I. J. Computer Network and Information Security, **2020, 6, 30-43**
Published Online December 2020 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2020.06.03

Malware Classification with Improved Convolutional Neural Network Model

Sumit S. Lad
Dept. of CSE, Rajarambapu Institute of Technology, Rajaramnagar, Sangli, Maharashtra, India
E-mail: sumitx86@gmail.com

Amol C. Adamuthe
Dept. of CS&IT, Rajarambapu Institute of Technology, Rajaramnagar, Sangli, Maharashtra, India
E-mail: amol.admuthe@gmail.com

Received: 01 June 2020; Accepted: 24 June 2020; Published: 08 December 2020

Abstract: Malware is a threat to people in the cyber world. It steals personal information and harms computer systems. Various developers and information security specialists around the globe continuously work on strategies for detecting malware. From the last few years, machine learning has been investigated by many researchers for malware classification. The existing solutions require more computing resources and are not efficient for datasets with large numbers of samples. Using existing feature extractors for extracting features of images consumes more resources. This paper presents a Convolutional Neural Network model with pre-processing and augmentation techniques for the classification of malware gray-scale images. An investigation is conducted on the Malimg dataset, which contains 9339 gray-scale images. The dataset created from binaries of malware belongs to 25 different families. To create a precise approach and considering the success of deep learning techniques for the classification of raising the volume of newly created malware, we proposed CNN and Hybrid CNN+SVM model. The CNN is used as an automatic feature extractor that uses less resource and time as compared to the existing methods. Proposed CNN model shows (98.03%) accuracy which is better than other existing CNN models namely VGG16 (96.96%), ResNet50 (97.11%) InceptionV3 (97.22%), Xception (97.56%). The execution time of the proposed CNN model is significantly reduced than other existing CNN models. The proposed CNN model is hybridized with a support vector machine. Instead of using Softmax as activation function, SVM performs the task of classifying the malware based on features extracted by the CNN model. The proposed fine-tuned model of CNN produces a well-selected features vector of 256 Neurons with the FC layer, which is input to SVM. Linear SVC kernel transforms the binary SVM classifier into multi-class SVM, which classifies the malware samples using the one-against-one method and delivers the accuracy of 99.59%.

Index Terms: Malware classification, image classification, convolutional neural network, support vector machine.

1. Introduction

Malicious code is a code that is a part of the software, system, or scripts, causing harm to the system. In the last few years, growth in Internet usage has grown significantly. With more Internet usage, there is a rise in criminal activities and hacking code injections. For doing such activities, criminals are using malicious codes to perform illegal activities on devices connected to the Internet. Creating malicious codes with online available automated tools is easy for attackers. According to Symantec's report of the year 2019, every month, on average, attackers inject malicious codes into 4800+ retailer's websites to get credit card details [1]. Malwarebyte's research [2] reported that malware attacks on Windows operating systems increased by 18% in the year 2019.

Malicious codes are divided into several classes considering their working, namely virus, worms, logic bombs, trojans, keyloggers, and backdoors. These classes again further divided into different families of malware. FireEye [3] stated that in 2019, they discovered 500 new families of malicious code. With fast technological progress, malicious code analysis, detection, and classification malware became more critical from the computer and network security point of view. Performing newly arrived malware detection and classifying them to their respective families involves feature vectors that represent the essential features of malicious codes. There are two fundamental techniques for analyzing malware with the help of feature vectors, namely static analysis, and dynamic analysis. Static analysis examines the behavior of malicious code without executing them. The sequence of code or instructions is cross-checked to understand the purpose of it [4].

On the other hand, dynamic analysis involves the execution of the malware under strictly remote or virtual sandbox devices to understand the working of it [5]. Paper [6] shows how these traditional methods fail to deliver excellent classification results. Paper [7] proposed a new technique to classify malicious codes by visualizing the binary files. Samples of malicious code binary files are converted into malware gray-scale images with the help of 2D arrays [8]. Paper [9] showed different malware visualization techniques for malware classification, which are better than analysis techniques. Features of the image are a vital aspect of the image processing problem. Features hold essential information related to the image, which later helps in the classification to distinguish between the classes. Feature extraction is the process of representing image data in the form of a set of features for extracting the features of the image paper [8] used GIST descriptor. Paper [10] explained feature extraction and different techniques of it. Paper [11] showed that computing the features of the image with the help of various feature extractors like GIST, SURF, SIRF, and HOG requires more computational resources. When these extractors are applied to more massive datasets, they are less impactful, which causes misclassification. In the same paper, how CNN can be used as an automatic feature extractor and how it is better than using feature descriptors is presented. Paper [12] concluded that the feature computed by CNN is less prone to misclassification, which improves the overall results.

The main objective of the paper is to build an efficient malware classification model with the hybridization of machine learning algorithms. The challenges are avoiding image data imbalance, building the fine-tuned CNN model, replacing the activation function with suitable machine learning algorithms. During this research work, malware visualization and Convolutional neural network are used for automatic computation of features from the images. Paper [10, 11, 12] shows that when CNN is used as an automatic feature extractor, it gives better results for malware classification. To improve the results, machine learning algorithms are used in combination with CNN. Features computed by CNN are used as input to the Support vector machine algorithm. The replacement of the softmax activation function with SVM gives better results and minimal misclassification. The proposed CNN model is compared with CNN Architectures VGG16, ResNet50, InceptionV3, and Xception.

The research work is organized into sections as the following: Section 2 presents the related work of malware classification. Section 3 presents the usage of different existing CNN architectures and analysis of it. Section 4 represents the methodology used for building CNN and the hybrid CNN SVM model for malware classification. Section 5 compares the performance of proposed CNN, and the hybrid CNN+SVM models with different CNN architectures. Section 6 is conclusions.

2. Related Works

This section represents the related work for malware analysis and classification.

Cybersecurity professionals and communities analyze malicious codes and software for various security purposes. They use this analyzed information to study the infections caused by malware and find solutions for that. This analysis helps to address the issues during further infections. Paper [13] proposed a malware detection system based on static analysis for android, which shows that the false positive rate of this method is more. Paper [14] performed malware classification based on Application Program Interfaces (API), which involves static analysis and stated that applying static analysis for the classification of malware is not working as expected. Static analysis methods have some limitations and can be easily penetrated by using Obfuscation techniques [6]. Paper [15] used dynamic analysis technique for classification of malware and stated that the method with machine learning models work well. The dynamic analysis technique requires more resources, which makes it challenging to compute features for large size datasets [11].

Paper [8] proposed a novel approach for classifying the malware with the help of images. The technique involves reading the malware binary files bits and converting them into decimal values. These decimal values are stored in a 2D array and then visualized as a gray-scale image. Using the GIST descriptor for feature computation is done. Then the images are classified with the help of the K-Nearest Neighbours algorithm (KNN). Paper [9] followed the different visualization techniques then applied in paper [8] and applied feature extraction. These extracted features are given to supervised learning algorithms like KNN and SVM. Paper [16] used feature extractor for extracting features from binary files, which are generated with the help of malware executable programs. These features are then used with machine learning classifier Naive Bayes.

Deep learning is a subset of machine learning. These techniques are successful for many problems. Paper [17], is based on deep learning that applies to malware classification and proposed M-CNN architecture. Paper [18] applied a malware visualization technique with deep learning and Long-Short Term Memory (LSTM) for classifying malware. Paper [19] presented the architectural study of different CNN architectures and proposed IMCFN architecture based on CNN. Paper [20] proposed the Echo State Network (ESN) and Recurrent Neural Network (RNN) based approach for malware classification and stated ESN models are better than RNN. Paper [21] used a hybrid approach of Gated Recurrent Unit (GRU) and Support Vector Machine (SVM) to solve the malware classification problem using images. Paper [22] used CNN architecture ResNet50 for the classification of malware software in which transfer learning was used.

As the above approaches involve image texture feature extraction using different feature extractors techniques which require resources as well as time for computation. From the analysis of the existing methods available for malware, classification has its limitations like static analysis based methods are prone to obfuscation techniques.

Dynamic analysis is incapable of handling large datasets. Deep and Machine learning-based techniques achieved more accuracy but fell short when it comes to the misclassification rate. In the proposed method, A CNN model is used to compute the features of an image automatically with the help of multiple layers of the network. Taking inspiration from the above research work, proposed an approach to classify malware with deep learning and combined it with machine learning. This hybrid method delivers more accurate results and enhances it with the help of the ML algorithm. Proposed a fine-tuned CNN model that takes pre-processed images as input and computes features. In hybridization, the last layer of CNN is replaced by machine learning algorithms to enhance the results.

While working on the CNN model and considering the above research methodologies, handling over-fitting of the CNN model is more important. Achieving higher accuracy for classification is essential, but examining whether the model is overfitting is necessary too. Keeping these problems in mind, for this research work from the method of using Image Visualization and Deep learning are used [14]. The ability of CNN of automatic feature extraction is used in the proposed method. Classifying these features with the Softmax activation function works well or not is examined during our research by replacing it with machine learning algorithms and evaluating the performance of the hybridization.

3. Analysis of existing CNN Architectures for Malware Classification

This section presents an analysis of malware classification using existing CNN models. It gives a method of converting malware binary to the image, which is proposed by [8] and fine-tuned architectures of CNN.

CNN is widely used for problems based on computer vision like fast fashion guided clothing [24], safety harness detection [25], image reconstruction problems like HDR [26], and CT image reconstruction [27], color recognition problems like vehicle [28], and tongue [29] color recognition and image classification problems like indoor-outdoor images [30], and city-landscape images classification [31].

If CNN [32] is compared with a neural network, then the difference is that CNN layers are organized in such a way that each layer takes an input in the form of 3-Dimension. Neurons of CNN layers are not connected to every other neuron-like NN. The final output layer is converted into the vector, which contains probability scores for the classification. CNN includes a variety of layers that can be stacked like convolutional layers, pooling layers, dropout layers [32]. Paper [33] explained the working and math behind the Convolutional layers working. Convolution Layer takes the input in the form of height, width, and depth on which different filters are applied. These filters help to extract feature maps from the image. Parameters present in this layer are mostly user-specific. The Pooling Layer looks similar to the convolution layer. However, these layers perform the task of pooling values from filters, which might be Minimum Pooling, Average Pooling, or Maximum Pooling. This pooling of values helps in reducing the dimensionality of the network. Dropout layers take the portion weights from all the weights learned by the network and iteratively learn from them [33].

3.1. *Converting malware binaries into images*

In the last few years, researchers formulated the malware classification problem as an image classification problem [50]. Binary texture feature extraction is prone to binary obfuscation techniques that can be handled by converting binaries into gray-scale images [7]. Gray-scale images contain two colors, black and white, which help to capture details from the image regions. The process of converting malware files into images involves binary to decimal conversions of the binary bits and then storing these decimal values into a 2D array. Each value of this array is considered as the pixel value of an image [8]. Fig.1. Shows the malware binaries into the images conversion process.

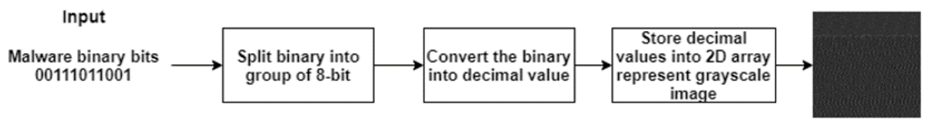
Input
Malware binary bits 00111011001
Split binary into group of 8-bit
Convert the binary into decimal value
Store decimal values into 2D array represent grayscale image

Fig.1. Malware binaries to the gray-scale image conversion process [8]

3.2. *CNN Architectures VGG16, ResNet50, InceptionV3 and Xception*

Paper [19] presented different architectures of CNN for malware detection and classification problems. Paper [22] shows the results of different CNN architectures for malware classification problems.

A. VGG16

VGG16 architecture is presented by K. Simonyan [34] for solving the problem of the ILSVRC-2012 dataset challenge of large scale visual recognition. The model contains five convolutional layers, five max-pooling layers, three fully-connected layers, a dense layer of 4096 neurons (softmax) in total 16-layers. The VGG16 model is investigated to solve different problems such as plankton classification [35], lung nodules classification [36], and plant image classification [37]. In our implementation for the input, an image of size 224*224 is given to the model, and the output layer has 160 neurons that are used for classification with a softmax activation function. The total number of parameters trained during the experimentation is 18,357,544.

B. ResNet50

Kaiming He [38] presented this architecture in 2015 for image recognition problems. It is 50 layers deeper with residual blocks, which shifts the input over the identity connection. The ResNet50 model is investigated to solve different issues such as disease classification [39], fault diagnosis [40], and malaria cell classification [41]. It takes a resolution of 224*224 size as input. During our research work, an input image of 200*200 is given to the model. For the output layer, 2048 neurons used with softmax activation function are used because of limited computation resources. The total number of parameters trained during the experimentation is 23,690,162.

C. InceptionV3

This architecture is presented by C Szegedy [42] in 2016, in which applying the Residual block in network improves the result is shown. This architecture contains 48 convolution layers in the network. It takes 224*224 size of the image as input. InceptionV3 model is investigated to solve different problems such as flower classification [43], traffic sign classification [44], skin cancer classification [45]. The critical part of this model is the factorization of convolutions, which reduces the number of parameter computations. This architecture is fine-tuned during research for our problem for input; 128*128 images are given, and for the output layer, 2048 neurons used for classification with a softmax activation function. The total number of parameters trained during the experimentation is 21,905,234.

D. Xception

This architecture is presented by Fran çois Chollet [46] in 2016 in which uses of depthwise separable convolutions are stated. The Xception model is investigated to solve different problems such as software fault injection and monitoring in processor functional units [47], audio event detection, and tagging [48]. This model takes the input of size 224*224. This architecture is referred to as an improvement over inception-v3 architecture. Both of these architectures have the same number of parameters. However, in the Xception model, the parameters are used more effectively. We used this model with the input image size of 128*128, and for the output layer, 2048 neurons used for classification with a softmax activation function because of limited computing resources. The total number of parameters trained during the experimentation is 20,963,930.

4. Methodology

This section represents the methodology of proposed CNN and Hybrid CNN+SVM. In the proposed architecture of CNN, pre-processing and data augmentation techniques are utilized by analyzing their impact in the research work [19] for boosting the results.

4.1. Preprocessing

Preprocessing the images is necessary to remove impurities and noise. Samples of images present in the dataset do not have the same size, so resizing of the images is essential. During this experimental work resizing of images is done on different sizes like 32, 64, 128, and 256.

4.2. Data augmentation

Data augmentation is a technique used when the dataset is not large enough. The Malimg dataset for malware classification consists of 9339 samples of 25 different families, and the distribution of samples in these 25 families is completely imbalanced. Data imbalance and data augmentation methods are used at the time of input data given to the CNN model to handle this problem. This technique creates a new dataset artificially from the given dataset. Image data augmentation creates a dataset by performing transformations of the given input image dataset. Transformations like rotation, shear, zoom, horizontal or vertical flip, and adjusting brightness levels of the images.

Keras provides a deep learning library ImageDataGenerator, which is used during experimental work to perform augmentation of the dataset by observing the improvement in the paper [7]. Parameters applied to the transformed dataset are presented in Table 1.

Table 1. Image data augmentation setting for the dataset

Method	Settings	Method	Settings
Rotation Range	0.1	Rescale	1/255
Width Shift	0.1	Shear Range	0.1
Height Shift	0.1	Zoom Range	0.1
Horizontal Flip	True	Fill Mode	Nearest

Fig.2. Shows the transformation of the image after applying parameters from Table 1.

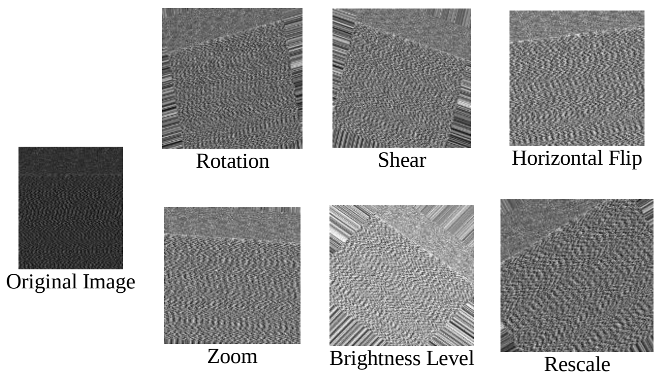

Fig.2. Malware samples after data augmentation

4.3. Proposed CNN architecture

CNN requires input in the form of 3-Dimensions Height, width, and depth. In particular, the input image of 32x32x1 is given to the CNN model for processing, followed by three convolution layers with filters 50, 70, 70, respectively. Each convolution layer is followed by LeakyReLU and Max Pooling layers, which reduce the input signals. Flatten layer normalizes the input into 256 neurons of the FC layer, which will later get classified with 25 neurons using the softmax activation function.

1. Input = 32 x 32 x 1
2. Convolutional Layer 1 = Filters = 50, Kernel = 5, 5, Padding = Same
3. LeakyReLU Layer 1 = Alpha = 0.1
4. Max Pooling Layer 1 = Pool Size = 2, 2 Padding = Same
5. Convolutional Layer 2 = Filters 70, Kernel = 3, 3 , Padding = Same
6. LeakyReLU Layer 2 = Alpha = 0.1
7. Max Pooling Layer 2 = Pool Size = 2, 2 Padding = Same
8. Convolutional Layer 3 = Filters = 70, Kernel = 3, 3 , Padding = Same
9. LeakyReLU Layer 3 = Alpha = 0.1
10. Max Pooling Layer 3 = Pool Size = 2, 2 Padding = Same
11. Flatten Layer
12. Dense Layer 1 = 256 Neurons
13. Dense Layer 2 = 25 Neurons of Output Classes

While proposing this architecture, every layer of this model is configured as per the dataset samples. Usage of LeakyReLU activation function is to make the model to train faster than other existing architectures. It also makes the slope of negative values smaller as compared to ReLU used in existing models. Proper analysis of parameters is done and selected only those parameters which are delivering good results. Table 2. Shows the parameters used for training this model. The parameters which are irrelevant for training are skipped from unnecessary training, which impacts the results. Similarities related to architecture are possible, but architecture developed for one problem cant suite for all problems, proper tuning and parameter selection is also essential. Applying techniques like Image data augmentation and data preprocessing before training the data helped us to get good results.

4.4. Hybrid CNN (CNN + SVM)

The proposed model uses CNN with machine learning algorithms as an intelligent system that classifies malware images efficiently to the respective classes. Papers [18, 23, 50] hybrid CNN works better than CNN. This paper [23] presents a hybrid CNN model with SVM to reduce the computational requirements and accuracy. Instead of using traditional softmax activation function for classifying malware and cross-entropy function for computation of loss in the proposed model, the second FC layer is replaced by the SVM algorithm. In this paper, the results computed for the L2-SVM method. Fig.3. Shows the architectural diagram of the proposed model where the experiments performed.

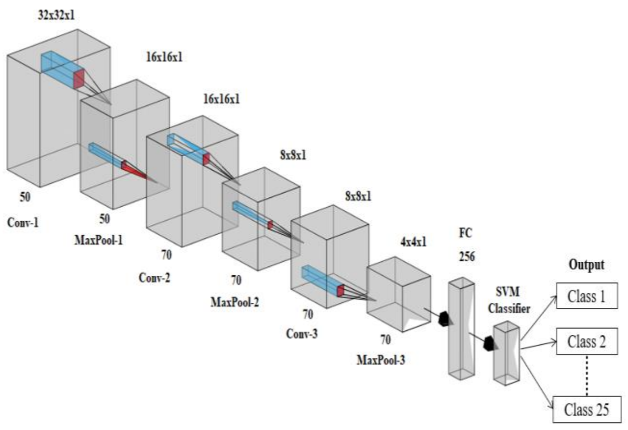

Fig.3. The Proposed hybrid CNN+L2-SVM model.

SVM is a supervised machine learning algorithm [51]. It uses the principle of finding hyperplanes into an N-dimensional sample space. These hyperplanes with more margin help to classify the samples correctly. The hyperplane is a line or boundary that separates data samples into distinct classes. SVM works on marginal boundaries, hyperplanes with maximum margins are selected. The data points which are near to these hyperplanes are called support vectors. The position of the hyperplane depends on these support vector points. SVM is capable of solving binary classification problems. Paper [52] shows different methods on multi-class SVM, in which stated that the one-against-one method works better than one-against-all methods. The SVM kernel functions help transform input data into the required format; that's why in our research work LinearSVC is used based on the one-against-one method. According to paper [52], a one-against-one approach is more suitable for Multi-class problems. This approach creates binary classifiers based on the number of classes present in the problem using the following eq.(1)

$$num\_classes \times \frac{(num\_classes-1)}{2} \tag{1}$$

In this problem, number of classes = 25 then 25* (25 - 1) / 2 = 300 binary classifiers are created. While taking the prediction decisions, "max wins" [53] algorithm is used in which each classifier votes to its predicted class, and at the final stage, most votes obtained class is considered as a predicted class. When two classes got the same number of votes, then the class which has the smallest index value will get selected.

L2-SVM is a Soft-margin support vector machine that uses the sum of squared slack variables as compared to L1-SVM as per paper [54]. The optimization function working described in [23] for L2-SVM is as follows.

$$Minimize\ Q(w,\xi) = \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 \tag{2}$$

$$Subject\ to\ yi(xi \cdot w + b) \geq 1 - \xi_i \qquad for\ i = 1,2,\ldots.,m \tag{3}$$

where, *w* = n-dimensional weight vector, *b* = bias value or term, i = Slack variable, *C* = penalty parameter.

The first term of equation (2) tries to minimize the margin, whereas the second term handles the misclassification. Penalty parameter *'C'* seeks the balance between these two. In the proposed methodology, L2-SVM works on extracted features vectors from CNN. The feature vectors are then classified for better accuracy.

5. Experimental Details, Results, and Discussion

This section represents the experimental details, results, and discussion of the proposed CNN approach. The experimental analysis is carried out by using the Python programming language for the implementation. Keras library is used to implement the CNN model. For executing the programs, Nvidia GeForce 940M 2GB GPU for training and Intel Core i5-4500 processor with 8 GB RAM is used. For resource-intensive tests, the Google Colab platform is used.

5.1. Dataset

The dataset used for experimental work is the Malimg dataset [55]. This dataset is generated from malware binary (.asm) files. These binaries are converted into an 8-bit vector and then transformed into the [0, 255] range where '0' is black, and '255' is white. A total of 9339 samples are present in this dataset. The dataset comprises 25 malware families (classes) with a varying number of samples per family and is available for download [55]. Fig.4 shows the details of the dataset with the family name and number of samples.

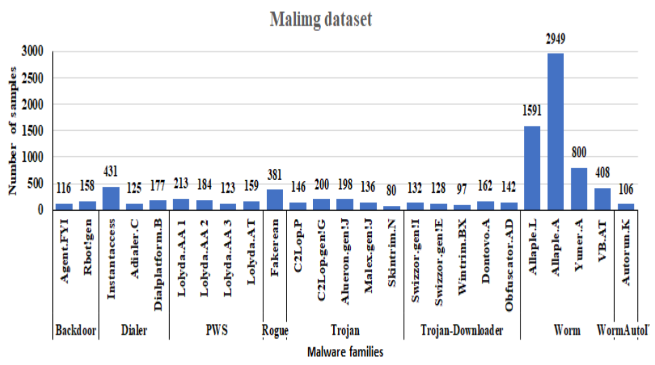

Fig.4. Dataset description

5.2. Results and Discussion

In this paper, to evaluate the performance of the models following evaluation metrics are considered. Researchers most commonly use these measures for assessing performance [19].

Precision: It is the ratio of true positive samples of the class to the sum of true positive and false positive samples of the predictions [56].

$$Precision = \frac{TP}{(TP+FP)} \tag{4}$$

Recall: It is the ratio of labeled data samples to the correctly predicted samples [56].

$$Recall = \frac{TP}{(TP+FN)} \tag{5}$$

$$TP = \frac{TP}{(TP+FN)} \tag{6}$$

$$TN = \frac{TN}{TN+FN} \tag{7}$$

$$FP = \frac{FP}{(FP+TN)} \tag{8}$$

$$FN = \frac{FN}{(TP+FN)} \tag{9}$$

$$FPR = \frac{FP}{FP+TN} \tag{10}$$

$$FNR = \frac{FN}{TP+FN} \tag{11}$$

$$FDR = \frac{FP}{TP+FP} \tag{12}$$

F1 Score: It is the average weighted values of precision and recall [56].

$$F1\ Score = 2 \times \frac{(Precision*Recall)}{(Precision+Recall)} \tag{13}$$

Accuracy: It is the ratio of true positive samples of all classes to the sum of all samples present in the confusion matrix [56].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{14}$$

For applying the CNN approaches, the dataset is partitioned into different split ratios, like 60%-40%, 70%-30%, 80%-20%, and 90%-10% for training and testing. Images from the dataset are resized into the fixed-size like 32x32x1 to handle the problem of variable size images. Table 2 shows the CNN model parameters.

Table 2. CNN Model parameters

Input Image Size	32x32x1
Batch Size	64
Epochs	1000
Optimizer	Adam
Learning Rate	0.001
Total Parameters	370,441
Trainable Parameters	364,016
Non-Trainable Parameters	6425
FC Layer Features	256

Data augmentation is performed to handle the imbalance between the dataset. Table 3 shows the results of proposed CNN with and without data augmentation. The data augmentation technique is built to create heterogeneity of the available data. Data augmentation helps in training large neural networks without collecting new data. That helped in our experimental work to improve the accuracy of the proposed CNN model.

Table 3. Impact of using Data Augmentation

Without Data Augmentation		With Data Augmentation	
Training	Testing	Training	Testing
95.49	96.88	98.72	98.03

Results obtained with different CNN architectures and the proposed model using the softmax activation function are presented. All models are trained for 1000 epochs using Adam optimizer. Results are calculated with 10-Folds cross-validation, and the accuracies given below are average accuracies of the 10-folds. Time calculation is done based on the hardware allocated by Google's colab at that moment and may vary on more powerful hardware.

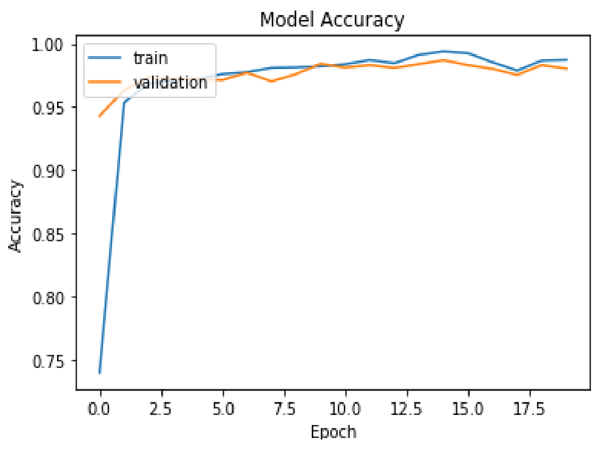

Fig.5. Accuracy of the proposed CNN model

The proposed CNN model is compared with VGG16, ResNet50, InceptionV3, and Xception. All the tests are performed with the help of Google's Colab, but in some cases, RAM of 32GB is not sufficient to handle. Table 4 presents the results of selected models with a 60-40 split ratio. The proposed CNN model shows better accuracy as compared to other models. The average execution time taken by the proposed CNN model is significantly better than other models.

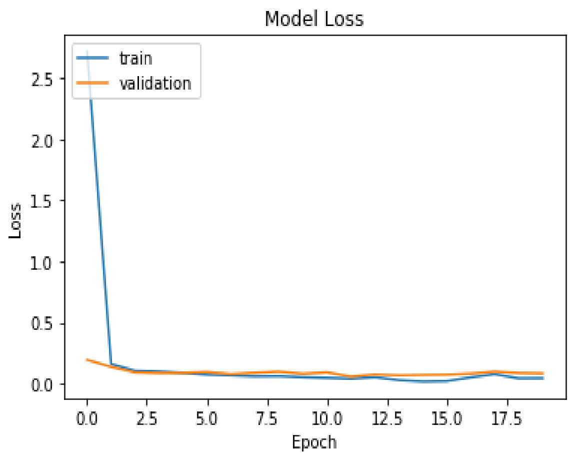

Fig.6. Loss of proposed CNN model

Table 4. Comparison of different CNN models.

	Proposed CNN	VGG16	ResNet50	InceptionV3	Xception
Accuracy (in %)	98.03	96.96	97.11	97.22	97.56
Average execution time (Hours)	1	1.10	19.89	9.7	7.5

The CNN model used for this image classification problem is trained using the Softmax activation function and SVM algorithm. Replacing the Softmax FC layer of 256 neurons with an ML algorithm improves the results. It does the accurate classification of malware samples with very little misclassification.

Table 5 shows the comparison of results for L2-SVM and softmax CNN models for malware classification. The accuracy of CNN+L2-SVM is more as compared to CNN+Softmax.

Table 5. Comparison of CNN using Softmax and replacing it with ML algorithms

CNN + Softmax (%)	CNN + L2-SVM (%)
98.03	99.59

Table 6 shows the result analysis by applying different split ratios. The split ratio indicates the Training and Testing ratio of the dataset. The results of 60-40% are considered during the research.

Table 7 shows the different statistical measures used for analyzing the performance of the proposed CNN+L2-SVM model. In which the obtained values are very less that prove the model is performing more accurately.

Table 6. Results of Hybrid CNN+L2-SVM

Split Ratios (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
60-40	99.59	99.22	99.04	99.12
70-30	99.53	98.98	98.78	98.87
80-20	99.51	98.85	98.77	98.80
90-10	99.78	99.16	99.16	99.16

Table 7. Statistical measures used for performance analysis

False Positive Rate(FPR)	False Negative Rate(FNR)	False Discovery Rate(FDR)	Negative Prediction Value(NPV)
0.0001	0.0095	0.0077	0.9998

Table 8 shows the cross-validation results which are used for obtained results analysis. For analysis, we used different values of cross folds. We tested up to the value of K=25.

Fig.7 shows the normalized confusion matrix, in which rows represent actual classes and columns, prediction classes. Diagonal elements show the degree of correct classification, whereas off-diagonal elements show the degree of incorrect classification. The percentage of the correctly classified sample is shown where 1 represents the malware samples belonging to that particular class and is 100% correctly classified according to the true labels present in the dataset. From this figure misclassification rate of malware samples can be determined where samples from only six families are incorrectly classified. However, the ratio of misclassified samples is very low.

Table 8. Cross-validation results using different K-values

K=3	K=5	K=7	K=10	K=25
99.50%	99.47%	99.55%	99.65%	99.57%

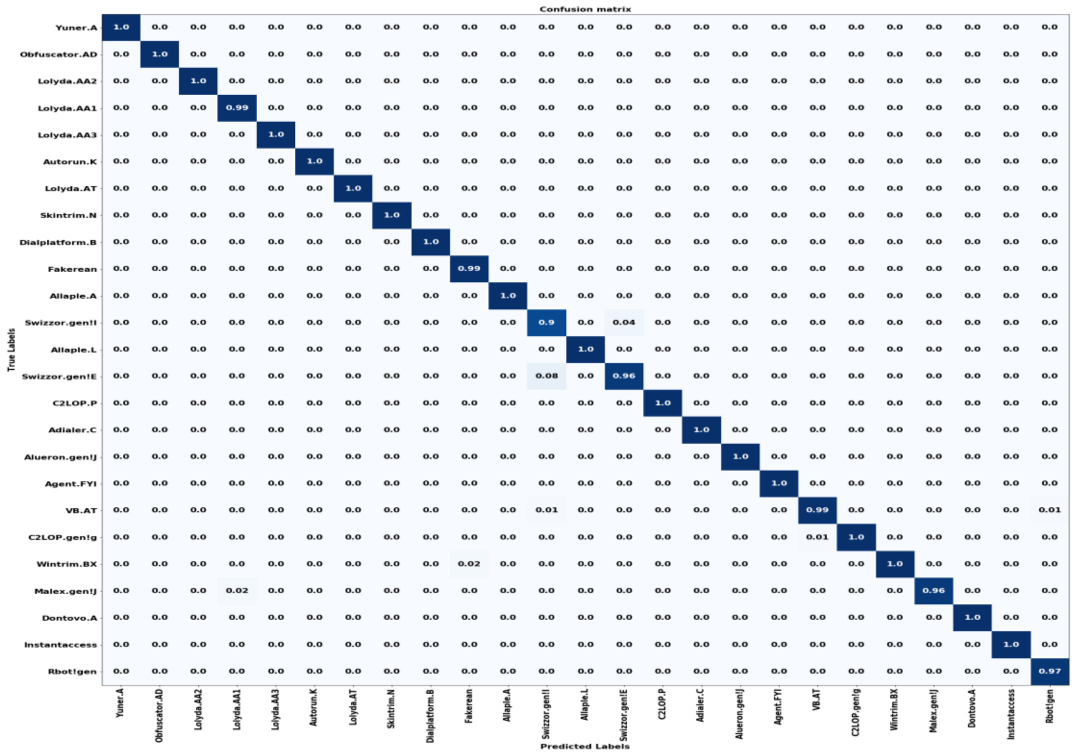

Fig.7. CNN+L2-SVM normalized confusion matrix

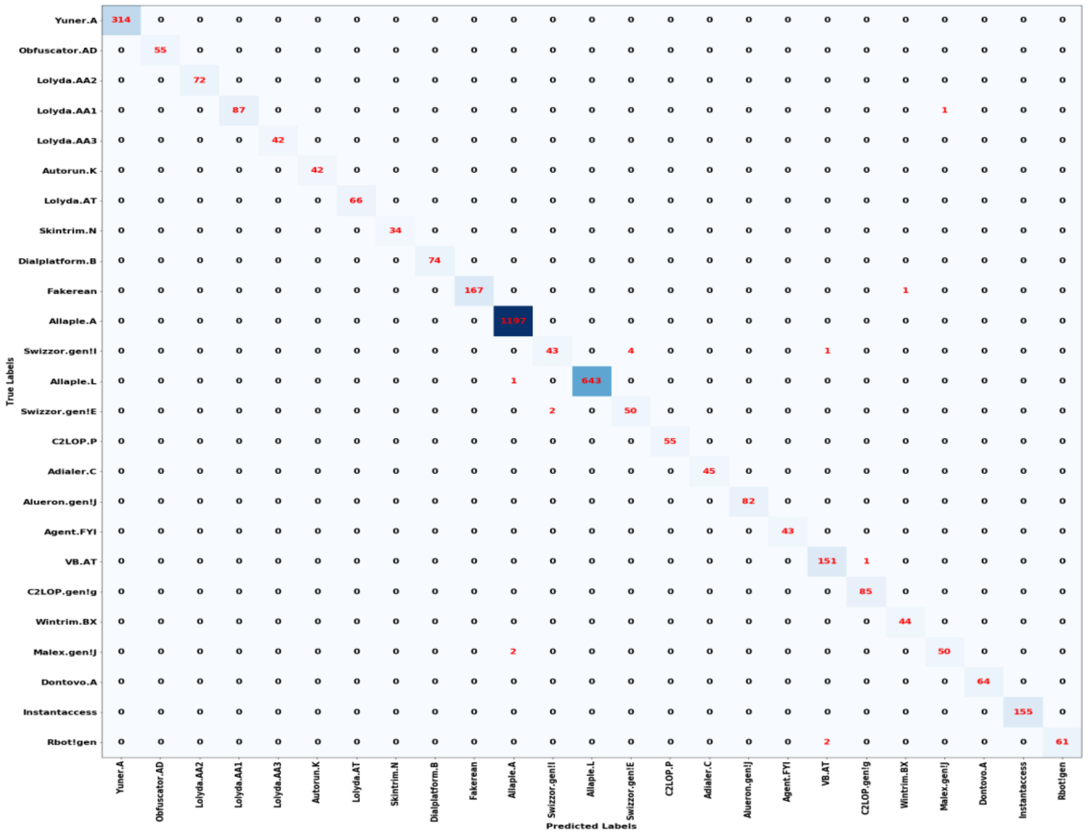

Fig.8. CNN+L2-SVM confusion matrix of test image sample distribution

Fig. 8 explains the test sample distribution from the test dataset. Diagonal elements show the degree of correct classified image samples. In contrast, off-diagonal elements show the degree of incorrect classified image samples. The diagonal numbers of correctly classified samples present in the dataset. The off-diagonal numbers show the incorrectly classified samples. From this image, we can see that hardly one or two images are incorrectly classified.

We compared the results of the proposed model with other studies. Table 8 presents a comparison of the results. The proposed model shows significant performance improvement.

Table 9. Results comparison

Reference	Visualization Method	Technique	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Nataraj et al.[7]	Gray-scale	KNN	97.18	-	-	-
Songqing Yue[60]	Gray-scale	Vgg-verydeep-19	97.32	-	-	-
S. Yajamanam et al.[57]	Gray-scale	GIST+CNN	98	-	-	-
N. Bhodia et al.[58]	Gray-scale	ResNet	94.8	-	-	-
Zhihua Cui[59]	Gray-scale	GIST+SVM	92.2	92.5	91.4	-
Zhihua Cui[59]	Gray-scale	GIST+KNN	91.9	92.1	91.7	-
Zhihua Cui[59]	Gray-scale	GLCM+SVM	93.2	93.4	93	-
Zhihua Cui[59]	Gray-scale	GLCM+KNN	92.5	92.7	92.3	-
Zhihua Cui[59]	Gray-scale	IDA+DRBA	94.5	94.6	94.5	-
Zhihua Cui[59]	Gray-scale	NSGA-II	97.6	97.6	88.4	-
D. Vasan[19]	Color	IMCFN	98.82	98.85	98.81	98.75
Proposed approach	Gray-scale	CNN+L2-SVM	**99.59**	**99.22**	**99.04**	**99.12**

6. Conclusions

Malware is most commonly used for performing attacks by cyber attackers. To control the attackers from performing attacks, stealing the information, and harming computer systems, security specialists and antimalware software companies are constantly working on identifying new methods. In this research work, the deep learning-based approach for malware classification is presented. Different researcher's methods for Malware classification were studied based on the Malimg dataset, which has 9339 samples of malware of 25 malware families. On this dataset, Considering the computational resources and time requirements existing CNN models, we proposed the CNN model by combining Data pre-processing and augmentation techniques and obtained results (98.03%) which are far better than different existing architectures of CNN such as VGG16 (96.96%), InceptionV3 (97.22%), Xception (97.56%), and ResNet50 (97.11%). The proposed CNN model requires less time (1hr) as compared to different architectures of CNN in Table 4. VGG16 (1.10hr), InceptionV3(9.7hrs), Xception (7.5hrs), and ResNet50 (19hrs). The hybridization technique is used for performing the task of classification. The Softmax activation function of the proposed CNN model is replaced with the Support Vector Machine and proposed hybrid CNN+L2-SVM model and obtained accuracy (99.59%). The proposed hybrid model of CNN+L2-SVM gives a 1.56% accuracy boost than the Softmax activation function and performs well in terms of misclassification of malware samples (see table 5). The proposed CNN+L2-SVM performance is compared with well-known models (see table 9). The proposed model achieves more accurate results with minimal rate of misclassification in less time and computational resources. The reason behind getting good results is a combination of several techniques like handling data imbalance, pre-processing and scaling of images, fine-tuning of the CNN model, using multi-class SVM, and proper replacement of SVM with Softmax. L2-SVM uses squared slack variables than L1-SVM, which helped us to get generalized results.

CNN models can further be improved with fine-tuning, and other machine learning algorithms to minimize the execution time.

References

[1] Symantec, Inc. Internet Security Threat Report Vol.24, 2020. https://docs.broadcom.com/docs/istr-24-2019-en
[2] Malwarebytes Inc. 2020 State of Malware Report February,2020 https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf
[3] F. M. Services and S. Report, "HR-Trends 2020," HR-Trends 2020, 2019, doi: 10.34157/9783648132616.
[4] Moser, Andreas, Christopher Kruegel, and Engin Kirda. "Limits of static analysis for malware detection." In Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), pp. 421-430. IEEE, 2007, doi: 10.1109/ACSAC.2007.21
[5] Damodaran, Anusha, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H. Austin, and Mark Stamp. "A comparison of static, dynamic, and hybrid analysis for malware detection." Journal of Computer Virology and Hacking Techniques 13, no. 1 (2017): 1-12, doi: 10.1007/s11416-015-0261-z
[6] Sharif, Monirul, Vinod Yegneswaran, Hassen Saidi, Phillip Porras, and Wenke Lee. "Eureka: A framework for enabling static

malware analysis.” In European Symposium on Research in Computer Security, pp. 481-500. Springer, Berlin, Heidelberg, 2008, doi: 10.1007/978-3-540-88313-5_31.
[7] Nataraj, Lakshmanan, Vinod Yegneswaran, Phillip Porras, and Jian Zhang. “A comparative assessment of malware classification using binary texture analysis and dynamic analysis.” In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, pp. 21-30. 2011, doi: 10.1145/2046684.2046689.
[8] Nataraj, Lakshmanan, Sreejith Karthikeyan, Gregoire Jacob, and Bangalore S. Manjunath. “Malware images: visualization and automatic classification.” In Proceedings of the 8th international symposium on visualization for cyber security, pp. 1-7. 2011, doi: 10.1145/2016904.2016908.
[9] Makandar, Aziz, and Anita Patrot. “Overview of malware analysis and detection.” International Journal of Computer Applications 975 (2015): 8887.
[10] Kumar, Gaurav, and Pradeep Kumar Bhatia. “A detailed review of feature extraction in image processing systems.” In 2014 Fourth international conference on advanced computing & communication technologies, pp. 5-12. IEEE, 2014, doi: 10.1109/ACCT.2014.74.
[11] Shaheen, Fatma, Brijesh Verma, and Md Asafuddoula. “Impact of automatic feature extraction in deep learning architecture.” In 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pp. 1-8. IEEE, 2016, doi: 10.1109/DICTA.2016.7797053.
[12] Horn, Z. C., L. Auret, J. T. McCoy, Chris Aldrich, and B. M. Herbst. “Performance of convolutional neural networks for feature extraction in froth flotation sensing.” IFAC-PapersOnLine 50, no. 2 (2017): 13-18, doi: 10.1016/j.ifacol.2017.12.003
[13] Schmidt, A-D., Rainer Bye, H-G. Schmidt, Jan Clausen, Osman Kiraz, Kamer A. Yuksel, Seyit Ahmet Camtepe, and Sahin Albayrak. “Static analysis of executables for collaborative malware detection on android.” In 2009 IEEE International Conference on Communications, pp. 1-5. IEEE, 2009, doi: 10.1109/ICC.2009.5199486.
[14] Iwamoto, Kazuki, and Katsumi Wasaki. “Malware classification based on extracted api sequences using static analysis.” In Proceedings of the Asian Internet Engineering Conference, pp. 31-38. 2012, doi: 10.1145/2402599.2402604
[15] Anderson, Blake, Daniel Quist, Joshua Neil, Curtis Storlie, and Terran Lane. “Graph-based malware detection using dynamic analysis.” Journal in computer Virology 7, no. 4 (2011): 247-258, doi: 10.1007/s11416-011-0152-x
[16] Schultz, Matthew G., Eleazar Eskin, F. Zadok, and Salvatore J. Stolfo. “Data mining methods for detection of new malicious executables.” In Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, pp. 38-49. IEEE, 2000, doi: 10.1109/SECPRI.2001.924286.
[17] Kalash, Mahmoud, Mrigank Rochan, Noman Mohammed, Neil DB Bruce, Yang Wang, and Farkhund Iqbal. “Malware classification with deep convolutional neural networks.” In 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1-5. IEEE, 2018, doi: 10.1109/NTMS.2018.8328749
[18] Akarsh, S., K. Simran, Prabaharan Poornachandran, Vijay Krishna Menon, and K. P. Soman. “Deep Learning Framework and Visualization for Malware Classification.” In 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 1059-1063. IEEE, 2019, doi: 10.1109/ICACCS.2019.8728471.
[19] Vasan, Danish, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. “IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture.” Computer Networks 171 (2020): 107138, doi: 10.1016/j.comnet.2020.107138.
[20] Pascanu, Razvan, Jack W. Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas. “Malware classification with recurrent networks.” In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1916-1920. IEEE, 2015, doi: 10.1109/ICASSP.2015.7178304.
[21] Agarap, Abien Fred. “An architecture combining convolutional neural networks (CNN) and support vector machine (SVM) for image classification.” arXiv preprint arXiv:1712.03541 (2017).
[22] Rezende, Edmar, Guilherme Ruppert, Tiago Carvalho, Fabio Ramos, and Paulo 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), pp. 1011-1014. IEEE, 2017, doi: 10.1109/ICMLA.2017.00-19
[23] Tang, Yichuan. “Deep learning using linear support vector machines.” arXiv preprint arXiv:1306.0239 (2013).
[24] He, Yuhang, and Long Chen. “Fast fashion guided clothing image retrieval: Delving deeper into what feature makes fashion.” In Asian Conference on Computer Vision, pp. 134-149. Springer, Cham, 2016, doi: 10.1007/978-3-319-54193-8_9
[25] Fang, Weili, Lieyun Ding, Hanbin Luo, and Peter ED Love. “Falls from heights: A computer vision-based approach for safety harness detection.” Automation in Construction 91 (2018): 53-61, doi: 10.1016/j.autcon.2018.02.018
[26] Eilertsen, Gabriel, Joel Kronander, Gyorgy Denes, Rafał K. Mantiuk, and Jonas Unger. “HDR image reconstruction from a single exposure using deep CNNs.” ACM Transactions on Graphics (TOG) 36, no. 6 (2017): 1-15, doi: 10.1145/3130800.3130816.
[27] Gupta, Harshit, Kyong Hwan Jin, Ha Q. Nguyen, Michael T. McCann, and Michael Unser. “CNN-based projected gradient descent for consistent CT image reconstruction.” IEEE transactions on medical imaging 37, no. 6 (2018): 1440-1453, doi: 10.1109/TMI.2018.2832656.
[28] Rachmadi, Reza Fuad, and I. Purnama. “Vehicle color recognition using convolutional neural networks.” arXiv preprint arXiv:1510.07391 (2015).
[29] Hou, Jun, Hong-Yi Su, Bo Yan, Hong Zheng, Zhao-Liang Sun, and Xiao-Cong Cai. “Classification of tongue color based on CNN.” In 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(, pp. 725-729. IEEE, 2017, doi: 10.1109/ICBDA.2017.8078731
[30] Szummer, Martin, and Rosalind W. Picard. “Indoor-outdoor image classification.” In Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database, pp. 42-51. IEEE, 1998, doi: 10.1109/CAIVD.1998.646032
[31] Vailaya, Aditya, Anil Jain, and Hong Jiang Zhang. “On image classification: City images vs. landscapes.” Pattern recognition 31, no. 12 (1998): 1921-1935, doi: 10.1016/S0031-3203(98)00079-X
[32] LeCun, Yann, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. “Backpropagation applied to handwritten zip code recognition.” Neural computation 1, no. 4 (1989): 541-551, doi:

10.1162/neco.1989.1.4.541.
[33] Samek, Wojciech, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. “Evaluating the visualization of what a deep neural network has learned.” IEEE transactions on neural networks and learning systems 28, no. 11 (2016): 2660-2673, doi: 10.1109/TNNLS.2016.2599820
[34] Simonyan, Karen, and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556 (2014).
[35] Tindall, Lucas, Cuong Luong, and Andrew Saad. “Plankton classification using vgg16 network.” (2015).
[36] Zhao, Defang, Dandan Zhu, Jianwei Lu, Ye Luo, and Guokai Zhang. “Synthetic medical images using F&BGAN for improved lung nodules classification by multi-scale VGG16.” Symmetry 10, no. 10 (2018): 519, doi: 10.3390/sym10100519.
[37] Abas, Mohamad Aqib Haqmi, Nurlaila Ismail, Ahmad Ihsan Mohd Yassin, and Mohd Nasir Taib. “VGG16 for plant image classification with transfer learning and data augmentation.” International Journal of Engineering and Technology (UAE) 7 (2018): 90-94.
[38] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition.” In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016, doi: 10.1109/CVPR.2016.90.
[39] Ray, Suhita. “Disease Classification within Dermascopic Images Using features extracted by ResNet50 and classification through Deep Forest.” arXiv preprint arXiv:1807.05711 (2018).
[40] Wen, Long, Xinyu Li, and Liang Gao. “A transfer convolutional neural network for fault diagnosis based on ResNet-50.” Neural Computing and Applications (2019): 1-14, doi: 10.1007/s00521-019-04097-w.
[41] Reddy, A. Sai Bharadwaj, and D. Sujitha Juliet. “Transfer Learning with ResNet-50 for Malaria Cell-Image Classification.” In 2019 International Conference on Communication and Signal Processing (ICCSP), pp. 0945-0949. IEEE, 2019, doi: 10.1109/ICCSP.2019.8697909
[42] Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. “Inception-v4, inception-resnet and the impact of residual connections on learning.” In Thirty-first AAAI conference on artificial intelligence. 2017.
[43] Xia, Xiaoling, Cui Xu, and Bing Nan. “Inception-v3 for flower classification.” In 2017 2nd International Conference on Image, Vision and Computing (ICIVC), pp. 783-787. IEEE, 2017.
[44] Lin, Chunmian, Lin Li, Wenting Luo, Kelvin CP Wang, and Jiangang Guo. “Transfer learning based traffic sign recognition using inception-v3 model.” Periodica Polytechnica Transportation Engineering 47, no. 3 (2019): 242-250, doi: 10.3311/PPtr.11480.
[45] Ridell, Patric, and Henning Spett. “Training set size for skin cancer classification using Google's inception v3.” (2017).
[46] Chollet, François. “Xception: Deep learning with depthwise separable convolutions.” In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251-1258. 2017, doi: 10.1109/CVPR.2017.195
[47] Carreira, Joao, Henrique Madeira, and João Gabriel Silva. “Xception: Software fault injection and monitoring in processor functional units.” Dependable Computing and Fault Tolerant Systems 10 (1998): 245-266.
[48] Gajarsky, Tomas, and Hendrik Purwins. “An Xception residual recurrent neural network for audio event detection and tagging.” In Sound and Music Computing Conference. 2018.
[49] Kim, Hae-Jung. “Image-based malware classification using convolutional neural network.” In Advances in Computer Science and Ubiquitous Computing, pp. 1352-1357. Springer, Singapore, 2017, doi: 10.1007/978-981-10-7605-3_215.
[50] Singh, Ajay, Anand Handa, Nitesh Kumar, and Sandeep Kumar Shukla. “Malware classification using image representation.” In International Symposium on Cyber Security Cryptography and Machine Learning, pp. 75-92. Springer, Cham, 2019, doi: 10.1007/978-3-030-20951-3_6.
[51] Cortes, Corinna, and Vladimir Vapnik. “Support-vector networks.” Machine learning 20, no. 3 (1995): 273-297.
[52] Hsu, Chih-Wei, and Chih-Jen Lin. “A comparison of methods for multi-class support vector machines.” IEEE transactions on Neural Networks 13, no. 2 (2002): 415-425, doi: 10.1007/BF00994018.
[53] Platt, John C., Nello Cristianini, and John Shawe-Taylor. “Large margin DAGs for multi-class classification.” In Advances in neural information processing systems, pp. 547-553. 2000.
[54] Koshiba, Yoshiaki, and Shigeo Abe. “Comparison of L1 and L2 support vector machines.” In Proceedings of the International Joint Conference on Neural Networks, 2003., vol. 3, pp. 2054-2059. IEEE, 2003, doi: 10.1109/IJCNN.2003.1223724.
[55] L. Nataraj, “Malimg Dataset.” https://www.dropbox.com/s/ep8qjakfwh1rzk4/malimg_dataset.zip?dl=0.
[56] Powers, David Martin. “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.” (2011).
[57] Yajamanam, Sravani, Vikash Raja Samuel Selvin, Fabio Di Troia, and Mark Stamp. “Deep Learning versus Gist Descriptors for Image-based Malware Classification.” In ICISSP, pp. 553-561. 2018, doi: 10.5220/0006685805530561.
[58] Bhodia, Niket, Pratikkumar Prajapati, Fabio Di Troia, and Mark Stamp. “Transfer learning for image-based malware classification.” arXiv preprint arXiv:1903.11551 (2019).
[59] Cui, Zhihua, Fei Xue, Xingjuan Cai, Yang Cao, Gai-ge Wang, and Jinjun Chen. “Detection of malicious code variants based on deep learning.” IEEE Transactions on Industrial Informatics 14, no. 7 (2018): 3187-3196, doi: 10.1109/TII.2018.2822680.
[60] Yue, Songqing. “Imbalanced malware images classification: a CNN based approach.” arXiv preprint arXiv:1708.08042 (2017).

Authors' Profiles

Sumit S. Lad, pursuing the Master of Technology in Computer Science and Engineering, from Rajarambapu Institute of Technology, Rajaramnagar, Sangli, MS, India. Completed B.E. from Shivaji University, Kolhapur, MS, India. His areas of interest are cloud computing, machine learning and classification problems.

Amol C. Adamuthe received Ph.D. from Mumbai University and Master of Technology in Computer Engineering from Dr. B. A. Technological University, Lonere, MS, India. He is currently an Assistant Professor at Rajarambapu Institute of Technology, Sakharale, Sangli, MS, India. His areas of interest are technology forecasting, optimization algorithms, soft computing, and cloud computing. He has published more than 45 papers at the international and national level. He is the recipient of the Institution of Engineers (India) Young Engineers Award for year 2018-19 in Computer Engineering discipline.

How to cite this paper: Sumit S. Lad, Amol C. Adamuthe, "Malware Classification with Improved Convolutional Neural Network Model", International Journal of Computer Network and Information Security(IJCNIS), Vol.12, No.6, pp.30-43, 2020. DOI: 10.5815/ijcnis.2020.06.03