

Malware Classification Using Dynamic Cluster Adaptive Aggregation Graph Neural Network with Prototypical Loss Function

Janardhana D. R.¹, Jayantkumar A. Rathod², Shivanna³ K., Shruthi Shetty J.¹, Niranjan Murthy C.⁴

¹Department of Computer Science and Engineering, Nitte Meenakshi Institute of Technology, Nitte (Deemed to be University), Bengaluru, India

²Department of Computer Science and Engineering, ATME College of Engineering, Mysuru, India

³Department of Computer Science and Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, India

⁴Department of Computer Science and Engineering, Sri Krishna Institute of Technology, Bengaluru, India

E-mails: Jdr5414@gmail.com jayantkumarrathod@gmail.com Shivannak4phd@gmail.com
Shruthi.shetty@nmit.ac.in Nirucs190@gmail.com

Abstract: Malware classification is the process of categorizing malicious software into specific families based on its structural and behavioral features. However, malware classification remains challenging due to high code similarity across different families, leading to suboptimal performance. To address this issue, this research proposes a Dynamic Cluster Adaptive Aggregation Graph Neural Network with Prototypical Loss Function (DCAAGNN-PLF) for effective malware classification. DCAA is incorporated in the GNN to adaptively group similar nodes that effectively capture both global and local structural variations. PLF ensures robust intra-class clustering while enhancing inter-class separation in the feature space that minimizes misclassification. Min-max normalization scales all features to a common range that avoids larger numerical values and enhances model performance. Therefore, the proposed DCAAGNN-PLF achieves better accuracies of 99.90% and 99.25% on the Malimg dataset and the Microsoft BIG2015 dataset, respectively, compared to existing techniques such as Convolutional Neural Network (ConvNet).

Keywords: Dynamic Cluster Adaptive Aggregation Graph Neural Network (DCAAGNN), Inter-class separation, Intra-class clustering, Malware classification, Prototypical Loss Function (PLF).

1. Introduction

Malware refers to types of code that perform unauthorized processes in a computer system, which damage information and compromise the confidentiality, integrity, and functionality of the digital system. Malware attacks are growing significantly and are characterized by multiple variants, rapid propagation, and severe impact, while emerging as one of the most significant security problems on the Internet [1]. Various devices, particularly those with inadequate security measures and limited computational resources, are vulnerable to malware attacks. Hence, accurate malware detection is critical for ensuring robust cybersecurity [2, 3]. The most conventional method for identifying malware is manual analysis, which is challenging and time-consuming. Malware detection relies on two primary approaches: Static and dynamic analysis [4]. Static analysis is performed without executing the malware, which is an effective step to identify malicious patterns [5]. In this type, the application's binary file is analyzed by utilizing specialized tools to convert the machine code into an assembly language for analysis [6, 7]. Dynamic analysis executes a malware sample in a controlled environment, such as a sandbox, to observe the runtime behavior and analyze the generated artifacts using software [8]. In 2020, an average of 360,000 new malware files was identified daily, which represents an increase of 5.2% [9, 10].

Malware proliferation provides the utilization and availability of sophisticated and automated malware tools such as Zeus, SpyEye, and Denial of Service (DoS) attacks [11, 12]. The system is exposed to different attacks, namely Man-In-The-Middle (MITM), Denial of Service (DoS), malware, and command injection; hence, robust security measures are designed to mitigate these threats [13]. The visualization of malware represents a significant alternative way to analyze it. It transforms binary malware files into 2-dimensional gray-level images, and later, the malware is converted into 8-bit vectors, generally strings of ones and zeros [14-16]. These vectors are then organized into 2D matrices, resulting in grayscale images [17]. Machine Learning (ML) and Deep Learning (DL) techniques greatly enhance malware detection by using large datasets to effectively identify malicious patterns [18]. However, ML techniques are heavily based on manual feature engineering that struggles to identify sophisticated and evolving malware [19, 20]. Moreover, DL automatically learns intricate and high-dimensional feature representations, which enables a more adaptive and accurate process for advanced threats.

1.1. Motivation and research gaps

Recent studies explore numerous ML and DL methods for malware classification, namely Convolutional Neural Network (CNN), encoding-based, and graph-based

methods. CNNs extract spatial features from malware images effectively but struggle to capture intricate structural dependencies. Encoding-based methods conserve semantic information but suffer from high computational complexity and scalability problems. Graph-based methods, such as Graph Neural Networks (GNN), capture structural relationships more effectively but employ uniform aggregation that considers all neighboring nodes equally, which affects discriminative and structural patterns in highly similar malware families. Despite these advancements, existing methods struggle in distinguishing malware families with high structural similarity, maintaining intra-class compactness, and enhancing inter-class separation in embedding space. Hence, a robust graph-based method is required to aggregate structurally similar components, which enhance discriminative feature representation.

1.2. Goal and objective

The primary goal is to propose a robust and discriminative malware classification to distinguish highly similar malware families effectively through preserving structural dependencies and enhancing feature separation. To achieve this goal, the objective is to develop a Dynamic Cluster Adaptive Aggregation Graph Neural Network with Prototypical Loss Function (DCAAGNN-PLF) that captures structural relationships within malware representation more effectively than classical GNNs. DCAA is employed to group structurally similar nodes and minimize redundant feature aggregation, preserving discriminative patterns in malware graphs. Moreover, PLF is used to enhance intra-class and inter-class separation in the embedding space, which maximizes classification robustness. The proposed DCAAGNN-PLF is evaluated on benchmark datasets such as Maling and Microsoft BIG2015, and its performance is compared with existing ConvNets using evaluation metrics.

1.3. Contribution

The main contribution of this research is as follows:

GNN effectively learns structural and relational information of processing data as a graph and aggregating features from nearby nodes. This enables GNN to capture both global and local dependencies that lead to enhanced representation learning and more accurate malware classification.

DCAA is incorporated in GNN to enhance feature learning by dynamically clustering similar nodes, which capture both global and local dependencies effectively. DCAA preserves discriminative structural features by minimizing redundant information through adaptive aggregation.

PLF enhances the model performance by providing intra-class clustering and maximizing inter-class separation in the feature space, which results in highly accurate and robust malware family classification.

The remaining sections are organized as follows: Section 2 provides the literature survey. Section 3 defines the proposed methodology, Section 4 demonstrates the experimental results, and the conclusion is presented in Section 5.

2. Literature survey

This section is organized into three subsections to provide a structured overview of existing methods for malware classification. CNN-based methods, encoding-based methods, and graph-based methods are the three subsections presented, which are explained below in detail.

2.1. CNN-based methods

Kim, Paik and Cho [21] presented an attention-based cross-model Convolutional Neural Network (CNN) to determine malware families by utilizing static features of different modalities. The presented approach had two modalities: structural entropies and malware images that were extracted from obtained files. Both modalities had different granularities of bytes and chunks that complemented each other. However, attention-based cross-model CNN overfitted specific malware patterns, which minimized their ability to generalize to new malware variants.

Singh et al. [22] established multilevel feature extraction and classification using DL for classifying malware families. Gated Recurrent Unit (GRU) was used to extract the features and fed into a CNN, which produced a feature vector. Nevertheless, a multilevel model led to redundant feature representation, which facilitated feature overlap among different malware families and made it harder to distinguish.

Liu et al. [23] developed a Convolutional Neural Network (ConvNet) to detect malware effectively. To provide uniform learning of the model, a Gray-Level Co-occurrence Matrix (GLCM) was employed to design three channels that allowed the model to extract the features for grayscale images. However, ConvNet suffers from detecting malware effectively due to its limited ability to capture contextual dependencies in code, which led to misclassification.

Guan and Zhang [24] introduced malware classification using an improved ResNet50 and Visual Geometry Group-16 (VGG16) Feature Fusion (MC-KDFF) to identify attacks. The improved ResNet50 was used to extract the features, and an improved VGG16, as well as a triple attention mechanism, were employed, which enhanced the feature extraction stage by capturing the significance of the cross-

dimensional tensor features. Furthermore, a global attention block was incorporated to improve the tracking features that enhanced robustness. Nevertheless, MC-KDFF struggled to distinguish highly similar malware variants, which minimized the model performance.

Hafeth and Abdullahi [25] established a Hybrid ResNet-Transformer Network (HRT-Net) and an Improved Grasshopper Optimization Approach (IGOA) to detect malware. ResNet50 was used for extracting local features, and the transformer concentrated on intricate patterns and long-range dependencies using multihead attention. GOA was enhanced by utilizing dynamic inertia motion, weight, and dynamic mutation to choose an ideal subset of features. Finally, an ensemble learning method was utilized for classifying malware samples accurately.

2.2. Encoding-based method

Yu et al. [26] presented a lossless encoding technique to visualize the code structure, which enabled unlimited processing of malicious code images of any size. A bytecode file was converted into semantically lossless images, and then interleaving encoding was utilized to preserve the semantic information. The lossless encoding outcomes were integrated with global contextual and local receptive field features. Finally, feature maps were segmented into numerous row patch blocks, and the transformer input elements were adjusted to cache and combine the hidden states of each block. Nevertheless, lossless encoding produced extremely large image files for complex malware, which resulted in inefficient storage.

2.3. Graph-based method

Someya, Otsubo and Otsuka [27] presented a Self-Attention Function Embedding-GNN (SAFE-GNN) to classify malware. SAFE-GNN refined function embedding using function call relationships in the graph structure, which enabled more accurate malware classification. The self-attention network processed an instruction vector as sequential data by utilizing a Recurrent Neural Network (RNN) that assigned higher weights for appropriate information. However, SAFE-GNN struggled to classify heavily packed malware due to code transformation distorting instruction semantics and function call patterns.

In overall analysis, existing techniques suffer from overfitting, redundant feature representation, limited ability to capture contextual dependencies, and high similarity in code patterns across different families. To address this issue, DCAAGNN-PLF is proposed for malware classification by dynamically establishing node clusters and aggregating features that minimize redundant

representation and capture contextual dependencies. PLF enables intra-class clustering and maximizes inter-class separation, which minimizes misclassification.

3. Proposed methodology

This research proposes a DCAAGNN-PLF for effective malware classification, which enhances accuracy against diverse malware variants. Initially, malware code is gathered from two benchmark datasets to determine the model's performance. These codes are converted into colorful image representations using a color-map module. Later, the converted images are preprocessed using min-max normalization, and finally, the proposed method is used to classify the malware effectively. Fig. 1 depicts the workflow of the overall methodological process.

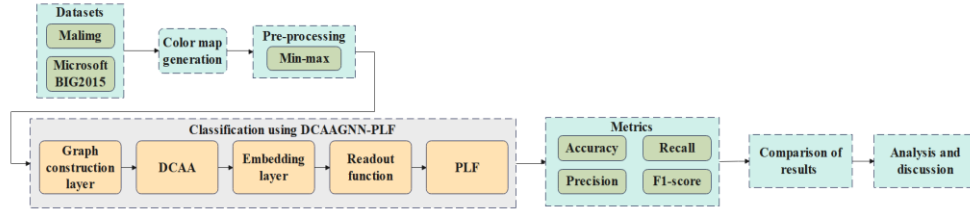


Fig. 1. Workflow for the overall methodology process

3.1. Dataset

The Maling dataset and the Microsoft BIG2015 dataset are used to assess the effectiveness of the proposed method. These datasets provide various malware samples, which provide a comprehensive evaluation of the classification performance. The explanation of these datasets is provided below:

Maling. This is a benchmark dataset to categorize malware that is available on Kaggle. It involves 9435 [28] executable files of malware from 25 different families, each represented as grayscale images generated from the binary content of files, which is appropriate for malware classification.

Microsoft BIG2015. This dataset has 10,868 [29] malware assemblies and binary files from nine families. The files are generated using an Interactive DisAssembler (IDA) tool without a file header. All files have a hash value of 20 characters, which is utilized to uniquely determine a malware file. Table 1 provides an overview of the dataset, where 80% of the samples are used for training; the rest of the files are used for testing. The obtained files are passed as input to the color map module.

Table 1. Overview of the dataset with the number of samples per malware family

Description	Maling	Microsoft BIG2015
Number of samples	9435 files	10,868 files
Number of malware families	25	9

3.2. Color map generation

After gathering files, color map generation is performed to convert executable malware files into colorful images, which allows DL to detect and classify malware more effectively. The visualization of executable malware binaries is used to convert raw hexadecimal byte sequences into image representations, which enable DL methods to effectively capture patterns. A malware binary bit string is divided into substrings of 8 bits each. Each of these substrings associates to a pixel as 8 bits that indicate an unsigned integer in the range of 0 to 255. After converting into binary, the malware bit string is transformed into a 1-dimensional vector in decimal form. Based on a specified width, this 1-dimensional array is reshaped into a 2-dimensional matrix of defined width. For uniformity, each sample is standardized to an image dimension of 256×256 pixels, which ensures consistent input size across all malware families. Later, the matrix corresponding to the malicious code is converted into a grayscale image. To enhance visual interpretability and show subtle variations, a color map module is used to map single-channel grayscale intensities into a 3-channel Red Green Blue (RGB) color representation, which enables spatial information to be more prominent. The entire byte sequence without padding retained full data integrity and obtained stable classification accuracy with less than 0.3% variation compared to resized versions, which confirms that the conversion process does not distort critical information.

3.3. Preprocessing

Following the color map module, min-max normalization is employed to ensure that all pixel intensity values across RGB channels are scaled consistently within the $[0, 1]$ range. Due to the applied color transformation, uneven intensity distribution is introduced among channels. Hence, min-max normalization is used to rescale the pixel values, which enhances stability and the training process by preventing large gradient updates and ensuring rapid convergence. A mathematical formula for min-max normalization [30] is formulated by employing the next equation,

$$(1) \quad X_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}},$$

where X_{scaled} indicates the normalized value, x demonstrates the original pixel value, and $x_{\min}$ and $x_{\max}$ denote the minimum and maximum values, respectively. Thus, min-max maintains a consistent data scale and enables the model to perform

better for classifying malware, which leads to enhanced training reliability and classification accuracy. These preprocessed data are fed into the classification stage using the DCAAGNN-PLF.

3.4. Classification

The DCAAGNN-PLF is performed to extract and categorize malware from preprocessed images. These images are split into small fixed patches and considered as nodes and edges, which are created according to the spatial adjacency between neighboring patches. These nodes and edges form a graph indicate global and local structural patterns within the malware image. Subsequently, this graph is processed through GNN for learning meaningful dependencies for accurate malware classification. Unlike CNN fails to capture hidden structural dependencies in malware patterns, malware is not only defined by local patterns but also by complex dependencies that exist over various functional binary regions. However, GNNs represent malware samples as graphs, where nodes indicate malware regions and edges capture structural dependencies, which ensures that the model integrates both global structural relationships and local neighborhood features. DCAA is incorporated into GNN by dynamically grouping nodes into clusters of structural similarity at each layer, which emphasizes inter-cluster and intra-cluster communication. This strengthens the informative structural pattern while minimizing redundant features. Through this process, DCAAGNN-PLF performs robust and structurally aware embedding for malware samples, which enhances the classification accuracy. The process of the proposed DCAAGNN-PLF is explained as follows.

DCAAGNN-PLF: GNN considers the input as a graph G , where edges and nodes are related to feature vectors x_i and $x_{(i,j)}^e$. The message-passing process in an L -layered GNN contains two significant stages: node embedding and message aggregation update, which enable the network to learn meaningful node representations. In a conventional GNN, all neighboring nodes equally contribute during feature aggregation, which results in the dilution of discriminative malware features and minimizes overall performance. DCAA solves this issue by employing soft k -means clustering with a trainable cluster centroid, where each node obtains probabilistic membership. Unlike hard k -means that require discrete assignments and interrupt gradient flow, soft assignments enable the network to dynamically adjust clusters as feature representations evolve, which helps capture fine-grained relationships among data points. DCAA initializes all cluster centroids by utilizing Xavier uniform initialization, depending on the embedding dimension, to enable stable variance during training.

During each iteration, centroids are refined by utilizing updated node embeddings, and then cluster assignments are recalculated. This iterative process enables the nodes to shift among clusters as representation evolves, which ensures that the clustering structure remains aligned with a learned feature space. The update frequency for these clusters occurs at every training step, where Exponential Moving Average (EMA) is used for gradually updating the centroid, which ensures that the centroid remains stable. Furthermore, the objective for cluster assignments concentrates on minimizing intra-cluster distances to enable embedding in each cluster to be more compact, thereby maximizing inter-cluster separation to facilitate clear discrimination among diverse feature groups. This provides better performance of latent representation and supports accuracy and interpretability. The gating mechanism employs a learnable bilinear mechanism that considers the cluster centroid as input and outputs a gating score between 0 and 1, which controls the amount of information and enables the model to emphasize appropriate inter-cluster relationships and minimize noisy ones. Fig. 2 illustrates the structure diagram of the proposed DCAAGNN-PLF, and Fig. 3 depicts the structure of the proposed DCAA.

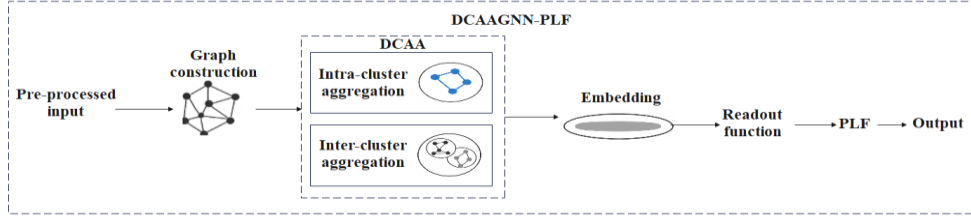


Fig. 2. Structure of the proposed DCAAGNN-PLF

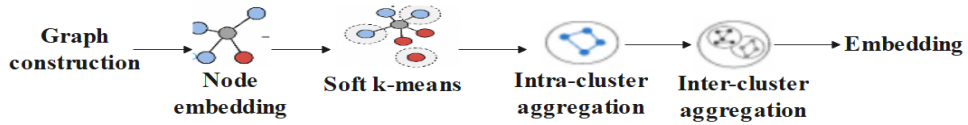


Fig. 3. Structure diagram of DCAA

At each layer, nodes are grouped dynamically into clusters C_i^l , which demonstrates higher-level functional patterns in the malware. During aggregation, neighbors in the same cluster provide more robustness to node representation, which ensures that coherent functional features are reinforced. Moreover, neighbors from various clusters are not ignored, but their influence is managed by a learnable inter-cluster, which allows the information to be shared only when the connection

among clusters is significant. The mathematical formula of the message aggregation is provided in the following equation, for the node i at layer l ,

$$(2) \quad h_i^l = \sum_{j \in N(i)} \alpha_{i,j}^l \cdot f_{\text{message}}^l(h_i^l, h_j^l, x_{(i,j)}^e).$$

Particularly, the hidden node representation is updated by aggregating messages from neighboring nodes. Each message is generated by a parametric function that is considered as input to the hidden representation for the target and source nodes. Each neighboring node is modulated by an aggregation weight, which enables adaptive information propagation. The aggregation weight is specified in the following equation, which distinguishes between intra-cluster and inter-cluster messages,

$$(3) \quad \alpha_{i,j}^l = \begin{cases} \lambda \text{Intra}(C_i^l, C_j^l) & \text{if } C_i^l = C_j^l, \\ (1 - \lambda) \text{Inter}(C_i^l, C_j^l) & \text{if } C_i^l \neq C_j^l, \end{cases}$$

where C_i^l illustrates cluster assignment of a node i at layer l , C_j^l defines the cluster centroid of neighbor node j in layer l , $\lambda \in [0, 1]$ represents the balance of intra vs inter aggregation, $\text{Intra}(\cdot)$ indicates a weighting function that emphasizes neighbors within the same cluster, $\text{Inter}(\cdot)$ demonstrates a gating function that learns which clusters are communicated, h_i determines the hidden representation of nodes i , h_i^l represents the hidden representation of nodes i at layer l , $\alpha_{i,j}^l$ demonstrates the aggregation weight between node i and neighbor node j at layer l , $x_{i,j}^e$ illustrates edge embedding with a connecting node i and neighbor node j , and $N(i)$ denotes a neighborhood set with an incoming edge to the node i . The update function f_{update} and message function f_{message} are parametric functions shared by every node, and each node embedding contains information from neighboring nodes. The mathematical formula of the node update is indicated in the following equation. The neighbor-dependent index j is aggregated via a summation process, which ensures that the resulting node representation depends only on node i .

$$(4) \quad h_i^{l+1} = f_{\text{update}}^l(h_i^l, \sum_{j \in N(i)} h_{(i,j)}^l, C_{\text{summary}}^l),$$

where C_{summary}^l represents the cluster summary embedding, which provides a global cluster during the updates, h_i^{l+1} refers to the updated representation of hidden node i at layer $l + 1$, f_{update}^l illustrates the update function at layer l , which combines node, edge, and cluster-level embedding to generate a refnode representation, and $h_{(i,j)}^l$ represents the hidden representation between node i and neighbor node j at layer l . In the node classification task, a GNN categorizes node labels based on the final representation h_i . In the graph classification task, a GNN categorizes graph labels depending on the global graph representation h_G . Hence, the readout function f_{readout} is mathematically expressed in the following equation,

which means that the final graph embedding is derived from cluster summaries, not just node embeddings.

$$(5) \quad h_G = f_{\text{readout}}(\{h_i^L | v \in G\}).$$

Thus, this formulation enables the learning of robust and cluster-aware embeddings that represent specific malware patterns, thereby making classification more accurate across diverse malware variants.

Moreover, prototypical loss is used to learn a discriminative embedding space, where all classes are indicated by the prototype, which makes accurate classification of test samples possible by using fewer labeled training samples per class in each training episode. This enhances generalization, which is effective in managing diverse attack variants. To update the network, the PLF is calculated within an episode using

$$(6) \quad L_{\text{PLF}} = \sum_{\{x_i, y_i \in Q\}} -\log p_{\theta}(y = y_i | x_i),$$

where x_i, y_i represents query samples x_i with ground truth label y_i , Q indicates the query set, and $p_{\theta}(y = y_i | x_i)$ determines the model's predicted probability that the query sample x_i is assigned to the correct class based on distance to the class prototype. Table 2 represents hyperparameter values for the proposed DCAAGNN-PLF, which are selected using grid search to enable optimal performance and stability. The Adam optimizer is used for adaptive learning, which ensures rapid convergence, and a learning rate of 0.001 offers an optimal balance by enabling the model to learn rapidly while avoiding unstable parameter updates. The PLF loss function is used to perform multi-class classification by enforcing compact intra-class clusters, whereas a hidden size of 256 provides sufficient representation ability without overfitting. A batch size of 32 maintains efficient Graphics Processing Unit (GPU) utilization and a stable gradient. Moreover, three layers are used to provide adequate network depth for capturing hierarchical features that minimize the computational complexity.

Table 2. Hyperparameter values of the proposed DCAAGNN-PLF method

Parameter	Value
Optimizer	Adam
Learning rate	0.001
Loss function	PLF
Hidden size	256
Batch size	32
Number of layers	3
Clustering method	Soft k-means with EMA updates
Gating mechanism	Learnable bilinear
Hyperparameter tuning method	Grid search

4. Experimental results

DCAAGNN-PLF is simulated by employing the Windows 10 operating system, 64 GB Random Access Memory (RAM), and an Intel i5 processor. Accuracy, recall, F1-score, and precision are utilized to estimate the proposed method using equations (7) to (10). Accuracy evaluates the overall correctness of prediction, and precision measures the proportion of correctly predicted positive instances. Recall determines the proportion of actual positives correctly identified, whereas the F1-score analyzes the harmonic mean of precision and recall.

$$(7) \quad \text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100,$$

$$(8) \quad \text{Recall} = \frac{TP}{TP+FN} \times 100,$$

$$(9) \quad \text{F1-score} = \frac{2TP}{2TP+FP+FN} \times 100,$$

$$(10) \quad \text{Precision} = \frac{TP}{TP+FP} \times 100,$$

where TN is the number of cases with True Negative, FN refers to a False Negative, FP denotes a False Positive, and TP demonstrates True Positive.

4.1. Performance analysis

Table 3 presents different DL methods evaluation using the Malimg dataset and the Microsoft BIG2015 dataset. Compared to existing methods such as Deep Neural Networks (DNN), Deep Belief Networks (DBN), CNN, ResNet-Vision Transformer (ResNet-ViT), Zero Shot Learning (ZSL), and GNN, the proposed DCAAGNN-PLF attains better accuracies of 99.90% and 99.25% because it emphasizes meaningful intra-cluster relationships and selectively regulates inter-cluster data flow, which preserves discriminative structural dependencies in malware. Moreover, PLF minimizes intra-family variance by clustering variants of the same malware family while simultaneously enhancing inter-family separation. ResNet-ViT lacks relational reasoning among malware instances, while ZSL struggles to generalize due to limited semantic alignment. Therefore, the proposed DCAAGNN-PLF achieves better performance compared to existing methods.

Table 4 shows the ablation study analysis for each component of DCAAGNN-PLF. This method achieves better accuracy of 99.90% and 99.25% due to the integration of the strengths of both prototype-based learning and dynamic cluster aggregation. The conventional GNN treats all neighbors equally, which leads to obfuscated regions, whereas DCAAGNN without PLF lacks intra-family clustering, which causes overlaps among families. Furthermore, GNN-PLFs without DCAA miss the relational structural dependencies in malware graphs. By integrating

DCNN and PLF, the proposed method captures fine-grained structural relationships that obtain better performance.

Table 3. Performance analysis of different DL models

Dataset	Method	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
Maling	DNN	92.16 $\pm$ 1.83	91.99 $\pm$ 1.47	93.33 $\pm$ 1.56	92.65 $\pm$ 1.87
	DBN	94.50 $\pm$ 1.76	94.03 $\pm$ 1.25	93.98 $\pm$ 1.48	94.00 $\pm$ 1.48
	CNN	96.79 $\pm$ 1.54	95.62 $\pm$ 1.12	94.18 $\pm$ 1.35	94.89 $\pm$ 1.32
	ResNet-ViT	96.82 $\pm$ 1.42	95.89 $\pm$ 1.05	95.36 $\pm$ 1.48	95.62 $\pm$ 0.45
	ZSL	97.12 $\pm$ 0.87	96.20 $\pm$ 0.95	95.45 $\pm$ 0.65	95.82 $\pm$ 0.15
	GNN	97.63 $\pm$ 0.53	97.94 $\pm$ 0.52	95.49 $\pm$ 0.32	96.70 $\pm$ 0.14
	DCAAGNN-PLF	99.90 $\pm$ 0.02	99.85 $\pm$ 0.06	99.87 $\pm$ 0.04	99.85 $\pm$ 0.02
Microsoft BIG2015	DNN	91.76 $\pm$ 1.96	90.78 $\pm$ 1.36	91.01 $\pm$ 1.52	90.89 $\pm$ 1.96
	DBN	92.93 $\pm$ 1.54	91.14 $\pm$ 1.05	90.57 $\pm$ 1.28	90.85 $\pm$ 1.85
	CNN	94.78 $\pm$ 1.02	94.91 $\pm$ 0.48	94.39 $\pm$ 1.10	94.64 $\pm$ 1.12
	ResNet-ViT	94.86 $\pm$ 0.58	94.99 $\pm$ 0.41	94.45 $\pm$ 0.26	94.71 $\pm$ 0.85
	ZSL	95.05 $\pm$ 0.48	93.45 $\pm$ 0.35	94.86 $\pm$ 0.14	94.14 $\pm$ 0.73
	GNN	95.28 $\pm$ 0.36	94.27 $\pm$ 0.26	94.54 $\pm$ 0.09	94.40 $\pm$ 0.23
	DCAAGNN-PLF	99.25 $\pm$ 0.05	98.59 $\pm$ 0.07	98.85 $\pm$ 0.03	98.71 $\pm$ 0.06

Table 4. Ablation study evaluation for individual components of proposed DCAAGNN-PLF

Dataset	Method	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
Maling	GNN	97.63	97.94	95.49	96.70
	DCAAGNN without PLF	98.05	98.65	96.11	97.36
	GNN with PLF	98.71	98.87	97.06	97.95
	DCAAGNN-PLF	99.90	99.85	99.87	99.85
Microsoft BIG2015	GNN	95.28	94.27	94.54	94.40
	DCAAGNN without PLF	96.58	96.08	95.65	95.86
	GNN with PLF	97.44	98.61	97.72	98.16
	DCAAGNN-PLF	99.25	98.59	98.85	98.71

Table 5. Performance of time complexity and statistical analysis

Dataset	Method	Training time (s)	Inference time (s)	Memory consumption (MB)	p-value from t-test
Maling	GNN	110	106	236	0.074
	DCAAGNN without PLF	136	125	205	0.056
	GNN with PLF	126	113	186	0.045
	DCAAGNN-PLF	104	98	168	0.024
Microsoft BIG2015	GNN	137	126	221	0.048
	DCAAGNN without PLF	149	134	206	0.043
	GNN with PLF	150	186	198	0.040
	DCAAGNN-PLF	116	105	184	0.036

Table 5 presents the performance evaluation of computational time complexity and statistical analysis. DCAAGNN-PLF achieves a training time of 104 s and 116 s on both datasets because DCAA focuses only on relevant nodes, which leads

to minimized computation. Moreover, PLF speeds convergence by directly guiding embedding toward class prototypes, thereby avoiding the prolonged process of scattered representations. Hence, the model performs fewer redundant operations and attains quicker training and inference despite having more components.

Table 6 demonstrates the performance evaluation of k -fold cross-validation, which prevents data leakage and contamination by ensuring that the model is evaluated on unseen data during each fold. The dataset is split into k folds; in each fold, one fold is for testing while the remaining $k - 1$ folds are utilized for training. In this analysis, $k=5$ achieves optimal performance due to providing sufficient training data in each fold, which maintains a representative validation set. When $k=3$, the model has higher variance due to limited validation diversity, whereas larger values such as $k=7$ and $k=9$ lead to less generalization ability. Hence, $k=5$ attains high classification performance compared to others.

Table 6. Performance evaluation of k -fold cross-validation

Dataset	k -fold	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
Maling	$k=3$	94.82	93.30	93.74	93.51
	$k=5$	99.90	99.85	99.87	99.86
	$k=7$	97.23	97.99	97.99	97.99
	$k=9$	96.29	95.07	95.66	95.36
Microsoft BIG 2015	$k=3$	95.15	94.14	94.14	94.13
	$k=5$	99.25	98.59	98.85	98.71
	$k=7$	98.25	98.18	97.00	97.58
	$k=9$	96.64	96.96	95.44	96.19

Table 7. Performance analysis using different training/testing split configurations for the proposed method

Training/testing split	Dataset	Accuracy (%)	Precision (%)	F1-score (%)
70:30	Maling	95.78	96.18	95.47
	Microsoft BIG 2015	94.23	93.78	93.89
75:25	Maling	94.87	95.10	94.43
	Microsoft BIG 2015	93.75	91.05	91.75
85:15	Maling	95.68	94.36	94.25
	Microsoft BIG 2015	96.48	96.78	96.13
80:20	Maling	99.90	99.87	99.85
	Microsoft BIG 2015	99.25	98.85	98.71

Table 7 demonstrates the performance analysis of different training and testing split configurations for the proposed method. The 80:20 training-testing split achieves better performance because 80% of the data for training provides the model with sufficient samples for learning intricate feature representation and class distribution. The remaining 20% facilitate statistically significant and unbiased analysis of generalization ability. The splits, such as 70:30, 75:25, and 85:15, minimize the amount of training data that limits the model’s learning ability and

leads to unstable convergence. Hence, an 80:20 split achieves optimal performance and leads to obtaining higher robustness.

Table 8. Class-wise result performance on Maling

Class	Precision (%)	Recall (%)	F1-score (%)
Adialer. C	99.86	99.84	99.85
Agent.FYI	99.88	99.86	99.87
Allaple.A	99.87	99.85	99.86
Allaple. L	99.89	99.86	99.87
Alueron.gen!J	99.84	99.83	99.83
Autorun.K	99.88	99.85	99.86
C2LOP.P	99.87	99.85	99.86
C2LOP.gen!g	99.86	99.84	99.85
Dialplatform. B	99.88	99.86	99.87
Dontovo.A	99.86	99.85	99.85
Fakerean	99.89	99.86	99.87
Instantaccess	99.87	99.85	99.86
Lolyda.AA1	99.86	99.83	99.84
Lolyda.AA2	99.88	99.86	99.87
Lolyda.AA3	99.86	99.84	99.85
Lolyda.AT	99.88	99.85	99.86
Malex.gen!J	99.87	99.85	99.86
Obfuscator.AD	99.86	99.84	99.85
Rbot!gen	99.88	99.86	99.87
Skintrim. N	99.87	99.85	99.86
Swizzor.gen!E	99.88	99.86	99.87
Swizzor.gen!I	99.86	99.84	99.85
VB.AT	99.88	99.86	99.87
Wintrim.BX	99.87	99.85	99.86
Yuner.A	99.88	99.86	99.87
Average	99.87	99.85	99.85

Table 9. Class-wise result performance on Microsoft BIG2015

Class	Precision (%)	Recall (%)	F1-score (%)
Ramnit	98.86	98.60	98.73
Lollipop	98.85	98.58	98.71
Kelihos ver3	98.87	98.61	98.74
Vundo	98.84	98.57	98.70
Simda	98.86	98.59	98.72
Tracur	98.85	98.58	98.71
Kelihos ver1	98.88	98.62	98.75
Obfuscator.ACY	98.84	98.57	98.70
Gatak	98.87	98.60	98.73
Average	98.85	98.59	98.71

Tables 8 and 9 represent the class-wise result performance on the Maling dataset and the Microsoft BIG dataset. For both datasets, the proposed DCAAGNN-PLF exhibits consistent and robust performance for all classes. The proposed method obtains better performance and effectively captures both global and local structural dependencies in malware via DCAA. DCAA emphasizes meaningful

intra-cluster relationships while minimizing redundant inter-cluster information, which preserves discriminative structural features. Furthermore, PLF enhances intra-cluster and inter-cluster relationships that minimize overlap among classes. Therefore, the proposed method obtains high performance across all families, thereby reflecting robustness.

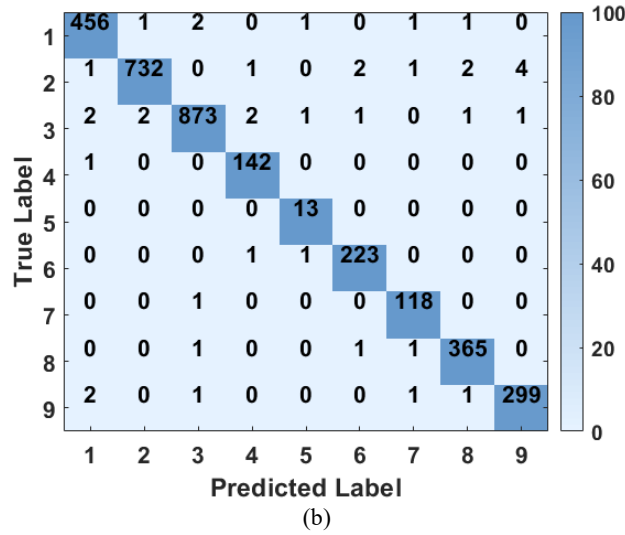
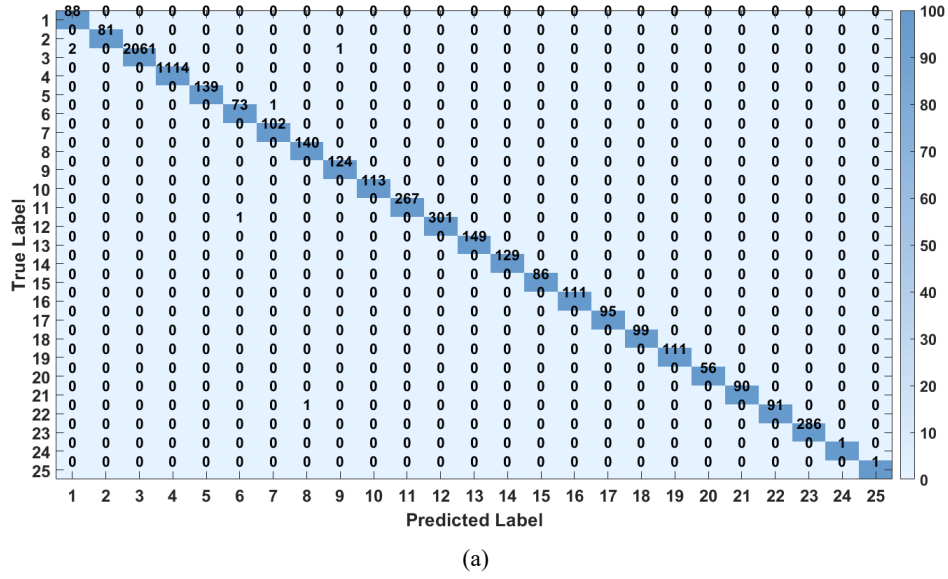
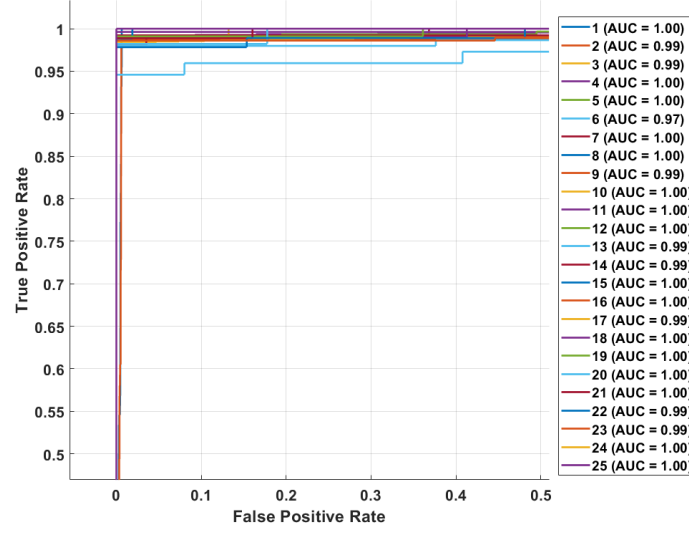
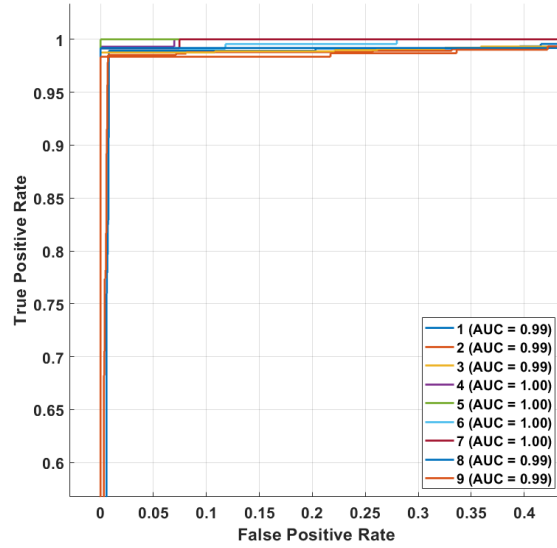


Fig. 4. Confusion matrix of DCAAGNN-PLF: Maling (a); Microsoft BIG2015 (b)



(a)



(b)

Fig. 5. RoC curve evaluation of DCAAGNN-PLF: Malimg (a); Microsoft BIG2015 (b)

Fig. 4 displays the confusion matrix for the proposed DCAAGNN-PLF on two different datasets: a) Malimg and b) Microsoft BIG2015. In the Malimg dataset, the diagonal indicates that most samples from 25 malware families are classified correctly with fewer misclassifications. Likewise, the classifier attains high performance across nine malware families for the Microsoft2015 dataset. A minimal off-diagonal in both datasets represents lower classification rates, which demonstrates model robustness. The confusion matrix shows the effectiveness of the proposed method in distinguishing malware families with high reliability.

Fig. 5 indicates the Receiver Operating Characteristic (ROC) curve of DCAAGNN-PLF: a) Maling and b) Microsoft BIG2015. In Maling, all classes obtain an Area Under Curve (AUC) of 1.0, with minor variations. Likewise, most of the classes also obtain a high AUC in the Microsoft BIG2015 dataset, which provides better discriminative classifier performance. Overall, these outcomes validate the reliability and model efficiency of malware family classification.

4.2. Comparative analysis

Tables 10 and 11 illustrate a comparative analysis of the existing methods. The accuracy values from the existing attention-based cross-model CNN [21] and the F1-score value from the existing CNN+SPP+Transformer [26] are represented in decimal, which are converted into percentages based on the proposed method value. The obtained datasets are split into 80% training and 20% testing. DCAAGNN-PLF obtains better accuracies of 99.90% and 99.25% while effectively capturing the local structure and dynamically grouping the same nodes, which enhances feature representation. PLF enhances classification by providing robust intra-class compactness and enhancing inter-class separation, which minimizes misclassification. This integration balances both local and global graph information, which increases the discrimination among malware families and results in high accuracy.

Table 10. Comparative evaluation of existing techniques on Maling

Method	Accuracy (%)	Recall (%)	F1-score (%)	Precision (%)
Attention-based cross-model CNN [21]	99.09	N/A	97.6	N/A
GRU + CNN + CSBW-RF [22]	99.518	99.52	99.52	99.52
ConvNet [23]	94.22	N/A	N/A	N/A
MC-KDFF [24]	99.50	99.44	99.20	98.97
HRT-Net-IGOA-Ensemble learning [25]	99.85	99.74	99.77	99.80
Proposed DCAAGNN-PLF	99.90	99.85	99.85	99.87

Table 11. Comparative evaluation of existing methods on the Microsoft BIG2015 dataset

Method	Accuracy (%)	Recall (%)	F1-score (%)	Precision (%)
GRU + CNN + CSBW-RF [22]	93.74	93.74	93.69	93.75
ConvNet [23]	96.19	N/A	N/A	N/A
MC-KDFF [24]	97.52	97.39	97.37	97.36
CNN + SPP + Transformer [26]	96.7	N/A	N/A	N/A
SAFE-GNN [27]	99.10	N/A	N/A	N/A
Proposed DCAAGNN-PLF	99.25	98.59	98.71	98.85

4.3. Discussion

The experimental results demonstrate that the proposed DCAAGNN-PLF achieves better performance compared to existing methods. The attention-based cross-model CNN [21] method extracts spatial features effectively but suffers from complex

structural dependencies that lead to misclassification. The lossless encoding method with CNN+SPP+transformer [26] conserves semantic information but has high computational complexity, which limits scalability. In contrast, the proposed DCAAGNN-PLF captures both global and local relationships through cluster-aware aggregation and PLF. This enhances intra-class compactness and inter-class separation, which maintains high accuracy across different malware families. This facilitates robust feature representation, minimizes misclassification, and obtains high accuracy across different malware families. When structural dependencies are complex and intra-class similarity is high, graph-based methods, namely the proposed DCAAGNN-PLF, are preferred over CNN and encoding-based methods. As a result, the proposed method obtains high reliability and superior accuracy compared to the existing methods on both datasets.

5. Conclusion

In this research, DCAAGNN-PLF was proposed to classify malware in both datasets. DCAA enabled dynamic grouping of nodes that vary structural patterns in malware data, which enhanced feature representation by conserving both global and local relationships efficiently. PLF enhanced the model's discriminative ability by clustering samples in the same class while enhancing separation among different classes. This minimized the overlap in the feature space, reduced misclassification, and enabled the learned embedding to be robust across different malware families. Hence, DCAAGNN-PLF obtained better accuracy of 99.90% and 99.25% on Maling and Microsoft BIG2015 datasets compared to ConvNet. Since certain malware families had significantly fewer samples, models exhibited biased learning toward majority classes, which affected generalization performance for minority classes. Also, this research did not evaluate robustness against adversarial attacks, which limited the reliability of the proposed method. In the future, the imbalanced issue will be addressed by utilizing sampling techniques for all malware families. Additionally, future work will focus on extending the evaluation to include robustness against adversarial attacks, resilience to obfuscation methods, and scalability on large datasets. This analysis will demonstrate additional security-oriented metrics to comprehensively assess the model's practical applicability in malware detection.

References

1. Li, J., J. He, W. Li, W. Fang, G. Yang, T. Li. SynDroid: An Adaptive Enhanced Android Malware Classification Method Based on CTGAN-SVM. – *Computers & Security*, Vol. **137**, 2024, 103604.
2. Guo, W., J. Xue, Y. Lin, W. Du, J. Hu, N. Shi, W. Han. MalFSLDF: A Few-Shot Learning-Based Malware Family Detection Framework. – *International Journal of Intelligent Systems*, Vol. **2025**, 2025, No 1, 7390905.
3. Chaganti, R., V. Ravi, T. D. Pham. A Multi-View Feature Fusion Approach for Effective Malware Classification Using Deep Learning. – *Journal of Information Security and Applications*, Vol. **72**, 2023, 103402.
4. Alsubai, S., A. K. Dutta, A. Rahaman Wahab Sait, Y. A. A. Jaish, B. H. Alamer, H. E. H. Saad, R. Ayub. Enhanced Slime Mould Optimization with Convolutional BLSTM Autoencoder-Based Malware Classification in Intelligent Systems. – *Expert Systems*, Vol. **42**, 2024, No 1, e13557.
5. Andriani, S., S. Galantucci, A. Iannacone, A. Maci, G. Pirlo. CNN-AutoMIC: Combining Convolutional Neural Network and Autoencoder to Learn Non-Linear Features for KNN-Based Malware Image Classification. – *Computers & Security*, Vol. **156**, 2025, 104507.
6. Farfoura, M. E., I. Mashal, A. Alkhatib, R. M. Batyha, D. Rosiyadi. A Novel Lightweight Machine Learning Framework for IoT Malware Classification Based on Matrix Block Mean Downsampling. – *Ain Shams Engineering Journal*, Vol. **16**, 2025, No 1, 103205.
7. Vasan, D., M. Hammoudeh, M. Alazab. Broad Learning: A GPU-Free Image-Based Malware Classification. – *Applied Soft Computing*, Vol. **154**, 2024, 111401.
8. Nazim, S., M. M. Alam, S. Rizvi, J. C. Mustapha, S. S. Hussain, M. M. Su'ud. Multimodal Malware Classification Using the Proposed Ensemble Deep Neural Network Framework. – *Scientific Reports*, Vol. **15**, 2025, 18006.
9. Kaithal, P. K., V. Sharma. African Vulture Optimization-Based Decision Tree (AVO-DT): An Innovative Method for Malware Identification and Evaluation through the Application of a Meta-Heuristic Optimization Algorithm. – *Cybernetics and Information Technologies*, Vol. **24**, 2024, No 2, pp. 142-155.
10. Thakur, A. S., K. Muppavaram. Feature Selection Using Adaptive Weight-Based Grey Wolf Optimization for Malware Detection in Android. – *International Journal of Intelligent Engineering & Systems*, Vol. **17**, 2024, No 3, pp. 635-646.
11. Gulatas, I., H. H. Kilinc, A. H. Zaim, M. A. Aydin. I-MCM: IoT Malware Counter Measures for Cross-Architecture IoT Malware Detection. – *IEEE Access*, Vol. **13**, 2025, pp. 95524-95534.
12. Wang, P., H. C. Li, H. C. Lin, W. H. Lin, N. Z. Xie. A Transductive Zero-Shot Learning Framework for Ransomware Detection Using Malware Knowledge Graphs. – *Information*, Vol. **16**, 2025, No 6, 458.
13. Polat, O., A. A. Ahmad, S. Oyucu, E. Algül, F. Doğan, A. Aksöz. Temporal-Spatial Feature Extraction in IoT-Based SCADA System Security: Hybrid CNN-LSTM and Attention-Based Architectures for Malware Classification and Attack Detection. – *IEEE Access*, Vol. **13**, 2025, pp. 102109-102132.

14. Naseer, M., F. Ullah, S. Ijaz, H. Naeem, A. Alsirhani, G. N. Alwakid, A. Alomari. Obfuscated Malware Detection and Classification in Network Traffic Leveraging Hybrid Large Language Models and Synthetic Data. – *Sensors*, Vol. **25**, 2025, No 1, 202.
15. Nwasra, N., M. Daoud, Z. H. Qaisar. ANFIS-AMAL: Android Malware Threat Assessment Using Ensemble of ANFIS and GWO. – *Cybernetics and Information Technologies*, Vol. **24**, 2024, No 3, pp. 39-58.
16. Ismail, S. J. I., B. Rahardjo, T. Juhana, Y. Musashi. MalSSL-Self-Supervised Learning for Accurate and Label-Efficient Malware Classification. – *IEEE Access*, Vol. **12**, 2024, pp. 58823-58835.
17. Al-Masri, B., N. Bakir, A. El-Zaart, K. Samrout. Dual Convolutional Malware Network (DCMN): An Image-Based Malware Classification Using Dual Convolutional Neural Networks. – *Electronics*, Vol. **13**, 2024, No 18, 3607.
18. El Youssofi, C., K. Chougali. YoloMal-XAI: Interpretable Android Malware Classification Using RGB Images and YOLOv11. – *Journal of Cybersecurity and Privacy*, Vol. **5**, 2025, No 3, 52.
19. Aliguliyev, R., R. Aliguliyev, L. Sukhostat. Radon Transform-Based Malware Classification in Cyber-Physical Systems Using Deep Learning. – *Results in Control and Optimization*, Vol. **14**, 2024, 100382.
20. Zhang, D., Y. Song, Q. Xiang, Y. Wang. IMCMK-CNN: A Lightweight Convolutional Neural Network with Multi-Scale Kernels for Image-Based Malware Classification. – *Alexandria Engineering Journal*, Vol. **111**, 2025, pp. 203-220.
21. Kim, J., J. Y. Paik, E. S. Cho. Attention-Based Cross-Modal CNN Using Non-Disassembled Files for Malware Classification. – *IEEE Access*, Vol. **11**, 2023, pp. 22889-22903.
22. Singh, S., D. Krishnan, V. Vazirani, V. Ravi, S. A. Alsuhbany. Deep Hybrid Approach with Sequential Feature Extraction and Classification for Robust Malware Detection. – *Egyptian Informatics Journal*, Vol. **27**, 2024, 100539.
23. Liu, Y., H. Fan, J. Zhao, J. Zhang, X. Yin. Efficient and Generalized Image-Based CNN Algorithm for Multi-Class Malware Detection. – *IEEE Access*, Vol. **12**, 2024, pp. 104317-104332.
24. Guan, X., G. Zhang. A Malware Classification Method Based on Knowledge Distillation and Feature Fusion. – *IEEE Access*, Vol. **13**, 2025, pp. 51268-51276.
25. Hafeth, A. A., A. I. Abdullahi. An Efficient Malware Detection Method Using a Hybrid ResNet-Transformer Network and IGOA-Based Wrapper Feature Selection. – *Electronics*, Vol. **14**, 2025, No 13, 2741.
26. Yu, Y., B. Cai, K. Aziz, X. Wang, J. Luo, M. S. Iqbal, P. Chakrabarti, T. Chakrabarti. Semantic Lossless Encoded Image Representation for Malware Classification. – *Scientific Reports*, Vol. **15**, 2025, No 1, 7997.
27. Somya, M., Y. Otsubo, A. Otsuka. Graph Neural Network-Based Function Call Graph Embedding for Malware Classification. – *Journal of Surveillance, Security and Safety*, Vol. **4**, 2023, No 2, pp. 47-61.
28. Maling dataset (Accessed on 22 September 2025)
<https://www.kaggle.com/datasets/manmandes/maling>
29. Microsoft BIG2015 dataset (Accessed on 22 September 2025).
<https://www.kaggle.com/c/malware-classification>

30. Hasan, R., B. Biswas, M. Samiun, M. A. Saleh, M. Prabha, J. Akter, F. H. Joya, M. Abdullah. Enhancing Malware Detection with Feature Selection and Scaling Techniques Using Machine Learning Models. – Scientific Reports, Vol. **15**, 2025, No 1, 9122.
31. Liu, Z., D. Yang, Y. Wang, M. Lu, R. Li. EGNN: Graph Structure Learning Based on Evolutionary Computation Helps More in Graph Neural Networks. – Applied Soft Computing, Vol. **135**, 2023, 110040.

*Received: 16.12.2025; First Revision: 07.02.2026; Second Revision 24.02.2026;
Accepted: 12.03.2026*

© 2026 by the authors. Licensee IICT-BAS, Sofia, Bulgaria. This article is an open access article distributed under the terms and conditions of the **Creative Commons Attribution-NonCommercial-NoDerivatives** license.