DOI 10.15622/ia.23.6.11

H. Dong

CONVOLUTIONAL-FREE MALWARE IMAGE CLASSIFICATION USING SELF-ATTENTION MECHANISMS

Dong H. **Convolutional-free Malware Image Classification using Self-attention Mechanisms.**

Abstract. Malware analysis is a critical aspect of cybersecurity, aiming to identify and differentiate malicious software from benign programmes to protect computer systems from security threats. Despite advancements in cybersecurity measures, malware continues to pose significant risks in cyberspace, necessitating accurate and rapid analysis methods. This paper introduces an innovative approach to malware classification using image analysis, involving three key phases: converting operation codes into RGB image data, employing a Generative Adversarial Network (GAN) for synthetic oversampling, and utilising a simplified Vision Transformer (ViT)-based classifier for image analysis. The method enhances feature richness and explainability through visual imagery data and addresses imbalanced classification using GAN-based oversampling techniques. The proposed framework combines the strengths of convolutional autoencoders, hybrid classifiers, and adapted ViT models to achieve a balance between accuracy and computational efficiency. As shown in the experiments, our convolutional-free approach possesses excellent accuracy and precision compared with convolutional models and outperforms CNN models on two datasets, thanks to the multi-head attention mechanism. On the Big2015 dataset, our model outperforms other CNN models with an accuracy of 0.8369 and an AUC of 0.9791. Specifically, our model reaches an accuracy of 0.9697 and an F1 score of 0.9702 on MALIMG, which is extraordinary.

Keywords: malware classification, cybersecurity, deep learning, transformer.

1. Introduction. Malware analysis is essential for identifying malicious software on a host system and distinguishing it from benign programs. Despite significant advancements in cybersecurity measures, malware remains a potent hazard in cyberspace. Given the challenges posed by malware, accurate classification techniques with efficient computation are vital for safeguarding computers against infection or effectively removing malware. The field of malware analysis faces two primary challenges: the swift development and distribution of malware and the use of sophisticated evasion techniques. The internet has accelerated the development and dissemination of new malware, creating a vulnerability that can infect numerous systems despite antivirus defences [1]. To meet the ever-evolving threats of cyberattacks, recent advancements in research focus on real-time ransomware detection and combining machine learning techniques in autonomous protection to combat malware development and distribution and counter sophisticated evasion techniques.

Understanding malware attacks and their methodologies is crucial for developing robust defences against such threats, and this logic is utilised in much of the existing research. Malware files can be analysed using static and dynamic approaches. Static analysis involves examining the code without executing the file, enabling quick diagnosis [2, 3], while dynamic analysis scrutinises the system and network behaviour of malicious files in a controlled environment for a more thorough analysis [2, 4]. A newer method, in-memory analysis, involves analysing memory snapshots of active processes to detect unusual process activation consistently [4, 5]. In the domain of malware analysis, numerous techniques have been deployed; traditional methods like signature-based detection, string analysis, and malware binaries hashing still retain a certain level of effectiveness. However, the sophisticated nature of malware necessitates advanced detection strategies, as traditional ML methods often struggle with high-dimensional data and extracting intricate patterns. A recent survey [6] highlights that DL often outperforms traditional ML techniques in malware detection and analysis tasks, attributed to its automatic feature extraction, capacity to learn non-linear relationships, and prowess in image analysis. DL has long been employed in network intrusion detection [7, 8] and emerged as a viable option for extracting insights in malware analysis, with convolutional neural networks (CNN) gaining popularity due to their strong capability in feature representation learning and working with imagery inputs [2, 4, 9]. The progress in contemporary malware analysis research is indeed valuable; however, the development of malware classification techniques with better accuracy, fast computation, and generalisability is worth researching. Overall, effective malware classification should provide highly accurate performance and scalability to protect against network-wide attacks in various system environments, as well as demonstrate adaptability to evolving threat landscapes.

This paper introduces an innovative malware classification method through image analysis, comprising three phases: translating operation codes into RGB image data, deploying a Generative Adversarial Network (GAN) for adaptive synthetic oversampling, and implementing a self-attention block-based classifier for image analysis. Initially, the binary files of analysed programmes are transformed into RGB image data for imagery analysis. The adoption of the 3-channel RGB format is justified due to its widespread prevalence as the standard colour model in DL frameworks. Pre-trained CNNs are typically trained on large-scale datasets like ImageNet, and this standard enables seamless integration of pre-existing models without extensive modifications, as demonstrated by the practicality of the transfer learning approach in this context, which simplifies implementation and fine-tuning processes. To tackle the challenge of imbalanced software sample classification, we utilise GAN-based oversampling techniques to create synthetic samples for minority classes. Within this framework, the generator is designed as a convolutional autoencoder (AE) with denoising capabilities, pre-trained on the comprehensive dataset; conversely, the discriminator functions as a hybrid classifier to discern the “fake” samples produced by the generator; this strategy mitigates the effects of imbalanced data on model performance. Finally, we adapt the self-attention blocks for image classification. Instead of employing the original ViT model with 16 transformer blocks, we adjust its structure and depth to achieve an equilibrium between accuracy and computational efficiency. The experiments demonstrate that the light transformer model outperforms all CNN models in malware classification challenges, with an accuracy of 0.80 on Big2015 and 0.966 on MALIMG, respectively.

The novelty of the proposed methodology lies in its multifaceted approach to enhancing malware classification via advanced image analysis techniques. First, the integration of GAN for adaptive synthetic oversampling is particularly noteworthy; it not only alleviates the prevalent issue of class imbalance in malware training datasets but does so by generating high-quality, diverse synthetic samples that improve the robustness of the classifier. The use of AE as the generator with denoising capabilities refines this process by ensuring that the synthetic malware images contribute positively to model training without introducing noise. Secondly, the strategic modification of the self-attention block-based classifier by optimising the ViT model structure showcases an advanced approach with both accuracy and efficiency. This tailored adaptation offers a scalable, efficient solution to contemporary cybersecurity challenges.

The paper is structured as follows: Section 2 provides a review of the existing literature on malware analysis, mainly image-based techniques. Section 3 outlines the proposed workflow and techniques. Section 4 details the experimental setup, results, and discussions. Finally, Section 5 offers conclusions and discusses potential future research directions.

2. Related Work. Recent advancements in data mining, machine learning, and deep learning algorithms have significantly enhanced the effectiveness of malware analysis. While various deep learning (DL) models can be utilised for security domains, there is a growing trend towards the application of AE. In study [10] the authors employed an AE model for network-based anomaly intrusion detection and malware classification, aiming to improve performance across different evaluation metrics. Paper [11] conducted an analysis of Android malware using image classification, employing AE with three distinct structures: a feed forward network, CNN, and VGG19, for representation learning. The experimental results underscore the exceptional representation extraction capabilities of CNN-based models. In paper [12] the authors introduced a hybrid DL approach that combines

CNN with bidirectional LSTM for malware detection. Study [13] integrated attention mechanisms with CNN to direct the model's focus towards regions of higher importance for the classification task during the learning process. Paper [9] developed a malware detection framework using Depthwise Efficient Attention Module and DenseNet, using spatial pyramid pooling to improve detection performance and overcome obfuscation sensitivity and computational overhead. The attention-based approach in these models is built upon convolutional layers. Despite the promising results in the referenced paper, there are limitations associated with using convolutional-based attention mechanisms. CNN may struggle to learn representations from long-range dependencies due to the constrained receptive field of convolutional layers, potentially hindering the model's ability to grasp global context and relationships among distant elements in the input sequence.

Recently, transfer learning (TL) has emerged as a prevalent methodology within security research. This technique, by applying knowledge gained from addressing one issue to a similar and related one, can mitigate the issue of inadequate training data and enhance the capability to detect malware or network attacks with great robustness. A notable application of TL involves an automated vulnerability detection method that converts source code into a minimal intermediate representation, employing pre-trained convolutional classifiers for analysis, demonstrating high granularity [14]. Research employing TL frequently favours CNN-based models, not only for malware but also for conducting network traffic analysis. Study [15] introduced a ConvNet model that employs transfer learning for network intrusion detection, markedly improving the detection accuracy for both known and novel attacks, as verified through experiments on the NSL-KDD dataset. Additionally, a ConvNet-based malware detection model, through a novel framework that incorporates a deep unsupervised pre-training clustering technique, surpassed the performance of ConvNets with a shallower structure [16]. TL-based approaches often necessitate deep models, especially for malware imagery analysis. While a shallow CNN model with merely three convolutional layers and one feed forward layer may suffice for network intrusion detection due to fewer features [17], opcode-based malware images typically demand more layers for effective representation learning. In the Malbert framework [18], a deep model comprising twelve encoder blocks for representation learning, a pre-classifier layer for anomaly detection, and a malware classification layer were used. Despite its complex structure, Malbert surpassed other deep learning models, such as LSTM, and ensemble learning models, such as Random Forests. A study comparing pre-trained CNN models for malware classification underscored the efficacy of TL and examined prevalent challenges like over-fitting and high resource consumption, utilising simpler CNN models like MobileNet and ResNet50 [19]. Ultimately, the study revealed the importance of employing deep and complex models, where MobileNet showed more stable outcomes than ResNet50, yet with fewer parameters, indicating a potential equilibrium between classification efficacy and resource efficiency.

The existing literature underscores the effectiveness of CNNs and AEs in security domains, particularly malware analysis capabilities. Furthermore, the adoption of TL strategies, leveraging data from related fields, has been a cornerstone in advancing security solutions. Building upon the remarkable progress spotlighted in recent studies on malware detection through DL techniques, this paper proposes a novel malware analysis framework integrating self-attention mechanisms and GAN oversamplers with convolutional AE as generators. This research aims to utilise the inherent strengths of ViT self-attention frameworks while implementing a relatively lightweight model structure. Moreover, by employing convolutional AEs within GANs to generate oversampled data, this study seeks to address the challenges of imbalanced datasets, a recurrent issue in malware classification tasks.

3. Methodology. Figure 1 illustrates the overall workflow of the proposed method, which comprises three major steps: malware conversion, GAN-based oversampling, and image classification. Transforming malware binaries into images is a common initial step in imagery-based malware analysis approaches, aimed at visually representing binary data for pattern detection using image analysis techniques. Malware files in any binary PE format will have each byte read as an 8-bit unsigned integer before being organised into an imagery array for further processing. Starting with the original malware samples, the binary files are opened in binary reading mode. Each byte of the binary data is then converted into its hexadecimal representation. Subsequently, the hexadecimal values are used to create images: in greyscale images, each hexadecimal value corresponds directly to a pixel value, where 0x00 represents black and 0xFF represents white, with intermediate values translating to shades of grey; in RGB images, the hexadecimal data is distributed across the three-colour channels (Red, Green, Blue). Finally, the numeric array dimensions are reshaped to the desired image array size, typically from 128*128 to 256*256 pixels per channel.

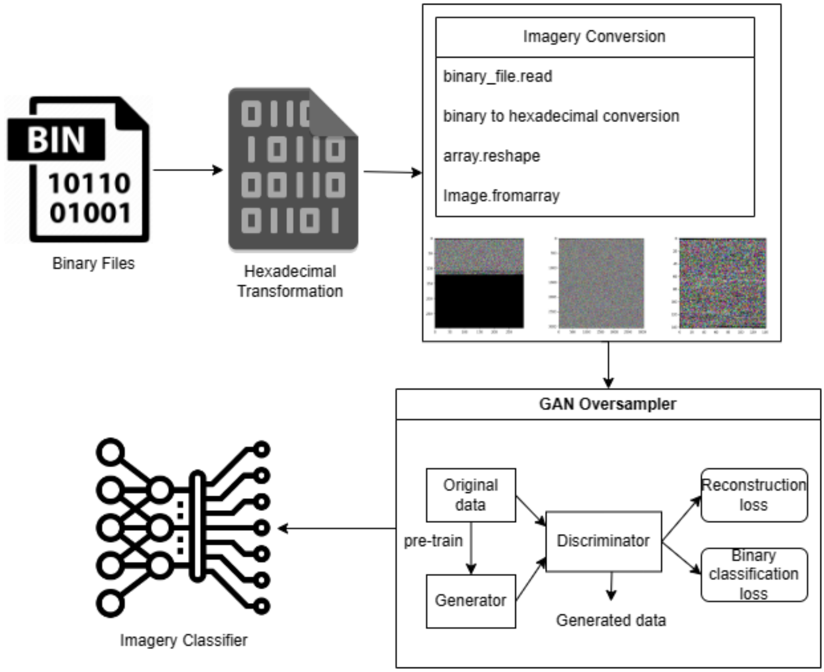


Fig. 1. The workflow from imagery generation to classification

Following data preparation, the initial step involves analysing the distribution of different classes. Since certain types of malware are relatively scarce, data imbalances emerge. To address this issue, a GAN-based oversampling technique is employed to rectify the skewed class distribution by generating synthetic samples. Lastly, a self-attention-based convolution-free image classifier is devised for the classification task. A comprehensive elaboration of these pivotal stages is provided in the following sub-sections.

3.1. GAN-based Oversampling. GANs are a class of artificial intelligence algorithms proposed by the authors in [20]. GANs consist of two neural networks, the generator and the discriminator, which are concurrently trained in a competitive manner. The generator's objective is to create data that closely resembles the original input, while the discriminator's role is to distinguish between generated and authentic data. This adversarial framework compels both networks to enhance their performance, culminating in the generator producing remarkably realistic results. GANs can acquire deep representations by propagating back-propagation signals through the competitive training of the generator (G) and discriminator (D). Through pre-training, G learns from original input X and endeavours to generate synthetic samples X' that closely match X, hence aiming to minimise the reconstruction loss. On the other hand, D receives both generated data and original input and aims to identify fake data from the real ones, leading to the objective of minimising the loss for the binary classification task. The objective of the loss function of the GAN framework can be represented as follows:

$$\min_G \max_D \mathscr{L}(D,G) = \mathbb{E}_{x\sim P(x)}[\log D(x)] + \mathbb{E}_{z\sim P_Z(z)}[\log(1-D(G(z)))], \quad (1)$$

where $P(X)$ represents the distribution of the original data and $P(z)$ is the distribution of the generator's noise input z. Function $G(z)$ maps random noise to generator models with weights learned from original data, namely learns from X and generates X'; meanwhile, function $D(x)$ represents the probability of identifying that x is real data rather than generated. During the optimisation process, the former should be minimised, and the latter should be maximised. By playing this min-max game, the generator G is forced to produce more realistic samples matching the training data distribution to fool the discriminator, allowing GAN to generate new synthetic samples for minority class enhancement.

A GAN-based oversampling technique could work well with sequential network traffic data, in which a simple model structure with only one-dimensional layers is utilised [21]. For malware imagery analysis, a more complicated model structure is essential. Figure 2 illustrates the GAN oversampler's model structures. The generator is designed as a deep convolutional AE, where both the encoder and decoder comprise two convolutional blocks. Each encoding block consists of three Conv2D layers and one Pooling2D layer. In the latent space, an additional convolutional layer is added. Subsequently, the decoder commences the reconstruction process in two blocks, each featuring three Conv2D layers and one UpSampling2D layer. The convolutional layer units are consistent within each block, with the units across the four blocks specified as "64, 128, 256 for the latent space, 128, 64," respectively. Finally, the reconstruction phase yields the output using a Conv2D layer, producing a 3-channel output representing the reconstructed image. The discriminator initiates with 128-unit initial convolutional layers, followed by batch normalisation and global average pooling. Batch normalisation aids in stabilising the model training and accelerating convergence. Global Average Pooling, instead of flattening the multi-dimensional layers directly, reduces the total parameters and enhances computational efficiency by leveraging a globally learned parameter set. Moreover, it directs the network to prioritise crucial features, thereby improving interpretability and generalisation. Subsequently, a dropout layer is incorporated to prevent over-fitting, followed by two feed forward layers, among which the last layer is for binary classification output.

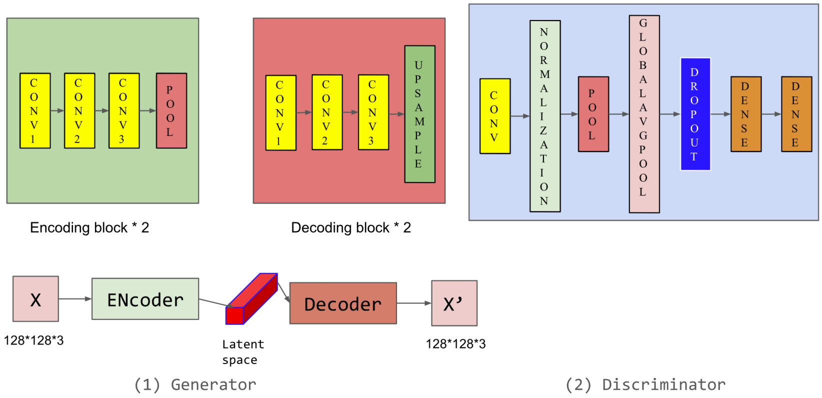


Fig. 2. The workflow from imagery generation to classification in GAN Oversampler

Algorithm 1 presents the training and oversampling process of GANOversampler, primarily focusing on enhancing the representation of minority classes in the dataset. This process generally includes GAN initialisation, a pre-training generator, and oversampling by training and applying the discriminator. The input of the oversampler takes feature sets X_train and label sets y_train as input. During initialisation, the default minority class threshold is set at 1000, which is adjustable. This quantification is selected based on the observations across selected experimental datasets, in which a threshold of 1000 sufficiently equalises the representation of minority samples. In the oversampling phase, the discriminator differentiates samples by assessing their predicted probabilities. A threshold of 0.5 is employed as a critical distinction, where samples above this threshold are deemed indistinguishable from genuine data by the discriminator. These selected samples are then used in the oversampling process to enhance minority class representation. The final outputs of the algorithm are an oversampled feature set and an oversampled label set.

Algorithm 1. GANOversampler

```
1: G = $generator()$
2: D = $discriminator()$
3: define $GAN(G,D)$
4: decide an array of the class id of minorities $minority\_class\_id$
5: decide the threshold of minorities $threshold = 1000$
6: Prepare the input data, feature set $X\_train$ and label set $y\_train$
7: Train the discriminator on the original data $G.fit(X\_train)$
8: for $i \in minority\_class\_id$ do
9:     Create subset filtering by minority id $X(i)$, $y(i)$
10:    Count the current volume of $i$ class $cnt = len(y(i))$
11:    while $cnt < threshold$ do
12:        Generate synthetic samples $sample\_raw = G.predict(noise)$
13:        Keep only those samples predicted to be True $new\_samples = sample\_raw[D.predict(sample\_raw) > 0.5]$
14:        Add $new\_samples$ into $X\_train$ and $y\_train$ and update cnt
15:    end while
16: end for
17: Shuffle the updated $X\_train$ and $y\_train$
```

The generator learns the complex distributions and generates new data instances, while the discriminator evaluates their similarity to real data. This adversarial process continuously improves the quality of synthetic samples to closely mimic real data characteristics, demonstrating the generator's ability to learn and replicate the target distribution. With a customised threshold indicating the preferred volume of minorities, the oversampler can generate synthetic samples of under-represented classes from original volumes to the desired one, thereby balancing the dataset and providing equal training opportunities for all classes. Among traditional oversampling methods, simple random oversampling only copies the original samples. K-nearest-neighbour (KNN)-based methods like the Synthetic Minority Oversampling Technique (SMOTE) and the Adaptive Synthetic (ADASYN) fall short when working with high-dimensional data and can be time-consuming due to the computation of KNN. GANs, on the other hand, can navigate these high-dimensional spaces to produce more accurate and feasible synthetic data. The model is trained extensively before performing oversampling; hence, it will not be as time-consuming as KNN-based oversampling methods. Overall, the introduced oversampling technique excels not only in enhancing the sample size but also in preserving the quality and diversity of synthetic data, thereby ensuring that the augmented dataset supports effective model training.

3.2. Self-attention-based Classifier. As mentioned in the previous session, while convolutional-based attention mechanisms are effective for modelling local dependencies, they may struggle to effectively capture and utilise long-range relationships across the input data. On the other hand, self-attention mechanisms, as employed in ViT, allow for the modelling of interactions between all input elements simultaneously without being constrained by fixed receptive fields. This enables ViT to capture long-range dependencies more effectively and facilitates better integration of global context into the model's representations.

Our study suggests utilising the self-attention mechanism for the classification of malware imagery. This mechanism was initially introduced in the transformer model for machine translation [22], comprising an encoder and a decoder. The encoder consists of stacked self-attention and fully connected layers, while the decoder integrates multi-head attention over the encoder output. The attention mechanism employed is known as Scaled Dot-Product Attention, which calculates dot products of queries and keys, scales them, and applies a SoftMax function. With several input sets, queries, and keys with dimensions of d_k and values with a dimension of d_v, a set of queries can be packed into Q, K, and V respectively. Afterwards, the attention matrix can be calculated as follows, in which the scaling factor $1/\sqrt{d_k}$ is to counteract the potential issue of large value dot products in case of large values of d_k.

$$Attention(Q,K,V) = softmax(QK^T/\sqrt{d_k})V. \tag{2}$$

Within the classifier, the initial input image undergoes patch tokenization to divide it into multiple patches. These patches, referred to as tokens, are subsequently embedded to derive the value vectors (V). Following this, distinct linear projections are utilised on the value vectors to create query (Q) vectors and key (K) vectors, which are essential for subsequent self-attention operations. This query-key-value mechanism enables ViT models to concentrate attention on the most pertinent areas of the input by assessing the similarity between queries and keys [22]. In ViT, images are split into small patches and treated as tokens [23], which are embedded and fed to the transformer architecture. The position embedding, which is essential to retaining the imagery patches' positional information, is combined with patch embedding as input too.

Figure 3 illustrates the model architecture of the proposed malware classifier.

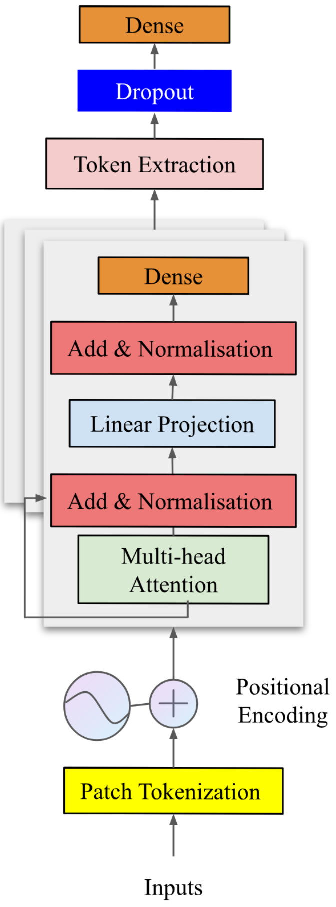


Fig. 3. The model structure of malware classifier with multiple self-attention-based blocks

Following image patching and embedding, the model incorporates multiple self-attention blocks. Within each block, multi-head attention applies separate linear projections to queries, keys, and values, enabling parallel attention computations. Each multi-head attention module contains eight attention layers in parallel, with reduced dimensions to ensure computational efficiency. Layer normalisation is applied after each block to stabilise training and mitigate issues such as vanishing gradients. Two feed forward layers utilising GELU activation top each transformer block for non-linear transformation, and the block outputs are concatenated to represent the input image. Residual connections link stacked transformer blocks. Subsequently, a token extraction layer flattens the outputs, followed by dropout to prevent overfitting. Finally, an additional feed forward layer extracts features while the last layer classifies inputs. We optimise the block numbers based on performance, aiming to find a balance between performance and relatively low model complexity.

3.3. The Optimised Training Process. We employ several techniques to enhance stable and efficient training, including the following: TL with pre-trained weight parameters to leverage prior knowledge and avoid lengthy training from scratch; implementing class weight initialisation based on the proportion of classes in the dataset to address imbalance classification; utilising adaptive learning rate during training.

3.3.1. Transfer Learning. It is common for software to contain diverse types of malware, necessitating the classification of different malware labels. Therefore, malware image classification inherently involves numerous, imbalanced labels. Our approach pre-trains the classifier on the ImageNet dataset, which is proven beneficial for learning generalised features [24]. Initially, the base model is trained on ImageNet with a 1000-class output setting to learn sufficient parameters for complex classification tasks. Subsequently, the output layer is removed, a dropout layer is added, and feed forward layers with corresponding output units specified in the malware imagery set are incorporated. This leverages the pre-trained model for feature extraction from malware images, utilising the learned feature representations from ImageNet. However, the final layers of the pre-trained model require retraining for the new task to acquire task-specific weights. The base model layers are initially frozen by setting their trainable parameter to "false," transforming the pre-trained model into a fixed feature extractor for the new dataset. During this phase, only the classification head layers, initialised randomly, are trained to discern patterns from the extracted features, as outlined in Table 1. The model undergoes training in the frozen setting for 20 epochs. Following the training of the classification head, the base model layers become trainable by setting their trainable parameter to True. Subsequently, the complete model, encompassing both pre-trained and randomly initialised layers, undergoes end-to-end fine-tuning on the target data for 20 epochs with a reduced learning rate, as detailed in Table 1. This process facilitates further optimisation of feature representations to achieve tailored adaptation to new tasks.

Table 1. Classification Model Settings

Model Structure
InputLayer Transformer_block (stacked) LayerNormalization ExtractToken – output size(768) Dropout – rate(0.2) (Dense(256,activation="gelu", kernel_regularizer=regularizers.l2(0.01)) Dense(num_classes, activation='softmax', kernel_regularizer=regularizers.l1(0.01))
Optimistion

Stage 1:
optimizers.AdamW(weight_decay=0.01, learning_rate=0.005)
Stage 2:
optimizers.AdamW(weight_decay=0.05, learning_rate=0.0001)

3.3.2. Weight Initialisation. To mitigate the impact of imbalanced data, in addition to employing oversampling techniques to equalise the distribution of different classes, we also incorporated class weights to prioritise the minority classes during training. The initialisation of class weights entails computing weights for each class according to their distribution in the training dataset, aiming to tackle class imbalances. Algorithm 2 outlines the key steps. The initialised class weights dictionary is then passed to the model during compilation to account for class imbalance during training.

Algorithm 2. WeightInit

```
Prepare input: Original label set $y\_train$
Extract the unique class labels $unique\_classes$
Calculate the frequency of unique class labels $class\_counts$
Calculate the total number of samples $total = len(y\_train)$
**for** $class\_id \in unique\_classes$ **do**
    $weight(class\_id) = total/(num\_classes * count)$
**end for**
Output: Standardised and return the class weight array $W$
```

3.3.3. Optimisation with Adaptive Learning Rate. Adam [25], an adaptive gradient algorithm, has been widely favoured for compiling DL models. However, there are variations that can enhance its performance. According to an optimisation-focused study [26], L2 regularisation in Adam is ineffective, and weight decay is only applied after parameter updates, even though weight decay is crucial for preventing over-fitting. This research explores both L2 regularisation and weight decay regularisation, demonstrating that the proposed Decoupled Weight Decay Regularisation (DWDR) is more effective. The main concept behind DWDR involves adding an extra term for weight decay during the parameter update step. With true label y and predicted label y', the original binary cross-entropy function can be presented as (3), while the regularisation addition to the loss function can be represented as (4):

$$BCE\_loss = -\frac{1}{m}\sum_{i=1}^{m}\left(y^i \log y'^i + (1-y^i)\log(1-y'^i)\right), \tag{3}$$

$$loss = \text{BCE\_loss} + \frac{\lambda}{2m}\sum_{w} w^2. \tag{4}$$

In which w represents the weight learned and λ is a hyper-parameter that need to be initialised manually. When using DWDR, the weight update process from w_t to w_{t+1} is updated with a subtraction from the weights as (5), in which lr represents the learning rate and β is another hyper-parameter for weight decay. The term in the square bracket is the original updating step. In this way, we reduce the updated weight by a small portion at each step.

$$w_{t+1} = [w_t - lr \times gradient] - lr \times \beta \times w_t. \tag{5}$$

As illustrated in Table 1, we initially freeze the base model and train it with a larger learning rate and a smaller decay hyper-parameter, followed by fine-tuning it with a smaller learning rate and a larger decay hyper-parameter. In the first stage of training, this configuration accelerates the training process, enabling the model to rapidly learn general patterns and features from new data. It also enhances generalisation by allowing the model to adapt more effectively to the new dataset. During the fine-tuning stage, the smaller learning rate and larger decay contribute to stabilising the training process and preventing over-fitting. We employ this approach to maximise the benefits of adopting transfer learning.

4. Experiments

4.1. Dataset Description. To thoroughly validate the proposed methods, we conduct experiments on two malware imagery datasets within a multi-classification setting. Both datasets are divided into training, validation, and testing subsets as pre-defined by the original authors. The first dataset, Big2015 [27], was proposed by Microsoft and comprises 10,470 malware files from nine distinct families. Each file is characterised by an identifier, a 20-character hash value, and a class label. The class labels, malware types, and their original proportions in the training set, as well as after the application of oversampling, are presented in Table 2. A backdoor is a type of malware that permits unauthorised access to a computer system, typically bypassing standard authentication mechanisms to facilitate remote control or data theft. Adware constitutes unwanted software that displays advertisements on a user's device. Obfuscated malware is malicious software with code deliberately obfuscated to evade detection by security software. A worm is a self-replicating type of malware that exploits security vulnerabilities and spreads across computer networks. A Trojan is a type of malware commonly disguised as legitimate software. Similarly, a Trojan downloader is a type of Trojan horse designed to download and install additional malware onto the infected system.

Table 2. Data Summary for Big2015

Labels	Malware Types	Initial Weight	Weight after oversampling
Gatak	Backdoor	9.3%	12.3%
Kelihos_ver1	Backdoor	3.7%	9.6%
Kelihos_ver3	Backdoor	27.1%	17.8%
Lollipop	Adware	22.8%	15.0%
Obfuscator.ACY	Any obfuscated malware	11.3%	7.4%
Ramnit	Worm	14.1%	9.3%
Simda	Backdoor	0.4%	8.0%
Tracur	TojanDownloader	6.9%	9.1%
Vundo	Tojan	4.4%	11.5%

The second dataset, MALIMG [28], was proposed in the paper for malware imagery visualisation research. It consists of 9,458 samples from 25 different malware families. The class labels and their original proportions in the training set, as well as after the application of oversampling, are presented in Table 3. Apart from the malware types covered in Big2015, MALIMG contains several new types: Dialer malware is malicious software designed to connect a system to a network or phone number for a fraudulent purpose. Rogue malware is generally characterised by deceptive behaviour; it pretends to be legitimate but can cause significant harm once installed. PWS, short for Password Stealing Ware, aims at stealing sensitive information such as login credentials, passwords, and other personal data.

The malware images contain transformed binary data, with sections such as .text holding the executable code, .rdata containing read-only data like constant values and strings, .data storing initialised data, and .rsrc housing specific resources used by the executable and version information. Various code sections will be mapped to distinct textures within the images, as illustrated in Figure 1. Our approach relies on learning these anomalous patterns to inform the classification process by identifying similarities among the patterns.

Table 3. Data Summary for MALIMG

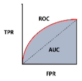
Labels	Malware Types	Initial Weight	Weight after oversampling
Allaple.A	Worm	31.6%	13.5%
Allaple.L	Worm	17.1%	7.3%
Yuner.A	Worm	8.6%	3.7%
Instantaccess	Dialer	4.6%	3.4%
VB.AT	Worm	4.4%	3.4%
Fakerean	Rogue	4.1%	3.4%
Lolyda.AA1	PWS	2.3%	3.4%
C2LOP.gen!g	Trojan	2.1%	3.4%
Alueron.gen!J	Trojan	2.1%	3.4%
Lolyda.AA2	PWS	2.0%	3.4%
Dialplatform.B	Dialer	1.9%	3.4%
Dontovo.A	TojanDownloader	1.7%	3.4%
Lolyda.AT	PWS	1.7%	3.4%
Rbot!gen	Backdoor	1.7%	3.4%
C2LOP.P	Trojan	1.6%	3.4%
Obfuscator.AD	TojanDownloader	1.5%	3.4%
Malex.gen!J	Trojan	1.4%	3.4%
Swizzor.gen!I	TojanDownloader	1.4%	3.4%
Swizzor.gen!E	TojanDownloader	1.4%	3.4%
Lolyda.AA3	Dialer	1.3%	3.4%
Adialer.C	PWS	1.3%	3.4%
Agent.FYI	Backdoor	1.2%	3.4%
Autorun.K	Worm	1.1%	3.4%
Wintrim.BX	TojanDownloader	1.0%	3.4%
Skintrim.N	Trojan	0.9%	3.4%

When resizing the imagery data for the classification task, we consider the original image size in both datasets. For Big2015, where the original images were standardised to the shape of (128, 128, 3), we utilise this shape as well. For MALIMG, the original size varies, so we resize the data into a shape of (224, 224, 3), as this is the standard input size for the original ViT model.

4.2. Baselines and Evaluation Metrics. In our comparative analysis, we select three prominent CNN-based image classifiers to compare with our convolution-free approach. The first model, Inception [29], leverages parallel convolutional operations to effectively capture spatial hierarchies and patterns, characterised by its impressive depth of 22 layers. The second model, MobileNet [30], is equipped with a novel efficient segmentation decoder, specifically designed for semantic segmentation, which delivers optimal performance even on mobile CPUs. The third model, Xception [31], employs a linear stack of depth-wise separable convolution layers with residual connections, which is followed by a point-wise convolution (1*1), implemented after the spatial convolution over each channel.

In the process of evaluation, we consider not only the overall accuracy but also the detection capability for minor classes and the equilibrium of performance. The metrics used are presented in Table 4, in which TP, FP, TN, and FN stand for true positive, false positive, true negative, and false negative, respectively.

Table 4. Basic evaluation metrics

Name	Equation
Recall/Detection Rate	$\frac{TP}{TP+FN}$
Precision	$\frac{TP}{TP+FP}$
AUC	
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$
F1-score	$\frac{2*recall*precision}{recall+precision}$

Accuracy, a prevalent performance measure, compares the correctly predicted observations to the total observations. However, it can be misleading in imbalanced class distributions, particularly in security sectors such as malicious traffic detection and malware classification. The Area Under Curve (AUC) evaluates the performance across all conceivable classification thresholds. Precision, the ratio of correctly predicted positive observations to the total predicted positive observations, signifies a low false positive rate when high. Recall, on the other hand, is the ratio of correctly predicted positive observations to all actual positives, indicating sensitivity and detection ability. The F1 score, being the harmonic mean of precision and recall, serves as a better measure in cases of imbalanced classes. In multi-label classification tasks with imbalanced classes, such as our task of malware analysis, precision, recall, and F1 score are generally deemed more crucial metrics for evaluation.

4.3. Result Comparison. To evaluate the impact of the transformer block implemented in the classifier on the final performance, we test three variations with 4, 5, and 6 blocks implemented in the classifier, which are represented by labels Trans-4, Trans-5, and Trans-6, respectively. Table 5 presents the overall performance on the test set of the Big2015 task. Generally, the convolution-free approaches outperform all the CNN models. Not only do our approaches have higher accuracy and AUC scores, but they also have relatively higher recall. While a model like Xception has a recall of only 0.3242, our approaches, Trans-4, Trans-5, and Trans-6, boast much higher recall scores. Furthermore, it is suggested that it may not be necessary to make the model extremely deep as the Trans-5, namely the model with 5 transformer blocks implemented, achieves the highest accuracy of 0.8369, AUC of 0.9791, and recall of 0.7959. Although the Trans-6 model has equally high accuracy, the highest precision of 0.8989, and F1 of 0.8359, the difference is not significant, indicating that the performance is not compromised by a lighter-weight model.

Table 5. Overall Performance on Test Set– Big2015

	Accuracy	AUC	Precision	Recall	F1
Inception	0.7592	0.9580	0.8480	0.6758	0.7522
MobileNet	0.7399	0.9470	0.8235	0.6563	0.7304
Xception	0.6774	0.9150	0.8384	0.3242	0.4676
Trans-4	0.8223	0.9698	0.8820	0.7666	0.8203
Trans-5	**0.8369**	**0.9791**	0.8792	**0.7959**	0.8355
Trans-6	**0.8369**	0.9777	**0.8989**	0.7813	**0.8359**

Conversely, the training process of our models could potentially be further improved. As presented in Figure 4, the trend of enhancing evaluation metrics stabilises during the fine-tuning phase; for example, our models consistently achieve lower loss rates. Specifically, Trans-5 exhibits the least loss and highest accuracy and AUC post the 46th epoch. Though our models showcase improved accuracy and reduced loss, it is evident that our model encounters notable variations throughout training, particularly in the validation loss curves. These fluctuations in loss optimisation may stem from the weight initialisation technique we employ. Nonetheless, it is worth noting that there is no sign of overfitting in our models. This indicates that despite the training fluctuations, our model remains dependable and robust, underscoring its efficient design and execution.

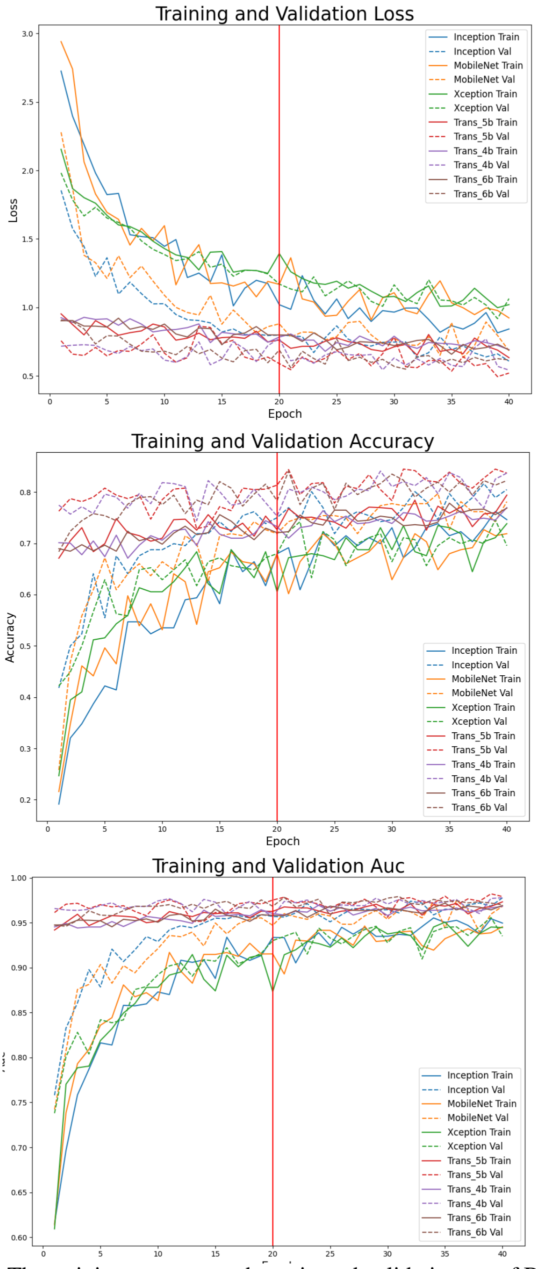


Fig. 4. The training process on the train and validation set of Big2015

Table 6 presents the outcomes of the test set for MALIMG, a particularly challenging task owing to its 25 categories. The findings reveal that all CNN models, along with Trans-4, slightly struggled with learning and accurate prediction, as indicated by their low accuracy and recall scores. In contrast, Trans-5 and Trans-6 exhibit superior performance. Although Trans-6, the deepest model, achieves the highest accuracy, recall, and F1 score, the discrepancies between Trans-5 and Trans-6 are not significant. With an accuracy of 0.9686, an AUC of 0.9992, a recall of 0.9686, and an F1 score of 0.9686, Trans-5 delivers satisfactory outcomes compared to the CNN models.

Table 6. Overall Performance on Test Set– MALIMG

	Accuracy	AUC	Precision	Recall	F1
Inception	0.8537	0.9011	0.8701	0.8418	0.8557
MobileNet	0.8537	0.9911	0.8701	0.8418	0.8557
Xception	0.8174	0.9008	0.8311	0.8311	0.8311
Trans-4	0.8581	0.9899	0.8601	0.8527	0.8564
Trans-5	0.9686	0.9992	0.9686	0.9686	0.9686
Trans-6	**0.9697**	**0.9998**	**0.9707**	**0.9697**	**0.9702**

Figure 5 presents the optimisation process for both the training and validation sets. In terms of loss and accuracy scores, all models exhibit continuous improvement during the initial training stage, although CNN models lag behind compared to Trans models. During the fine-tuning stage, the pronounced fluctuation of all curves highlights the increased challenge of training on tasks with more class labels. As MobileNet achieves the best training loss and accuracy but relatively poor validation loss and accuracy, there is a suggestion that this CNN-based model may be susceptible to overfitting. For Trans models, while the loss optimisation curve displays a consistent decrease, the accuracy experiences more significant fluctuations. This can be attributed to the weight initialisation process, where higher loss weights are assigned to minorities. Considering that accuracy evaluates overall performance, these fluctuations are to be expected. Nevertheless, a key point of optimism is that our model maintains an accuracy rate above 80% for the majority of the time, indicating its promising and reliable performance.

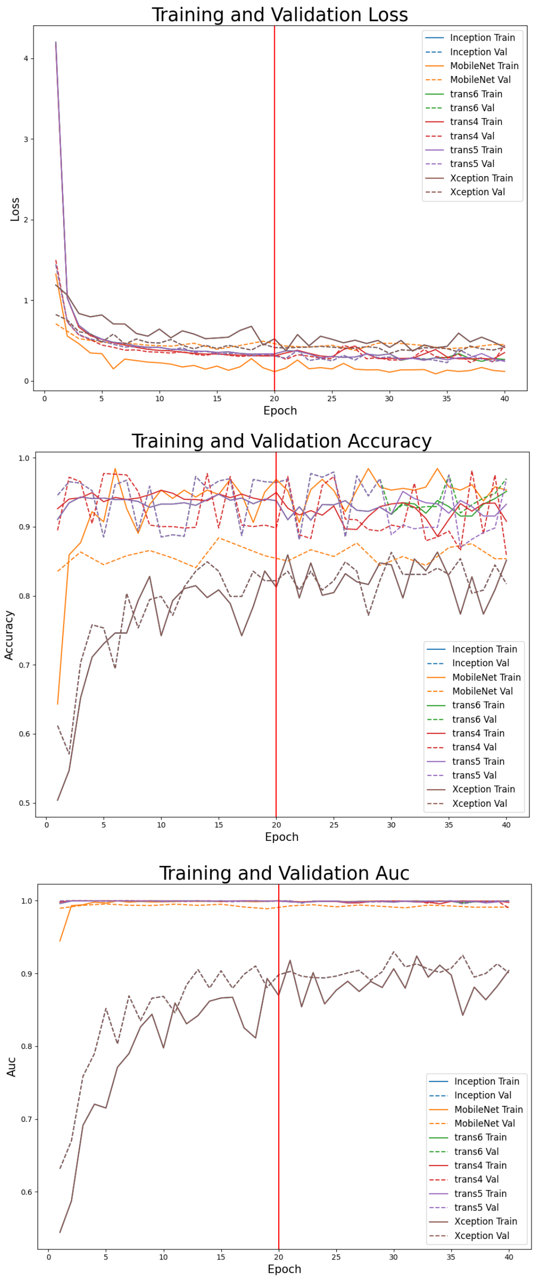


Fig. 5. The training process on the train and validation set of the MALIMG dataset

AUC presents a distinctive case. Except for Xception, all other models exhibit consistently high AUC scores across the entire training process. The reason behind the elevated AUC scores and the varying performance levels of the different models may be linked to the inherent nature of the AUC metric itself. While AUC evaluates the model's capacity to differentiate between positive and negative classes, the high AUC scores could possibly stem from the intrinsic characteristics of the data. Conversely, due to the imbalanced class distribution, the assessment of models' effectiveness through recall and the precision-recall equilibrium, known as the F1 score, can offer a more accurate evaluation of the performance.

CNN models exhibit satisfactory performance on the Big2015 task but fall short on the MALIMG task. In contrast, our approach consistently delivers superior results. This could be attributed to the capacity of self-attention-based models to manage long-range dependencies between pixels in images by assigning weights and prioritising patches for predictions. Consequently, these models can capture complex patterns and structures that may elude CNNs, particularly in intricate tasks requiring the prediction of 25 labels. Moreover, attention-based blocks are more readily parallelised, facilitating faster training times and enhanced performance. In scenarios involving extensive data, attention-based blocks also exhibit a less pronounced inductive bias. Nonetheless, Transformers may surpass CNNs in specific tasks or datasets. As highlighted in the original ViT paper, CNNs may still outperform transformers when dealing with smaller datasets.

4.4. Discussion on Future Research. Despite the demonstrated reliability and robustness of our model, there remain areas for potential enhancement. Primarily, the model's representation learning ability and detection capability could be further improved, as suggested by the results of the Big2015 task. Recent research has investigated various methods for transformer optimisation. For example, the Swin Transformer employs a shifted window-based self-attention mechanism, enabling it to capture both local and global dependencies in images [32]. Another notable study is DynamicViT [33], which introduces a technique that allows the model to adaptively adjust its computational complexity based on the complexity of the input image.

Another limitation is the insufficient consideration of obfuscation techniques, which can significantly impact the performance of malware classification systems. The technique employed for concealing malicious code makes accurate threat identification challenging. One possible enhancement involves augmenting the proposed method's resilience through feature engineering specifically tailored for obfuscation detection. In [34] the authors undertook further static analysis to discern obfuscation patterns and devise discriminating features that distinguish benign applications from malware; this strategy enhanced the detector's efficacy in identifying obfuscated malware. An alternative approach is exploring adversarial ML techniques to simulate attacks on the classification model using obfuscated malware samples. Random noise can be generated and incorporated into image arrays to mimic obfuscation. This method can assist in identifying weaknesses within the classifier and subsequently refining the model to more effectively resist obfuscation techniques. Furthermore, extensive research into information-hiding techniques can be applied to image-based malware analysis. Several techniques exist for mitigating blurring to unveil hidden content or anomalies, including patch-line and fuzzy clustering-line priors for dehazing [35], as well as noise-aware filtering reversal through modified Landweber iterations [36].

Lastly, from the perspective of usable security, we intend to refine and expand upon our current methodology for malware classification. While the existing approach exhibits commendable performance in tasks involving multi-label classification, it encounters substantial challenges when confronted with novel or previously unidentified malware variants. Such instances are prone to misclassification or, more concerning, being erroneously identified as benign behaviours. Hence, our future endeavours will be directed towards the development and implementation of an advanced two-module framework. This innovative strategy will amalgamate a sophisticated multi-label classifier with an anomaly detection model. The latter component is planned to either embody an AE or leverage a probabilistic model specifically designed for outlier detection. In scenarios where the classifier is unable to accurately categorise a sample, defaulting to label it as benign, and concurrently, the anomaly detection module identifies it as an anomaly diverging from the established distribution patterns, such instances will be flagged as "potential new attacks". This designation will trigger alerts, thereby facilitating timely intervention and analysis. This strategic enhancement aims not only to bolster the accuracy and reliability of malware classification but also to establish a proactive defence mechanism against emerging cyber threats. By incorporating this two-pronged approach, our system will be better equipped to adapt to the evolving landscape of cybersecurity threats, ensuring enhanced protection for digital ecosystems.

5. Conclusion. In this study, we introduce a convolutional-free malware classifier, complemented by a GAN-based oversampler. This oversampling technique significantly amplifies the minority classes with realistic samples, while the classifier consistently surpasses CNN models such as MobileNet, Inception, and Xception on a range of public datasets. Looking ahead, we aim to investigate the potential for enhancing the malware classification capability of our model and reducing its complexity. Nevertheless, the superior performance of our model across various tasks emphatically attests to its reliability and robustness. This not only substantiates our model's credibility but also positions it as a promising solution for tackling diverse and complex malware imagery classification tasks.

References

1. Altan G. SecureDeepNet-IoT: A deep learning application for invasion detection in industrial internet of things sensing systems. Transactions on Emerging Telecommunications Technologies. 2021. vol. 32. no. 4. DOI: 10.1002/ett.4228.
2. Tien C.-W., Chen S.-W., Ban T., Kuo S.-Y. Machine learning framework to analyze iot malware using elf and opcode features. Digital Threats: Research and Practice. 2020. vol. 1. no. 1. pp. 1–19. DOI: 10.1145/3378448.
3. Rizvi S., Aslam W., Shahzad M., Saleem S., Fraz M. Proud-mal: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable. Complex & Intelligent Systems. 2022. pp. 1–13.
4. Jung B., Kim T., Im E. Malware classification using byte sequence information. Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems. 2018. pp. 143–148.
5. Alrawi O., Lever C., Valakuzhy K., Snow K., Monrose F., Antonakakis M., et al. The circle of life: A large-scale study of the IoT malware lifecycle. 30th USENIX Security Symposium (USENIX Security 21). 2021. pp. 3505–3522.
6. Smmarwar S., Gupta G., Kumar S. Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review. Telematics and Informatics Reports. 2024. vol. 14. DOI: 10.1016/j.teler.2024.100130.
7. Branitskiy A., Kotenko I. Network attack detection based on combination of neural, immune and neuro-fuzzy classifiers. IEEE 18th International Conference on Computational Science and Engineering. 2015. pp. 152–159.
8. Desnitsky V., Kotenko I., Nogin S. Detection of anomalies in data for monitoring of security components in the internet of things. XVIII International Conference on Soft Computing and Measurements. 2015. pp. 189–192.
9. Wang C., Zhao Z., Wang F., Li Q. A novel malware detection and family classification scheme for IoT based on deam and densenet. Security and Communication Networks. 2021. vol. 2021. no. 1. pp. 1–16. DOI: 10.1155/2021/6658842.
10. Yousefi-Azar M., Varadharajan V., Hamey L., Tupakula U. Autoencoder-based feature learning for cyber security applications. In 2017 International Joint Conference on Neural Networks (IJCNN). 2017. pp. 3854–3861.
11. Bakır H., Bakır R. Droidencoder: Malware detection using auto-encoder based feature extractor and machine learning algorithms. Computers and Electrical Engineering. 2023. vol. 110. DOI: 10.1016/j.compeleceng.2023.108804.
12. Venkatraman S., Alazab M., Vinayakumar R. A hybrid deep learning image-based analysis for effective malware detection. Journal of Information Security and Applications. 2019. vol. 47. pp. 377–389.
13. Yakura H., Shinozaki S., Nishimura R., Oyama Y., Sakuma J. Malware analysis of imaged binary samples by convolutional neural network with attention mechanism. Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. 2017. pp. 55–56.

14. Li X., Wang L., Xin Y., Yang Y., Chen Y. Automated vulnerability detection in source code using minimum intermediate representation learning. Applied Sciences. 2020. vol. 10. no. 5. DOI: 10.3390/app10051692.

15. Wu P., Guo H., Buckland R. A transfer learning approach for network intrusion detection. IEEE 4th International Conference on Big Data Analytics. 2019. pp. 281–285.

16. Qiang Q., Cheng M., Zhou Y., Ding Y., Qi Z. Malup: A malware classification framework using convolutional neural network with deep unsupervised pre-training. In 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). 2021. pp. 627–634.

17. Hu J., Liu C., Cui Y. An improved cnn approach for network intrusion detection system. International Journal of Network Security. 2021. vol. 23. no. 4. pp. 569–575.

18. Xu Z., Fang X., Yang G. Malbert: A novel pre-training method for malware detection. Computers & Security. 2021. vol. 111(2). DOI: 10.1016/j.cose.2021.102458.

19. Habibi O., Chemmakha M., Lazaar M. Performance evaluation of cnn and pre-trained models for malware classification. Arabian Journal for Science and Engineering. 2023. vol. 48. no. 8. pp. 10355–10369.

20. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems. 2014. vol. 27. pp. 2672–2680.

21. Dong H., Kotenko I. Hybrid multi-task deep learning for improved iot network intrusion detection: Exploring different cnn structures. 2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS). 2024. pp. 7–12.

22. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser Ł., Polosukhin I. Attention is all you need. Advances in neural information processing systems. 2017. pp. 5998–6008.

23. Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. 2020. arXiv preprint arXiv:2010.11929.

24. Huh M., Agrawal P., Efros A. What makes imagenet good for transfer learning? 2016. arXiv preprint arXiv:1608.08614.

25. Kingma D., Ba J. Adam: A method for stochastic optimization. 2014. arXiv preprint arXiv:1412.6980.

26. Loshchilov I., Hutter F. Decoupled weight decay regularization. 2017. arXiv preprint arXiv:1711.05101.

27. Ronen R., Radu M., Feuerstein C., Yom-Tov E., Ahmadi M. Microsoft malware classification challenge. 2018. arXiv preprint arXiv:1802.10135.

28. Nataraj L., Karthikeyan S., Jacob G., Manjunath B. Malware images: visualization and automatic classification. Proceedings of the 8th International Symposium on Visualization for Cyber Security. 2011. pp. 1–17.

29. Szegedy C., Ioffe S., Vanhoucke V., Alemi A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence. 2017. vol. 31. no. 1. DOI: 10.1609/aaai.v31i1.11231.

30. Howard A., Sandler M., Chen B., Wang W., Chen L.-C., Tan M., Chu G., Vasudevan V., Zhu Y., Pang R., Adam H., Le Q. Searching for mobilenetv3. IEEE/CVF International Conference on Computer Vision (ICCV). 2019. pp. 1314–1324.

31. Chollet F. Xception: Deep learning with depthwise separable convolutions. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

32. Liu Z., Lin Y., Cao Y., Hu H., Wei Y., Zhang Z., Lin S., Guo B. Swin transformer: Hierarchical vision transformer using shifted windows. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2021. pp. 10012–10022.
33. Rao Y., Zhao W., Liu B., Lu J., Zhou J., Hsieh C.-J. Dynamicvit: Efficient vision transformers with dynamic token sparsification. Advances in Neural Information Processing Systems. 2021. vol. 34. pp. 13937–13949.
34. Kim D., Majlesi-Kupaei A., Roy J., Anand K., ElWazeer K., Buettner D., Barua R. DynODet: Detecting Dynamic Obfuscation in Malware. Proceedings of the Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA): 14th International Conference, DIMVA. 2017. pp. 97–118.
35. Liao M., Lu Y., Li X., Di S., Liang W., Chang V. An unsupervised image dehazing method using patch-line and fuzzy clustering-line priors. IEEE Transactions on Fuzzy Systems. 2024. vol. 4. pp. 1–15. DOI: 10.1109/TFUZZ.2024.3371944.
36. Wang L., Fayolle P., Belyaev A. Reverse image filtering with clean and noisy filters. Signal, Image and Video Processing. 2023. vol. 17. no. 2. pp. 333–341.

Huiyao Dong — Postgraduate, Faculty of information security, ITMO University; Programmer, laboratory of computer security problems, St. Petersburg Federal Research Center of the Russian Academy of Sciences. Research interests: applied data science (particularly multi-task deep learning, autoencoder, imagery analysis, reinforcement deep learning), network security, IoT. The number of publications — 8. hydong@itmo.ru; 49 A, Kronverksky Av., 197101, St. Petersburg, Russia; office phone: +7(812)508-3311.

УДК 004.056 **DOI** 10.15622/ia.23.6.11

Х. Дун

КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ ВРЕДОНОСНЫХ ПРОГРАММ БЕЗ ИСПОЛЬЗОВАНИЯ СВЕРТОК С ИСПОЛЬЗОВАНИЕМ МЕХАНИЗМОВ ВНУТРЕННЕГО ВНИМАНИЯ

Дун Х. **Классификация изображений вредоносных программ без использования сверток с использованием механизмов внутреннего внимания.**

Аннотация. Анализ вредоносных программ является важнейшим аспектом кибербезопасности, направленным на выявление и дифференциацию вредоносного ПО от безвредных программ для защиты компьютерных систем от угроз безопасности. Несмотря на достижения в мерах кибербезопасности, вредоносные программы продолжают представлять значительные риски в киберпространстве, требуя точных и быстрых методов анализа. В этой статье представлен инновационный подход к классификации вредоносных программ с использованием анализа изображений, включающий три ключевых этапа: преобразование кодов операций в данные изображений RGB, использование генеративно-состязательной сети (GAN) для синтетической передискретизации и использование упрощенного классификатора на основе визуального трансформера (ViT) для анализа изображений. Данный метод повышает богатство функций и объяснимость с помощью данных визуальных изображений и устраняет несбалансированную классификацию с использованием методов передискретизации на основе GAN. Предложенная структура сочетает в себе преимущества сверточных автоэнкодеров, гибридных классификаторов и адаптированных моделей ViT для достижения баланса между точностью и вычислительной эффективностью. Как показали эксперименты, наш подход без использования сверток обладает превосходной точностью и прецизионностью по сравнению со сверточными моделями и превосходит модели CNN на двух наборах данных благодаря механизму многоголового внимания. На наборе данных Big2015 наша модель превосходит другие модели CNN с точностью 0,8369 и площадью под кривой (AUC) 0,9791. В частности, наша модель достигает точности 0,9697 и оценки F1 0,9702 на MALIMG, что является экстраординарным результатом.

Ключевые слова: обнаружение вредоносных программ, кибербезопасность, глубокое обучение, автоэнкодер.

Литература

1. Altan G. SecureDeepNet-IoT: A deep learning application for invasion detection in industrial internet of things sensing systems. Transactions on Emerging Telecommunications Technologies. 2021. vol. 32. no. 4. DOI: 10.1002/ett.4228.
2. Tien C.-W., Chen S.-W., Ban T., Kuo S.-Y. Machine learning framework to analyze iot malware using elf and opcode features. Digital Threats: Research and Practice. 2020. vol. 1. no. 1. pp. 1–19. DOI: 10.1145/3378448.
3. Rizvi S., Aslam W., Shahzad M., Saleem S., Fraz M. Proud-mal: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable. Complex & Intelligent Systems. 2022. pp. 1–13.
4. Jung B., Kim T., Im E. Malware classification using byte sequence information. Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems. 2018. pp. 143–148.

5. Alrawi O., Lever C., Valakuzhy K., Snow K., Monrose F., Antonakakis M., et al. The circle of life: A large-scale study of the IoT malware lifecycle. 30th USENIX Security Symposium (USENIX Security 21). 2021. pp. 3505–3522.
6. Smmarwar S., Gupta G., Kumar S. Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review. Telematics and Informatics Reports. 2024. vol. 14. DOI: 10.1016/j.teler.2024.100130.
7. Branitskiy A., Kotenko I. Network attack detection based on combination of neural, immune and neuro-fuzzy classifiers. IEEE 18th International Conference on Computational Science and Engineering. 2015. pp. 152–159.
8. Desnitsky V., Kotenko I., Nogin S. Detection of anomalies in data for monitoring of security components in the internet of things. XVIII International Conference on Soft Computing and Measurements. 2015. pp. 189–192.
9. Wang C., Zhao Z., Wang F., Li Q. A novel malware detection and family classification scheme for IoT based on deam and densenet. Security and Communication Networks. 2021. vol. 2021. no. 1. pp. 1–16. DOI: 10.1155/2021/6658842.
10. Yousefi-Azar M., Varadharajan V., Hamey L., Tupakula U. Autoencoder-based feature learning for cyber security applications. In 2017 International Joint Conference on Neural Networks (IJCNN). 2017. pp. 3854–3861.
11. Bakır H., Bakır R. Droidencoder: Malware detection using auto-encoder based feature extractor and machine learning algorithms. Computers and Electrical Engineering. 2023. vol. 110. DOI: 10.1016/j.compeleceng.2023.108804.
12. Venkatraman S., Alazab M., Vinayakumar R. A hybrid deep learning image-based analysis for effective malware detection. Journal of Information Security and Applications. 2019. vol. 47. pp. 377–389.
13. Yakura H., Shinozaki S., Nishimura R., Oyama Y., Sakuma J. Malware analysis of imaged binary samples by convolutional neural network with attention mechanism. Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. 2017. pp. 55–56.
14. Li X., Wang L., Xin Y., Yang Y., Chen Y. Automated vulnerability detection in source code using minimum intermediate representation learning. Applied Sciences. 2020. vol. 10. no. 5. DOI: 10.3390/app10051692.
15. Wu P., Guo H., Buckland R. A transfer learning approach for network intrusion detection. IEEE 4th International Conference on Big Data Analytics. 2019. pp. 281–285.
16. Qiang Q., Cheng M., Zhou Y., Ding Y., Qi Z. Malup: A malware classification framework using convolutional neural network with deep unsupervised pre-training. In 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). 2021. pp. 627–634.
17. Hu J., Liu C., Cui Y. An improved cnn approach for network intrusion detection system. International Journal of Network Security. 2021. vol. 23. no. 4. pp. 569–575.
18. Xu Z., Fang X., Yang G. Malbert: A novel pre-training method for malware detection. Computers & Security. 2021. vol. 111(2). DOI: 10.1016/j.cose.2021.102458.
19. Habibi O., Chemmakha M., Lazaar M. Performance evaluation of cnn and pre-trained models for malware classification. Arabian Journal for Science and Engineering. 2023. vol. 48. no. 8. pp. 10355–10369.
20. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems. 2014. vol. 27. pp. 2672–2680.

21. Dong H., Kotenko I. Hybrid multi-task deep learning for improved iot network intrusion detection: Exploring different cnn structures. 2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS). 2024. pp. 7–12.
22. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser L., Polosukhin I. Attention is all you need. Advances in neural information processing systems. 2017. pp. 5998–6008.
23. Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. 2020. arXiv preprint arXiv:2010.11929.
24. Huh M., Agrawal P., Efros A. What makes imagenet good for transfer learning? 2016. arXiv preprint arXiv:1608.08614.
25. Kingma D., Ba J. Adam: A method for stochastic optimization. 2014. arXiv preprint arXiv:1412.6980.
26. Loshchilov I., Hutter F. Decoupled weight decay regularization. 2017. arXiv preprint arXiv:1711.05101.
27. Ronen R., Radu M., Feuerstein C., Yom-Tov E., Ahmadi M. Microsoft malware classification challenge. 2018. arXiv preprint arXiv:1802.10135.
28. Nataraj L., Karthikeyan S., Jacob G., Manjunath B. Malware images: visualization and automatic classification. Proceedings of the 8th International Symposium on Visualization for Cyber Security. 2011. pp. 1–17.
29. Szegedy C., Ioffe S., Vanhoucke V., Alemi A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence. 2017. vol. 31. no. 1. DOI: 10.1609/aaai.v31i1.11231.
30. Howard A., Sandler M., Chen B., Wang W., Chen L.-C., Tan M., Chu G., Vasudevan V., Zhu Y., Pang R., Adam H., Le Q. Searching for mobilenetv3. IEEE/CVF International Conference on Computer Vision (ICCV). 2019. pp. 1314–1324.
31. Chollet F. Xception: Deep learning with depthwise separable convolutions. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.
32. Liu Z., Lin Y., Cao Y., Hu H., Wei Y., Zhang Z., Lin S., Guo B. Swin transformer: Hierarchical vision transformer using shifted windows. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2021. pp. 10012–10022.
33. Rao Y., Zhao W., Liu B., Lu J., Zhou J., Hsieh C.-J. Dynamicvit: Efficient vision transformers with dynamic token sparsification. Advances in Neural Information Processing Systems. 2021. vol. 34. pp. 13937–13949.
34. Kim D., Majlesi-Kupaei A., Roy J., Anand K., ElWazeer K., Buettner D., Barua R. DynODet: Detecting Dynamic Obfuscation in Malware. Proceedings of the Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA): 14th International Conference, DIMVA. 2017. pp. 97–118.
35. Liao M., Lu Y., Li X., Di S., Liang W., Chang V. An unsupervised image dehazing method using patch-line and fuzzy clustering-line priors. IEEE Transactions on Fuzzy Systems. 2024. vol. 4. pp. 1–15. DOI: 10.1109/TFUZZ.2024.3371944.
36. Wang L., Fayolle P., Belyaev A. Reverse image filtering with clean and noisy filters. Signal, Image and Video Processing. 2023. vol. 17. no. 2. pp. 333–341.

Дун Хуэйяо — аспирантка, факультет информационной безопасности, Университет ИТМО; программист, лабораторией проблем компьютерной безопасности, Санкт-Петербургский Федеральный исследовательский центр Российской академии наук. Область научных интересов: прикладная наука о данных (в частности, многозадачное глубокое обучение, автокодирование, анализ изображений, глубокое обучение с подкреплением), сетевая безопасность, IoT. Число научных публикаций — 8. hydong@itmo.ru; Кронверкский проспект, 49 А, 197101, Санкт-Петербург, Россия; р.т.: +7(812)508-3311.