Image Based Malware Classification with Multimodal Deep Learning

Mustafa Umut Demirezen

Artificial Intelligence and Technology Management Department, ROKETSAN Missiles Inc., Ankara, TURKEY
Corresponding Author: umut@demirezen.tech

Research Paper Received: 11.05.2021 Revised: 10.06.2021 Accepted: 10.06.2021

Abstract—Today, there are many different methods for analyzing and detecting malware. Some of these methods are basically based on statistical analysis, some on static and dynamic analysis methods, and some on machine learning methods. The studies carried out to classify malware with statistical machine learning-based analysis methods are generally based on complex and challenging feature extraction methods, and manual feature extraction is a very tedious process. However, the capability of deep learning methods to automatically extract complex features in a way simplifies this arduous process. In this study, a novel multimodal convolutional neural network-based deep learning architecture and singular value decomposition-based image feature extraction method are proposed to classify malware files using intermediate-level feature fusion. In addition to this, the performances of classical machine learning algorithms, neural networks, and the proposed multimodal convolutional neural networks-based deep learning algorithm are compared, and their performance is revealed. The performance of the proposed algorithm was also compared with the results of studies conducted with the same data set in the literature. The experimental results concluded that the proposed method is more successful than other methods or showed the same performance even though it did not use manual feature extraction techniques. It has been observed that with architecture, intermediate fusion approaches have the ability to obtain more specific features more effectively than other methods, thus improving performance values more than other methods.

Keywords—Malware, image, deep learning, singular value decomposition, multimodal

1. Introduction

Owing to the amount and variety of malicious software (malware) attacks, analysts have evolved techniques for automated malware identification and classification, rather than manually reviewing malware files in a time-consuming endeavor [1], [2]. Simultaneously, threat actors have created methods to circumvent signature-based identification techniques. Both analysts and malware writers have shown that malware detectors are, sadly, constrained in their capabilities and are easily managed to evade by convenient obfuscation methods [3]. Malware software and malicious technology not only cause significant expenses and damages, but can have a detrimental effect on the integrity of devices and networks. Malware writers, hackers, and computer technology professionals are constantly refining their tactics for defeating one another. Regrettably, there is no universal valid solution or suitable approach for detecting and removing malware. This condition is exacerbated even by undiscovered bugs in computer applications and internet resources [4].

Such flaws would stay undetected until they are completely exploited by malware creators, resulting in significant reputational, operational and financial damages.

The most efficient means of defending against malware is by the security software, which is focused chiefly on signature-based identification [5] to secure individual users. A signature is a sequence of function codes that serves as the sole means of identifying a piece of software or an application that can be differentiated from other application or software codes. Malware writers, on the other hand, can avoid this identification approach by variant creation strategies [6]. There are two primary categories of generation technologies: shared fundamental and obfuscation technology. The popular fundamental technology is that an attacker creates exploit code derivatives by recycling a basic unit. Obfuscation technology [7] was designed to supplement emerging protection and identification systems. It has become more commonly implemented and is classified into two groups based on the design philosophy. One is reverse engineering's intrusion uncertainty, which prohibits reverse engineering from obtaining the right study outcome. Another is command and control flow [8] inconsistency, which is often used to disguise malicious code's inner call logic by packaging, adding junk code, deleting instruction equivalents, or reallocating registers. The fundamental technology for identifying malicious variations is to isolate and reflect the malicious behavior's primary characteristics. So far, malware function manifestations have been mostly grouped into two types: static and dynamic [9]. The features are derived from malicious code depend on the static depiction by evaluating the Portable Executable data format [10], binary byte and disassembled codes and device call following structure. The static depictions, on the other hand, is often susceptible to obfuscation technology. Unlike static depictions, dynamic depictions are based on the concept of enclosing the target software or code in a controllable virtual environment such as a sandbox, and determining if it really is infected or malicious by observing the actions of the executing operation. For instance, identification can be accomplished by evaluating the series of Application Programming Interface calls or instruction streams in terms of certain activities. In comparison to static depictions, the dynamic depictions does not require complex reverse engineering techniques such as decryption and disassembly [11]. While dynamic identification is far more resistant to generic obfuscation, it is really a time and resource-intensive process, requiring significant processing time and disk capacity. Additionally, since the complex execution requirements are not always resolved, certain malicious functions and calls should be shown, impairing malicious code detection.

This article is organized into five sections. The first section provides a detailed literature review for research and methods on past malware classification and detection. In the second section, the definition of the specific problem aimed to be solved within the scope of this study, and the contributions of the proposed method for the solution are explained in detail. The multimodal singular value decomposition-based convolutional neural network method, which is basically based on the conversion of malware files to images, has been proposed as an innovative approach, and this method is explained in detail in the third section. In the fourth section, the results of the experimental studies performed on the data sets widely used in the literature are presented to examine the performance of the proposed method and to prove its accuracy. In the last section, in the light of the information obtained based on the experimental findings obtained as a result of this study, predictions are expressed in terms of increasing the performance of the suggested methodology and its use for solving various problems.

2. Machine Learning Methods for Malware Classification

According to the literature review carried out, the researches conducted in this field focus on the methods and approaches explained in the following titles:

2.1. Feature Based Machine Learning Methods

To fix the shortcomings of the classical approaches and in light of the possibility that variations of malware groups usually exhibit common behavioral behaviors [5], security authorities began developing more advanced classification methods focused on machine learning and data mining technologies [14]. These strategies use a variety of feature extracting approaches in order to create more smart malware security mechanisms. Numerous ML techniques are used to detect and identify malware into its forms and families, with the aim of excluding those who exhibit unusual behavior for comprehensive study. An early study [12] suggests a flexible paradigm in that they can distinguish infected files from normal files utilizing machine learning techniques. They used one-sided and kernelized one-sided neurons in the study to reduce false positives and differentiate between uninfected and files containing malware. Another research [13] suggested a framework in that they used machine learning algorithms to conduct data analysis, malware recognition, and new malware identification. The data processing is carried out using gray scale pictures, Opcode n-grams, and feature extraction. To find novel malicious software groups, the identification part of the method employs the shared nearest neighbor clustering. In a study, the usage of ML and data mining to identify and categorize malicious executables on the fly was reported [14]. They encoded malicious executables using n-grams, and then selected the most appropriate features for prediction; they tested a number of inductive techniques, including NB, DT, SVM, and boosting. Finally, they found that boosted decision trees outperformed other models in terms of a good classification metric as AUC. They also measured how well the methods categorized executables depending on the role of their payloads, with an AUC score of about 0.9 for detecting payload function.

The entropy based analysis was performed since files containing packed or encrypted code fragments usually have a greater entropy value than original code. Numerous studies have been conducted on entropy-based research. In [15], a method was proposed to autonomously calculate the degree that changes according to the file's systemic entropy raise suspicions. The value mentioned earlier was computed with the executable's structural entropy's wavelet-based energy spectrum and then fitted separate LR models over the different stages to generate a series of parameters that weigh the intensity of all resolution energy as a probability. In another study [16], Again, metamorphic malware was detected using entropy-based features. They used wavelet transform to identify regions with large differences in entropy values. They also used the Levenshtein distance to determine the resemblance between two data. As a result, provided an unfamiliar section of code, it could be categorized as the group related to the most identical example of the training set. A typical technique for reducing a file's entropy is to pad no-operation commands in the required sections in a file. Nonetheless, files that contain native, compressed, padded, or encrypted sections usually have different and special entropy values. Therefore, analysts began studying a file's anatomical entropy [17]. To identify malicious file, the intrinsic entropy of an undisclosed file was contrasted to the file in the training data.

Application Programming Interfaces implemented as function calls are widely recognized since highly distinguishing characteristics then they can be used as features in ML studies. The literature indicates that invocation of API functions may be used to model the actions of a program. API functions and device calls are mostly concerned on the services offered by operating systems. Since programs cannot access system resources in any other way but by API functions, the execution of specific API functions offers critical knowledge about malware activity. In a recent research [18] it was suggested a three-step classification system for Portable Executable (PE) files dependent on the API calls they use. They began by analyzing the PE files and obtaining a collection of acquired API calls. Second, they used the Clospan algorithm to reduce the function vector. Finally, a Random Forest algorithm was trained using the subset of features obtained. In another study [19], a rule-based method for malware categorization was suggested. The framework was composed of three modules: a PE translator, a rule builder for Objective-Oriented Associations (OOA), and a malware identification module. The executable's PE parser was charged with decoding it and retrieving the static operation requests for the relevant API functions. These requests were then utilized to create signatures for the PE data, which were then placed in a signature database. Then, using an OOA algorithm, class association rules were generated and saved in the rule repository. Finally, the malware identification unit was transferred the function calls and guidelines in order to decide if a file is benevolent or harmful.

2.2. Deep Learning Based Methods

Recently, deep learning has been applied to the classification of malware. A recent study reduced the dimensions of the input function vectors by utilizing a Deep Belief Network (DBN) architectures as to learn efficient data depictions in an unsupervised way (autoencoders) [20]. By training the DBN on unlabeled results, they outscored the K-NN, SVM, and DT algorithms in terms of classification accuracy. In [21], a DNN based malware detection architecture using static analysis results was proposed. In the study Byte/Entropy Histogram, PE Import, String 2D Histogram, PE Metadata Features were used to form a new feature vector. By using this feature vector they trained a deep learning model with Bayesian calibration. A novel multi-task, DL architecture for binary malware categorization was suggested by using feature extraction [22]. Their multi-task design combines both the binary and malware family classification loss functions. Additionally, they suggested a common malware family classification architecture in their study. Both models were trained using 4.5 million files gathered through complex file processing of malicious and benign files, and models were tested on a 2 million file holdout data. Their algorithms were reached an error rate of 0.358 percent, their regular (non-multitasking) model reached a 2.94 percent error rate. In a recent study [23], both convolutional and recurrent neural networks are used to acquire the features for classification problems. They obtained a hierarchical characteristic extraction structure using this approach, which blends n-gram convolution with complete sequential modeling. Their results showed that the proposed solution outperforms commonly used approaches for malware classification, with an overall precision value of 85.6% and recall value of 89.4% utilizing suggested hybrid neural network architecture.

2.3. Novel Malware File to Image Conversion Based Methods

In [1], pioneered an intriguing method for malware representation, in which malware's binary material is translated and rendered as a grayscale picture. This is accomplished by translating each byte into a pixel in a picture of values ranging from 0 to 255. Following that, the resultant collection is reorganized into a two-dimensional array. The image depiction of malware relating to a particular malicious software family is very close to that of malware connected to a related malicious software family. This visual resemblance is the outcome of malware code pieces reutilization when creating novel malware binaries. Thus, if previously implemented binaries are recycled to create new ones, the resultant binaries could become identical. In most instances, by depicting a malware file as a grayscale image, minor differences between malicious software codes belonging to the same family may be detected. In another study with the same approach [24], after the malware-to-image conversion, several hand-made features were derived dependent on intensity, Wavelet, and Gabor filters. By using these features, machine learning algorithms are applied for classification. On a specific dataset consisting of 12000 benign and 15000 malicious samples, the output of SVM as a machine learning model was evaluated using 70% for training and 30% for research.

Thanks to the approach of converting malware files to images, it has made it possible to use deep learning architectures such as convolutional neural networks in this field. The use of CNN-based algorithms in the solution of this problem is an increasingly popular approach. For example, in [25], They suggested a CNN-based classification architecture for malware samples. They transformed malware files to grayscale pictures and then equipped a CNN architecture to classify them. According to their experiments on two difficult-to-classify malware datasets, this suggested approach outperforms the SOTA methods. On the Malimg and MS datasets, the proposed approach reached an accuracy of 98.52 percent and 99.97 percent, individually. More novel approach has been proposed in [26]. They intended to extend deep networks to malware image detection in light of recent progress with Siamese NN for one-shot image recognition. They converted malware examples to rescaled monochrome images and categorized them according to their hash value within the same species. The siamese architecture was taught to rate association among examples during the training and testing periods. Their networks outperformed the baseline approaches in the trial. Additionally, their study showed that their architectures were more eligible for 1-shot learning of malware images than traditional DL models. In another research [27], They proposed a novel classifier, called IMCFN, that uses CNN-based deep learning to recognize variants of malicious software species and advance malware disclosure. Their approach is unique in that it proposes a novel strategy for multi-class classification problems using an adjusted CNN architecture for detecting and identifying malware classes. They contrasted IMCFN's efficiency to that of Inception Version III, Res-Net50, and VGG-16. They discovered that their approach would efficiently identify embedded code, obscured malware, and malware class modifications whilst requiring minimal work-time. The proposed approach was stated to be resistant to straightforward obfuscation techniques typically utilized by attackers to hide malicious code, such as packing and encryption; moreover, it shows that malware depicted in colored images outperformed grayscale malware images in terms of accuracy. Another study by the same authors [28], suggested a novel architecture focused on ensemble convolutional neural networks (IMCEC) for the successful identification of packed and unpacked malware. Their primary claim is that since various CNNs have different deeper architectures, they have distinct semantic depictions of the image; hence, a collection of CNN architectures enables the extraction of features with superior quality than conventional approaches. According to their result, using malware raw as an input, they accomplished a higher detection performance with a low false discovery rate with greater than 99 percent accuracy in detecting unpacked malwares and greater than 98 percent accuracy in detecting packaged malwares.

As a result of the literature review, the solution methods for the problem of malware detection and classification, which was examined with different perspectives, the approach of using classical machine learning algorithms by extracting manual attributes has been replaced by methods of classifying malicious software files using deep learning algorithms by converting them into images [29]. However, the detection and classification of malicious software and its variants that are encrypted or packaged with some methods and software is still a problem that needs to be dwelled on. In this study, a new image processing based methodology and method is proposed by focusing on this subject.

3. Image Based Malware Classification with Singular Value Decomposition Features

The grayscale picture depiction of malicious software has a number of disadvantages that are directly linked to the image generation process. Primarily, files are not two-dimensional images, and converting them into such introduces superfluous predictions. At the first step, an image width value must be chosen, which introduces a new hyper-parameter to tune. Take note that the image's height is determined by the binary's scale by specifying the dimension. Second, it introduces spatial correlations among pixels in distinct rows that do not occur. Additionally, it tends to suffer from code obfuscation methods, as most of the static malware features. Encryption and compression, in particular, will totally alter the inner layout of a binary code, causing methods relying on this depiction to struggle to identify it accurately. In Figure 1, example malware images are shown for two different malware families (Fakerean and Swizzor.gen!E) after conversion process was completed. As seen from the figure, After the malware files are converted to images, the malware belonging to the same family is very similar when they are represented as images, while the malware belonging to different families are depicted by different images and shows significant dissimilarity. Obviously, the images per each class exhibit significant variations that help us distinguish samples from one class from samples from the other malware family.

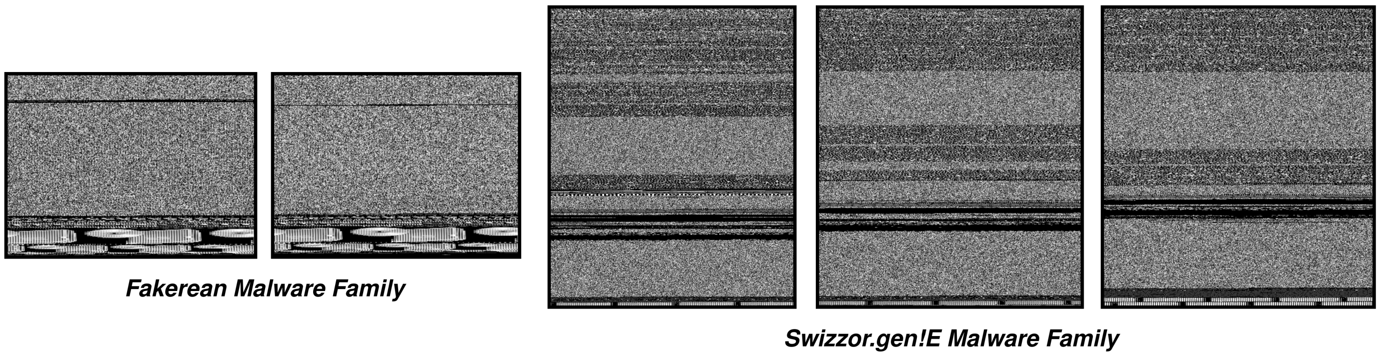

Fig. 1. Example Malware Images After Conversion From Binary Files [1].

To fight malware proliferation, it is critical to develop novel techniques for rapidly identifying and classifying malicious files so that their actions can indeed be evaluated. Machine learning-based malware classification approaches for detecting and classifying malware depend heavily on a single form of function or modality of data. However, malware identification is a multimodal research challenge, as it involves several data modalities. Multimodal learning is the study of how to process certain multimodal signs in combination. While mixing various modalities or types of knowledge to improve efficiency can sound attractive, integrating the differing amounts of noise and tension between modalities is extremely difficult.

Multimodal techniques may be classified into several categories based on the way the various modalities are integrated. Earlier fusion techniques combined the features derived from different modalities into a single depiction. Concatenating these function vectors produces a fused representation. Following that, a single model is equipped to discover the correlations and associations between the modality's characteristics. Through combining the intermediate features extracted from the different models, intermediate fusion approaches provide a common depiction. After concatenating such intermediate features, a machine learning model is equipped to catch relationships among modalities. Fusion at the decision-making level or at a later stage. In comparison to early fusion, late fusion approaches train a single algorithm for each module and combine the trained individual values using a fusion process like averaging, polling, or a final trained algorithm or model. The primary benefit of late fusion is that it enables the use of several templates on multiple modalities, thus increasing flexibility. Additionally, since projections are rendered independently for each modality, it is simpler to deal with incomplete modalities.

In this study, an intermediate fusion-based approach has been proposed as a multimodal method for detecting and classifying malware and this process is given in Figure 2. As the suggested method, malware files are converted to color and gray scale images. Three different matrices are obtained by applying the single value decomposition of the gray scale images. The previously produced color image and these three matrices are given as input to 4 different non-complex convolutional neural network models. The convolution layer outputs in the last layer of each convolutional network are combined on a single vector, and the intermediate layer fusion

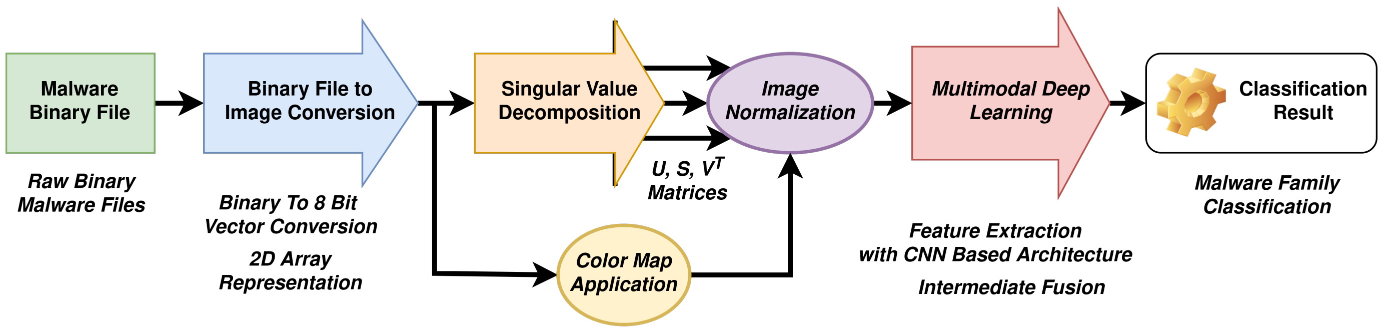

Fig. 2. Proposed Pipeline for Image Based Malware Classification with Matrix Decomposition and Multimodal Deep Learning.

process is performed. At this stage, the features obtained by convolutional neural networks are combined on a vector and a new feature vector will be obtained by combining the information produced by each model output. In the next step, this new fusion vector obtained is input to a classical multilayer perceptron neural network. As a result, with the proposed architecture, by obtaining 4 different representations of a single malware image data, it will be provided to train a neural network type classifier with the new information obtained by fusing these new representations.

3.1. Malware to Image conversion

Malware versions that are members of the same family usually share a typical texture. To benefit from this idea, the binary malware file must first be converted to an image to obtain image-based functionality from a malware sample. This idea is a basic but efficient method for visualizing raw malware binary files into a color picture to represent the problem in 2-dimensional space. Also, This approach is devoid of the need for feature engineering or domain specialist expertise and provides the usage of computer vision algorithms specific to this problem. In order to convert binary malware files to images, the binary file of a specified malware is first read into a vector of 8-bit unsigned integers. Following that, each part of this vector's binary value is translated to its decimal counterpart, which is then stored in a new numerical vector symbolic of the malware sample. Finally, the resulting decimal vector is transformed into a two-dimensional matrix and rendered as a grayscale image. The spatial resolution of the image is the width and height of the 2D matrix as a grayscale image, and their values are determined primarily by the size of the malware binary file. These decimal vectors are resized using the spatial resolution according to idea from [1] and given in Table 3.1. Finally, A color map to obtain a colored version of the images is applied to these two-dimensional sequences. All necessary steps are shown in Figure 3.

3.2. Singular Value Decomposition

Singular Value Decomposition [30] is a well known and generic linear algebra method that is used to analyze matrices. A matrix Z of size n × n can be represented by three special matrices as defined in Equation (1). It Is a sophisticated linear algebra procedure that yields a foundation for the matrix's row and column spaces as well as an indicator of the matrix's rank.

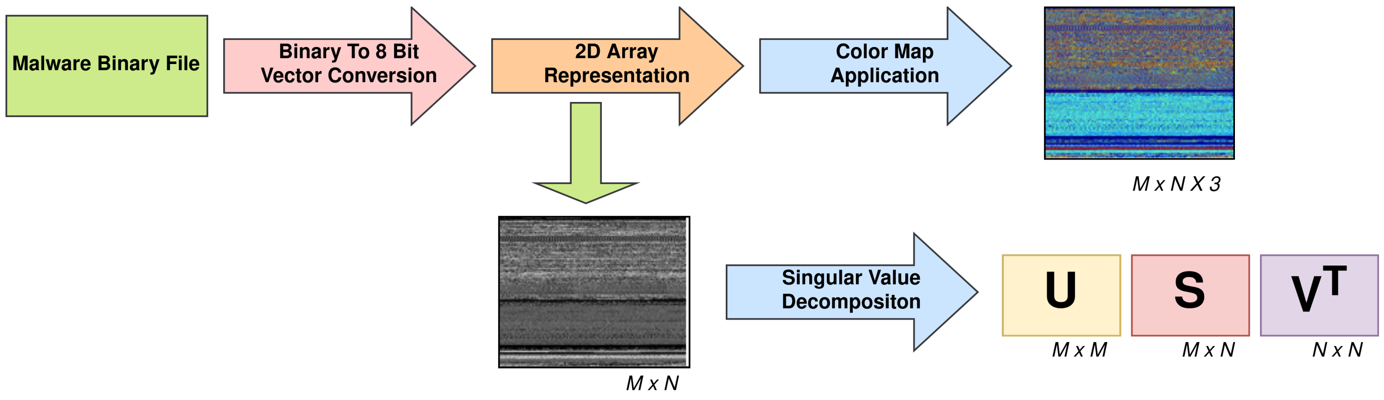

Fig. 3. Malware to Image Conversion and SVD Feature Extraction.

TABLE 1
File Size vs. Image Width for Malware to Image Conversion [1].

File Size (Kilo Bytes)	Image Width (Pixels)
0 kBytes <F.S. <10 kBytes	32
10 kBytes <F.S. <30 kBytes	64
30 kBytes <F.S. <60 kBytes	128
60 kBytes <F.S. <100 kBytes	256
100 kBytes <F.S. <200 kBytes	384
200 kBytes <F.S. <500 kBytes	512
500 kBytes <F.S. <1000 kBytes	768
F.S. >1000 kBytes	1024

$$Z_{m\times n} = U_{m\times m} \times S_{m\times n} \times V^T_{n\times n} \tag{1}$$

Where U is an m × m orthogonal matrix, V is an n × n orthogonal matrix, and S is an m × n matrix with the diagonal elements representing the singular values, $\sigma_1 > \sigma_2 > \cdots > \sigma_n$, of Z. The matrix S can be represented in Equation (2).

In Equation (2), The columns of U and V matrices are named the left singular vectors, and right singular vectors respectively. The left singular vectors of matrix Z are defined as the eigen vectors of $Z.Z^T$, and the right singular vectors are named as the eigen vectors of $Z^T.Z$.

Singular Value Decomposition may be used for a variety of purposes, including watermarking images, calculating weighted least squares, and optimizing prediction. Each singular value corresponds to an image layer's light intensity, whereas the related pair of singular vectors corresponds to the image's geometry. U and V are unitary orthogonal matrices, whereas S is a diagonal matrix of decreasing singular values. Each eigen-image's singular value is literally its L_2-norm. Due to the fact that SVD gives the maximum the highest singular values, the first eigen-image is the template that explains the highest level of variance-covariance form.

$$\begin{pmatrix} z_{11} & & z_{1n} \\ & \ddots & \\ z_{m1} & & z_{mn} \end{pmatrix} = \begin{pmatrix} u_{11} & & u_{m1} \\ & \ddots & \\ u_{1m} & & u_{mm} \end{pmatrix} \begin{pmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_r & \\ 0 & & & \ddots \end{pmatrix} \begin{pmatrix} v_{11} & & v_{1n} \\ & \ddots & \\ v_{n1} & & v_{nn} \end{pmatrix} \tag{2}$$

By evaluating the biggest singular values that constitute the majority of the image's resources, SVD may provide low-rank estimations that may be ideal sub-rank estimates [31]. SVD demonstrates how to view a matrix as a sum of lower rank matrices especially rank-one. A matrix Z can be generated approximately by a shortened matrix $Z_k$of unique rank k and can be seen in Equation (3). Utilizing SVD for matrix approximation has a range of functional benefits, including the ability to store the approximation Z_k of a matrix rather than the entire matrix Z, which is the case in image compression and, more lately, image watermarking frameworks.

$$Z = \sum_{i=1}^{t} s_i u_i v_i^{\mathrm{T}} \approx s_1 u_1 v_1^T + s_2 u_2 v_2^T + \ldots . + s_t u_t v_t^T \tag{3}$$

In Equation (3), Z equals to summation of the product of t rank-1 matrices. The fractional summation extracts as much energy of Z as a matrix with at maximum rank-r can. The Frobenius (L_2) norm is used to describe the level of energy in this situation. Each outer product ($u_i.v_i^T$) is a basic rank one matrix that can be held in m+n numbers (m and n are the number of the rows and columns of the matrix Z respectively), as opposed to the m×n of the initial matrix. The memory contains (m+n+1)×t for truncated SVD transformations with rank-t elements [31].

After the malware transformation process is completed, the 2-dimensional matrix obtained is parsed into U, S and V^T matrices by the singular value decomposition method. The U and V matrices show the matrices consisting of orthonormal vectors belonging to the image representing the malware, while the S matrix shows the eigenvalues for the same image. As a result of this process, the most distinctive features on the image can be revealed. The sizes of the 3 matrices obtained may differ. For this reason, these three matrices are also subjected to the normalization process.

3.3. Image Normalization

Normalization is a pre-processing phase employed to rescale the input data in regards to the CNN configuration to make inputs similar scale and statistical properties. Malicious software images are yielded from binary malware files, and are not pre-configured in terms of scale. To address this problem, malware images were resized and set the size of 224×224 pixels. The primary advantage of the normalization method was that it reduced the scale of the input images, which was beneficial for CNN training. Additionally, certain critical characteristics were missing during the dimensionality reduction phase. As a result, the significant number of malware images in the dataset retained their texture characteristics throughout the normalization phase.

Since the U, S and V^T matrices obtained by the singular value decomposition method will be used as an input in the convolutional neural network architecture in the next step, the normalization process is applied to these matrices as well. As with the approach applied to previous malware images, these matrices are resized to 224×224 dimensions.

3.4. Convolutional Neural Networks

A Convolutional Neural Network (CNN) [32], is a form of Deep Learning model that can accept an image as input, allocate significance by using learnable parameters as weights and biases to various objects in the image, and distinguish between them. CNNs need far less pre-processing than other classification algorithms. Although basic techniques include hand-engineering of filters, CNNs may acquire these filters with sufficient preparation. A CNN's configuration is similar to the connection structure of Neurons in the Human Brain and was influenced by the Visual Cortex's configuration. Neural circuits react to a stimulus only within a small area of the field of vision referred to as the Receptive Field. A set of such fields will converge to fill the visual space fully. A high level architecture diagram of CNN is shown in Figure 4.

Many layers comprise a convolutional neural network. These layers can take on a variety of forms, but the most fundamental and popular are as follows:

Convolutional layers (CL) are made up of a grid of neurons in lattice shape. It is therefore essential for the preceding layer to be a grid of neurons in similar shape. Every neuron accepts inputs from lattice parts of the preceding part of the architecture; the trainable weight parameters of this lattice section are equal for all neurons that existed in the CL. Therefore, the CL is simply the preceding layer's 2 dimensional convolution, in which the weights define a convolution filter. Additionally, each convolutional layer can include several meshes; each mesh takes inputs from every other nodes in the former layer, utilizing theoretically distinctive filters.

Following each CL, a pooling layer (PL) can be added. The PL subsamples tiny square chunks from the CL to generate an individual contribution from each chunk. This pooling can be accomplished in a variety of forms, either by taking the average or limit of the neurons in the row, or by learning a linear mixture of the units in the lattice. PLs might all be maximum-pooling operations; that is, they will always calculate the maximum amount of the pooled sections. But other type of pooling operations can also be used such as average pooling, and global pooling.

Eventually, following to the many convolutional and max pooling layers, the neural network's high-grade inference is carried out using fully-connected layers. A fully-connected layer executes a flattening operation, and binds all neurons in the preceding layer to each and every neuron in the subsequent layer in the architecture. Due to the fact that fully connected layers are no longer spatially positioned, behind a fully connected layer, there may be no convolutional layers. They can be visualized as one-dimensional after a flattening operation.

With the implementation of suitable filters, a CNN is capable of effectively capturing the spatial and temporal relations in a picture. Owing to the reduced set of elements and reusability of weights, the model achieves a closer match to the image or tensor type of data. In other terms, the network may be programmed to recognize the image's sophistication.

Assume that a layer of N×N size neurons preceded by a layer of convolutional neurons. If we select to use an m×m size filter ω, the convolutional layer output will be of size

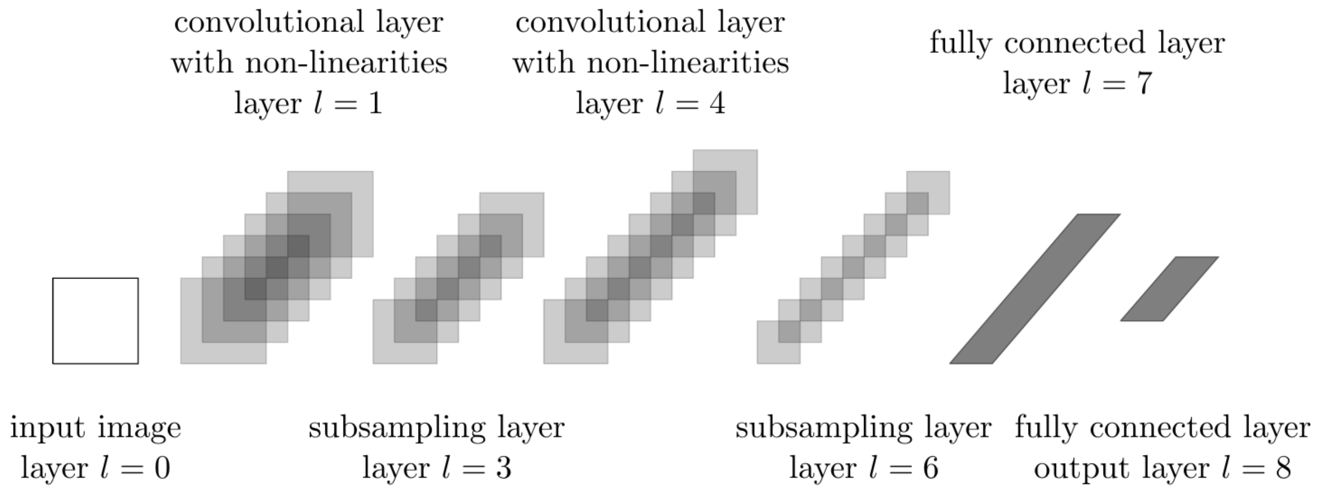
convolutional layer with non-linearities layer $l = 1$
convolutional layer with non-linearities layer $l = 4$
fully connected layer layer $l = 7$
input image layer $l = 0$
subsampling layer layer $l = 3$
subsampling layer layer $l = 6$
fully connected layer output layer $l = 8$

Fig. 4. Convolutional Neural Network High Level Diagram [33].

(N−m+1)×(N−m+1). In order to calculate the pre-activation function input to some unit $x^{\ell}{}_{ij}$ in the layer, It is required to add up all the individual contributions as weighted by the filter components from the previous layer cells. This operation is given in Equation (4).

$$o_{ij}^{\ell} = \sum_{x=0}^{m-1} \sum_{y=0}^{m-1} \omega_{xy} f_{(i+x)(j+y)}^{\ell-1} \tag{4}$$

After this operation is conducted for a layer, a nonlinear activation function is applied to the output of the each neuron in the layers. This operation is given in Equation (5).

$$y_{ij}^{\ell} = \sigma\left(o_{ij}^{\ell}\right) \tag{5}$$

In Equation (5), σ is an activation function such as ReLU, hyperbolic tangent, Sigmoid or similar type of nonlinearities for neurons. The max-pooling layers do not perform any learning operation. Other than that, then, reduce the problem's scale by adding sparsity. During the training, at the forward propagation step, it reduces k×k blocks to a single value. Then, at the backpropagation stage, from the proceeding layers, this single value receives an error and gradient. This error value is then simply redirected to the location from which it originated. Due to the fact that it originated from a single location in the k×k block, the gradients obtained by backpropagation operation from max-pooling layers are very sparse.

In this study, CNNs are used to provide feature extraction for image data. However, by applying an architecture different from the standard convolutional neural networks architectures, it has been used to automatically extract and fuse different features on multiple images.

3.5. Multimodal Convolutional Neural Network with Intermediate Fusion

The paper propose an intermediate fusion-based solution as a multimodal system for detecting and classifying malware, and this architecture is given in Figure 5. Malware files are translated to color and grayscale photos using the recommended process. Three distinct matrices are obtained by decomposing the gray scale representations into single values. The previously produced color image and these three matrices are fed into four separate uncomplicated CNN models. The outputs of each convolutional network's final layer are merged into a single vector, and the intermediate layer fusion step is done. At this point, the CNN features are merged into a vector, and a new feature vector is created by integrating the information provided by each model's performance. Following that, this newly created fusion vector is fed into a classical multilayer perceptron neural network. As a consequence, utilizing the proposed architecture, it would be possible to train a neural network style classifier using the new knowledge gained from fusing these new representations.

In this study, 2 different types of convolutional neural networks are used for 4 different images. The main reason for this is that different types of features are obtained by CNNs by making use of the proposed multimodal architecture and then fused. For this purpose, VGG16 [34] architecture was used for the image obtained by converting the malware to image and normalizing it. This architecture was chosen because it is simpler compared to the others, the selected CNN architecture is less prone to overfitting problems due to the limited number of malware samples in the data set, and also because it can successfully extract basic image attributes on the malware image. On the other hand, determining the textures on the image, especially performing this analysis on the SVD matrices, will allow extracting important features in the classification of malware. On the other hand, obtaining texture based features in multiple resolution will provide important information in solving the classification problem. Therefore, for each matrix obtained after SVD process, architectures are used in Pyramid CNN [35], [36], [37] structure. In this way, both vision-based and texture-based features on the image data will be obtained at the same time and will be combined in the last layer of CNN architectures and subjected to fusion process. The biggest advantage of this proposed method is that it enables to obtain more effective and distinctive features by using both different solutions and classical image features together. These advantages provide the usage of multi-resolution information and both global and local features together.

4. Experimental Results

Malimg malware dataset [1], which includes 9339 malware representations as images from 25 distinct malware groups, is used for this study. Malimg is in an image-based format, and therefore the malware samples do not need any pre-processing operations prior to performing image-related analyses; furthermore, binary files related to Malimg images really are not easy to access. As expected, the dataset's groups are highly unbalanced: the largest Allaple.A family includes 2949 images, whereas the smallest Allaple.B family includes just 80. For the creation of training and test sets, the stratified sampling method [38] was used, with 70% training set and 30% test set of all data. The proposed architecture in Figure 5 has been trained using the ADAM optimizer [39] using the cross entropy loss function given in Equation (6). In Equation (6), $y_{o,c}$ shows the grand truth labels and $p_{o,c}$ is the output probability value of the CNN. By using this loss function, backpropagation algorithm was used to update the whole trainable parameters for the architecture. For the training process,

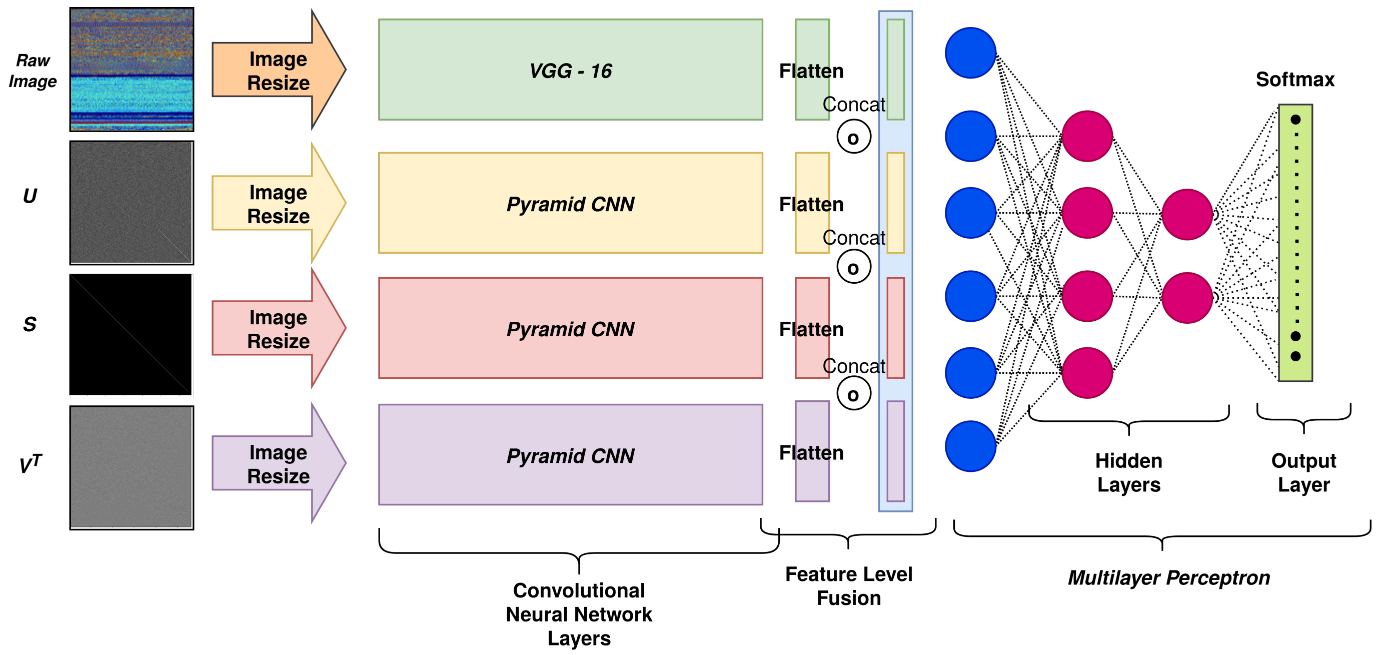

Fig. 5. Multimodal Deep Learning Architecture.

the learning rate 1e-4 was selected and the architecture 1000 epochs were trained with 64 batch size. To prevent the model from being overfitted, the early-stopping method was applied and a value of 0.5 was used in the dropout layers. ReLU activation function [40], and He initialization method [41] were used for VGG-16 and Pyramid CNN models included in the proposed architecture.

$$Loss = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{6}$$

The performance of the proposed architecture has been measured with the accuracy, precision, recall and F_1 measure given in Equations (7), (8), (9), (10) and (11). In these equations, TP, TF, FP, FN are true positive, true negative, false positive, false negative values of the model output respectively.

$$Accuracy = \frac{TP + TF}{TP + TF + FP + FN} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{10}$$

$$F_1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{11}$$

The model was trained with the previously mentioned parameter values, and then its performance was tested on a test data set that it had never seen before. The results showing the success of the proposed method with similar studies using deep learning and classical methods on the same data set in the literature are given in Table 4.

TABLE 2
Overall Results with Comparative Studies.

Methods	Accuracy	Precision	Recall	F_1 Measure
[42] ResNet + Softmax	98.62	-	-	-
[43] Google + Softmax	98.00	-	-	-
[44] Deep Learning Based	98.80	-	-	-
[45] CNN + Mul. Objective	97.60	88.40	-	-
[46] Light-weight DL	94.00	-	-	-
[28] IMCEC	99.50	99.50	99.46	99.48
[1] Nataraj et al. (Gist-KNN)	97.18	-	-	-
[25] M-CNN	98.52	-	-	-
[47] CS. CNN-2 Layer-LSTM	95.50	95.50	95.50	95.50
[27] IMCFN	98.82	98.85	98.81	98.75
[48] AlexNet, Inception v3	99.30	-	-	-
Prosed Model (MM-DL)	99.72	99.72	99.60	99.66

It is seen from Table 4 that proposed method has the best scores for each performance metric. It has been observed that the proposed architecture provides slightly better performance than the IMCEC model obtained by combining different deep learning architectures, and the proposed method uses simpler architectures for this. Although it is seen that the performance of classical deep learning architectures is high when compared to other methods, it is understood that the proposed method yields better results than classical methods and deep learning architectures used in the literature, and classifies packaged malware especially better. Based on the experimental results, it can be deduced that the main reason of this result is the benefit of teaching texture-based features to the algorithm with the intermediate fusion approach. Finally, when the experimental results are examined, it is observed that the more complexity of the architectures and methods used, the better results are obtained, and the performance does not improve when hybrid architectures [47] (such as CNN + LSTM) are used for classification.

5. Conclusion

In this study, a method is proposed to classify binary files with artificial intelligence algorithms by converting binary files to images for the classification of malware. The proposed method allows obtaining more important and silent features from malware images at different resolutions and using texture information using intermediate fusion without using any data approach. Two different types of deep learning architecture were used within the proposed approach. In this method, in addition to the features that architecture can obtain with the deep learning architectures used in the classical computer vision field, it is suggested that the other architecture focuses on texture information, extracting other valuable features from it, and further improving the classification performance with the fusion of all this information obtained. The experimental results revealed that the deep learning approach is more successful than classical machine learning, deep learning architectures widely used in the field of computer vision, and other hybrid methods proposed for the classification problem. However, as in the proposed method, it has been observed that with architecture, intermediate fusion approaches have the ability to obtain more specific features more effectively than other methods, thus improving performance values more than other methods. Data augmentation was not used in any way within the scope of the proposed method [49]. As a result, the experimental findings were obtained to support the conclusion that approaches are more successful than others, as the classification studies conducted on the same in the literature.

Acknowledgments

This research has been supported by ROKETSAN Missiles Inc., System Technologies Development Directorate, Artificial Intelligence and Technology Management Unit.

References

[1] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware Images: Visualization and Automatic Classification,” in Proceedings of the 8th International Symposium on Visualization for Cyber Security. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: 10.1145/2016904.2016908

[2] J. Singh and J. Singh, “A survey on machine learning-based malware detection in executable files,” Journal of Systems Architecture, vol. 112, p. 101861, 2021. [Online]. Available: https://doi.org/10.1016/j.sysarc.2020.101861

[3] D. Gibert, C. Mateu, and J. Planes, “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges,” Journal of Network and Computer Applications, vol. 153, p. 102526, 2020. [Online]. Available: https://doi.org/10.1016/j.jnca.2019.102526

[4] H. V. Nath and B. M. Mehtre, “Static Malware Analysis Using Machine Learning Methods,” in Recent Trends in Computer Networks and Distributed Systems Security, G. M. Pérez, S. M. Thampi, R. Ko, and L. Shu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 440–450.

[5] A. Ojugo and A. Eboka, “Signature-Based Malware Detection Using Approximate Boyer Moore String Matching Algorithm,” International Journal of Mathematical Sciences and Computing, vol. 5, pp. 49–62, 2019.

[6] P. Harshalatha and R. Mohanasundaram, “Classification Of Malware Detection Using Machine Learning Algorithms: A Survey,” International Journal of Scientific & Technology Research, vol. 9, pp. 1796–1802, 2020.

[7] R. Kumar and A. R. E. Vaishakh, “Detection of Obfuscation in Java Malware,” Procedia Computer Science, vol. 78, pp. 521–529, 2016. [Online]. Available: https://doi.org/10.1016/j.procs.2016.02.097

[8] Y. Ding, W. Dai, S. Yan, and Y. Zhang, “Control flow-based opcode behavior analysis for Malware detection,” Computers & Security, vol. 44, pp. 65–74, 2014. [Online]. Available: https://doi.org/10.1016/j.cose.2014.04.003

[9] A. Damodaran, F. D. Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, “A comparison of static, dynamic, and hybrid analysis for malware detection,” Journal of Computer Virology and Hacking Techniques, vol. 13, no. 1, pp. 1–12, Feb 2017. [Online]. Available: 10.1007/s11416-015-0261-z

[10] A. Kumar, K. S. Kuppusamy, and G. Aghila, “A learning model to detect maliciousness of portable executable using integrated feature set,” Journal of King Saud University - Computer and Information Sciences, vol. 31, no. 2, pp. 252–265, 2019. [Online]. Available: https://doi.org/10.1016/j.jksuci.2017.01.003

[11] B. Anderson, C. Storlie, M. Yates, and A. McPhall, “Automating Reverse Engineering with Machine Learning Techniques,” in Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop. New York, NY, USA: Association for Computing Machinery, 2014, pp. 103–112. [Online]. Available: 10.1145/2666652.2666665

[12] D. Gavriluţ, M. Cimpoeşu, D. Anton, and L. Ciortuz, “Malware detection using machine learning,” in 2009 International Multiconference on Computer Science and Information Technology, 2009, pp. 735–741. [Online]. Available: 10.1109/IMCSIT.2009.5352759

[13] L. Liu, B. sheng Wang, B. Yu, and Q. xi Zhong, “Automatic malware classification and new malware

detection using machine learning," Frontiers of Information Technology {&amp;} Electronic Engineering, vol. 18, no. 9, pp. 1336–1347, Sep 2017. [Online]. Available: 10.1631/FITEE.1601325

[14] J. Z. Kolter and M. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild," Journal of Machine Learning Research, vol. 7, pp. 2721–2744, 2006.

[15] M. Wojnowicz, G. Chisholm, M. Wolff, and X. Zhao, "Wavelet decomposition of software entropy reveals symptoms of malicious code," Journal of Innovation in Digital Ecosystems, vol. 3, no. 2, pp. 130–140, 2016. [Online]. Available: https://doi.org/10.1016/j.jides.2016.10.009

[16] D. Baysa, R. M. Low, and M. Stamp, "Structural entropy and metamorphic malware," Journal of Computer Virology and Hacking Techniques, vol. 9, no. 4, pp. 179–192, Nov 2013. [Online]. Available: 10.1007/s11416-013-0185-4

[17] I. Sorokin, "Comparing files using structural entropy," Journal in Computer Virology, vol. 7, no. 4, p. 259, Jun 2011. [Online]. Available: 10.1007/s11416-011-0153-9

[18] A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, and A. Hamze, "Malware Detection Based on Mining API Calls," in Proceedings of the 2010 ACM Symposium on Applied Computing. New York, NY, USA: Association for Computing Machinery, 2010, pp. 1020–1025. [Online]. Available: 10.1145/1774088.1774303

[19] Y. Ye, D. Wang, T. Li, D. Ye, and Q. Jiang, "An intelligent PE-malware detection system based on association mining," Journal in Computer Virology, vol. 4, no. 4, pp. 323–334, Nov 2008. [Online]. Available: 10.1007/s11416-008-0082-4

[20] D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," Neural Computing and Applications, vol. 31, no. 2, pp. 461–472, Feb 2019. [Online]. Available: 10.1007/s00521-017-3077-6

[21] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in 2015 10th International Conference on Malicious and Unwanted Software (MALWARE), 2015, pp. 11–20. [Online]. Available: 10.1109/MALWARE.2015.7413680

[22] W. Huang and J. W. Stokes, "MtNet: A Multi-Task Neural Network for Dynamic Malware Classification," in Detection of Intrusions and Malware, and Vulnerability Assessment, J. Caballero, U. Zurutuza, and R. J. Rodríguez, Eds. Cham: Springer International Publishing, 2016, pp. 399–418.

[23] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep Learning for Classification of Malware System Call Sequences," in AI 2016: Advances in Artificial Intelligence, B. H. Kang and Q. Bai, Eds. Cham: Springer International Publishing, 2016, pp. 137–149.

[24] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," in 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), 2013, pp. 40–44. [Online]. Available: 10.1109/CICYBS.2013.6597204

[25] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks," in 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2018, pp. 1–5. [Online]. Available: 10.1109/NTMS.2018.8328749

[26] S.-C. Hsiao, D.-Y. Kao, Z.-Y. Liu, and R. Tso, "Malware Image Classification Using One-Shot Learning with Siamese Networks," Procedia Computer Science, vol. 159, pp. 1863–1871, 2019. [Online]. Available: https://doi.org/10.1016/j.procs.2019.09.358

[27] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," Computer Networks, vol. 171, p. 107138, 2020. [Online]. Available: https://doi.org/10.1016/j.comnet.2020.107138

[28] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-Based malware classification using ensemble of CNN architectures (IMCEC)," Computers & Security, vol. 92, p. 101748, 2020. [Online]. Available: https://doi.org/10.1016/j.cose.2020.101748

[29] B. Yadav and S. Tokekar, "Recent Innovations and Comparison of DeepLearning Techniques in Malware Classification : A Review," International Journal of Information Security Science, vol. 9, no. 4, pp. 230–247, 2020.

[30] J. J. M. Cuppen, "The singular value decomposition in product form," SIAM Journal on Scientific and Statistical Computing, vol. 4, no. 2, pp. 216–222, 1983.

[31] R. A. Sadek, "SVD Based Image Processing Applications: State of The Art, Contributions and Research Challenges," International Journal of Advanced Computer Science and Applications, vol. 3, no. 7, 2012. [Online]. Available: 10.14569/IJACSA.2012.030703

[32] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Computation, vol. 1, pp. 541–551, 1989.

[33] D. Stutz, "Illustrating (Convolutional) Neural Networks in LaTeX with TikZ," 06 2020. [Online]. Available: https://davidstutz.de/illustrating-convolutional-neural-networks-in-latex-with-tikz/

[34] S. Tammina, "Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images," International Journal of Scientific and Research Publications (IJSRP), vol. 9, no. 10, pp. 9420–9420, 2019. [Online]. Available: https://dx.doi.org/10.29322/ijsrp.9.10.2019.p9420

[35] H. Liu, S.-I. Kamata, and Y. Li, “Hybrid Featured based Pyramid Structured CNN for Texture Classification,” in 2019 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), 2019, pp. 170–175. [Online]. Available: 10.1109/ICSIPA45851.2019.8977773

[36] P. Burt and E. Adelson, “The Laplacian Pyramid as a Compact Image Code,” IEEE Transactions on Communications, vol. 31, no. 4, pp. 532–540, 1983.

[37] E. Adelson, P. Burt, C. Anderson, J. M. Ogden, and J. Bergen, “Pyramid methods in image processing,” RCA engineer, vol. 29, no. 6, pp. 33–41, 1984.

[38] F. J. Gallego, “Stratified sampling of satellite images with a systematic grid of points,” ISPRS Journal of Photogrammetry and Remote Sensing, vol. 59, no. 6, pp. 369–376, 2005.

[39] H. Zhong, Z. Chen, C. Qin, Z. Huang, V. W. Zheng, T. Xu, and E. Chen, “Adam revisited: a weighted past gradients perspective,” Frontiers of Computer Science, vol. 14, pp. 145 309–145 309, 2020.

[40] H. Ide and T. Kurita, “Improvement of learning for CNN with ReLU activation by sparse regularization,” in 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2684–2691.

[41] Y.-M. Kwon, Y. woo Kwon, D.-K. Chung, and M.-J. Lim, “The Comparison of Performance According to Initialization Methods of Deep Neural Network for Malware Dataset,” International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 4S2, pp. 57–62, 2019.

[42] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. de Geus, “Malicious Software Classification Using Transfer Learning of ResNet-50 Deep Neural Network,” in 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017, pp. 1011–1014.

[43] R. U. Khan, X. Zhang, and R. Kumar, “Analysis of ResNet and GoogleNet models for malware detection,” Journal of Computer Virology and Hacking Techniques, vol. 15, pp. 29–37, 2018.

[44] N. Bhodia, P. Prajapati, F. D. Troia, and M. Stamp, “Transfer Learning for Image-Based Malware Classification,” in ICISSP, 2019.

[45] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, “Malicious code detection based on CNNs and multi-objective algorithm,” Journal of Parallel and Distributed Computing, vol. 129, pp. 50–58, 2019.

[46] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, “Lightweight Classification of IoT Malware Based on Image Recognition,” in 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol. 02, 2018, pp. 664–669.

[47] S. Akarsh, K. Simran, P. Poornachandran, V. K. Menon, and K. P. Soman, “Deep Learning Framework and Visualization for Malware Classification,” in 2019 5th International Conference on Advanced Computing Communication Systems (ICACCS), 2019, pp. 1059–1063.

[48] M. Nisa, J. H. Shah, S. Kanwal, M. Raza, M. A. Khan, R. Damaševičius, and T. Blažauskas, “Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features,” Applied Sciences, vol. 10, no. 14, 2020.

[49] F. O. Catak, J. Ahmed, K. Sahinbas, and Z. H. Khand, “Data augmentation based malware detection using convolutional neural networks,” PeerJ. Computer science, vol. 7, pp. 346–346, 2021. [Online]. Available: 10.7717/peerj-cs.346