Iraqi Journal of Information and Communications Technology(IJICT)
Conference Series: The 1st Conference of Applied Researches in Information Engineering(ARIE2021), 2021

ISSN:2222-758X
e-ISSN:2789-7362

MALWARE DETECTION SYSTEM BASED ON DEEP LEARNING TECHNIQUE

Zahraa Z. Edie [1], Ammar D. Jasim [2]
[1,2] College of Information Engineering, Al-Nahrain University, Baghdad, Iraq
zahraazedan20@gmail.com [1], ammar@coie-nahrain.edu.iq [2]
Received:5/5/2021, Accepted:22/6/2021

***Abstract*- Malicious software (malware) has long been used for illegal purposes; however, the number and variety of malware has increased dramatically in recent years, highlighting the need to enhance automated malware detection and classification. Neural network methodology has now progressed to the point that it might be able to outperform previous machine learning approaches such as Hidden Markov Models and Support Vector Machines (SVM). As a result, convolutional neural networks (CNNs) have outperformed conventional learning methods in a variety of tasks, including image recognition. Motivated by this success, we propose a CNN-based architecture for malware classification. In this paper, we propose a malware classification and detection framework using transfer learning based on existing Deep Learning models that have been pre-trained on massive image datasets, we applied a deep Convolutional Neural Network (CNN) based on the Xception model to perform malware image classification. The Xception model is a recently developed special CNN architecture that is more powerful with less overfitting problems than the current popular CNN models such as VGG16, The experimental results on a Malimg Dataset which is comprising 9,821 samples from 26 different families, Malware samples are represented as byteplot grayscale images and a deep neural network is trained to freeze the convolutional layers of Xception model adapting the last layer to malware family classification, The performance of our approach was compared with other methods including KNN, SVM, VGG16, etc. , the Xception model can effectively be used to classify and detect malware families and achieve the highest validation accuracy than all other approaches including VGG16 model which are using image-based malware, our approach does not require any features engineering, making it more effective to adapt to any future evolution in malware, and very much less time consuming than the champion's solution.**

I. INTRODUCTION

Malware is a term used to describe a computer program (code, scripts, and software) that is intended to disrupt computer operations or collect data [1] Malware compromises one's privacy, grants unauthorized access to device resources, and engages in other nefarious activities. Kaspersky Lab found 360,000 new malicious files every day in 2017 [2] . This worryingly high number is likely to increase in the future, given the motivation of malware authors to write such programs for illegal financial gain. Traditional antivirus products have occupied the most significant share of the security market for many years and are generally very effective. However, over decades of development, the ability of malware authors to invent technology and tactics has grown exponentially [3] . Research shows that in the past decade, traditional antivirus products have rarely succeeded in detecting unknown malware. Nowadays, most antivirus vendors have used machine learning to solve this problem and applied these techniques throughout the industry, for example using statistical characteristics like API calls [4] or N-grams [5]. Many of them rely on a wide range of domain knowledge for malware analysis and feature extraction, which is often time-consuming. Deep convolutional neural networks (CNN) based on the VGG16 architecture have recently been investigated for malware classification [6]. The malware classification problem was transformed into an image classification problem to be solved by CNN. The VGG16 model is the most popular CNN, which has been widely applied to image classification. Over the years, there has been a trend where the deeper the model is, the better performance the model can get. In 2014, at the ImageNet competition, the VGG16 model with a depth of 23 resulted in a top-1 classification error of 28.5%. In 2015, the Residual Network (ResNet) model with depth 168 resulted in a top-1

ISSN:2222-758X
e-ISSN:2789-7362

classification error of 24.1%. In 2016, the Inception V3 model with depth 159 resulted in a top-1 classification error of 21.8%. And finally, in 2017, the Xception model with 126 layers resulted in a top-1 classification error of 21.0% [7]. In this paper, we proposed a novel method to classify malware families based on the deep CNN with the Xception model. The Xception model used in this paper is a pre-trained model on ImageNet, released on Keras. Therefore, the popular transfer learning strategy (the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned) has been used to transfer the pre-trained Xception model on the malware classification task. The experiments results demonstrate that the validation accuracy of the Xception model on the Malimg Dataset [8] is higher 99.20% compared with the previous winner of the VGG model on malware image classification. Additionally, the proposed method reduces the chance of over-fitting compared with the VGG model due to the Global Average Pooling layer minimizes over-fitting by reducing the total number of parameters in the Xception model [9]. Furthermore, the proposed method uses the CNN model to extract the image features automatically, which is much less time-consuming than the methods that use manual feature engineering.

II. Related Work

The use of machine learning for automatically classifying malware families has been extensively studied in the working paper. Kolter and Maloof [10] extracted byte n-grams from Windows executables and trained several classifiers. They used a one-versus-all classification approach and combined the predictions of the individual classifiers. Shabtai et al. [11] Evaluated various settings of opcode n-gram sizes and classifiers. The authors concluded that the 2-gram opcodes outperformed the others and the use of byte n-grams appears to produce less accurate classifiers than using opcode n-grams. Some approaches based on the use of visualization techniques have been proposed to support malware analysis concerning feature extraction and pattern recognition of malware samples. Nataraj et al. [8] proposed a method for classifying malware represented as byteplot grayscale images using image processing techniques. Using Gabor filters to extract GIST descriptors from the byteplot grayscale images and then using a k-nearest neighbors (knn) classifier, they obtained an accuracy of 97.18% in a dataset consisting of 25 malware families, totaling 9458 malware samples. In the last few years, researchers have applied deep learning techniques to learn patterns in a set of features extracted from static and dynamic malware analysis to classify new samples. Kolosnjaji et al. [12] used a hierarchical feature extraction architecture that combines convolutional and recurrent neural network layers for malware classification using system call n-grams obtained from dynamic analysis. Their evaluation results achieved an average accuracy of 89.4% in a dataset containing 4753 malware samples from ten different families. Unlike previous work, our approach does not require any feature engineering, using raw pixel values of byteplot images as our underlying malware representation. Additionally, we employ knowledge transfer from a deep neural network trained for object detection tasks on a different dataset to discover good malware representations, improving the classification and detection results.

III. Methodology

The proposed method consists of the following basic steps:

ISSN:2222-758X
e-ISSN:2789-7362

1) Data Collection and Preprocessing, this step includes (decompiling or binary unpacking, Image construct as a grayscale image, classify the image according to Malware type) .
2) In the second step we convert the grayscale image to RGB and rescale it to a fixed resolution of 224×224 . Additionally, we subtract the mean RGB value computed on the ImageNet dataset from each pixel of the resulting $224 \times 224 \times 3$ images, as suggested by Krizhevsky et al. [13]
3) Build a deep convolutional neural network (DCNN) based on the xception architecture, transferring the parameters of xception convolutional layers to the convolutional layers of the DCNN model and replacing the last 1000 fully connected softmax layers at first by 26 fully connected softmax layers since we have 26 classes dataset and then apply Machin learning classifier algorithm.
4) Freeze the transferred convolutional layer's parameters and train the DCNN model to classify each sample into its malware family.
5) Malware Detection System, by using the trained model, the system will be able to classification and detection malware files from (malware origin file or malware image file), as it will be explained in. An overview of the entire method's pipeline is given in Fig. 1 .

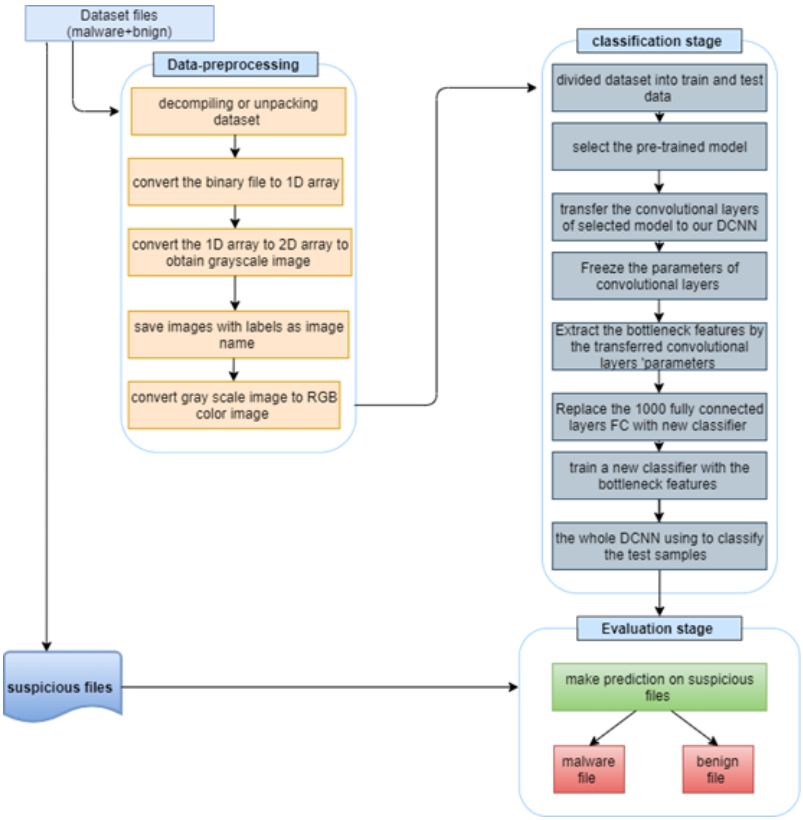

Figure 1: General diagram for malware detection and classification

ISSN:2222-758X
e-ISSN:2789-7362

A. Converting Binary to Image

The binary visualization method was initially proposed by Conti et al. [14] to represent binary data objects as grayscale images, where each byte corresponds to one image pixel color rendered as a grayscale (zero is black, 255 is white and other values are intermediate shades of gray). They presented a visual reverse engineering system arguing that visual analysis of binary data presented as grayscale graphical depictions helps distinguish structurally different regions of data and thus facilitates a wide range of analytic tasks such as fragment classification, file type identification, Location of regions of interest, and other tasks that require an understanding of the primitive data types. Later, Nataraj et al. [8] observed significant visual similarities in image texture for malware belonging to the same family, as shown in Fig. 2 , possibly explained by the common practice of reusing code to create new malware variants. To transform malware samples into binary images, a given malware binary is read as a vector of 8-bit unsigned integers and then organized into a 2D array, where the width is defined by the file size, based on empirical observations made by Nataraj et al. [8]. The height is allowed to vary depending on the width and the file size.

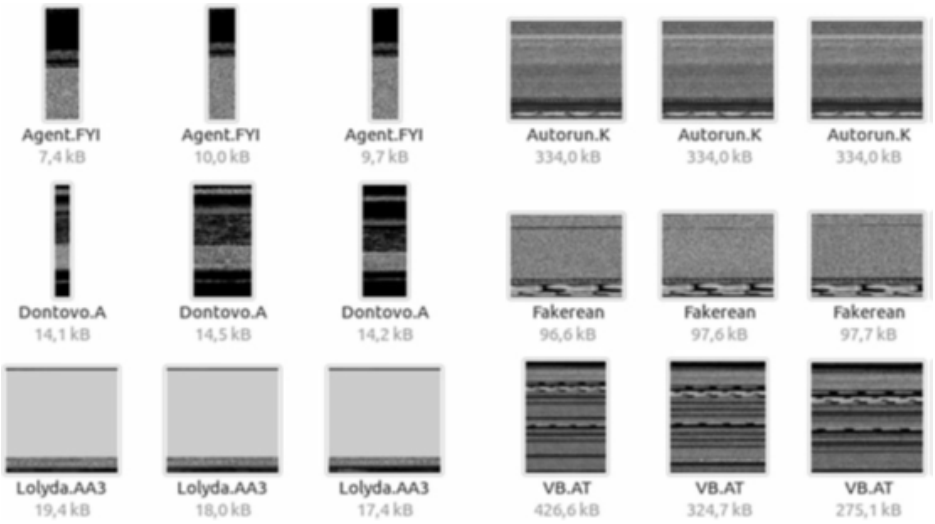

Figure 2: Binary visualization of malware samples from six different families

B. Deep Convolutional Neural Network (DCNN) Architecture

CNN [15] is a type of neural network that has proven to be very effective in different areas such as image recognition and classification and has been successfully applied in different fields including skin cancer detection, face recognition, and object detection. CNN architectures: a typical CNN architecture consists of an input layer, convolutional layers, RELU layers, pooling layers, fully connected layers:

INPUT Layer: this layer is responsible for the input of raw pixel values (image width, height, with three-colour channels R, G, B) of the image.

CONV layer: the input image is placed into a set of convolution filters, each of which activates certain features in the image.

ISSN:2222-758X
e-ISSN:2789-7362

RELU layer :achieves faster, more efficient training by mapping negative values to zero and maintaining positive values. Sometimes this is called activation because only activated features can be transferred to the next layer.

POOL layer: : downsampling on spatial dimensions (width and height) to reduce memory usage.

FC (fully connected) layer: the fully connected layer is to flatten the previous results and then receive the most basic neural network, Fig. 3 shows the structure of CNN. A special CNN topology **(GoogLeNet Architecture)**: The GoogLeNet architecture was introduced as GoogLeNet (Inception V1), later refined as Inception V2, and recently as Inception V3 [16] Fig. 4 show the architecture of the Google network. While Inception modules are conceptually convolutional feature extractors, they empirically appear to be capable of learning richer representations with fewer parameters. A traditional Convolutional layer attempts to learn filters in a 3D space, with 2 spatial dimensions (width and height) and a channel dimension. Thus, a single convolution kernel is tasked with simultaneously mapping cross-channel correlations and spatial correlations. The idea behind the Inception module is to make this process easier and more efficient by explicitly factoring it into a series of operations that would independently look at cross-channel Correlations and spatial correlations. The Xception architecture [17] is an extension of the Inception architecture which replaces the standard Inception modules with depthwise separable convolutions. Instead of partitioning input data into several compressed chunks, it maps the spatial correlations for each output channel separately and then performs a 1-1 depthwise convolution to capture cross-channel correlation. This is essentially equivalent to an existing operation known as a "depthwise separable convolution", which consists of a depthwise convolution (a spatial convolution performed independently for each channel) followed by a pointwise convolution (a 1-1 convolution across channels). We can think of this as looking for correlations across a 2D space first, followed by looking for correlations across a 1D space. Intuitively, this $2D + 1D$ mapping is easier to learn than a full 3D mapping. Xception slightly outperforms InceptionV3 on the ImageNet dataset, and vastly outperforms it on a larger image classification dataset with 17,000 classes. Most importantly, it has a similar number of parameters as Inception V3, implying a greater computational efficiency. Xception has 22,855,952 trainable parameters while Inception V3 has 23,626,728 trainable parameters, Fig. 5 shows the structure of the xception model.

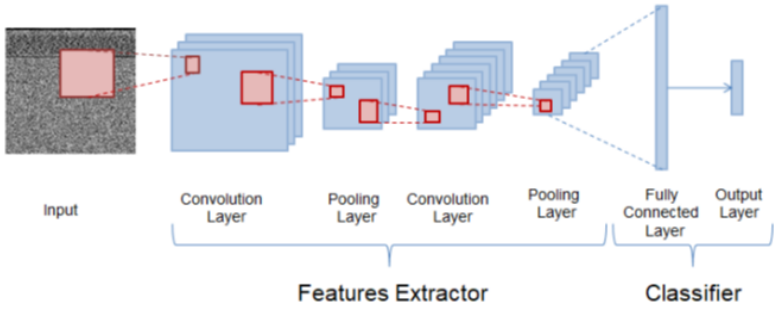

Figure 3: CNN architecture

ISSN:2222-758X
e-ISSN:2789-7362

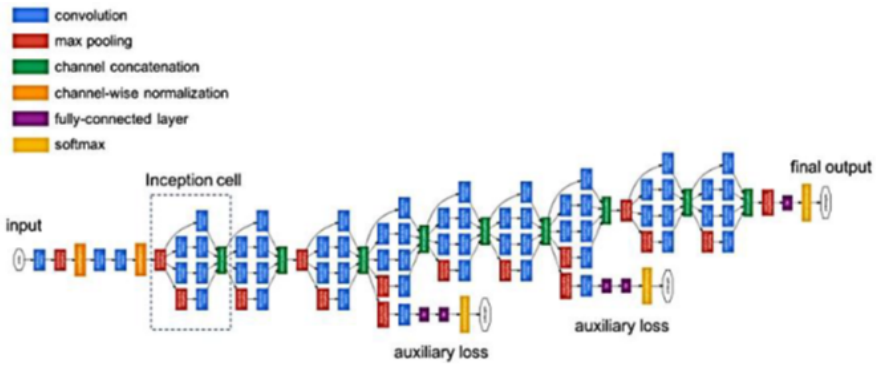

Figure 4: Google network architecture

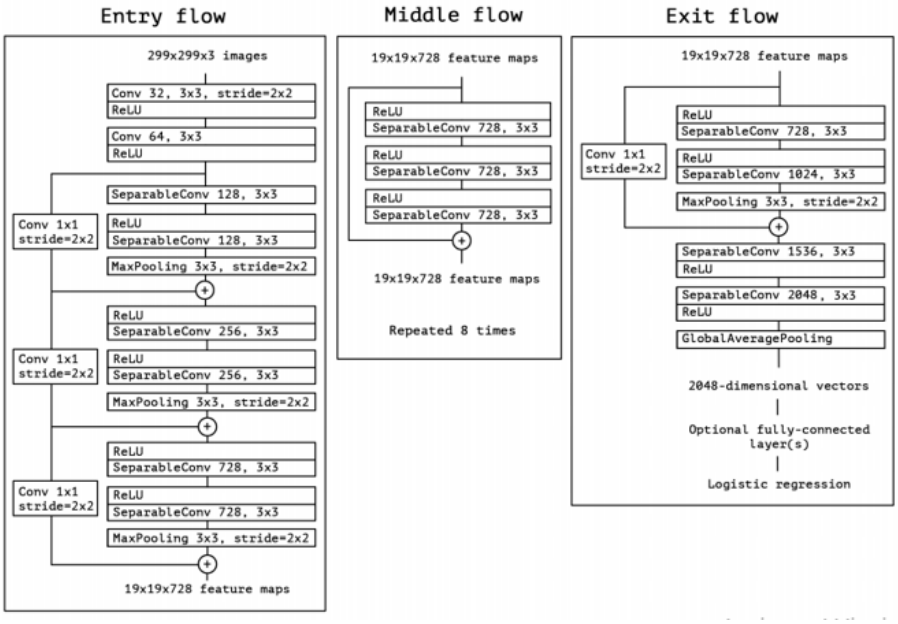

Figure 5: xception model implementation

C. Transfer Learning

Transfer learning consists of transferring the parameters of a neural network trained with one dataset and task to another problem with a different dataset and task [18]. Many deep neural networks trained on natural images exhibit a curious phenomenon in common: on the first layers they learn features that appear not to be specific to a particular dataset or task, but general in that they apply to many datasets and tasks. Features must eventually transition from general to specific by the last layers of the network. When the target dataset is significantly smaller than the base dataset, transfer learning can be a powerful tool to enable training a large target network without over-fitting. In this proposed transfer learning approach,

ISSN:2222-758X
e-ISSN:2789-7362

we have used xception as the base model, which is pre-trained for object detection tasks on the ImageNet dataset. The ImageNet is a public dataset containing 1.28 million natural images of 1,000 classes; we transferred convolutional layers of the xception model, which left frozen on the malware classification task. These layers can be seen as learned feature extraction layers. The activation maps generated by those learned feature extraction layers are usually called bottleneck features. Using the bottleneck features of our malware binary images as input to train different types of classifiers such as 26 fully connected softmax, and machine learning classifiers (SVM, KNN).

D. Top Classifier

In the proposed method, the last layer of the Xception was replaced by another classifier and we have evaluated three different classifiers for this task.

1) Softmax

   The softmax function [19] is widely used in deep learning architectures and consists in the generalization of the binary logistic regression to multiple classes. This function is particularly interesting because it provides an intuitive output with a probabilistic interpretation. The outcome of the function is a vector containing the probability for each class. Typically, the softmax is used for classification as the activation function on the last fully connected layer of CNNs.

2) k-Nearest Neighbor (kNN)

   The k-Nearest Neighbor (kNN) [19] classifier is one of the simplest and most popular supervised classifiers. It consists of classifying a sample based on the k nearest samples from the training set. Usually, the Euclidean distance function is used, but other distance functions may be chosen. The sample is then classified by the majority voting or other similar function among the k nearest samples. The advantage of kNN is that it is very simple, does not require an explicit training step and yet it is very effective for many applications, especially when the training set is large. The main drawbacks of this method are that

   - the method requires the distance computation to all samples from the training dataset, which makes it computationally heavy; and
   - the fact that it also requires a lot of memory since the whole dataset has to be loaded for comparison.

3) SVM

   Support Vector Machines (SVM) [19] is one of the most popular supervised classifiers and operates by finding the optimum hyperplane that best separates two classes. Originally designed for binary classification, it can be extended for multi-class problems by reducing one multi-class task to multiple binary classification problems using techniques known as one-versus-one or one-versus-all, among other methods. Although the original SVM is a linear classifier, it can be applied for non-linear problems by using kernel functions which nonlinearly map the feature vector to a new space. In the present work, we evaluate both the linear SVM, for the top classifier of the architecture.

IV. EXPERIMENTS AND RESULTS

1) **Implementation Details**

ISSN:2222-758X
e-ISSN:2789-7362

The proposed system is implemented during implementation and evaluation using PyCharm and the proposed system implemented on the Window 10 O.S 64-bit operating, and the Intel platform Core i7, CPU with 6 GB RAM.

2) **Dataset**

The malware dataset is gathered from Vision Research Lab and is called Malimg Dataset, which is created by [8] . The dataset consists of 25 malware families while the number of samples in each family is distinct and this dataset contains 9339 malware images in total. The samples in the Malimg dataset are all originally converted to image pixels as grayscale with a size of 64 by 64 in PNG file format. Using the database in the current form is not suitable to create a capable distinguishing system to detect the malware files, because it does not contain the benign files class. Therefore, the researcher added a new type of benign data to the database by collecting more than 500 files of the most used files type malicious files, and check them by using Kaspersky internet security to examine the records and isolate the benign files. After the benign files were collected, the benign files are converted into image form by identical use algorithms were used to create the original database.

3) **Validation Protocol**

To evaluate the proposed model performance we used a Scikit-learn's model selection train-test-split module, split the dataset into train sets with size =0.9 and test sets with the size =0.1, each set containing roughly the different proportions of the class labels (test each model on the data never trained) , fixing random-state = 43 to evaluated algorithms on the same subsets of the dataset, with stratified=dataframe.labels to split the dataset into train and test sets in a way that preserves the same proportions of samples in each class as observed in the original dataset and finally reported the classification accuracy, the accuracy and the execution time.

4) **The performance metric**

To train our two models we will use an unbalanced dataset (Malimg). Furthermore, accuracy is not the best metric to use when evaluating unbalanced datasets as it can be very misleading. However, for our comparative study we will use the following metrics: precision, recall, and F1 score and confusion matrix:

Precision (P): is the number of true positives predictions (Tp) divided by all true positive predictions (Tp) plus the number of false positives (Fp) .

$$P = \frac{Tp}{Tp + Fp} \tag{1}$$

Recall (R): is the number of true positives (Tp) divided by the number of true positives plus the number of false negatives (Fn)

$$R = \frac{Tp}{Tp + Fn} \tag{2}$$

F1 Score: the weighted average of precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

ISSN:2222-758X
e-ISSN:2789-7362

TABLE I
Malware Family Labelling

Malware Family	Label numbering
'Adialer.C'	0
Agent.FYI'	1
'Allaple.A'	2
'Allaple.L'	3
'Alueron.gen!J',	4
'Autorun.K'	5
'Benign',	6
'C2LOP.P'	7
'C2LOP.gen!g'	8
'Dialplatform.B'	9
'Dontovo.A'	10
Fakerean	11
Instantaccess	12
'Lolyda.AA1'	13
'Lolyda.AA2'	14
'Lolyda.AA3'	15
'Lolyda.AT'	16
'Malex.gen!J'	17
'Obfuscator.AD'	18
'Rbot!gen'	19
'Skintrim.N'	20
'Swizzor.gen!E'	21
'Swizzor.gen!I'	22
'VB.AT'	23
'Wintrim.BX',	24
'Yuner.A'	25

ISSN:2222-758X
e-ISSN:2789-7362

Macro averaged F1 Score: is the average of the individual F1 scores obtained for each class.

$$marco - F1 = \frac{1}{q}\sum_{i=1}^{q} F_1^i \tag{4}$$

Where q = number of classes in the dataset $F_1^i = F1$ score of class i .

Confusion Matrix: is a table showing the correct predictions and the incorrect types of predictions.

V. RESULTS AND DISCUSSION

For the corresponding multiclass classification problem, we attempt to classify the malware samples into their respective families, with the Windows benign set treated as an additional "Family." Since there are 25 malware families in the Malimg dataset, for this classification problem, we have 26 classes. As described, our method takes advantage of the transfer learning process to train our DCNN model on the malware classification task, we first use the Xception transferred convolutional layers to extract the bottleneck features of our binary images (The features called bottleneck features are extracted from the first layer up to the Global Average Pooling layer), in the second step we convert the grayscale image to RGB and rescale it to a fixed resolution of $299 \times 299 \times 3$ to match the input image size required by Xception model, Feature reduction was applied to the 2048-dimensional output feature vector of the last convolution layer which is then used as input to train three types of classifier algorithms: 26 fully connected softmax layer and machine learning classifier (SVM, KNN), The trained classifiers is stacked on top of the layers transferred from Xception model to build our DCNN model, which is used to classify the test set. This process intends to generate a set of features with a better degree of reparability, which could allow the top classifier to achieve a higher classification accuracy.

A. Classification by Softmax Algorithm

At this round of experiments, we classify our samples using a deep CNN architecture similar to the original Xception. The only difference is that we use 2 fully-connected softmax top layers instead of using the original 1000 fully-connected softmax layers. Our top layer has been trained with sparse categorical cross-entropy loss function and Adam optimizer for a limit of 10 epochs, the batch size has been initialized to 32. The weights have been initialized using glorot uniform approach [20] and the bias terms were initialized to zero. The results show that 98.06% accuracy was achieved.

B. Classification by SVM Algorithm

In the second experiment, we use the xception transferred convolutional layers but use an SVM Algorithm top layer instead of using the original 1000 fully-connected softmax layer, to classify images represented by bottle-neck features. For the SVM classifier, we use a linear kernel, where the parameter C has been initialized to =1.0, we obtained an accuracy of 99.20%.

C. Classification by KNN Algorithm

In the last experiments, we replace the last fully connected softmax layer of the Xception model with KNN classifier to classify images represented by bottle-neck features. For knn classifier, we use a k = 1, we obtained an accuracy of 98.39% in the third experiment.

ISSN:2222-758X
e-ISSN:2789-7362

D. Comparison of Models Performance:

To compare the performance of our experiments, we will use the following metrics already explained above. However, in previous sections, we reported results obtained with our deep CNN model using a softmax, SVM, and a KNN top layer. Our better result of 99.20% is achieved when using an SVM as a top layer, others performance metrics are summarized in the Table II

TABLE II
Comparison of Performance Metrics for Our Models

Method	Accuracy	Precision	Recall	F1 score
xception+2 fully connected layer	98.06%	98.06	98.06	98.06
Xception+svm	99.20%	99.20	99.20	99.20
Xception +knn	98.72%	98.72	98.72	98.72

Table III presents a comparison of the accuracy performance of our model with the literature:

TABLE III
A comparison of The Accuracy Performance of Our Model with The Literature

Method	Techniques	Accuracy
Saja Ali Abttan et a1.[21]	CNN Techniques (1C1D)	97.54%
Saja Ali Abttan et a1.[21]	VGG16 from scratch	91.72%
Bouhorma mohammed et a1.[22]	Transfer learning using VGG16	98%
Nataraj et al [8]	GIST+KNN	96.97%
Our model	Transfer learning of xception model	99.20%

VI. CONCLUSION

In this work, we propose a malware classification mechanism using binary malware images and deep learning techniques. We evaluated our approach on Xception model which is a pre-trained deep learning image recognition model based on transfer learning, using a Malimg dataset consisting of 9,339 samples from 25 malware families. Our approach does not require any feature engineering making it more flexible and effective in adapting to any future evolution in malware; it is also a very much less time-consuming approach. Using raw pixel values of byteplot images as our underlying malware representation. Moreover, we demonstrated that the knowledge obtained in the ImageNet classification task can be successfully transferred to malware classification tasks. The Xception model can achieve the highest validation accuracy than all other approaches, we obtaining an average accuracy of 98.62%, Compared with the VGG16 model, Xception model has less overfitting problem. This study can be considered as an introduction to many new experiments in the field of using transfer learning for malware classification.

ISSN:2222-758X
e-ISSN:2789-7362

REFERENCES

[1] R. Moir, "Defining Malware: FAQ" , Microsoft Wind. Serv. , 2003.
[2] Y. Dai, H. Li, Y. Qian, and X. Lu, "A Malware Classification Method Based on Memory Dump Grayscale Image" ,Digit. Investig. , vol. 27, pp. 30-37, 2018.
[3] "How Traditional Antivirus Works" , https://blogs.blackberry.com/en/2017/05/how-traditional-antivirus-works, accessed May 04, 2021.
[4] K. S. Han, I. K. Kim, and E. G. Im, "Malware Classification Methods Using API Sequence Characteristics" , in Proceedings of the International Conference on IT Convergence and Security, pp. 613-626, 2012.
[5] C. Liangboonprakong and O. Sornil, "Classification of Malware Families Based on N-Grams Sequential Pattern Features" , in 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA) , pp. 777-782, 2013.
[6] E. K. Kabanga and C. H. Kim, "Malware Images Classification Using Convolutional Neural Network" , J. Comput. Commun. , Vol. 6, No. 1, pp. 153-158, 2017.
[7] From https://keras. io/ja/applications/entatio, for individual Models, DocumModels, D, for individual, (n. d.) , Applications -Keras Documentation, Retrieved May 4, 2021, "Applications Keras Documentation" , https://keras.io/ja/applications/ , accessed May 04, 2021.
[8] 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, pp. 1-7, 2011.
[9] M. Lin, Q. Chen, and S. Yan, "Network in Network" , arXiv Prepr. arXiv1312. 4400, 2013.
[10] J. Z. Kolter and M. A. Maloof, "Learning to Detect and Classify Malicious Executables in The Wild" , J. Mach. Learn. Res. , Vol. 7, No. 12, 2006.
[11] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting Unknown Malicious Code by Applying Classification Techniques on Opcode Patterns" , Secur. Inform. , Vol. 1, No. 1, pp. 1-22, 2012.
[12] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep Learning for Classification of Malware System Call Sequences" , in Australasian Joint Conference on Artificial Intelligence, , pp. 137-149, 2016.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks" , Adv. Neural Inf. Process. Syst. , Vol. 25, pp. 1097-1105, 2012.
[14] G. Conti, E. Dean, M. Sinda, and B. Sangster, "Visual Reverse Engineering of Binary and Data Files" , in International Workshop on Visualization for Computer Security, pp. 1-17, 2008.
[15] A. Das, Das, A. (n.d.), CNN Architectures - CS60010: Deep Learning.CNN Architectures - CS60010, "Deep Learning" .
[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision" , in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826, 2016.
[17] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions" , in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251-1258, 2017.
[18] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable are Features in Deep Neural Networks ? " , arXiv Prepr. arXiv1411.1792, 2014.
[19] C. M. Bishop, "Pattern Recognition and Machine Learning" , springer, 2006.
[20] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks" , in Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249-256, 2010.
[21] S. A. Abttan, "Malware Detection and Classification Using Machine Learning Techniques" , pp. 1-17, 2018, [Online] , Available: https://pdfs.semanticscholar.org/5565/df617c90ff6477aa3d6584031d1a5622fe1a.pdf.
[22] B. Prima and M. Bouhorma, "Using Transfer Learning for Malware Classification" , Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. , Vol. 44, pp. 343-349, 2020.