Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

TC 11 Briefing Papers

A novel malware classification and augmentation model based on convolutional neural network

Adem Tekerek[a], *Muhammed Mutlu Yapici*[b,*]

[a] *Gazi University, Technology Faculty, Computer Engineering Department, Ankara, Turkey*
[b] *Ankara University, Ankara, Turkey*

ARTICLE INFO

Article history:
Received 18 April 2021
Revised 26 August 2021
Accepted 19 October 2021
Available online 23 October 2021

Keywords:
Malware classification
Convolutional neural network
Cybersecurity

ABSTRACT

The rapid development and widespread use of the Internet have led to an increase in the number and variety of malware proliferating via the Internet. Malware is the general nomenclature for malicious software. Malware classification is an undecidable problem and technically NP hard problem because the halting problem is NP hard. In this study, we proposed a convolutional neural network based novel method for malware classification. Since CNN models use the images as input, bytes files are transformed to gray separately and RGB image formats for the classification process. A new approach called B2IMG is developed for the transformation of bytes file. Moreover, a new CycleGAN-based data augmentation method is proposed to address the problem of imbalanced data size between malware families. The proposed system was tested on the BIG2015, and DumpWare10 datasets. According to the experimental results, classification performance increased thanks to the proposed data augmentation method. The accuracy of the classification is 99.86% for the BIG2015 dataset and 99.60% for the dataset.

1. Introduction

Malware is malicious software or code that damages information systems on which it is executed. Malware damage can have a devastating effect such as deleting data on a personal computer or causing a breakdown at a nuclear power plant. Therefore, information systems must be protected against malware attacks. Many researchers and cyber security companies are working to develop new methods and tools in order to protect information systems against malware and to detect and prevent malicious software. Many methods are used to detect and prevent malware. Malware analysis can be performed as static, dynamic and memory analysis techniques. Static analysis aims to gather information without running malicious applications. Dynamic analysis is gathering information by running the malware in a test environment. Memory analysis collects and analyzes memory structures to learn more about malware (Egele et al., 2008). These methods can generally be categorized into two categories. One of them is signature-based detection method and the other is anomaly-based detection method (Tekerek, 2021). The signature-based detection method includes the logic of defining known attack types as a black list and comparing the malware identified in the black list. Signature-based detection method is not

* Corresponding author.
E-mail address: mutlu.yapici@ankara.edu.tr (M.M. Yapici).
https://doi.org/10.1016/j.cose.2021.102515

so effective due to ever changing malware types. Anomaly-based method used to detect malware that does not conform to the known file system structure and has unusual features. Anomaly-based methods are effective in detecting and preventing zero-day attacks. Machine learning and data mining methods are used for anomaly-based methods. Recently, deep learning algorithms, which are multi-layered neural networks, are used and successfully applied in many learning processes. However, deep learning requires more computation time to train models (Liu et al., 2017). Machine learning algorithms are fast, but they are not successful as deep learning.

Deep learning is a machine learning research area that has achieved valuable success in many research areas. Deep learning approaches have significantly outperformed the-state-of-the-art approaches in many fields (Alom et al., 2019). Besides, deep learning has achieved on information security applications as well (Tekerek, 2021). Researchers have tried to apply deep learning to various subfields in information security, such as data privacy and protection, vulnerability detection, and malware detection. The success of deep learning is attributed to its high representational ability of input data by using various layers such as fully connected, dropout, and pooling. Deep learning discovers the intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation between layers (LeCun et al., 2015). Deep learning structure consists of an input layer, hidden layers and an output layer as in artificial neural network. The difference between deep learning and artificial neural network is the number of hidden layers, which is more than one and directly affects the depth of the algorithm in deep learning. Firstly, raw data is given to the input layer, and secondly, all values are calculated sequentially along with the network layers as output. The obtained output value is used as input data for each next layer. Output values are calculated via the sum of the multiplication of the input and the weight for each unit in the current layer. Then, a non-linear function such as a rectified linear unit (ReLU), hyperbolic tangent or sigmoid is applied to compute the output values of the layer (Yu et al., 2020). The output data obtained as it progresses towards the last layer contain slightly more abstract representations than the state of the raw input data. It allows a more successful classification with the data. Deep learning has a major impact in the field of information security. In malware analysis via image analysis, deep learning algorithms have obtained successful results in many tasks such as image enhancement, image retrieval, classification, object localization, object detection and segmentation.

In this study, byte files were transformed separately to gray and RGB image formats for the classification process. A new approach called B2IMG was developed for the transformation process. Moreover, a new CycleGAN-based data augmentation method was developed in the study to address the problem of inconsistent data size between malware families. The proposed model was tested on the BIG2015 and DumpWare10 datasets. Experimental results are presented at the section 5.

This paper is organized as follows. Section 2 presents the literature review. Section 3 presents the materials and methods. In Section 4, proposed CNN based deep learning model is described. In Section 5 experimental results are presented. Finally, Section 6 presents the conclusions of the paper.

2. Literature review

There are many studies in the literature to detect malicious code or malware. Many methods are used to develop a feasible and successful detection result for malware classification. In this study, deep learning based malware detection techniques are used for comparison.

Kalash et al. (Kalash et al., 2018) developed a CNN-based model for malware detection. The authors used Maling and Microsoft datasets to evaluate the performance of their model. Firstly, malware files are converted to grayscale images which are provided as input to the CNN. According to experimental results, they achieved an accuracy of 99.17 on Microsoft Dataset and 98.52 on Maling dataset.

Kim et al. (Kim et al., 2018) proposed a learning-based model to detect zero-day malware attacks using the BIG2015 dataset. Authors used Generative Adversarial Networks (GANs) for fake malware detection (Pan et al., 2019). Before GAN training, they used deep autocoder to extract generic features from the data, and then transferred the information to the GAN to gain stability within the trained network.

David et al. (David and Netanyahu, 2015) reported a deep learning-based malware generation and detection. In the authors proposed method, the malware is invariant and the compact representation of the malware is learned using the deep belief network with stacked denoising autocoders. Wozniak et al. (Wozniak et al., 2020) presented a recurrent neural network based model developed for network traffic analysis of various Internet of things solutions. According to the test results, the model has an accuracy rate of over 99%. Drew et al. (Drew et al., 2017) used BIG 2015 and reported an accuracy of 98%. The goal of the authors was to show how gene sequence classifier can be used for malware classification and how fast it works as compared to other hybrid techniques.

Huang et al. (Huang et al., 2020) proposed a malware detection method based on deep learning and malware visualization using VGG16. Authors combined malware visualization technology with convolutional neural network. Jang et al. (Jang et al., 2020) proposed a malware classification method based on CNN using BIG2015 dataset. The study introduced a Text-based local feature visualization method. Accuracy of the proposed method is 99.65%. Wang et al. (Wang et al., 2021) introduced a new module of CNN named as Depthwise efficient attention module (DEAM) using MalImg and BIG2015 dataset. They used DenseNet to propose malware detection and family classification model. Accuracy of the proposed method is 99.3%. Because signature-based malware detection approaches are inadequate, Kang et al. (Kang et al., 2019) proposed a word2vec based LSTM method to analyze opcodes and API function names using fewer dimensions. According to the experimental results, proposed word2vec based method was approximately 0.5% higher. Stiborek et al. (Stiborek et al., 2018) proposed a model of malware behavior observed through its interactions with the operating system and network resources (operations with files, mutexes, registry keys, operations with network servers or error messages provided by the operat-

ing system) based on multiple instance learning. The proposed model has 95.4% successful classification. Kang et al. (Kang et al., 2019) proposed a word2vec- based LSTM method to analyze opcodes and API function names using fewer dimensions. The proposed model has 97.59% successful classification result. Vasan et al. (D. Vasan et al., 2020) proposed a malware classification algorithm called IMCFN which combines malware visualization and fine-tuned CNN architecture already trained on ImageNet dataset. The classification accuracy is 98.82% for Malimg dataset. Vasan et al. (D. Vasan et al., 2020) proposed a new CNN-based architecture to detect both packaged and unpackaged malware classification. According the experimental result, proposed model demonstrated 99% accuracy for unpacked malware and 98% accuracy for packed malware. Gibert et al. (Gibert et al., 2020) developed a novel framework to address the task of malware detection and classification by combining different types of features to discover the relationships between distinct modalities. The authors proposed a baseline system consisting of both hand-engineered and end-to-end components to combine the benefits of feature engineering and deep learning so that malware characteristics can be represented effectively. Yuan et al. (Yuan et al., 2020) proposed a byte-level malware classification method based on markov images and deep learning to improve of the accuracy of gray scale images. The average accuracy rates are respectively 99.2% and 97.3% on Microsoft and the Drebin datasets. Zhang et al. (Zhang et al., 2019) proposed a feature-hybrid malware variants detection approach, which integrates multi-types of features to address these challenges. According to the experimental results, proposed approach achieves 95% malware detection. Ghanei et al. (Ghanei et al., 2021) presented a novel dynamic malware detection method, which utilizes hardware events during file execution status as features of the classification model. According to the experimental results presented CNN and LSTM combined method achieved 91.63% malware detection accuracy. Jain et al. (Jain et al., 2020) conducted experiments to evaluate CNN and extreme learning machines (ELM) machine learning models for malware classification. They visualized malware samples and used image processing methods, creating both two-dimensional and one-dimensional vectors derived from images. Catak et Al. (Catak et al., 2021) developed a CNN model enhanced with image augmentation to detect malware families in malware classification. The experimental results obtained 98% accuracy rate. Yan et al. (Yan et al., 2018) proposed a novel malware detection method that learns features automatically from the grayscale images. They used CNN and LSTM methods. The malware detection evaluation result has 99.88% accuracy rates.

In this study, a deep learning-based model is developed for data augmentation and malware classification. Moreover, a novel method named B2IMG is proposed to transform bytes files to jpg format. BIG2015 (Ronen et al., 2018) and Dumpware10 (Bozkir et al., 2021) datasets are used for evaluating the performance of the proposed system.

The contributions of this study to the literature are as follows:

- √ Developing a new novel method named B2IMG for byte file to image translation.
- √ Producing synthetic Gray and RGB images of malware families for the first time with the CycleGAN architecture.
- √ Determining the effect of synthetic data in CNN model for the malware classification.
- √ Classification of malware families with images obtained after data augmentation for the widely used BIG2015, and Dumware10 datasets.

3. Materials and methods

3.1. Datasets

In the study two different datasets was used: *Microsoft Malware Classification Challenge Dataset (BIG 2015)*, *Dumpware10 Dataset*. The datasets are polymorphic datasets. Detailed information about the datasets are presented below.

Microsoft Malware Classification Challenge Dataset (BIG 2015): Microsoft released a large malware dataset in 2015 WWW Conference. It is called Microsoft Malware Classification Challenge Dataset (BIG 2015)[27]. The Microsoft Malware Classification Challenge Dataset (BIG 2015) is almost 500 GB size and consists of a set of known malware files representing a mix of 9 different families. Family classification is an important part of the BIG2015 dataset. Because distinguishing between malware families is crucial to understanding how these software can infect computers, the level of threat they pose, and how to take action against them. Each malware file has an identifier, a 20-character hash value uniquely identifying the file, and a class label, which is an integer representing one of the 9 family names to which the malware may belong. For each file, the raw data contains the hexadecimal representation of the file's binary content, without the header (to ensure sterility). The dataset also includes a metadata manifest, which is a log containing various metadata information extracted from the binary, such as function calls, strings, etc. The details of the dataset are given in Table 1. We used the training part of the dataset in this work. The training dataset has 9 different classes of malware. These malware classes are (1) Ramnit, (2) Lollipop, (3) Kelihos_ver3, (4) Vundo, (5) Simda, (6) Tracur, (7)

Table 1 – Malware families in the training dataset.

Order	Family Name	#Train Samples	Type
1	Ramnit	1533	Worm
2	Lollipop	2478	Adware
3	Kelihos_ver3	2942	Backdoor
4	Vundo	474	Trojan
5	Simda	42	Backdoor
6	Tracur	751	TrojanDownloader
7	Kelihos_ver1	398	Backdoor
8	Obfuscator.ACY	1228	Any kind of obfuscated malware
9	Gatak	1013	Backdoor
	Total	10,859	

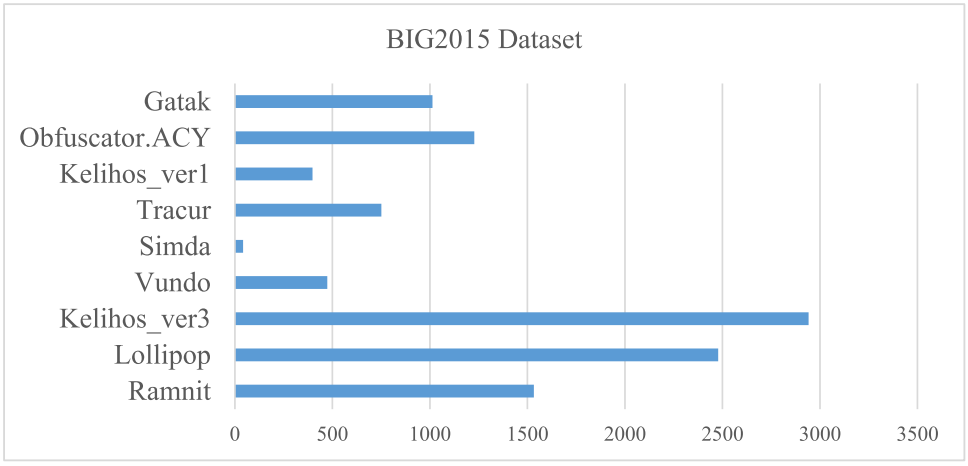


Fig. 1 – BIG2015 Dataset Frequency Distribution.

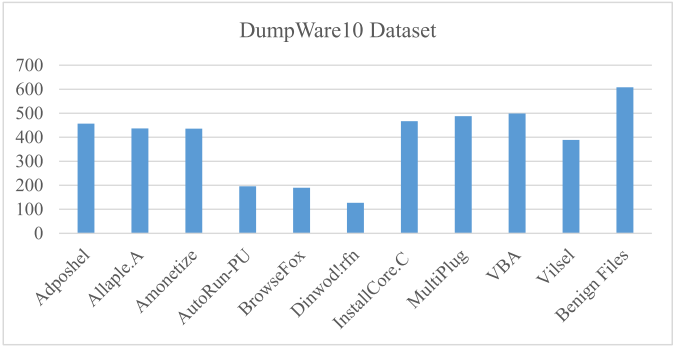


Fig. 2 – DumpWare10 Dataset Frequency Distribution.

Table 2 – The malware families and their categories in the dataset.

Family	Category	Training	Val	Total
Adposhel	Adware	364	93	457
Allaple.A	Worm	349	88	437
Amonetize	Adware	349	87	436
AutoRun-PU	Worm	158	38	196
BrowseFox	Adware	152	38	190
Dinwod!rfn	Trojan	98	29	127
InstallCore.C	Adware	376	91	467
MultiPlug	Adware	390	98	488
VBA	Virus	399	100	499
Vilsel	Trojan	311	78	389
Benign Files	–	487	121	608
	Total	3433	861	4294

Kelihos_ver1, (8) Obfuscator.ACY, (9) Gatak. The data distribution among these malware classes is shown in Fig. 1.

Dumpware10 Dataset: The dataset covering 4294 portable executables, which are grouped under 11 categories (10 malwares + 1 benign class). The dataset contains 608 benign samples as well as 3686 malware samples collected from different families. The distribution and characteristics of the dataset are given in Table 2 and Fig. 2.

In this dataset, malware files are in PNG format. Each malware binary has been translated to 4 different images of sizes 224px, 300px, 4096px and square widths. In this study, we use the image file of size 4096px because with this image size, authors of (Bozkir et al., 2021) obtained the best results when experimenting with various image sizes.

3.2. *Convolutional Neural network (CNN)*

Deep Learning (DL) models are an artificial neural network model. However, DL algorithms differ structurally and numerically from ANNs. Their main difference is that they have so many hidden layers, and therefore they have been called deep learning. They consist of tens, hundreds and even thousands of hidden layers, depending on the need of the system. Convolutional Neural Network (CNN) is a DL model in which the linear regression process performed for each snap in the artificial neural network is represented by a convolution process. CNN was first introduced by LeCun et al. (LeCun et al., 1990) for image processing. The first model consisted of two basic features: Spatial Shared Weights and Spatial Pooling. The same team (LeCun et al., 1998) developed CNN algorithms and published second CNN version as LeNet-5 in 1998. They applied the model on several bank images to increase handwriting number classification success, and achieved success in the first 7 levels.

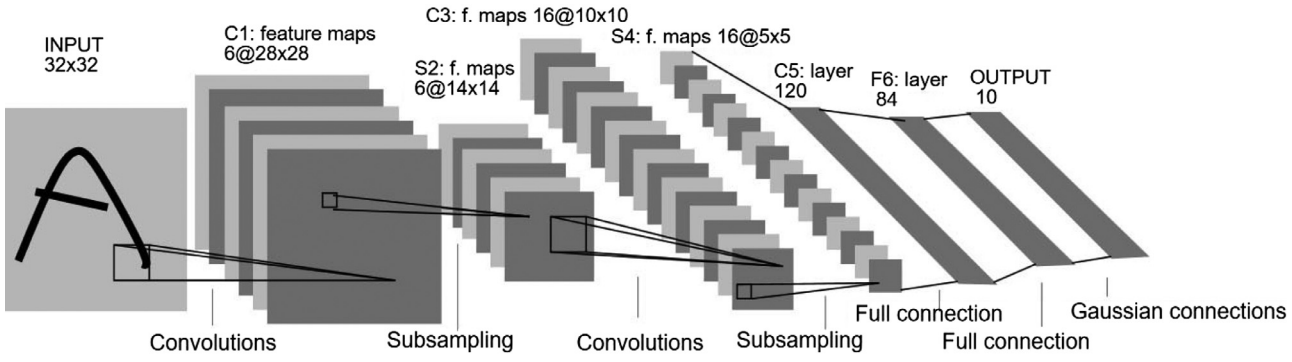


Fig. 3 – Basic CNN architecture introduced by LeCun (Catak et al., 2021).

CNN consists of three basic layers: convolution layer, pooling / sub-sampling layer, and fully-connected layer. An exemplary CNN architecture depicted by LeCun is shown in Fig. 3.

The way CNNs work can be summarized as follows: CNN aims to learn the abstract features of images by using convolution and pooling processes. Features learned in the first layers of the CNN system consist of basic properties such as the edges, curves and colors. The features learned by the last layers consist of the holistic structures, shapes and parts of the objects in the image. In this way, the object in the image becomes clearer as we move from the first layer to the last layer. It has been determined that this learning style has similar features to the cortex of the human eye. CNNs, like human beings, can see the small pieces of the image first and see the big table consisting of the combination of these pieces in the following layers.

In the CNN model, the input data are images digitalized and converted into matrix format. The matrix consists of width, length and color dimensions. If the input image has the gray color level, the matrix consists of a total of 2 dimensions: width, height and one color (such as 512×512×1, trailing 1 represents a gray color). If the image consists of a colored data in RGB format, the input matrix consists of three dimensions, one dimension for each color (such as 512×512×3, trailing 3 representing color tints of red, green and blue). The basic process that CNNs use to determine the attributes in an image is the convolution process. The system learns the features from the data in this matrix format. The basic process that CNNs use to determine the features is the convolution. The convolution takes place by moving the filter / core matrix, which is a smaller matrix, on the input matrix. In this way, the features of the input matrix are mapped. The convolution process is repeated to cover all image pixels by shifting right and down on the image matrix. The mathematical operation performed in the convolution process is the sum of multiplication of the overlapping image pixels and the filter pixels and, then adding the error value (bias) to these results. Thus, the result obtained in each process represents a pixel of the new feature map. The neuron representation is shown in Fig. 4 and the basic convolution process is shown in Fig. 5.

x represents the pixels in the input image, w represents the pixels in the filter image, b represents the bias value, y represents the feature map pixel, and f is the activation function.

An input matrix ($5 \times 5 \times 1$) and a filter matrix ($3 \times 3 \times 1$) are shown in the above example. In the example, a new feature map in dimensions ($3 \times 3 \times 1$) is created by moving the filter matrix over the input matrix. The size of the created feature map varies depending on the size of the input image, the filter size, the number of empty edges on the input, and the number of steps of the filter on the input. The size of the output image can be calculated according to the formula given in Eq.1.

$$O = \frac{(W - K + 2P)}{S} + 1 \tag{1}$$

O represents the size of the output image, W represents the size of the input image, K represents the size of the filter (kernel), P (padding) represents the number of spaces added to the edges of the input image, and S (stride) represents the number of moving steps for filter. The basic formula of the convolution process is given in Eq. (2). In the equation, the pixels of the output image, the pixels of the input image, the pixels of the filter (kernel) and the bias term are represented by y, x, w and b, respectively.

$$y_n = \sum_{n=1}^{9} x_n * w_n + b_n \tag{2}$$

An activation layer is used after every convolution layer to model nonlinear operations. Generally, tanh, sigmoid or ReLU functions are used as activation functions. With this layer, it is aimed to give a nonlinearity to the convolution layer that performs linear regression. Recently, the most used activation function is ReLU. This is because, although ReLU function does not change the classification success so much, it reduces the complexity of the process and provides an increase in performance. It also provides a solution to the gradient vanishing problem that causes the most processing complexity. The ReLU function is simple and calculated by the formula given in Eq.3.

$$ReLU(x) = \begin{cases} 0, & if\ x < 0 \\ x, & if\ x \geq 0 \end{cases} \tag{3}$$

One of the most effective and powerful layers used in convolutional neural networks is the pooling layer. With the pooling layer, the data in the previous layer can be represented

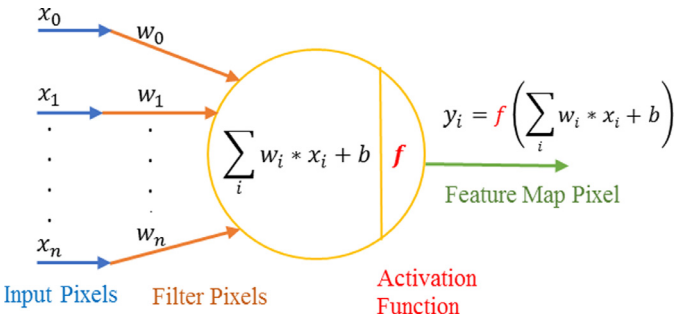


Fig. 4 – Simple neuron representation of CNN.

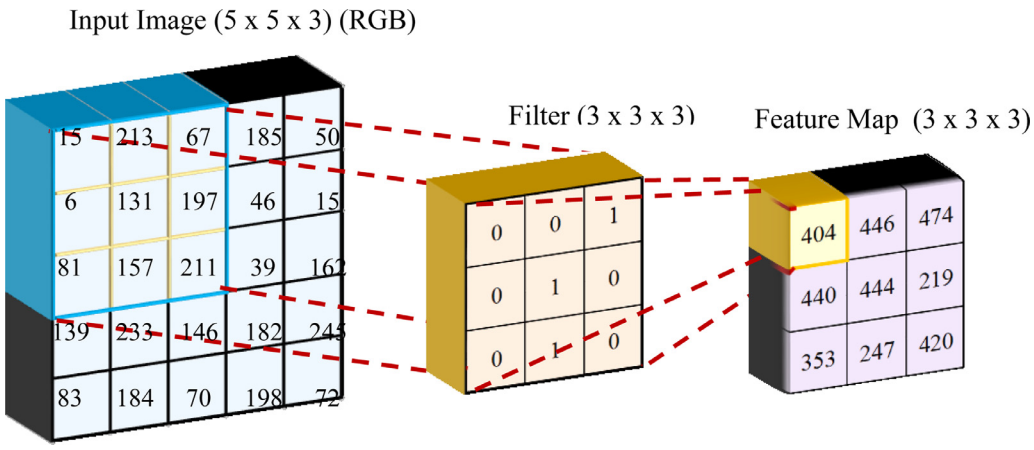


Fig. 5 – Basic convolution process, the convolution operation is obtained by multiplication of the input data matrix and the filter data matrix and adding a bias term.

by sub-samples; therefore, it is also called the sub-sampling layer. There are two frequently used pooling layer types, average pooling and maximum pooling. Average pooling layer produces a new pixel value by taking average values of the matching pixels in the input image. Maximum pooling layer produces a new pixel value by taking maximum values of the matching pixels in the input image. The advantage of the pooling layer is that it reduces the number of parameters by reducing the image to the sub-sample size. Thus, it contributes positively to system performance as the number of transactions decreases. While performing this process, classification success is not affected so much. Another advantage determined in the literature is that it prevents over-fitting (LeCun et al., 1990). Today, some disadvantages of the pooling layer have been identified. This issue can be briefly described as the problem of losing some attributes in the original data while creating a sub-sample based on its data.

The last layer in the CNN model is Fully-Connected (FC) layer, which is one of the basic building blocks of traditional neural networks. FC layers are formed by connecting the output neurons of the previous layers to each neuron in the next layer. The results obtained in this layer are then normalized to a probability distribution for the classification process using a SoftMax layer. FC layers are aimed at taking high-level filtered images and converting them into proportions representing classes. These rates are expressed as weight or link strength between each value and each class.

3.3. *DenseNet*

Huang et al. (Huang et al., 2017) developed DenseNet with the prediction that short connections between the first layers and subsequent layers in CNN models can be deeper, more accurate and more efficient. DenseNet consists of fully interconnected layers, one between each layer in the classical CNN model and the next layer. A normal N-layer CNN consists of N connections in DenseNet model since each layer is connected to each layer that comes after it, and it consists of N $(N + 1) / 2$ connection. For each layer, feature maps of all previous layers are used as input, and their feature maps are sent as input to all subsequent layers. In this way, DenseNets tries to solve the vanishing gradient problem like ResNet, strengthens feature propagation, encourages feature reuse and significantly reduces the number of parameters. DenseNet has types with 121, 169, 201, 264 layers. The DenseNet-121 model was used within the scope of the study. The figure of DenseNet model is given in Fig. 6.

3.4. *Cycle-GAN*

Generative Adversarial Networks (GANs) are machine-learning algorithms used to create images that are not real. GAN was first described by Goodfellow et al. (Goodfellow et al., 2014) emerged from an adversarial network-based study. For example, GANs can create images that look like the image

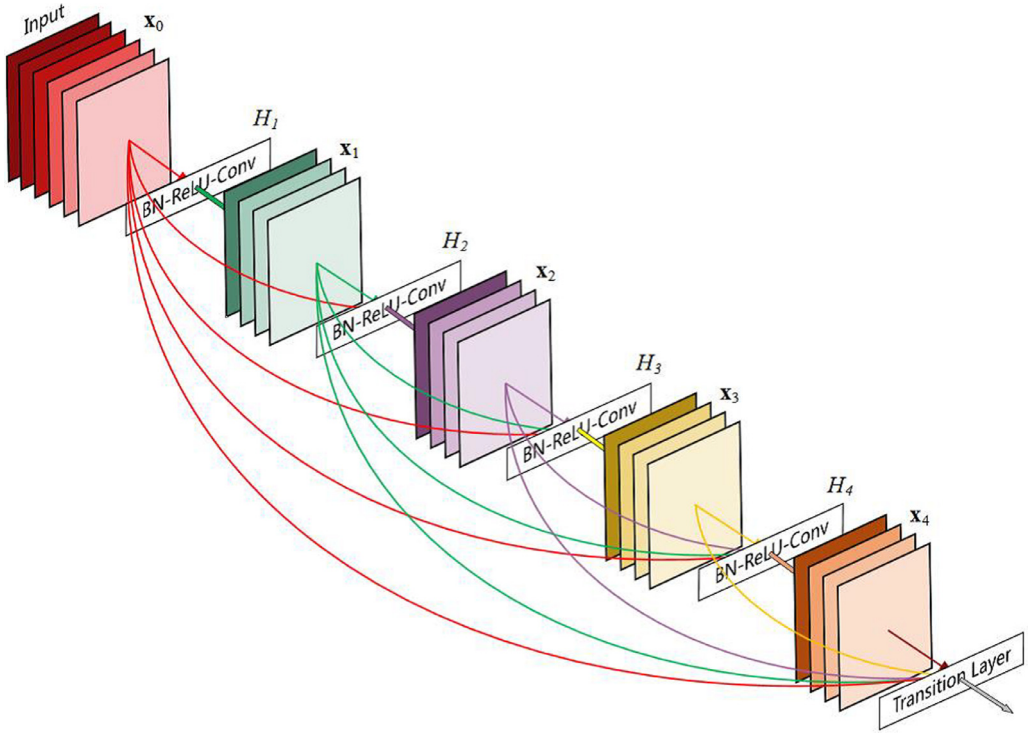


Fig. 6 – The structure of the DenseNet model (Yan et al., 2018).

of human faces, even if the faces do not belong to any real person. The Cycle Generative Adversarial Network or CycleGAN is an approach to training a deep convolutional neural network for image-to-image translation tasks. Unlike other GAN models, Cycle-GAN does not require a dataset of paired images. Cycle-GAN model has a cyclical structure and an inter-cycle consistency constraint (Tang et al., 2019). Image-to-image translation is a class of image and graphics problems where the purpose is to learn the mapping between an input image and an output image using a training dataset of aligned pairs of images. Image-to-image translation is the production of a new synthetic version of an image. Image-to-image translation usually requires a large paired sample dataset. These datasets can be difficult and expensive to prepare. CycleGAN is a technique that involves automatic training of image-to-image translation models. Models are trained unsupervised using a collection of images that do not need to be associated in any way from the source and target domain.

4. Proposed deep learning architecture

In this study, we propose a hybrid method for malware classification consisting of data augmentation and classification stages. Moreover, we propose a method to convert byte files to color and gray image format within the scope of the study. This method is named B2IMG (Bytes to Image). The proposed hybrid system was tested on two different databases used in the literature. These databases are *Microsoft Malware Classification Challenge Dataset (BIG 2015)* and *Dupmware10*. The data in the Dumpware10 database are in color image format, but the data in the BIG2015 database are in the form of asm and bytes files. Therefore, first, all data in the BIG2015 database were converted to JPG image format with the recommended B2IMG method. Then, the classification process was carried out separately on these RGB and gray images.

One of the most important problems affecting the classification success of malware families is the unbalanced data distribution between classes. The number of unbalanced data between classes in both data sets used in this study is clearly seen in Fig. 1 and Fig. 2. The big Classes reduce the classification success by causing the method to ignore other classes with less data. For this reason, the proposed classification method has been applied in two basic stages. In the first stage, a Generative adversarial network (GAN) based data augmentation method is proposed to tackle the imbalanced data problem between classes in the database. In the second stage, training was carried out on DenseNet network by using the augmented data. The system architecture of the proposed method is shown in Fig. 7.

4.1. *Bytes to image method*

Classification processes were carried out on the images of each malware. The data of each class of the Dumpware10 database are in PNG image format. However, the data in BIG2015 database consist of two types as ASM file and Bytes file. ASM files consist of assembly codes of the malware. On the other hand, the Bytes files consist of hexadecimal numbers, which are the compiled versions of the codes. Each row of Bytes files has a hexadecimal number consisting of 8 digits to represent the row number. Then, sixteen hexadecimal numbers between 00 and FF follow this row number. Each data in

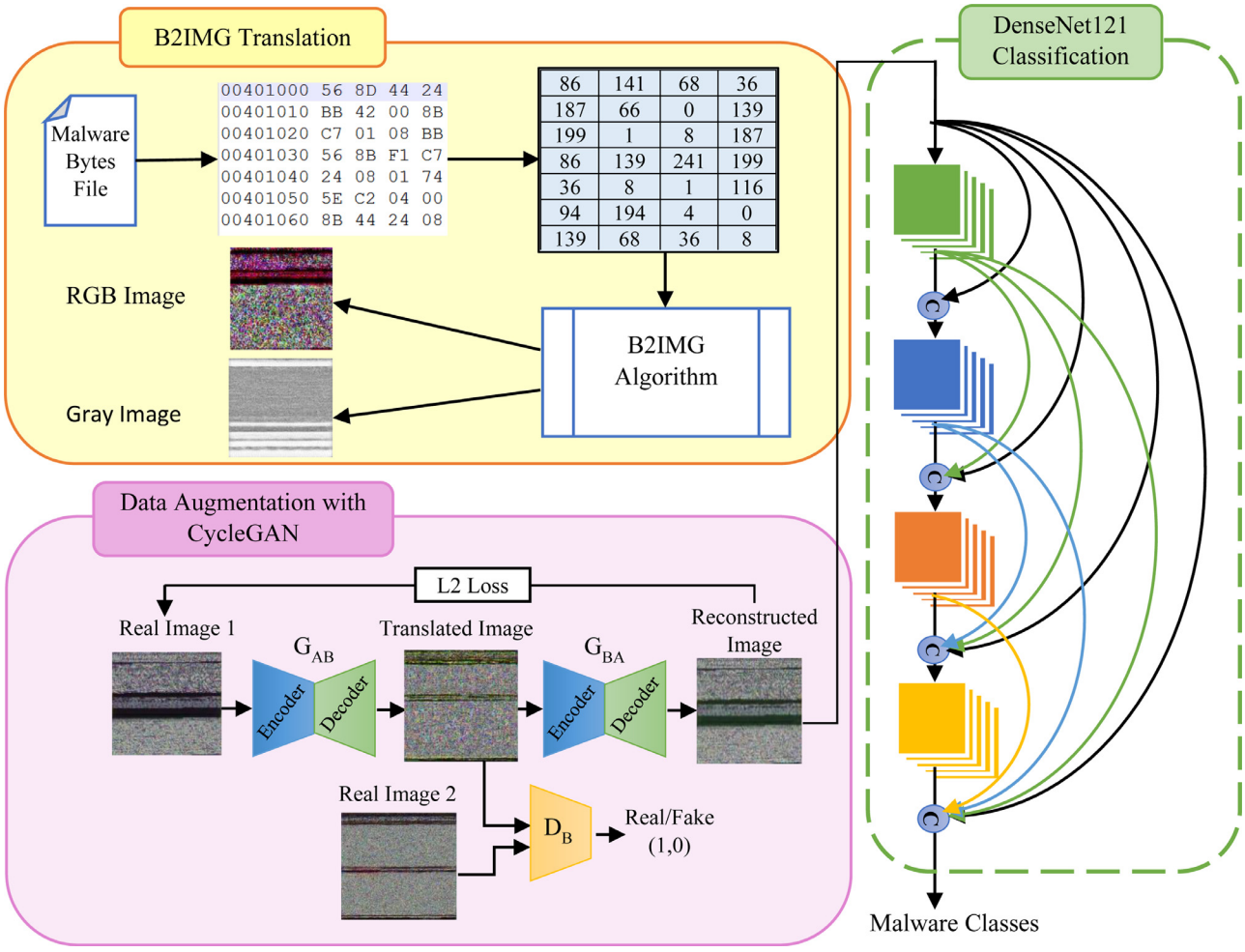


Fig. 7 – Architecture of the proposed method.

the row is separated by spaces. Within the scope of the study, RGB and gray images were obtained by using these Bytes files. We proposed two different methods to convert Bytes files to RGB and Gray images.

In both methods, we proposed, firstly, each data in the Bytes file were read line by line. The data were split into an array according to the spaces in the lines. The Bytes files include meaningless characters and numbers such as ?? and 00. These characters were detected and cleared. Since the first element of the array represents the row number, it was cleared too. The other 16 hexadecimal values remaining in the array were converted to decimal numbers between 0 and 255. These decimal numbers were used to create the pixels in the gray and RGB images by using the proposed B2IMG method. In order to equalize the aspect ratios of the created images, the number of elements of the array with decimal data was divided by the number of channels of the image to be converted. Then, the aspect ratio of the image was determined by using the square root of the obtained result. A matrix was created according to the aspect ratio and the number of channels, and all values were set to 0. After then, the values between 0 and 255 in the decimal array were loaded in this matrix. Since there are three channels in RGB images, a 3-dimensional matrix was created, and a 2-dimensional matrix was created in gray images. Finally, images of each malware were obtained using the matrix. The algorithm of the proposed B2IMG method is given in Algorithm 1.

4.2. *Data augmentation method*

In the study, a data augmentation method is proposed to eliminate the imbalance between the data belonging to the classes. The proposed data augmentation method is based on Generative Adversarial Networks (GAN). Cycle-GAN model, which is a type of GANs, has been developed for image to image transform. While transforming from image to image, it creates a feature map by learning the features between the image pairs which are given as input and output to the network. Then, image to image transform process is performed by using this feature map. We can exemplify the image to image transform process as follows: Let's take an orange and apple image pair, we can train the GAN network with orange images so that it learns the orange texture map. Then we can use this texture map to get a sample of the apple image converted to orange texture. In the Cycle-GAN method, these transformations are constantly repeated between each other and the net-

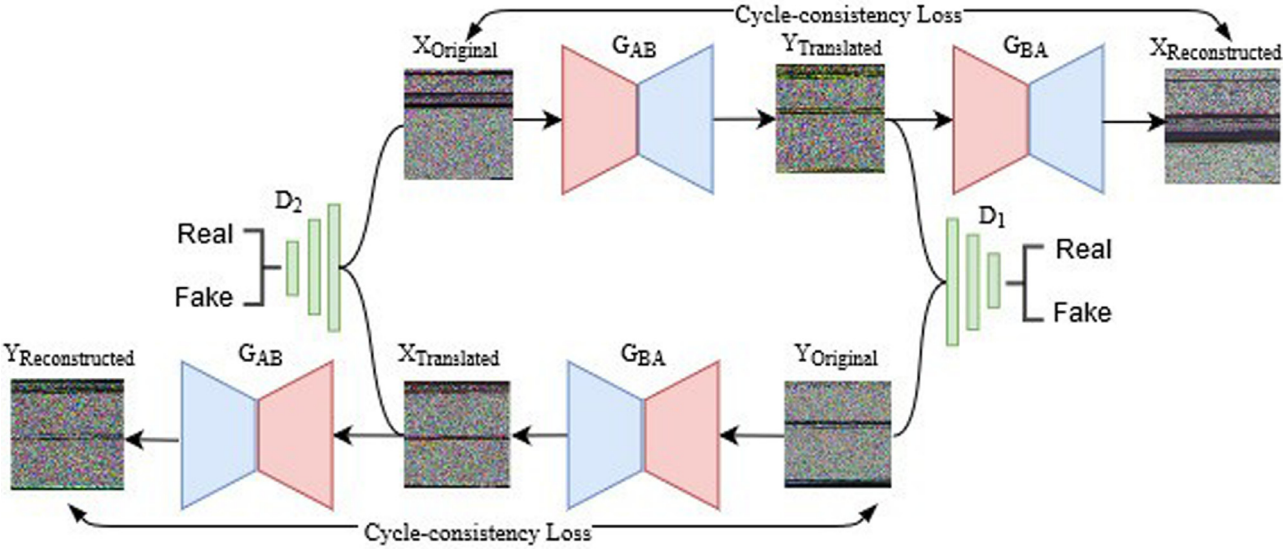


Fig. 8 – The structure of the proposed data augmentation method.

Algorithm 1 – Algorithm of bytes to image transformation.

Step 1 : While(Read Line with (filename))
Step 2 : Line split in pixel array according to the spaces
Step 3 : Foreach(item in pixel array)
Step 4 : IF(item ==?? OR item <=00)
Step 5 : Clear item
Step 6 : ELSE
Step 7 : Convert item hexadecimal to decimal
Step 8 : Load converted item in pixel array
Step 9 : End While
Step 10 : image size = Ceil($\sqrt{\frac{pixel\ array\ length}{color\ channel}}$)
Step 11 : Create matrix with size of (image size X image size X color channel)
Step 12 : Load 0 values in matrix
Step 13 : Reshape pixel array with (image size X image size X color channel)
Step 14 : Load pixel array in matrix
Step 15 : Convert matrix to image

work is provided to create realistic images. In this study, the ability of the CycleGAN model to learn the feature map between image pairs is rearranged and used for data augmentation. Cycle-GAN network was used on the malware images to learn the features between different images belonging to the same malware family. Similar malware samples were obtained from these learned features. The structure of the proposed data augmentation method is given in Fig. 8.

Data augmentation was carried out for only classes with fewer data to provide a balance in the number of data between classes. In the BIG2015 database, the number of data in Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator and ACY classes was augmented as many as 300, 738, 400, 100, 400, respectively. The number of trainings, augmented trainings, and test data for BIG2015 database is given in Table 3.

In the Dumpware10 database, augmented malware families are AutoRun, BrowseFox, Dinwod! Rfh, MultiPlug, Vilsel and Benign. The number of trainings, augmented trainings, and testing data in the Dumpware10 database is given in Table 4. In both databases, only training data were augmented. Data augmentation process was not performed on the test data.

5. Experiments and results

Malware classification was performed on two different databases named BIG2015 and, DumpWare10. In order to prove the success of the proposed method, the experiments were carried out in two phases separately for both databases. In the first phase, experiments were carried out without any data augmentation and the results were obtained. In the second phase, the experiments were repeated with the augmented data on the proposed hybrid model and the results were obtained. All experiments were coded using PYTHON language in UBUNTU operating system and executed on the graphics card (GPU) using the Tensorflow library in the KERAS

Table 3 – The number of trainings, augmented trainings, and test data for BIG2015.

	Ramnit	Lollipop	Kelihos_ ver3	Vundo	Simda	Tracur	Kelihos_ ver1	Obfuscator. ACY	Gatak
The Number of Trainings	1073	1735	2059	332	29	526	279	860	709
The Number of Augmented Trainings	0	0	0	300	738	400	100	400	0
The Number of Test Data	460	743	883	142	13	225	119	368	304

Table 4 – The number of trainings, augmented trainings, and test data for Dumpware10.

	Adposhel	Allaple.A	Amonetize	AutoRun-PU	BrowseFox	Dinwod!rfh	InstallCore.C	MultiPlug	VBA	Vilsel	Benign
The Number of Trainings	364	349	349	158	152	98	376	390	399	311	487
The Number of Augmented Training	0	0	0	600	598	598	0	510	0	490	577
The Number of Test Data	93	88	87	38	38	29	91	98	100	78	121

framework. The system specifications used in the experiments consisted of NVIDIA TITAN XP graphics card with 12GB DDR5 memory, I7 CPU, 32GB DDR3 RAM and 1 TB SSD hard memory.

Before the experiments were performed, the pre-processing stage was applied to the databases. All images in the BIG 2015 database were transformed to JPG format using the proposed B2IMG method. Since the images in the Dumpware10 database are in PNG format, the transformation process was not applied. In the pre-process stage, the images were loaded and converted into digital matrix format. All images on the BIG 2015 database were resized to 224×224. The gray and RGB images in the database were trained separately on the proposed system, and the results were obtained. Since the Dumpware10 database contains only RGB images, a gray level training was not carried out. Each image on the Dumpware10 database was resized to 300×300 since the authors reported that they achieved the best classification success on the DumpWare10 with 300×300 image size. Therefore, we used the same sizes in order to make a complete system comparison in this study.

Before the experiments were applied, first, the data were split into two as 80% training and 20% testing. Test data were not used during the training phase. All experiments performed were carried out with 10-fold cross validation (CV) to prevent the possibility of the overfitting. According to the working structure of the 10-fold CV model, 90% of the data at each fold training phase was used for training and the remaining 10% for the validation phase. The final results were obtained through test data that the system had never seen during the training phase. Moreover, dropout layer was also used to prevent the possibility of overfitting the system. In the proposed classification system, two more fully connected (FC) layers was added on the end of the DenseNet-121 model in the KERAS framework. The dimensions of these FC layers were 1024 and 512, respectively. After each FC layer, dropout layers and batch normalization layers were used. The dropout layers had ratios of 0.3 and 0.2, respectively. System training was carried out with 32 Batch sizes on 100 epochs. Stochastic Gradient Descent (SGD) method was used as optimization with the values of 0.9, 0.001 and 0.000001 for momentum, learning coefficient and, decay respectively.

The success of the system in the study was determined via accuracy (Acc), specificity (S), area under the curve (AUC), precision (P), recall (R) and F1-score metrics. These metrics were preferred to compare the system results with other studies in the literature. These metrics were given in Eq. (4) and, Eq. 5.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \; S = \frac{TN}{TN + FP} \tag{4}$$

$$P = \frac{TP}{TP + FP}, \; R = \frac{TP}{TP + FN}, \; F1 = 2.\frac{Precision.Recall}{Precision + Recall} \tag{5}$$

4 different experiments were performed on the BIG 2015 database. In order to prove the success of the proposed data augmentation method, the system was trained with the original data and results were obtained. Then, the data were augmented by the proposed method within the scope of the study. After then, the proposed system was also trained with the

Table 5 – Comparison of Malware families of BIG2015 dataset by using gray images

		Ramnit	Lollipop	Kelihos_ver3	Vundo	Simda	Tracur	Kelihos_ver1	Obfuscator.ACY	Gatak
Without Augmentation	Auc	0,9841	0,9908	0,9996	0,9918	0,9226	0,9720	0,9995	0,9721	0,9949
	Spectivity	0,9943	0,9964	0,9992	0,9978	0,9991	0,9974	0,9990	0,9958	0,9997
	Accuracy	0,9914	0,9939	0,9994	0,9972	0,9985	0,9939	0,9991	0,9905	0,9988
	Precision	0,9655	0,9879	0,9977	0,9524	0,7857	0,9638	0,9754	0,9668	0,9967
	Recall	0,9739	0,9852	1,0000	0,9859	0,8462	0,9467	1,0000	0,9484	0,9901
	F1-Score	0,9697	0,9865	0,9989	0,9689	0,8148	0,9552	0,9876	0,9575	0,9934
With Augmentation	Auc	0,9861	0,9896	1,0000	0,9920	0,9231	0,9767	0,9995	0,9686	0,9965
	Spectivity	0,9939	0,9980	1,0000	0,9981	1,0000	0,9934	0,9990	0,9969	0,9997
	Accuracy	0,9917	0,9942	1,0000	0,9975	0,9994	0,9911	0,9991	0,9905	0,9991
	Precision	0,9636	0,9932	1,0000	0,9589	1,0000	0,9153	0,9754	0,9746	0,9967
	Recall	0,9783	0,9812	1,0000	0,9859	0,8462	0,9600	1,0000	0,9402	0,9934
	F1-Score	0,9709	0,9871	1,0000	0,9722	0,9167	0,9371	0,9876	0,9571	0,9951

Table 6 – Comparison of malware families of BIG2015 dataset by using RGB images.

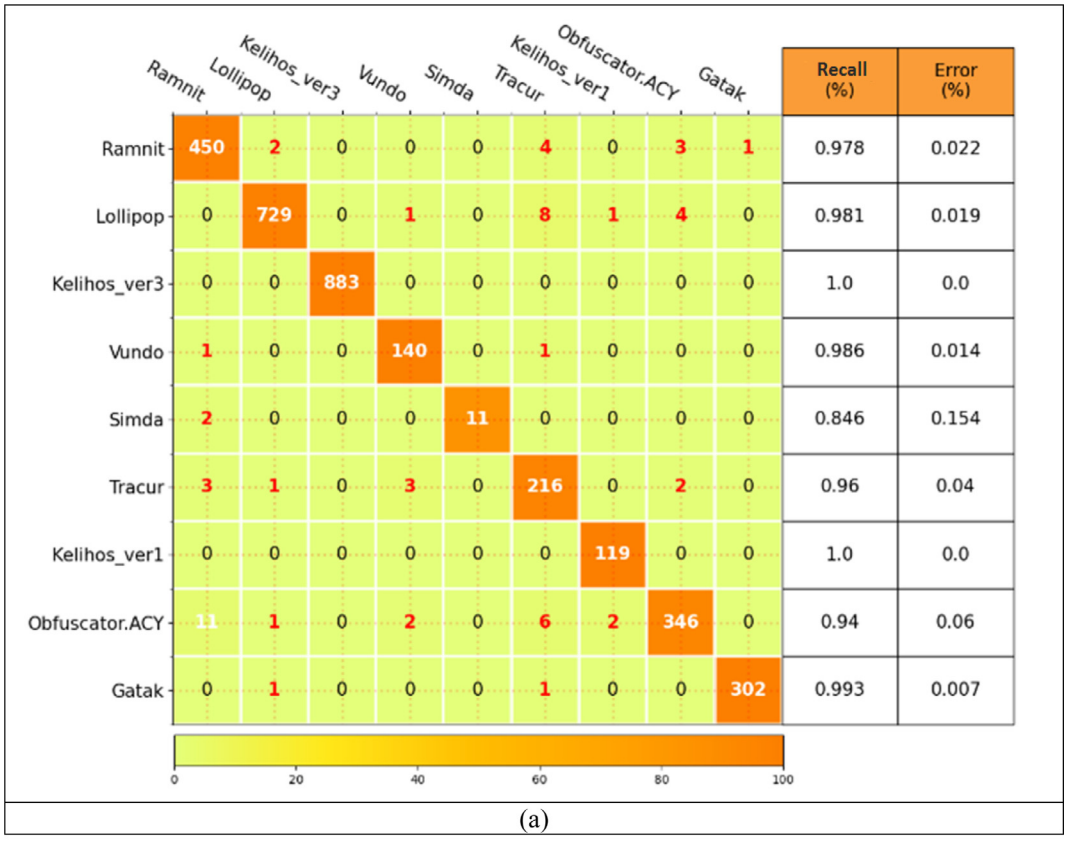
		Ramnit	Lollipop	Kelihos_ver3	Vundo	Simda	Tracur	Kelihos_ver1	Obfuscator.ACY	Gatak
Without Augmentation	Auc	0,9933	0,9889	0,9998	0,9994	0,9231	0,9802	0,9998	0,9694	0,9975
	Spectivity	0,9932	0,9980	0,9996	0,9987	1,0000	0,9960	0,9997	0,9986	0,9983
	Accuracy	0,9932	0,9939	0,9997	0,9988	0,9994	0,9939	0,9997	0,9920	0,9982
	Precision	0,9601	0,9932	0,9989	0,9726	1,0000	0,9476	0,9917	0,9886	0,9838
	Recall	0,9935	0,9798	1,0000	1,0000	0,8462	0,9644	1,0000	0,9402	0,9967
	F1-Score	0,9765	0,9864	0,9994	0,9861	0,9167	0,9559	0,9958	0,9638	0,9902
With Augmentation	Auc	0,9926	0,9942	1,0000	0,9992	0,9231	0,9876	0,9958	0,9753	0,9977
	Spectivity	0,9961	0,9992	1,0000	0,9984	1,0000	0,9974	1,0000	0,9969	0,9986
	Accuracy	0,9951	0,9969	1,0000	0,9985	0,9994	0,9960	0,9997	0,9920	0,9985
	Precision	0,9764	0,9973	1,0000	0,9660	1,0000	0,9649	1,0000	0,9750	0,9870
	Recall	0,9891	0,9892	1,0000	1,0000	0,8462	0,9778	0,9916	0,9538	0,9967
	F1-Score	0,9827	0,9932	1,0000	0,9827	0,9167	0,9713	0,9958	0,9643	0,9918

augmented data and results were obtained. These operations were applied separately for gray and RGB image data. Data augmentation was used only in the training phase. So, only the original data were used in the experiments without data augmentation while the merge of the augmented data and original data was used as training data in experiments with augmented data. The results obtained with gray images in the BIG 2015 database are given in Table 5. System achievement for each malware class is given separately in the table. In addition, the confusion matrix is given in Fig. 9 in order to better analyze the results of the classes.

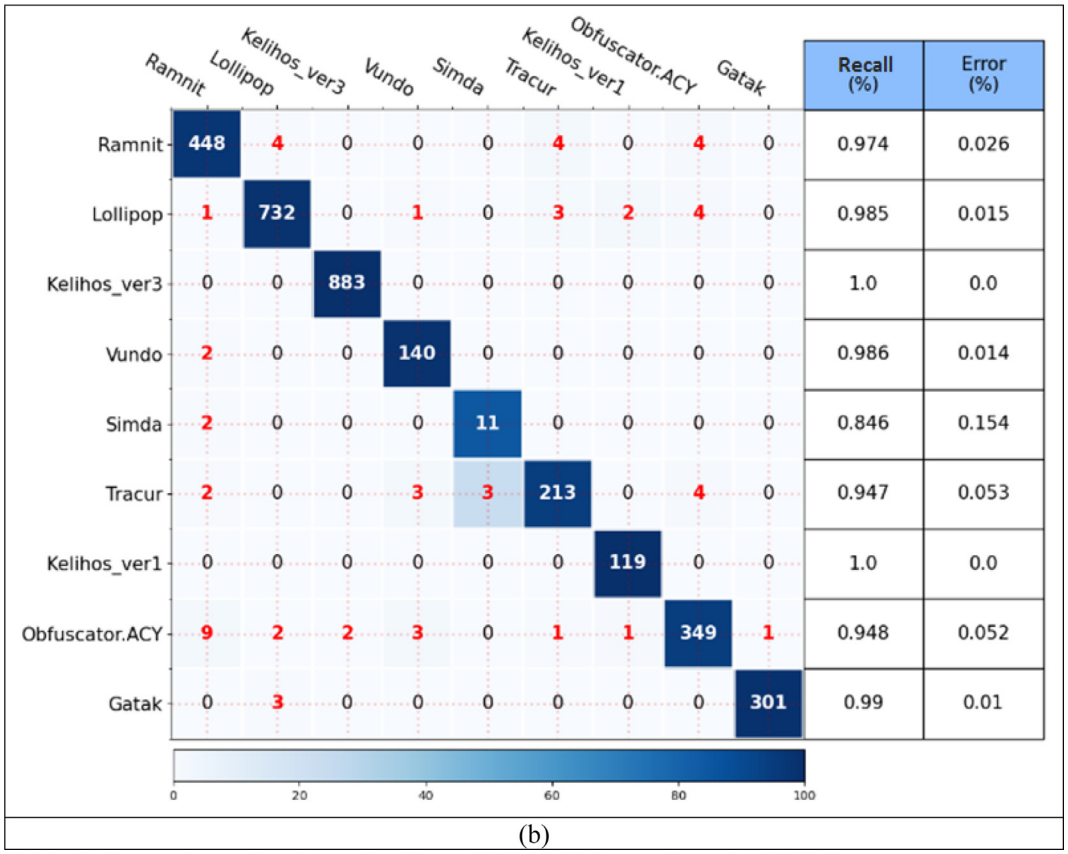
RGB images can represent more features than gray images. Hence, RGB images were used to sample more attributes of each malware on the BIG 2015 database. The number of used RGB images is the same as for gray images. An RGB image was created for each gray image. The results obtained with RGB images on the proposed system are given in Table 6. System achievement was reported separately for each malware class in the table. In addition, the confusion matrix graphics of the classes are given in Fig. 10. When the experimental results were examined, it was seen that the data augmentation on RGB images increases success. RGB images achieved better results than gray images since they contain more attributes. Classification success with gray images was 100% on only 2 malware classes while success with RGB images increased to 3. In RGB images, it was seen that the classification success increased on Ramnit, Lollipop, Vundo, Tracur, Obfuscator.ACY, Gatak families.

There are 11 different malware families in the Dumpware10 database, along with Benning. The data format is RGB on the database and, there are not any gray images. The database contains 4 different sample sizes with widths of 224px, 300px, 4096px and square for all data. The researchers of the DumpWare10 database reported that they obtained the best results with 300×300px sized images in 4096px folder. Therefore, within the scope of this study, the 300×300px RGB images in the folder of 4096 were used for classification. As with the BIG2015 database, the data augmentation was performed only on classes which had insufficient training data in DumpWare10 databases, too. The results obtained in the experiments are given in Table 7. System success for each malware class is given separately in the table. The confusion matrix showing the results of the classes is given in Fig. 11. in order to make better analysis. When the obtained results were examined, the achievement of the proposed data increase method was clearly seen. It was seen that the classification success of the augmented classes increased. It was also observed that success was increased especially on MultiPlug and Benign families. Moreover, 100% success was achieved in all metrics on Allaple.A and VBA families.

In order to better prove the contribution of the proposed method to the literature, it was compared with the studies

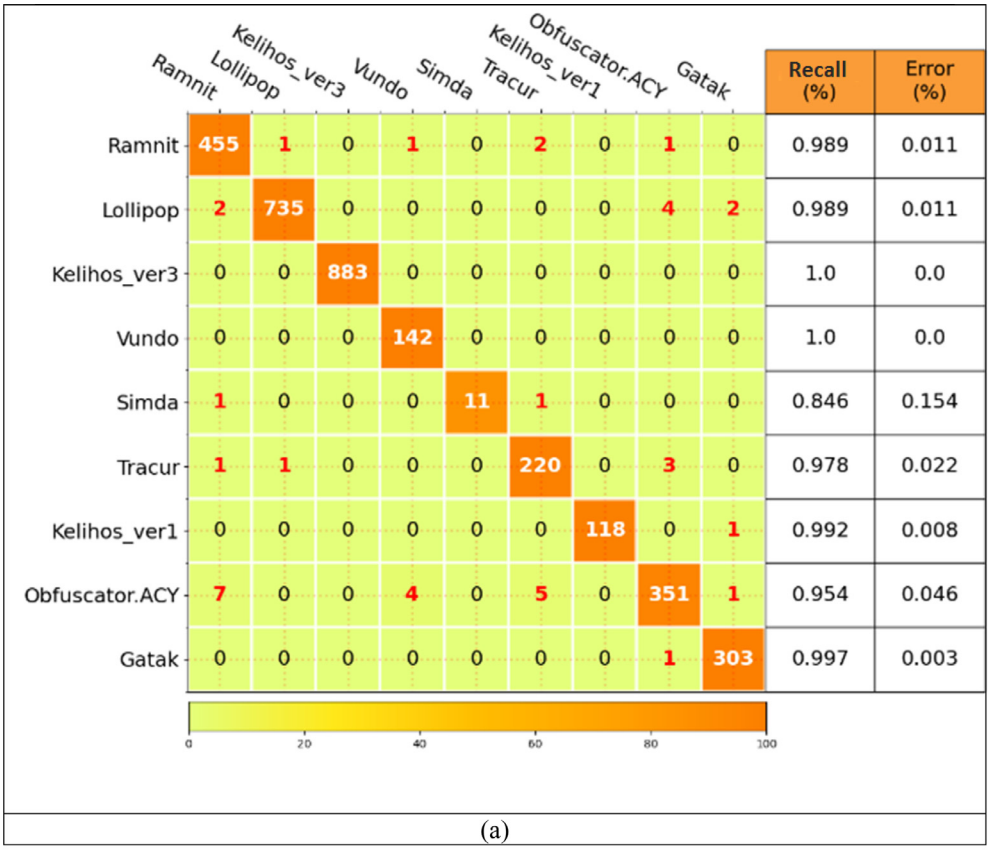


(a)

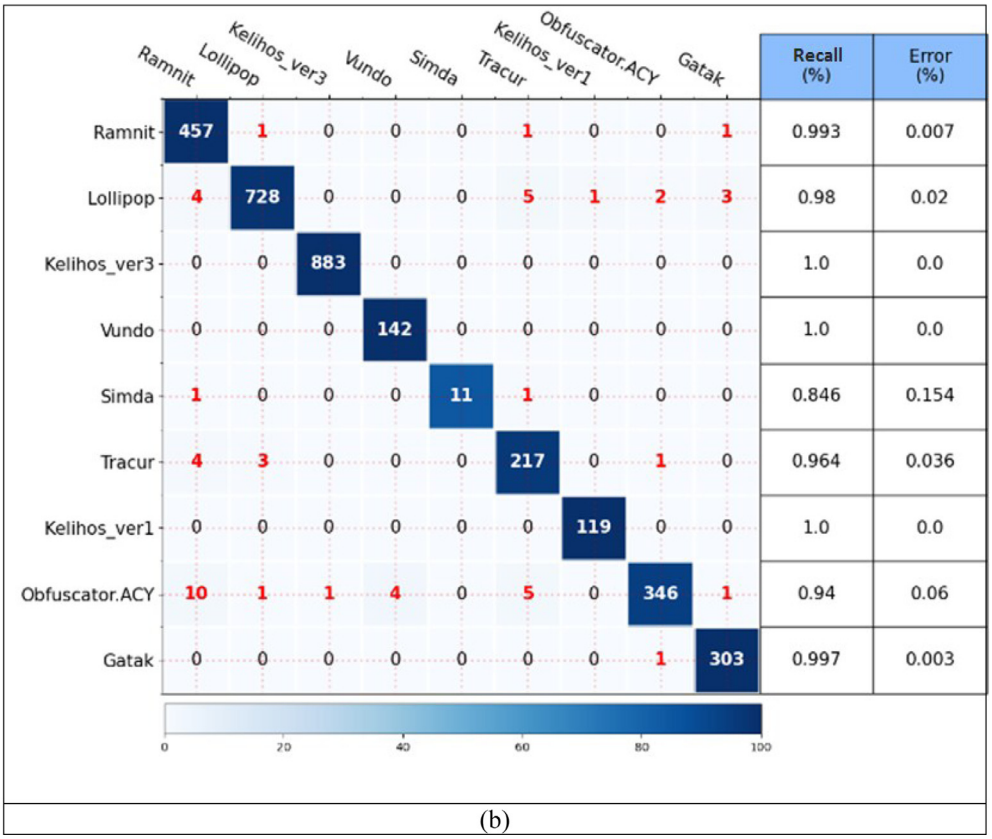


(b)

Fig. 9 – Confusion matrix of malware classes on gray image results, (a) with proposed data augmentation method, (b) without data augmentation process.

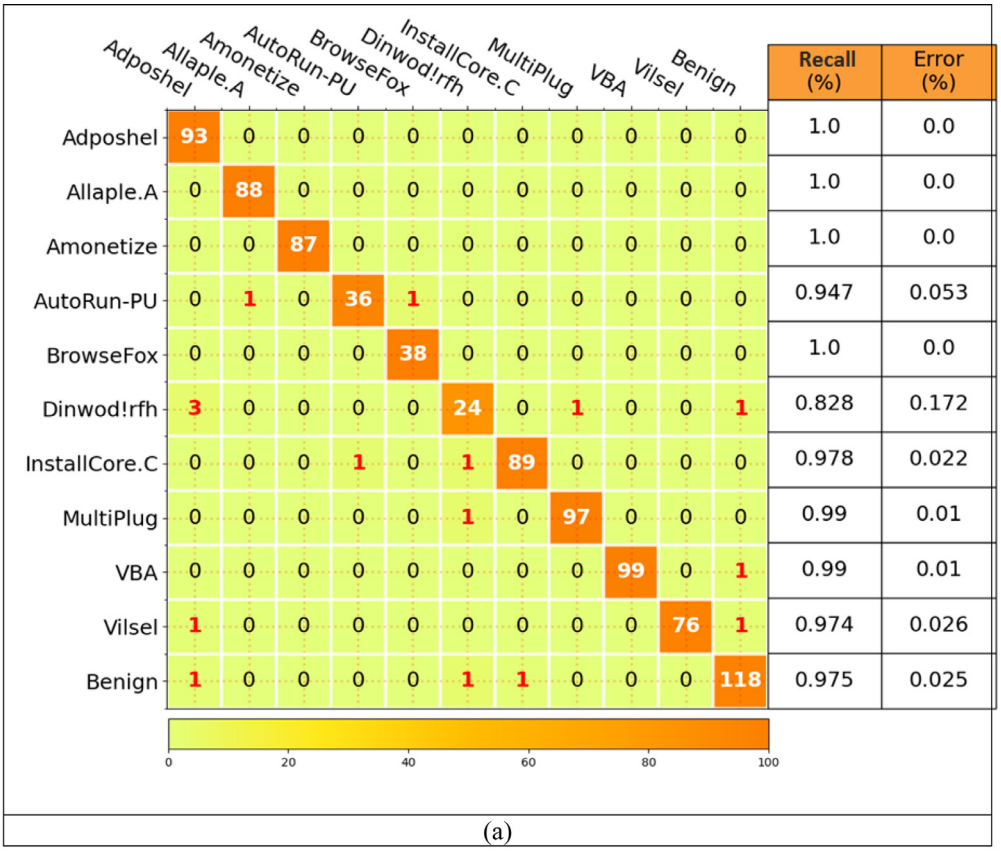


(a)

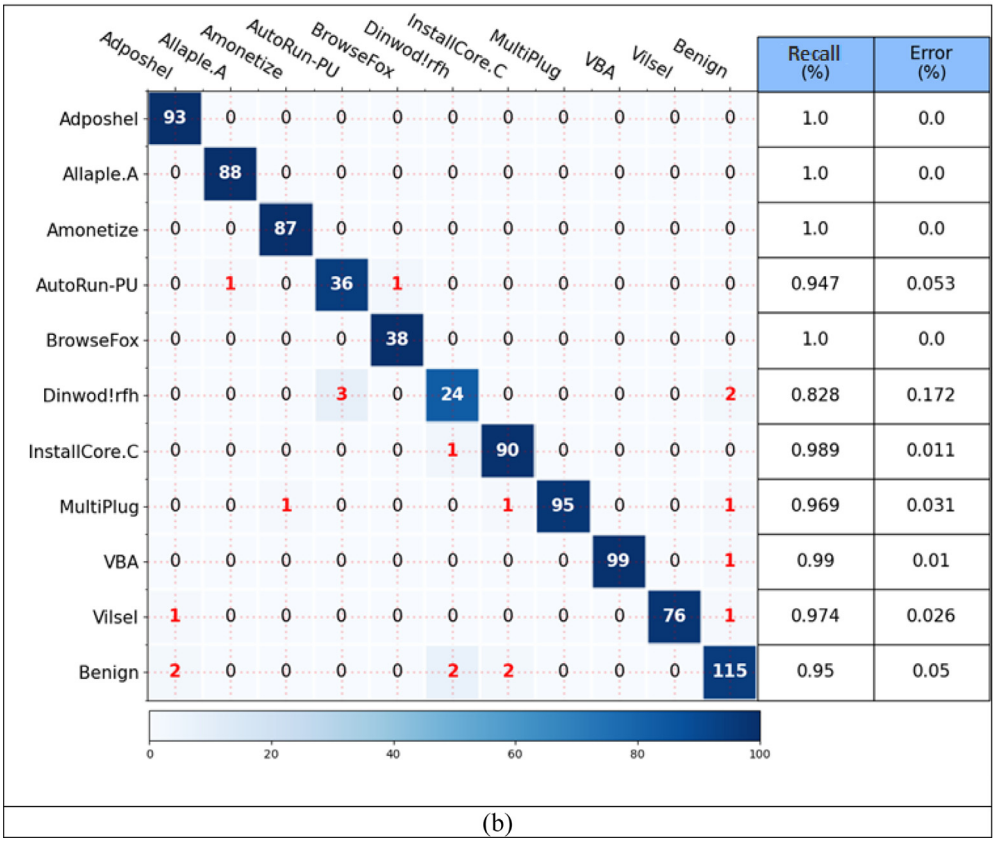


(b)

Fig. 10 – Confusion matrix of malware classes on RGB image results, (a) with proposed data augmentation method, (b) without data augmentation process.



(a)



(b)

Fig. 11 – Confusion matrix of malware classes on RGB image results, (a) with proposed data augmentation method, (b) without data augmentation process.

Table 7 – Comparison of malware families of DumpWare10 dataset by using RGB images.

		Adposhel	Allaple.A	Amonetize	AutoRun-PU	BrowseFox	Dinwod!rfh	InstallCore.C	MultiPlug	VBA	Vilsel	Benign
Without Augmentation	Auc	0,9980	0,9994	0,9994	0,9719	0,9994	0,9120	0,9926	0,9847	0,9950	0,9872	0,9718
	Spectivity	0,9961	0,9987	0,9987	0,9964	0,9988	0,9964	0,9961	1,0000	1,0000	1,0000	0,9932
	Accuracy	0,9965	0,9988	0,9988	0,9942	0,9988	0,9907	0,9954	0,9965	0,9988	0,9977	0,9872
	Precision	0,9688	0,9888	0,9886	0,9231	0,9744	0,8889	0,9677	1,0000	1,0000	1,0000	0,9583
	Recall	1,0000	1,0000	1,0000	0,9474	1,0000	0,8276	0,9890	0,9694	0,9900	0,9744	0,9504
	F1-Score	0,9841	0,9944	0,9943	0,9351	0,9870	0,8571	0,9783	0,9845	0,9950	0,9870	0,9544
With Augmentation	Auc	0,9980	1,0000	0,9994	0,9725	0,9988	0,9132	0,9877	0,9834	1,0000	0,9859	0,9766
	Spectivity	0,9961	1,0000	0,9987	0,9976	0,9976	0,9988	0,9974	0,9974	1,0000	0,9974	0,9946
	Accuracy	0,9965	1,0000	0,9988	0,9954	0,9977	0,9930	0,9954	0,9942	1,0000	0,9954	0,9895
	Precision	0,9688	1,0000	0,9886	0,9474	0,9500	0,9600	0,9780	0,9794	1,0000	0,9744	0,9667
	Recall	1,0000	1,0000	1,0000	0,9474	1,0000	0,8276	0,9780	0,9694	1,0000	0,9744	0,9587
	F1-Score	0,9841	1,0000	0,9943	0,9474	0,9744	0,8889	0,9780	0,9744	1,0000	0,9744	0,9627

in the literature. Studies conducted with BIG2015 and DumpWare10 databases were compared separately. When the studies in the literature are examined, it was seen that there were not many studies since Dumpware10 was a new database. Only the authors who created the DumpWare10 database had their study on the database. On the other hand, there were many studies in the BIG2015 database. These studies reported success with various classification methods such as SVM, CNN, RNN, LSTM. Comparison of the proposed method with the literature is given in Table 8 and Table 9 for the BIG2105 and DumpWare10 databases, respectively.

By using the proposed B2IMG method, the classification process was performed separately with the gray and RGB images obtained from the bytes files in the BIG2015 database. According to the results, the classification performance was better on RGB images. Moreover, it was seen that the results with the proposed data augmentation method were better than the results obtained without further augmentation. The proposed CNN method achieved 99.86% accuracy with the RGB images by using original malware data without data augmentation. The classification accuracy was 99.73% with the RGB images augmented by using the proposed CycleGAN based augmentation method. Although the accuracy performance of the proposed method without data augmentation was seen to be better than the accuracy performance of the augmented data, the results of Recall, F1-Score, Precision, and AUC were found to be higher with data augmentation. When compared with the literature, it is seen that the proposed CycleGAN + CNN hybrid method improves the classification performance by increasing approximately 2%. The proposed method achieved the best result in the literature with 97.16% Recall, 97.76% F1-Score, 98.52% Precision, 98.51% AUC, and 99.86% Accuracy in the BIG2015 database. In addition, it is seen that the proposed CycleGAN + CNN method also achieved the best result in literature with gray images.

DumpWare10 database is a new database, so there is only 1 studyperformed by the researchers of the database. Hence, the obtained results were compared only with the researchers' study. In the DumpWare10 study, the results were obtained by making classifications in different data sizes. We made the comparison with the results of 300×300px images in the 4096px folder, which is the best result reported by the DumpWare10 researchers. The researchers classified the images in the 4096px folder by using five different classification methods, namely Random Forest, SMO (RBF), SVM (Linear), XGBoost, and J48. In this study, the proposed method was compared with the results of the first four classification methods, in which the DumpWare10 researchers achieved the best results. Within the scope of this study, we performed two different classifications with original (without augmentation) and with augmented RGB images in the DumpWare10 database. The results obtained showed that the proposed hybrid system improved the accuracy in the literature by approximately 5%. Both the proposed data augmentation classification method and the non-data-augmentation classification method achieved better results in the DumpWare10 database than the study/studies? in the literature. The proposed CycleGAN + CNN method achieved the best results with 96.87% Recall, 97.08% F1-Score, 97.39% Precision, 98.32% AUC, and 99.60% Accuracy on the database. The proposed data aug-

Table 8 – Literature comparison of the proposed hybrid classification system on BIG2015 dataset.

Study	Method	File	AUC	Precision	Recall	F1-Score	Accuracy
Kim et al. (Kim et al., 2018)	tDCGAN	Bytes					95,74
Kang et al. (Kang et al., 2019)	LSTM	ASM	–	–	–	–	97,59
Kadri et al. (Al Kadri et al., 2019)	CNN	Bytes	–	–	–	–	98
Gao et al. (Gao et al., 2020)	XGBoost+RNN	Bytes	–	96,92	96,90	96,81	96,90
Jang et al. (Jang et al., 2020)	CNN	ASM					99,65
Proposed system without augmentation by using gray image	CNN	Bytes	98,00	95,00	96,00	96,00	99,76
Proposed system with augmentation by using gray image	CycleGAN+CNN	Bytes	98,13	97,53	96,50	96,93	99,58
Proposed system without augmentation by using RGB image	CNN	Bytes	98,00	98,00	97,00	97,00	**99,86**
Proposed system with augmentation by using RGB image	CycleGAN+CNN	Bytes	**98,51**	**98,52**	**97,16**	**97,76**	99,73

Table 9 – Literature comparison of the proposed hybrid classification system on DumpWare10 dataset.

Study	Method	File	AUC	Precision	Recall	F1-Score	Accuracy
Bozkir et al. (Bozkir et al., 2021)	Random Forest	Bytes		93,10	93,10	93,20	93,14
Bozkir et al. (Bozkir et al., 2021)	SMO (RBF)	Bytes	–	94,80	94,70	94,70	94,65
Bozkir et al. (Bozkir et al., 2021)	SVM (Linear)	Bytes	–	92,80	92,90	92,80	92,91
Bozkir et al. (Bozkir et al., 2021)	XGBoost	Bytes	–	91,90	91,60	91,70	91,63
Proposed system without augmentation by using RGB image	CNN	Bytes	97,28	95,80	95,40	95,82	98,89
Proposed system with augmentation by using RGB image	CycleGAN+CNN	Bytes	**98,32**	**97,39**	**96,87**	**97,08**	**99,60**

mentation method achieved better results than the non-data augmentation method. By the way, the obtained results once again proved the contribution of the proposed data augmentation method.

6. Conclusion

In this study, CNN based new hybrid system was proposed for malware classification. First, byte files were transformed separately to gray and RGB image formats for the classification process. A new approach called B2IMG was developed for the transformation process. Moreover, a new CycleGAN-based data augmentation method was proposed in the study to address the problem of inconsistent data size between malware families.

The proposed system was tested on the BIG2015 database which was the most used in the malware classification. It was also tested on the newly developed DumpWare10 database. In order to prove the performance of the proposed data augmentation method, the experiments were carried out twice on the malware data 1) with data augmentation, 2) without data augmentation. The performance of the proposed system was proven by comparing the experimental results for both databases with the literature studies.

According to the results obtained in the experiments, it was seen that the classification performance increased thanks to the proposed data augmentation method. In addition, it was observed that the proposed system achieved better performance with RGB images than with gray images. The proposed method has increased the success of the malware classification in the BIG2015 database by approximately 2%. The proposed method achieved the best result in the literature with 97.16% Recall, 97.76% F1-Score, 98.52% Precision, 98.51% AUC, and 99.86% Accuracy in the BIG2015 database. In the Dumpware10 database, the classification performance improved by about 5% thanks to the proposed method. The proposed CycleGAN + CNN method achieved the best results with 96.87% Recall, 97.08% F1-Score, 97.39% Precision, 98.32% AUC, and 99.60% Accuracy on the database.

In future studies, it is aimed to improve the performance of malware classification by using both bytes and ASM data together. It is also planned to contribute to the literature by designing a hybrid system where Bytes and ASM files are used together.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Adem Tekerek: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Visualization, Project administration. **Muhammed Mutlu Yapici:** Conceptualization, Investigation, Resources, Writing – original draft, Supervision, Funding acquisition.

Acknowledgements

This work has been supported by the NVIDIA Corporation. All experimental studies were carried out on the TITAN XP graphics card donated by NVIDIA. We sincerely thank NVIDIA Corporation for their supports.

REFERENCES

Al Kadri M, Nassar M, Safa H. Transfer learning for malware multi-classification. In: Proceedings of the 23rd International Database Applications & Engineering Symposium; 2019. p. 1–7.

Alom MZ, et al. A state-of-the-art survey on deep learning theory and architectures. Electronics (Basel) 2019;8(3):292.

Bozkir AS, Tahillioglu E, Aydos M, Kara I. Catch them alive: a malware detection approach through memory forensics, manifold learning and computer vision. Comput. Secur. 2021;103.

Catak FO, Ahmed J, Sahinbas K, Khand ZH. Data augmentation based malware detection using convolutional neural networks. PeerJ Comput. Sci. 2021;7.

David OE, Netanyahu NS. Deepsign: deep learning for automatic malware signature generation and classification. In: 2015 International Joint Conference on Neural Networks (IJCNN); 2015. p. 1–8.

Drew J, Hahsler M, Moore T. Polymorphic malware detection using sequence classification methods and ensembles. EURASIP J. Inf. Secur. 2017(1):1–12 2017.

Egele M, Scholte T, Kirda E, Kruegel C. A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv. 2008;44(2):1–42.

Gao X, Hu C, Shan C, Liu B, Niu Z, Xie H. Malware classification for the cloud via semi-supervised transfer learning. J. Inf. Secur. Appl. 2020;55.

Ghanei H, Manavi F, Hamzeh A. A novel method for malware detection based on hardware events using deep neural networks. J. Comput. Virol. Hacking Tech. 2021:1–13.

Gibert D, Mateu C, Planes J. HYDRA: a multimodal deep learning framework for malware classification. Comput. Secur. 2020;95.

Goodfellow I, et al. Generative adversarial nets. In: Advances in Neural Information Processing Systems; 2014. p. 2672–80.

Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. CVPR 2017;1(2):3.

Huang X, Ma L, Yang W, Zhong Y. A method for windows malware detection based on deep learning. J. Signal Process. Syst. 2020:1–9.

Jain M, Andreopoulos W, Stamp M. Convolutional neural networks and extreme learning machines for malware classification. J. Comput. Virol. Hacking Tech. 2020;16(3):229–44.

Jang S, Li S, Sung Y. Fasttext-based local feature visualization algorithm for merged image-based malware classification framework for cyber security and cyber defense. Mathematics 2020;8(3):460.

Kalash M, Rochan M, Mohammed N, Bruce NDB, Wang Y, Iqbal F. Malware classification with deep convolutional neural networks. In: 2018 9th IFIP international conference on new technologies, mobility and security (NTMS); 2018. p. 1–5.

Kang J, Jang S, Li S, Jeong Y-S, Sung Y. Long short-term memory-based malware classification method for information security. Comput. Electr. Eng. 2019;77:366–75.

Kim J-Y, Bu S-J, Cho S-B. Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. Inf. Sci. (Ny). 2018;460:83–102.

LeCun Y, et al. Handwritten digit recognition with a back-propagation network. In: Advances in Neural Information Processing Systems; 1990. p. 396–404.

LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc. IEEE 1998;86(11):2278–324.

LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44 Google Sch. Google Sch. Cross Ref Cross Ref.

Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE. A survey of deep neural network architectures and their applications. Neurocomputing 2017;234:11–26.

Pan Z, Yu W, Yi X, Khan A, Yuan F, Zheng Y. Recent progress on generative adversarial networks (GANs): a survey. IEEE Access 2019;7:36322–33.

R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," arXiv Prepr. arXiv1802.10135, 2018.

Stiborek J, Pevn\'y T, Rehák M. Multiple instance learning for malware classification. Expert Syst. Appl. 2018;93:346–57.

Tang H, Xu D, Liu G, Wang W, Sebe N, Yan Y. Cycle in cycle generative adversarial networks for keypoint-guided image generation. In: Proceedings of the 27th ACM International Conference on Multimedia; 2019. p. 2052–60.

Tekerek A. A novel architecture for web-based attack detection using convolutional neural network. Comput. Secur. 2021;100.

Vasan D, Alazab M, Wassan S, Naeem H, Safaei B, Zheng Q. IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture. Comput. Networks 2020a;171.

Vasan D, Alazab M, Wassan S, Safaei B, Zheng Q. Image-Based malware classification using ensemble of CNN architectures (IMCEC). Comput. Secur. 2020b;92.

Wang C, Zhao Z, Wang F, Li Q. A novel malware detection and family classification scheme for IoT based on DEAM and densenet. Secur. Commun. Networks 2021 2021.

Wozniak M, Silka J, Wieczorek M, Alrashoud M. Recurrent neural network model for IoT and networking malware threads detection. IEEE Trans. Industr. Inform. 2020.

Yan J, Qi Y, Rao Q. Detecting Malware With an Ensemble Method Based On Deep Neural Network. Security and Communication Networks; 2018.

Yu Y, Adu K, Tashi N, Anokye P, Wang X, Ayidzoe MA. Rmaf: relu-memristor-like activation function for deep learning. IEEE Access 2020;8:72727–41.

Yuan B, Wang J, Liu D, Guo W, Wu P, Bao X. Byte-level malware classification based on markov images and deep learning. Comput. Secur. 2020;92.

Zhang J, Qin Z, Yin H, Ou L, Zhang K. A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding. Comput. Secur. 2019;84:376–92.

Adem TEKEREK: He is assistant professor at Gazi University, Computer Engineering Department, Technology Faculty. He graduated from Technical Education Faculty, Computer Systems Education Department in 2007. He graduated from MSc program of Informatics Institute in 2010. His-master thesis is about Institutional Content Management Systems. He graduated from PhD program of Informatics Institute in 2016. His-PhD thesis is about Web Application Firewall algorithms. He published 28 papers on computer sciences. He research is artificial ıntelligence, machine learning, deep learning, data mining and their applications especially on information security.

Muhammed Mutlu YAPICI: He is a Instructor at Ankara University. He graduated from Faculty of Technical Education, Electronics and Computer Education Department in 2010. As a second bachelor degree, he completed computer enginering program in 2016. He graduated from MSc. program of Informatics Institute in 2012. His-master thesis subject is Development of Course Scheduling Software Using Genetic Algorithm. He graduated from Faculty of Technology Computer Engineering Ph.D. in 2020. His-research interests are machine learning, deep learning, optimization techniques, genetic algorithm, image processing, and their applications on signature verification, face recognition, autonomus vehichles and medical image analysis.