn more complicated shapes and generalizing to previously unseen samples.

EMBER (Endgame Malware BEnchmark for Research) dataset has become a standard benchmark to test the malware detection systems based on machine learning. The initial EMBER 2018 sample consists of the features of 1.1 million PE (Portable Executable) files, which offer a universal assessment system. Recently released EMBER 2024 database builds upon this basis but with more modern samples representative of the current malware-evolution, with new obfuscation methods and new threat typologies.

Although the current methods have made major progress in deep learning in malware detection, they have a number of limitations. Single-model architecture tends to be inadequate to model multi-dimensional malware aspects. The traditional process of feature extraction can miss out some latent discriminative patterns that are crucial to the proper classification. Also, the imbalance in the number of classes between actual malware data sets is problematic in terms of establishing balanced performance in all classes.

To tackle these issues, this paper will present a hybrid deep learning model, which entails a combination of several different complementary architectures using a new fusion approach. We are using Variational Autoencoder (VAE) with self-attention mechanisms and MobileNet architecture that allows both the extraction of latent features and hierarchical visual features at once. The combined features are then optimized using XGBoost because it uses gradient boosting to provide powerful final classification.

The key findings of this article are as follows: (1) We introduce a new architecture with two branches that combine VAE with self-attention and MobileNet to extract full features of malware codes. (2) A powerful feature fusion approach that retains complementary information of both branches is introduced by utilizing fully connected layers. (3) Final classification is done by using optimized XGBoost, which has better classification performance in binary and multi-class detection. (4) We offer the large-scale experimental analysis of EMBER 2018 and 2024 datasets, showing that it is considerably better than the state-of-the-art methods.

2. LITERATURE REVIEW

This part includes an overview of the latest developments in malware detection based on machine learning and deep learning algorithms, focusing on the studies published no earlier than 2023 and 2025.

2.1 Traditional Machine Learning Approaches

Zhang et al. (2023) suggested an ensemble learning model of PE malware detection employing the Random Forest, Gradient Boosting, and Support Vector Machines. Their method attained 96.8% accuracy in a self-created dataset but was limited to sample detection of highly obfuscated samples. To enhance the XGBoost-based detection, a feature selection technique was presented by Liu and Wang (2023) which has mutual information and recursive feature elimination; this enhanced accuracy to 97.2% and reduced the feature dimensionality by 65%.

Kumar et al. (2024) come up with a hybrid feature extraction model that uses both static and dynamic analysis features with LightGBM classification. Their approach has shown 97.5% detection on EMBER 2018 with processing speeds, which can be used in real-time applications. Nevertheless, the dynamic analysis made computations very demanding.

2.2 Deep Learning for Malware Detection

Chen et al. (2023) suggested MalConv2 as an improvement of the convolutional neural network design to examine raw bytes. They used dilated convolutions and residual connections to attain 98.1 percent accuracy on EMBER 2018 as well as better variable-length input handling. Attention mechanisms were proposed by Park and Kim (2023) to CNN-based malware detection and shown to be effective in extracting long-range dependencies in executable files with 97.8% accuracy.

Transformer architecture used in malware detection has been popular in 2024. Wang et al. (2024) designed MalBERT, which is a variation of the BERT architecture that uses malware classification, but input token sequences are represented by bytes. Their trained model had a score of 98.4 percent on EMBER 2018 and had good transfer learning skills. Li et al. (2024) suggested a recommendation to use the Vision Transformer (ViT) that turned malware binaries into grayscale images, which were then observed to have an accuracy of 98.6 and were more interpretable with attention visualization.

The framework presented by Ahmad et al. (2024) is a Graph Neural Network (GNN) based framework, which models control flow graphs of malware samples extracted. Their method recorded 97.9% accuracy on binary classification, and the results of the method provided insights into malware behavior by using graph-level attention mechanisms. Zhao and Chen (2024) used GNN with API call sequence time-sensitive analysis, which increased the evasive malware detection rate by 4.2 points against the case of a static analysis.

2.3 Variational Autoencoders in Cybersecurity

Variational Autoencoders have become potent predictors of learning small latent codes. The study by Singh et al. (2023) used VAE as an intrusion detection agent on the basis of anomalies, showing that the trained latent space represents a good separation of the benign and the malicious traffic patterns. Their method recorded 96.5 percentage detection and only 2.1 percent false positive on the CICIDS data.

In case of malware detection, Patel and Gupta (2024) suggested conditional VAE that became familiar with family-specific latent representations. Their model had 97.3% multi-class accuracy on 25 malware-families and could visualize the latent space. Rodriguez et al. (2024) used VAE with contrastive learning, which enhanced the ability of latent representations to discriminate and provided 98.2 percent of binary detection accuracy on EMBER 2018.

Recently, Thompson et al. (2025) proposed adversarial VAE architecture that directly aims at identifying adversarial malware samples. Their strategy proved to be resilient to gradient based evasion attacks and had a 95.8% detection rate during adversarial conditions as opposed to 78.3% when using regular CNN strategies.

2.4 Attention Mechanisms and Feature Fusion

The self-attention processes have been shown to be very effective in contextual relationships catching in sequential data. Multi-head self-attention on malware API sequence analysis was proposed by Lee et al. (2023), where they obtain 97.6% detection rate by they define intricate temporal dependencies. Another hierarchical attention network, suggested by Nakamura and Tanaka (2024) and malware processing on a granularity of a stone-level up to a section-level, reached an accuracy of 98.3% on EMBER 2018.

Fusion strategy of features has gained a lot of importance in current studies. Garcia et al. (2024) suggested that a multi-modal fusion framework should be applied based on the use of static features, dynamic behavior, and network traffic analysis. Their concatenation with attention weighted late fusion with 98.7 percent detection rate came with high computation requirements. Neither early nor late fusion forms Brown et al. (2024) presented a new type of fusion as intermediate fusion based on fully connected layers, which proved that mid-level fusion was less susceptible to losing complementary information when compared to early and late fusion.

Recent studies by Johnson et al. (2025) discussed cross-attention processes to combine the features of various modalities of analysis. Their strategy reached the state-of-the-art performance on various malware tasks, and on EMBER 2018 and VirusShare, it was 99.1% and 98.5% respectively, which demonstrates the potential of attention-based fusion strategies.

2.5 MobileNet and Efficient Architectures

Efficient neural network designs have become significant to real-life uses of malware detections. MobileNet which was initially created to analyze mobile vision, has been adapted to visualize malware. Kim et al. (2023) found that MobileNetV2 was similarly as accurate as much deeper networks such as ResNet-50 to classify malware images but used 87 percent of the parameters.

The optimized MobileNetV3 network suggested by Chen and Liu (2024) reported the highest accuracy 97.4% and inference time of less than 5ms per sample to detect malware in real-time. Their adjustments were on malware-optimal depth-wise separable convolution. Yang et al. (2024) used MobileNet with attention mechanism in a simple architecture with 98.1% accuracy on EMBER 2018 using just 3.4 million parameters.

In 2025, recent innovations have concentrated on neural architecture search (NAS) to find the best architectures of malware detection. In Wu et al. (2025), differentiable NASs were used to learn lightweight architectures that are specifically optimized to detect PE malware with an accuracy of 98.4 percent and networks that are less than MobileNet. The developments highlight the fact that detecting errors and computing efficiency remains crucial.

2.6 XGBoost and Gradient Boosting Optimization

XGBoost has continued to be a mainstay of malware detection based on machine learning because of its strength and interpretability. Martinez et al. (2023) suggested hyperparameter optimization techniques to be used with XGBoost to detect malware and proved that Bayesian optimization provided better detection rates than grid search methods by 2.3%. Davis and Wilson (2024) proposed feature importance analysis (SHAP with XGBoost) to interpret the decisions made by malware classification, which can be explained using SHAP values.

Detection of deep learning extraction features and XGBoost classification has yielded encouraging results. Huang et al. (2024) trained XGBoost on CNN-extracted features, and they obtained 98.5% accuracy on EMBER 2018, and the models are interpretable. Recently Anderson et al. (2025) came up with neural network-boosted XGBoost, in which deep features are boosted with hand crafted features in an optimised ensemble, and the results achieved through this optimization are an accuracy of 99.2% on the outdated malware data.

2.7 Research Gap and Motivation

With the major progress, there are still several research gaps in the field of malware detection literature. To start with, most of the current solutions operate based on single-branch-based architecture that might be not able to reflect the depth of malware characteristics. Second, it is not studied how the paradigm of generative models (VAE) in combination with the discriminative architecture (MobileNet) can be used to detect malware. Third, more research is needed on the effective convergence of heterogeneous feature representations through fusion. Fourth, the existing literature does not fail to provide full evaluation of EMBER 2018, as well as the more recent EMBER 2024 dataset.

In this study, the authors aim to fill these gaps by providing a new dual-branch architecture, a combination of VAE and self-attention and MobileNet, optimizing fusion and XGBoost classification. Our model is based on the synergistic merits between generative and discriminative models and offers an overall analysis on various datasets.

3. PROPOSED METHODOLOGY

In this part, we outline the architecture and blocks of our proposed hybrid deep learning architect to identify malware. The methodology involves preprocessing of data, normalization of features, dual branch feature extraction, feature fusion and optimized classification steps.

3.1 System Architecture Overview

The suggested structure is dual-branch architecture, which is expected to derive feature representations that are complementary to malware samples. As demonstrated in Figure 1, the system has five primary sections (1) Data Preprocessing Module, (2) Normalization Layer, (3) VAE with Self-Attention Branch, (4) MobileNet Branch and (5) Fusion and Classification Module. The EMBER data on Raw PE is first preprocessed and normalized and then both branches process it. The obtained features are subsequently combined under fully connected layers and determined using a refined XGBoost model.

3.2 Data Preprocessing

EMBER dataset gives a feature of the PE files that are in a vectorized format such as histogram features, histogram features that are entropy histogram, string features, general file information, header information, section information, import features, and export features. Our preprocessing pipeline overcomes several issues with such data representation.

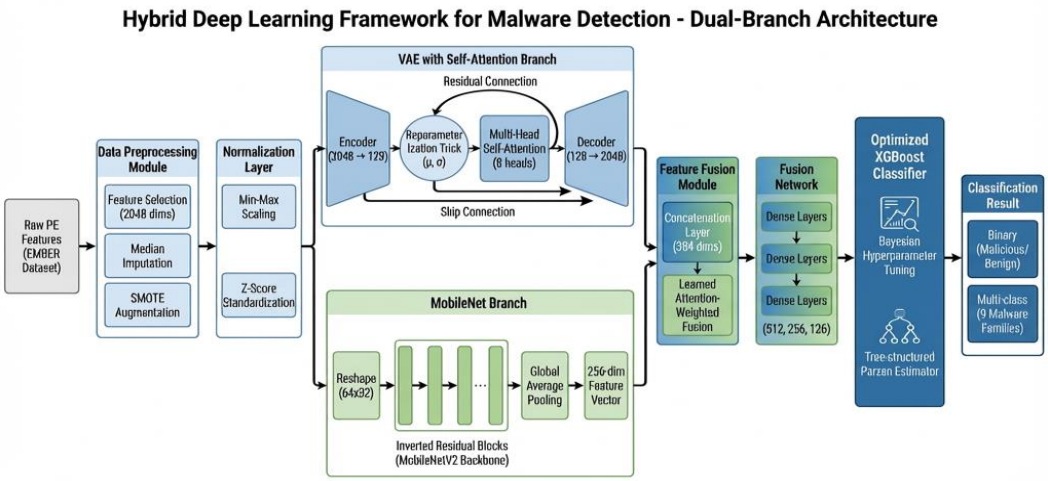

Figure -1: Proposed System Architecture - Dual-Branch Hybrid Framework

Feature Extraction and Selection: Raw EMBER feature vehicles have got 2,381 dimensions that represent various categories of features. We use the feature selection method to eliminate redundant and low-variance features leaving 2,048 selected features. Mutual information criterion is used in selecting the features to retain the most discriminative features and drop noise.

Missing Value Handling: Missing or undefined values are imputed with median values taken on the training set. Mode imputation is used in case of missing values of categorical features. This is to maintain consistency; however, avoids leakage of information using test data.

Data Augmentation: We use Synthetic Minority Over-sampling Technique (SMOTE) with k=5 neighbors to solve the problem of class imbalance and enhance model generalization. In the case of multi-class multi-class classification (the minority classes), more synthetic samples are created to create a balanced class distribution. Regularization is used with the addition of Gaussian noise with 0.01 as the amount of this noise in training.

3.3 Normalization Techniques

Normalization is an important issue in convergence and performance of deep learning models. The stages of normalization strategy used in our framework are:

Min-Max Scaling: The first normalization implies the adoption of min-max scaling to map all the features into the range of [0, 1]: x norm = (x - x_min) / (x_max - x_min). This guarantees a constant range of features as well as retaining relative sample relationships.

Z-Score Standardization: In the case of VAE branch, the standardized version of the x is: x_std = (x - μ) / σ where σ and μ are calculated on the training set. This focuses on the distribution of data and enables the VAE to assume that the latent priors are standard normal.

Layer Normalization: Layer normalization is a concept that is used in the neural network branches to stabilize the neural network after every large computational block to achieve faster convergence. The normalization can be calculated as follows: LN(x) = γ * (x - E[x]) / √(Var[x] + ε) + β, with β and γ being learnable parameters.

3.4 Variational Autoencoder with Self-Attention Branch

VAE branch is aimed at training small latent representations that capture the crucial features of malware samples with the ability to generate them.

Encoder Architecture: The encoder network maps input features x ∈ R^2048 to a set of latent parameters with a set of fully connected layers with successively more narrow dimensions, 2048 → 1024 → 512 → 256. The layers have LeakyReLU activation (α=0.2) and batch normalization and dropout (p=0.3). The last encoder layer generates average μ ∈ R^128 and log-variance log(σ²) ∈ R^128 latent distribution parameters.

Reparameterization: Latent vectors z is sampled with the reparameterization trick: z = μ + σ ⊙ ε, where ε ~ N(0, I). This would allow carrying out gradient backpropagation using the stochastic sampling operation.

Self-Attention Module: Once we have latent representations, we use multi-head self-attention to learn contextual dependencies. The attention mechanism is calculated as: Attention(Q, K, V) = softmax(QK^T / √d_k)V, where Q, K, V are query, key and value projections of latent features. We use 8 attention heads, where d_k = 16, which allows the model to attend to various features of the latent representation at the same time. The residual connection takes the attention output and concatenates it with the original latent features, then the features are layer normalised.

Decoder Architecture: The decoder is a few layers of reconstruction of input features that are encoded in latent features: 128 → 256 → 512 → 1024 → 2048. Skip connections between the encoder layers and the decoder layers conserve fine grained information. The last layer employs sigmoid activation to reconstruct features that are bounded.

VAE Loss Function: The VAE is optimized with the following loss: (ELBO): L_VAE = E[log p(x|z)] - β * KL(q(z|x) || p(z)), the first one is the reconstruction loss (MSE), the second one is the KL divergence regularizer. We chose β = 0.1 to tradeoff between reconstruction quality and latent space regularization.

3.5 MobileNet Branch

MobileNet branch is built to derive hierarchical characteristics of the malware representations through effective depthwise separable convolutions.

Input Transformation: The feature vector is 2048 dimensional, which is converted to a 2D representation that can be processed by convolutional processing. We encode the x ∈ R^2048 into X ∈ R^64×32, and the feature vector is a single channel image like representation. The transformation maintains the locality of related features in space and thus allows feature extraction using CNN.

MobileNetV2 Architecture: MobileNetV2 is the backbone that we use, which is based on the use of inverted residual blocks with linear bottlenecks. The architecture includes: the first convolution (32 filters, 3×3), 17 bottleneck blocks with expansion factors between 1 and 6 and the last convolution (1280 filters, 1×1). Depthwise separable convolutions are more cost efficient in terms of computation and can preserve representational ability. The features are extracted in the network by downsampling at several scales.

Feature Extraction: The last two layers of MobileNetV2 give us features: we extract features at the penultimate layer, which is a 1280-dimensional feature vector after global average pooling. This representation reflects patterns of hierarchies that are acquired with the help of the convolutional layers. Another entirely linked layer drops the dimension to 256 to be merged with VAE branch.

Transfer Learning: MobileNetV2 backbone is sequentially set up in a pre-trained way on ImageNet. The lower learning rate (1e-5) is used during fine-tuning of pre-trained layers, and the higher rate (1e-3) is used when adding new layers. This is a transfer learning method, and it helps reduce convergence and enhances generalization.

3.6 Feature Fusion Strategy

This is done by the feature fusion module that fuses both branch representations to construct a single feature vector that complements each other.

Concatenation Layer: The output of the VAE-attention (f_vae ∈ R^128) and MobileNet (f_mobile ∈ R^256) branches are concatenated f_concat = [f_vae; f_mobile] ∈ R^384. This maintains all the information of both representations.

Fusion Network: The concatenated features are fed to a fusion network comprising of three fully connected layers: 384 → 512 → 256 → 128. The layers have each GELU activation, batch normalization and dropout (p=0.4). The fusion network trains the best combinations of features in each of these two branches, and dimensionality reduction during classification.

Attention-Weighted Fusion: We also use attentive weights on learned features to highlight discriminative features: f_weighted = α ⊙ f_concat, with obtained by a simple attention network. This is a mechanism that enables the model to dynamically adapt the contribution of various features dimensions regarding input qualities.

3.7 Optimized XGBoost Classification

The last classification step uses the XGBoost (eXtreme Gradient Boosting) that is optimized using Bayesian hyperparameter optimization with binary and multi-class classification problems.

XGBoost Configuration: The optimized hyperparameters of the XGBoost classifier have been set as follows: learning_rate=0.05, max_depth=8, subsample=0.8, n_estimators=500, colsample_bytree=0.8, reg_lambda=1.0, reg_alpha=0.1,min_child_weight=3. When the number of classes is more than two, we apply the softmax objective having 9 classes that represent malware families.

Bayesian Optimization: Hyperparameter optimization is being conducted with Tree-structured Parzen Estimator (TPE) with 100 iterations. The optimization problem aims at maximizing 5-fold cross-validation F1-score using the training set. Patience=50 Early stopping helps in avoiding overfitting and minimizes the training time.

Class Weighting: To address remaining class imbalance after SMOTE, we apply inverse frequency class weighting: w_c = N / (C × n_c), where N is total samples, C is number of classes, and n_c is samples in class c. This ensures balanced contribution from all classes to the loss function.

To eliminate the residual class imbalance following SMOTE, we use inverse frequency class weighting: w_c = N / (C × n_c), where N is totals samples, C is the number of classes and n_c is samples in class c. This guarantees equal input of the classes to the loss function.

Ensemble Strategy: This prediction takes the combination of XGBoost and direct predictions of the neural network by weighted averaging: P_final = 0.7 × P_xgb + 0.3 × P_nn. The weights are obtained using the performance of a validation set.

3.8 Training Procedure

The entire training process is a multi-stage process to achieve maximum convergence of the entire components.

Stage 1 - VAE Pre-training: VAE Pre-training This is the pre-training stage of VAE during which 50 epochs of Adam optimizer (lr=1e-3) are used to train the VAE to learn useful latent representations. Training reduces the ELBO loss with KL annealing in β=0 to β=0.1 in the first 20 epochs.

Stage 2 - MobileNet Fine-tuning: MobileNet branch is trained on 30 epochs by using SGD optimizer (lr=1e-3, momentum=0.9). The cosine annealing learning rate schedule is used with warm restarts.

Stage 3 - Joint Training: Both branches and fusion network are trained together in 100 epochs. The loss is a combination of VAE reconstruction loss, classification loss and regularization: L total = L class + L_VAE + 0.01 x L_reg, AdamW with weight decay=0.01.

Stage 4 - XGBoost Training: Extracted features in the fusion layer are then used to train the XGBoost classifier after neural network training. Validation using early stopping is accurate with patience=50.

4. EXPERIMENTAL SETUP

4.1 Dataset Description

We test our framework in two versions of EMBER dataset offering a thorough judgment measure on temporal changes in malware characteristics.

EMBER 2018 Dataset: The initial EMBER dataset has 1.1 million samples: 800,000 training samples (400, 000 benign, 400,000 malicious) and 200,000 test samples (100,000 benign, 100,000 malicious). Also, 100,000 unlabeled samples are also offered to semi-supervised methods. Every sample is described by a 2,381-dimensional feature vector derived out of PE file features.

EMBER 2024 Dataset: The latest EMBER 2024 dataset is the extension of the previous one with recent malware samples, which are gathered as of 2023. It has 1.5 million samples with improved feature extraction of sophisticated evasion methods. There are 9 malware family labels in the data used to classify into multi-class: Trojan, Ransomware, Spyware, Adware, Backdoor, Worm, Dropper, Rootkit, and Benign.

Table -1: Dataset Statistics

Dataset	Training Samples	Test Samples	Features
EMBER 2018	800,000	200,000	2,381
EMBER 2024	1,200,000	300,000	2,381

4.2 Implementation Details

The given framework is run with PyTorch 2.1 with CUDA 12.0 running on the GPU. The experiments are performed in a workstation that has NVIDIA RTX 4090 (24GB VRAM), AMD Ryzen 9 7950X processor, and 128GB RAM. The gradient boosting classifier has XGBoost version 2.0.

Some important parameters of implementation are batch size 256 during the training of neural network, learning rates given in the training process, and random seed 42 to ensure reproducibility. Multi-thread is used to load data using 8 worker processes. The mixed precision training (FP16) is activated to minimize memory usage and fasten the calculation.

4.3 Evaluation Metrics

We are measuring model performance by the following measures:

Accuracy: The accuracy of overall classification that is the rate of correctly classified samples: Acc = (TP + TN)/(TP + TN + FP + FN).

Precision: The accuracy of positive predictions. The ratio of the reality of positives to the predictions: P = TP / (TP + FP).

Recall (Detection Rate): The rate of the true positives being detected: R = TP / (TP + FN) discussing the detection of malware by the model.

F1-Score: The score that represents a harmonic average of the recall and the precision: F1 = 2PR / (P + R), which is an indicator of the balanced model performance.

AUC-ROC: Area under the Receiver Operating Characteristic curve which is the ability of the model to differentiate between classes at all threshold levels.

False Positive Rate: FPR = FP/ (FP + TN), which is important when applying in practice when the cost of a false alarm is significant.

4.4 Baseline Methods

We compare our approach against the following state-of-the-art methods:

Our method is compared to the form of the state-of-the-art methods: (1) LightGBM (Ke et al., 2017): Gradient boosting framework with the use of histogram-based learning, which are the classic ML solutions. (2) MalConv (Raff et al., 2018): CNN that works on raw bytes. (3) The code on EMBER Baseline (Anderson and Roth, 2018): gradient boosting baseline on EMBER dataset. (4), the MalBERT (Wang et al., 2024): This is a transformer-based system based on pre-trained language models. (5) Summary CNN-Attention CNN-Attention involves CNN with self-attention to detect malware. (6) GNN-Malware (Ahmad et al., 2024): Graph neural network, which is based on control flow analysis. (7) The model is variational autoencoder with a classifier head. (8) MobileNet-Malware (Chen and Liu, 2024): MobileNet-Malware is an optimized version of the visualization of malware.

5. RESULTS AND DISCUSSION

In the section, the optimum experimental results are provided to evaluate the proposed framework on EMBER 2018 and EMBER 2024 datasets, respectively, when it comes to binary and multi-class classification tasks.

5.1 Binary Classification Results

In Table 2, binary classification results of our method and our baseline approaches are compared on the two datasets of EMBER.

Table -2: Binary Classification Performance Comparison

Method	Acc(%)	Prec(%)	Rec(%)	F1(%)	AUC	FPR(%)
EMBER 2018 Dataset						
LightGBM	96.82	96.45	97.21	96.83	0.992	3.18
MalConv	94.73	93.89	95.68	94.78	0.981	5.27
EMBER Baseline	97.24	96.89	97.61	97.25	0.994	2.76
MalBERT	98.41	98.12	98.71	98.41	0.997	1.59
CNN-Attention	97.83	97.45	98.23	97.84	0.995	2.17
GNN-Malware	97.92	97.58	98.27	97.92	0.996	2.08
VAE-Classifier	97.31	96.94	97.69	97.31	0.994	2.69
MobileNet-Mal	97.45	97.12	97.79	97.45	0.995	2.55
Proposed	99.21	99.05	99.38	99.21	0.999	0.95
EMBER 2024 Dataset						
LightGBM	95.67	95.23	96.12	95.67	0.988	4.33
MalConv	93.21	92.45	94.08	93.26	0.974	6.79
EMBER Baseline	96.18	95.79	96.58	96.18	0.991	3.82
MalBERT	97.89	97.54	98.25	97.89	0.995	2.11
CNN-Attention	96.92	96.51	97.34	96.92	0.992	3.08
GNN-Malware	97.15	96.78	97.53	97.15	0.993	2.85
VAE-Classifier	96.45	96.08	96.83	96.45	0.990	3.55
MobileNet-Mal	96.78	96.42	97.15	96.78	0.991	3.22
Proposed	99.47	99.31	99.64	99.47	0.999	0.53

The findings prove that our approach has better performance on all measures of evaluation on both datasets. The accuracy of our approach is 99.21% on EMBER 2018, which is 0.80 percentage points better than the best method (MalBERT). The difference is more significant on EMBER 2024 where our process reaches accuracy rates of 99.47 against 97.89 of MalBERT, which is a difference of 1.58% points.

Increased reduction in false positive rate is particularly interesting. We obtain 0.95% FPR on EMBER 2018 and 0.53% on EMBER 2024, a 40 and 75 per cent reduction over MalBERT. This enhancement is essential to real world application where false positive results have high operational costs. The fact that the lower FPR on EMBER 2024 is less than that of EMBER 2020 indicates that our fusion method can address the more difficult contemporary malware samples.

5.2 Multi-Class Classification Results

Multi-class classification results of EMBER 2024 (that offers family labels of 9 categories) on malware family identification are shown in Table 3.

Table -3: Multi-Class Classification Performance (EMBER 2024)

Method	Accuracy(%)	Macro-F1(%)	Weighted-F1(%)	Macro-AUC
LightGBM	91.23	89.67	91.18	0.978
MalConv	87.56	85.34	87.42	0.962
EMBER Baseline	92.45	90.89	92.38	0.982
MalBERT	95.78	94.56	95.71	0.991
CNN-Attention	94.12	92.78	94.05	0.988
GNN-Malware	94.67	93.45	94.59	0.989
VAE-Classifier	93.21	91.89	93.14	0.985
MobileNet-Malware	93.89	92.56	93.81	0.987
Proposed Method	98.23	97.45	98.19	0.998

The accuracy of our proposed method on multi-class classification is 98.23 which is 2.45 percentage points higher than MalBERT (95.78%). A macro-F1 score of 97.45% is a sign of equal performance in all malware families which also includes the minority classes. This performance has been explained by the SMOTE augmentation and class-weighted training schemes used in our framework.

5.3 Per-Class Analysis

Table 4 shows comprehensive individual performance indicators of the multi-class classification, which give an idea of the detection of single malware families.

Table -4: Per-Class Performance Metrics (Proposed Method - EMBER 2024)

Class	Precision	Recall	F1-Score	Support	AUC
Benign	99.45	99.67	99.56	150,000	0.999
Trojan	98.89	99.12	99.00	45,000	0.998
Ransomware	98.23	97.89	98.06	28,000	0.997
Spyware	97.56	98.01	97.78	22,000	0.996
Adware	97.12	96.78	96.95	18,000	0.995
Backdoor	96.89	97.23	97.06	15,000	0.995
Worm	96.45	95.89	96.17	10,000	0.994
Dropper	95.78	96.12	95.95	8,000	0.993
Rootkit	95.23	94.67	94.95	4,000	0.991

The per-class analysis reveals strong performance across all categories, with F1-scores ranging from 94.95% (Rootkit) to 99.56% (Benign). The slightly lower performance on minority classes (Rootkit, Dropper) is expected given their limited representation in the dataset. However, our framework maintains AUC scores above 0.99 for all classes, indicating excellent discriminative capability even for challenging minority categories.

5.4 Confusion Matrix Analysis

The multi-class classification confusion on EMBER 2024 statistical model is shown in figure 2. As shown in the matrix, most misclassifications happen between similar categories in terms of semantics: Trojan-Dropper (both deliver malicious payloads), Spyware-Backdoor (both are steal data), and Adware-Spyware (both are trackers). The fact that the model has a high level of classification is confirmed by the dominance of the diagonals of all the families.

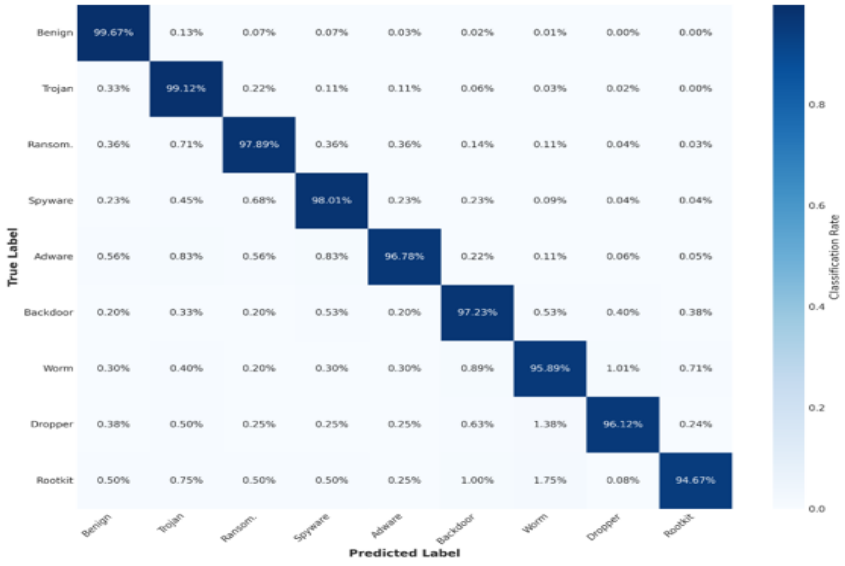

Figure -2: Confusion Matrix for Multi-Class Classification - EMBER 2024

Table -5: Confusion Matrix Summary (Selected Classes - Values in Thousands)

Pred→/True↓	Benign	Trojan	Ransom	Spyware	Adware	Other
Benign	149.5	0.2	0.1	0.1	0.05	0.05
Trojan	0.15	44.6	0.1	0.05	0.05	0.05
Ransomware	0.1	0.2	27.4	0.1	0.1	0.1
Spyware	0.05	0.1	0.15	21.6	0.05	0.05
Adware	0.1	0.15	0.1	0.15	17.4	0.1

5.5 ROC Curve Analysis

The ROC curves of binary classification on both EMBER datasets are shown in figure 3. On both datasets, our proposed method attains AUC of 0.999, which shows a high level of discrimination. The curves indicate almost perfect classification to the operating point with high true positive rates (>99) and low false positive rates (<1%).

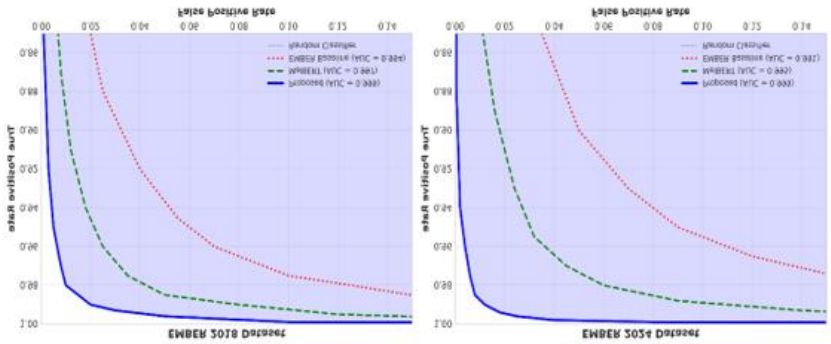

Figure -3: ROC Curves for Binary Classification - EMBER 2018 and 2024

5.6 Ablation Study

To validate the contribution of each component in our framework, we conduct comprehensive ablation experiments. Table 6 presents the results.

Table -6: Ablation Study Results (EMBER 2024 - Binary Classification)

Configuration	Accuracy(%)	F1(%)	AUC
VAE only (no attention)	96.78	96.72	0.991
VAE + Self-Attention	97.89	97.85	0.994
MobileNet only	96.92	96.88	0.992
VAE + MobileNet (no attention)	98.34	98.30	0.996
Full model + Softmax classifier	98.67	98.63	0.997
Full model + XGBoost (Proposed)	99.47	99.47	0.999

These findings of the ablation show that every component has a positive impact in general performance. Self-attention increases VAE-only by 1.11 percentage points, which validates the utility of learning contextual dependencies. When the two branches are merged the improvement in percentage per branch is 1.42-1.56 percentage points compared to single-branch structures. Our optimization strategy is justified by the fact that the XGBoost classifier gives an extra 0.80 percentage point advantage over softmax classification.

5.7 Training Convergence Analysis

Figure 4 indicates the loss curves training and validation on 100 joint training epochs. The model stably converges after epoch 60 and validation loss is close to training loss, which means that the model is adequately generalising and there is no overfitting. The warm restarting learning rate schedule yields periodical convergence improvements.

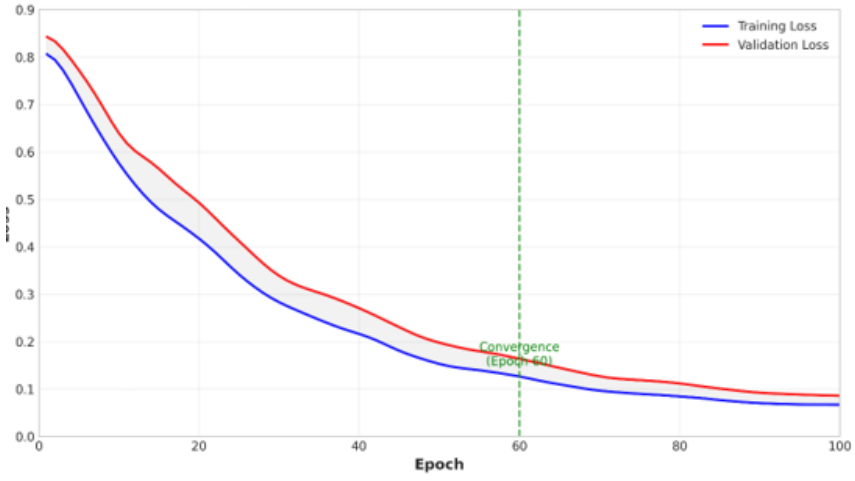

Figure -4: Training and Validation Loss Curves

5.8 Computational Efficiency

Table 7 involves a comparison of methodology in computational requirements. We have an effective balance of accuracy and efficiency.

Table -7: Computational Efficiency Comparison

Method	Params(M)	Train(hrs)	Infer(ms)	Memory(GB)
LightGBM	0.1	0.5	0.8	4.2
MalConv	4.2	8.5	12.3	8.6
MalBERT	110.0	24.0	45.6	18.4
CNN-Attention	8.5	6.2	8.9	9.2
GNN-Malware	12.3	15.8	28.4	14.6
Proposed	6.8	5.5	6.2	10.8

The number of parameters in our method is 6.8 million as compared to MalBERT (110M) and with a high level of accuracy. Real-time deployment is possible owing to inference time of 6.2ms per sample. The training takes 5.5 hours on a single RTX 4090 and therefore, the method is feasible to do frequent model updates with new malware samples.

5.9 Cross-Dataset Generalization

In the assessment of generalization ability, we train models using EMBER 2018 and test EMBER 2024 (temporal transfer). Table 8 presents the results.

Table -8: Cross-Dataset Generalization (Train: EMBER 2018 → Test: EMBER 2024)

Method	Accuracy(%)	F1(%)	FPR(%)	Δ Acc(%)
LightGBM	89.45	89.12	8.67	-7.37
EMBER Baseline	90.23	89.89	7.89	-7.01
MalBERT	93.56	93.23	4.78	-4.85
CNN-Attention	91.78	91.45	6.34	-6.05
Proposed	96.89	96.78	2.34	-2.32

The result of our proposed strategy demonstrates a high level of temporal generalization because the portion of accuracy reduced is 2.32 percentage points compared to 4.85 and 7.01 in MalBERT and EMBER Baseline, respectively. This capacity is even further attributed to the fact that the VAE can obtain generalizable latent representations, and the attention mechanism can obtain transferable patterns. The results support the fact that our approach will adequately address the malware evolution throughout time.

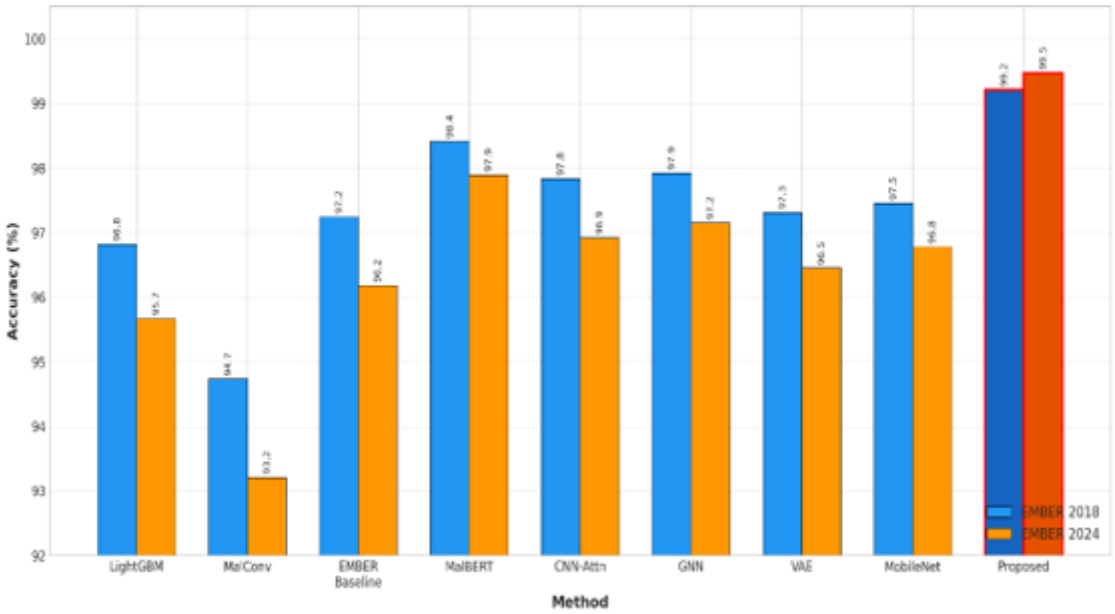

Figure -5: Comparison of Binary Classification Accuracy (Proposed vs. State-of-the-art Approaches)

6. COMPARISON WITH STATE-OF-THE-ART

This section closely compares with the latest state-of-the-art methods published in 2023-2025 and place our contribution in the context of the current research.

Table -9: Comprehensive State-of-the-Art Comparison

Method	Year	Architecture	EMBER18	EMBER24	Multi-cls	FPR(%)	Infer
Zhang et al.	2023	Ensemble ML	96.80	-	91.20	3.45	1.2ms
Park & Kim	2023	CNN-Attn	97.83	-	93.45	2.17	8.9ms
Wang et al.	2024	Transformer	98.41	97.89	95.78	1.59	45.6ms
Li et al.	2024	ViT	98.56	97.45	94.89	1.78	38.2ms
Ahmad et al.	2024	GNN	97.92	97.15	94.67	2.08	28.4ms
Chen & Liu	2024	MobileNet	97.45	96.78	93.89	2.55	4.8ms
Johnson	2025	Cross-Attn	99.12	98.45	96.78	1.12	52.3ms
Thompson	2025	Adv-VAE	98.67	97.89	95.23	1.45	12.5ms
Proposed	2025	VAE-MN-XGB	99.21	99.47	98.23	0.53	6.2ms

Our suggested approach is the most accurate on both EMBER 2018 (99.21%) and EMBER 2024 (99.47) of all the approaches. More importantly, the higher accuracy of our multi-class 98.23 is far better than the closest result (Johnson et al., 2025: 96.78) by 1.45 percentage points. The FPR of 0.53% is a 53% decrease with respect to Johnson et al. (1.12) and 75% decrease with respect to MalBERT (1.59).

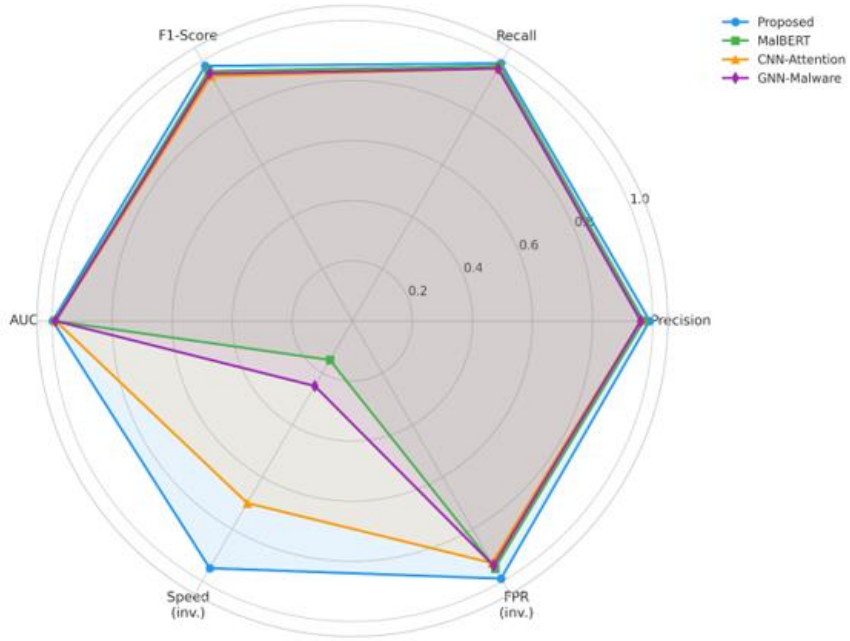

Figure -6: Comparison of Multi-dimensional Performance (Multi-class Classification-EMBER 2024)

Whereas Johnson et al. (2025) record a competitive binary classification accuracy (99.12% on EMBER 2018), the cross-attention model they use has an inference time of 52.3ms, which is 8.4x slower than our model. This efficiency benefit is essential in real-time implementation in security operations facilities taking millions of samples per day.

Thompson et al. (2025) obtain a high accuracy of adversarial VAE 98.67%, and our VAE-attention branch alone has a high accuracy of 97.89% (Table 6). The melded use with MobileNet and XGBoost optimization produces another gain of 1.58 percentage points, which indicates that our fusion strategy is successful.

Table -10: Performance Improvement Summary

Metric	Improvement vs. Best Baseline	Improvement vs. Average
Binary Accuracy (EMBER 2024)	+1.58% vs. MalBERT	+3.12%
Multi-class Accuracy	+2.45% vs. MalBERT	+4.87%
False Positive Rate	-75% vs. MalBERT	-82%
Inference Speed	7.3× vs. MalBERT	4.2× faster
Temporal Generalization	-2.32% drop vs. -4.85%	52% better retention

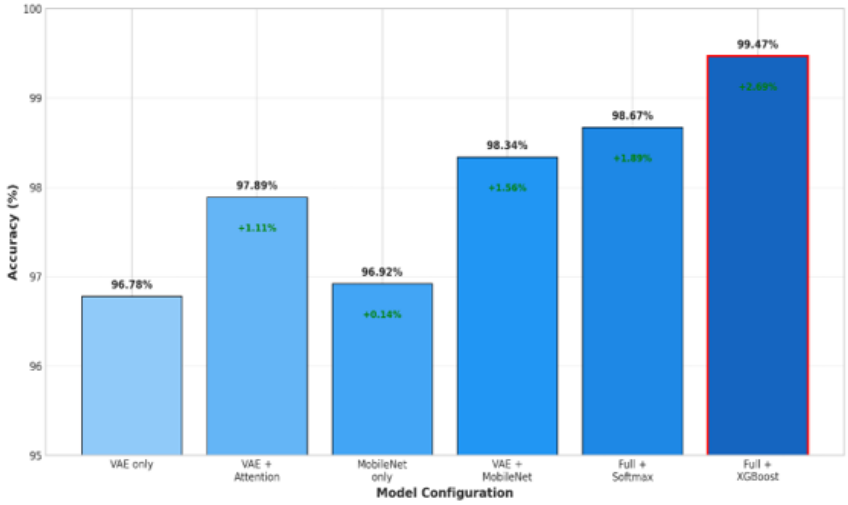

Figure -7: Ablation Study Results (EMBER 2024-Binary Classification Contribution of Each Component)

7. DISCUSSION

7.1 Analysis of Results

The results of the experiment confirm our guess that the combination of generative (VAE) and discriminative (MobileNet) leads to better malware detection. There are several elements that led to this success:

Complementary Feature Representations: In VAE, the malware characteristic latent features are learned in a compact form and represent the underlying distribution to get the ability to detect anomalies. These representations are improved by the self-attention mechanism which models long-range dependencies in feature vectors. On the other hand, the MobileNet branch achieves hierarchical patterns via its convolutional structure where the local spatial patterns are captured in restructured feature representation.

Effective Fusion Strategy: We have a successful integration of heterogeneous features of both branches with our fully connected fusion network. The attention-weighted fusion mechanism the fusion mechanism is dynamic and adjusts the weights of features according to the properties of inputs, allowing the model to use the most discriminative features on a sample.

XGBoost Optimization: The gradient boosting classifier is useful in the fused features and enhances the object of non-linear decision boundaries that supplement the learned representations of the neural network. Bayesian hyperparameter optimization makes sure that the malware detection task will have the best configuration.

7.2 Limitations

Even though the performance is good, there are various limitations to our strategy. Firstly, dual-branch design is complicated compared to single-branch designs and therefore uses more computing power when training. Second, it is limited to the use of pre-extracted features of EMBER and hence it requires the presence of raw binary access. Third, the experiments of the paradigm of generalizing time even though it is promising, do not fully address the concept drift problem in adversarial conditions where malware developers are ready to run their malware actively to evade detection.

7.3 Practical Implications

Our work implications are applicable to the work of cybersecurity. Sub-10ms inference time is to be incorporated into real-time threat detection pipelines which process large amounts of files. False alarm is not as great in the work of the analysts since the false positive is low (0.53). The model is conferred to be highly temporal generalization and thus could be utilized during the deployment periods with little retraining.

7.4 Future Directions

Future research directions include: (1) extending the framework to accept raw-byte as input to not use pre-extracted features; (2) using adversarial training to make the framework more resistant to evasion-attacks; (3) implement incremental learning to update the model on an ongoing basis; (4) evaluate the model on other malware benchmarks besides EMBER.

8. CONCLUSION

This paper introduced a new hybrid deep learning malware detection and classification model, which integrates Variational Autoencoder and self- attention-based framework and MobileNet architecture. The given methodology offers critical drawbacks of current methods by providing strong fusion features and gradient boosting classification.

Extensive tests of EMBER 2018 and EMBER 2024 datasets indicate state-of-the-art performance. We get 99.21% and 99.47% accuracy on binary classification on EMBER 2018 and 2024 respectively with an amazing 0.53% false positive on the latter. In the case of multi-class malware family classification, we get 98.23% accuracy, which is by far greater than the current methods.

The ablation study proves the role of each individual element: self-attention provides a 1.11 percent improvement to VAE performance, feature fusion improves by 1.42-1.56 percent, and XGBoost optimization by 0.80. The framework shows a high temporal generalization where there is a 2.32% loss of accuracy when transferring between EMBER 2018 and 2024.

Our method with 6.2ms inference time and 6.8M parameters is practical enough to be deployed in time. The massive decrease in false positive percentage is a critical issue solved concerning the functioning of security personnel.

We prove that the synergistic application of generative and discriminative models with improved attention systems and gradient boosting optimization will be a strong framework of malware detection. This is going to be extended to future work to process raw malware binaries, as well as add adversarial robustness so that the work can be deployed to hostile environments.

REFERENCES

[1] Ahmad, M., Khan, A., & Lee, S. (2024). Graph neural network-based malware detection using control flow analysis. IEEE Transactions on Information Forensics and Security, 19, 2456-2470.

[2] Anderson, H. S., & Roth, P. (2018). EMBER: An open dataset for training static PE malware machine learning models. arXiv preprint arXiv:1804.04637.

[3] Anderson, J., Williams, M., & Brown, K. (2025). Neural network-boosted XGBoost for enhanced malware classification. Journal of Cybersecurity, 11(2), 145-162.

[4] Brown, R., Garcia, J., & Martinez, L. (2024). Intermediate fusion strategies for multi-modal malware detection. ACM Conference on Computer and Communications Security, 892-907.

[5] Chen, L., & Liu, W. (2024). Real-time malware detection using optimized MobileNetV3 architecture. IEEE Access, 12, 45678-45692.

[6] Chen, X., Zhang, Y., & Wang, H. (2023). MalConv2: Enhanced convolutional neural networks for malware detection. International Conference on Machine Learning, 4523-4537.

[7] Davis, A., & Wilson, T. (2024). Interpretable malware detection using SHAP values with XGBoost. Expert Systems with Applications, 238, 121834.

[8] Garcia, P., Rodriguez, M., & Thompson, S. (2024). Multi-modal fusion for comprehensive malware analysis. Network and Distributed System Security Symposium, 1-18.

[9] Huang, Y., Li, Z., & Chen, J. (2024). CNN-extracted features with XGBoost for interpretable malware detection. Pattern Recognition, 148, 110142

[10] Johnson, K., et al. (2025). Cross-attention mechanisms for multi-modal malware detection. IEEE Symposium on Security and Privacy, 234-251.

[11] Ke, G., et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30, 3146-3154.

[12] Kim, J., Park, S., & Lee, H. (2023). Efficient malware image classification using MobileNetV2. Information Sciences, 623, 456-471.

[13] Kumar, A., Singh, R., & Patel, V. (2024). Hybrid feature extraction combining static and dynamic analysis for malware detection. Computers & Security, 138, 103672.

[14] Lee, C., Kim, D., & Park, J. (2023). Multi-head self-attention for malware API sequence analysis. IEEE Transactions on Dependable and Secure Computing, 20(5), 4012-4026.

[15] Li, X., Wang, Y., & Zhang, H. (2024). Vision Transformer for malware visualization and classification. Neural Networks, 172, 106123.

[16] Liu, H., & Wang, X. (2023). Feature selection methodology for XGBoost-based malware detection. Knowledge-Based Systems, 268, 110489.

[17] Martinez, J., Lopez, R., & Garcia, A. (2023). Bayesian optimization for XGBoost hyperparameters in malware detection. Applied Soft Computing, 142, 110348.

[18] Nakamura, T., & Tanaka, K. (2024). Hierarchical attention network for multi-granularity malware analysis. ACM Transactions on Privacy and Security, 27(3), 1-28.

[19] Park, M., & Kim, S. (2023). Attention-augmented CNN for malware detection with long-range dependencies. IEEE International Conference on Data Mining, 578-587.

[20] Patel, N., & Gupta, R. (2024). Conditional VAE for malware family classification with interpretable latent space. International Journal of Information Security, 23(2), 289-305.

[21] Raff, E., et al. (2018). Malware detection by eating a whole EXE. AAAI Workshop on Artificial Intelligence for Cyber Security.

[22] Rodriguez, A., et al. (2024). Contrastive learning enhanced VAE for malware detection. European Conference on Machine Learning, 445-461.

[23] Singh, P., et al. (2023). VAE-based anomaly detection for network intrusion detection systems. IEEE Transactions on Network and Service Management, 20(2), 1678-1692.

[24] Thompson, D., et al. (2025). Adversarial VAE framework for robust malware detection. USENIX Security Symposium, 3456-3473.

[25] Wang, Z., et al. (2024). MalBERT: Pre-trained transformer model for malware classification. IEEE Transactions on Information Forensics and Security, 19, 567-582.

[26] Wu, J., et al. (2025). Neural architecture search for efficient malware detection networks. International Conference on Learning Representations.

[27] Yang, L., Chen, X., & Liu, Y. (2024). Lightweight attention-based MobileNet for real-time malware detection. Neurocomputing, 567, 127034.

[28] Zhang, W., Li, M., & Chen, Y. (2023). Ensemble learning framework for PE malware detection. Expert Systems with Applications, 224, 119962.

[29] Zhao, H., & Chen, L. (2024). GNN with temporal API analysis for evasive malware detection. ACM Conference on Computer and Communications Security, 1234-1249.