USING TRANSFER LEARNING FOR MALWARE CLASSIFICATION

PRIMA Bouchaib [1], BOUHORMA Mohamed [1]

[1] Computer Science, Systems and Telecommunication Laboratory, Faculty of Sciences and Techniques, Abdelmalek Essaâdi University, Tangier 90000, Morocco -
bouchaib.prima@etu.uae.ac.ma
bouhorma@gmail.com

KEY WORDS: Cybersecurity, Malware, Machine Learning, Deep Learning, Transfer Learning, Convolutional Neural Network.

ABSTRACT:

In this paper, we propose a malware classification framework using transfer learning based on existing Deep Learning models that have been pre-trained on massive image datasets. In recent years there has been a significant increase in the number and variety of malwares, which amplifies the need to improve automatic detection and classification of the malwares. Nowadays, neural network methodology has reached a level that may exceed the limits of previous machine learning methods, such as Hidden Markov Models and Support Vector Machines (SVM). As a result, convolutional neural networks (CNNs) have shown superior performance compared to traditional learning techniques, specifically in tasks such as image classification. Motivated by this success, we propose a CNN-based architecture for malware classification. The malicious binary files are represented as grayscale images and a deep neural network is trained by freezing the pre-trained VGG16 layers on the ImageNet dataset and adapting the last fully connected layer to the malware family classification. Our evaluation results show that our approach is able to achieve an average of 98% accuracy for the MALIMG dataset.

1. INTRODUCTION

Malware and associated computer security threats have become more and more developed, and also malware developers have become more creative and use increasingly complex escape techniques (obfuscation, packers, cryptor, protector, Advanced Evasion Techniques (AET) and Network evasion) (Sibi Chakkaravarthy, Sangeetha, and Vaidehi 2019).
The latest Mcafe report indicates that the new PowerShell malwares increased by 689% in the 1st quarter of 2020 compared to the previous quarter, and the number of new macro malwares has increased by 412% in the first quarter of 2020.(McAfee Labs Threats Report, juillet 2020).

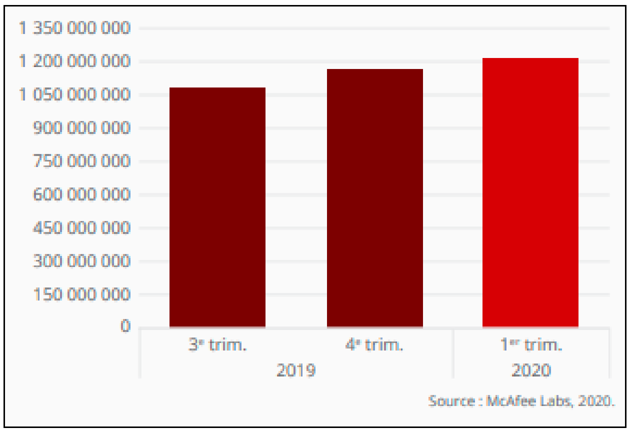


Figure 1. Augmentation of the total number of malwares (McAfee Labs, 2020)

This increase in the number of malware and the complexity of the escape techniques used, has led researchers to use detection and classification techniques based on machine learning, motivated by the success of this technique recently in the fields of computer vision and natural language processing.
The use of machine learning has shown favorable results compared to traditional malware analysis techniques that often require a lot of time and resources in feature engineering. Also, recently the Convolutional Neural Network (CNN) has been used for malware classification and this architecture has been able to achieve more satisfactory results in terms of accuracy. The principle of these techniques is presented in Section 2.
Based on the work of (Nataraj et al. 2011) who presented the malware presentation in grayscale images, we realize a malware classification system based on deep learning and we use transfer learning technique to train our CNN model based on VGG16 (Simonyan and Zisserman 2015) pre-trained model on larger dataset. Also, we make a comparative study of the different used techniques for malware classification.

We adapt VGG16 pre-trained model to make a malware classification and we make a comparative study of the obtained results with the literature, we prove that the transfer learning realizes a superior performance for malware classification then training our deep learning model from scratch.

2. RELATED WORK

In this section, we present the progress of the research as well as the techniques used to detect and classify malwares.

2.1 Static and Dynamic analysis

The objective of the malware analysis is to study the behavior and structure of malware, and there are two types: static analysis and dynamic analysis.

- **Static Analysis :**

The static analysis is performed without executing the malware, for Windows portable executable (PE) files we can proceed in two ways, either based on the binary file or on the disassembled malware program. This method of reverse engineering can be done on PE files executable by several tools the most used are: IDA Pro and Radare.

- **Dynamic Analysis :**

The dynamic analysis is performed by executing the malware on a testing environment (Sandbox) where we can analyse its behavior and have all traces made by this malware. This analysis is usually used if we were not able to collect much information about the malware by static analysis due to the complex obfuscation used by the malware developer or can be used as a complementary analysis to extract more features.
This scan should be performed on a completely isolated environment to avoid impacting our system, there are several environments to use, the most well-known is Cuckoo Sandbox.

(Talukder 2020; Sibi Chakkaravarthy, Sangeetha, and Vaidehi 2019) they summarize the tools used for each type of analysis and the extracted information.

2.2 Methods based on Machine Learning

The classification and detection of malware using Machine Learning (ML) is based on the following steps:
1- Features extraction.
2- Features selection.
3- Classification algorithm.
The work of (Ahmadi et al. 2016) is focused on extracting and selecting a new set of features from binary files and disassembled files to effectively represent malware samples.
Once the features are extracted and selected, they will be used to train the malware classification model or malware detection in case of binary classification (malicious or benign file) using a dataset of benign file features.(Ranveer and Hiray 2015)
There are several works that have performed the malware classification based on machine learning (ML) method such as:
(Nataraj et al. 2011) after presenting binary malware files as grayscale images, they performed a classification of the images based on GIST as features and they used machine learning algorithm k-nearest neighbors with Euclidean distance for malware classification.
(Kong and Yan 2013) based on the features (function call graphs) extracted from the malware they calculate the similarity of the two malwares using SVM, KNN.
(Abou-Assaleh et al. 2004) in this work, they used text classification techniques based on n-grams (is a subsequence of n elements built from a sequence of text), extracted from the signatures of malware, and they performed the KNN algorithm to perform the classification.

2.3 Methods based on Deep Learning

The malware visualization has successfully introduces deep convolutional neural networks into malware classification problems.
(Xiao et al. 2020) After they displayed the binary malware as entropy graphs they used deep learning to do feature extraction automatically and then used SVM to classify the malware based on the extracted features.
(Gibert et al. 2019) Based on the presentation of malware as an image, the following work presents a convolutional neural network (CNN) composed of three convolution layers followed by a fully-connected layer used for the classification of malware. They made a comparative study to prove that CNN has better results than KNN.

To resume, the methods based on traditional machine learning use a high computational cost because they often have to define and extract in advance a group of features and are not adapted for processing massive data. On the other hand the Deep learning automates the feature extracting and selecting, avoids the high computational cost. However, the literature has proved that the Deep Learning methods are more performant than the Machine Learning methods in term of accuracy.

3. METHODOLOGY

In this section we discuss the dataset and implementation details of our proposed models.

3.1 Visualizing Malware as an Image

Our work is based on the visualization of malware as an image, this approach initiated by (Nataraj et al. 2011) allowing to read a given malware binary as a vector of 8 bit unsigned integers and then organized into a 2D array. Finally this can be visualized as a gray scale image in the range [0,255] (0: black, 255: white).

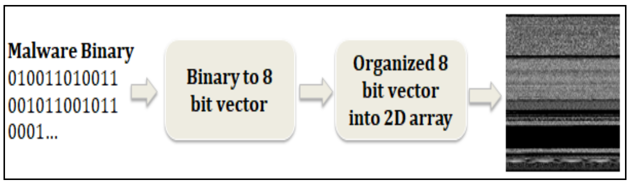


Figure 2. Visualizing malware as a grayscale image process

This presentation allows us to visualize malware belonging to the same family with a very similar image.
However, this malware visualization is based on the binary code, so if a malware developer is going to create a new malware by modifying the code of an old malware, with this approach the new malware will be visualised with a very similar image. Then we can use our classification model (CNN) presented later to easily classify it into the same family.

3.2 Dataset

The MalImg dataset was provided by (Nataraj et al. 2011) contains 9435 grayscale images of malwares packed with UPX, collected from 25 families:

Family	Family Name	Number of samples
Worm	Allaple.L	1591
Worm	Allaple.A	2949
Worm	Yuner.A	800
PWS	Lolyda.AA 1	213
PWS	Lolyda.AA 2	184
PWS	Lolyda.AA 3	123
Trojan	C2Lop.P	146
Trojan	C2Lop.gen!G	200
Dialer	Instantaccess	431
Trojan Downloader	Swizzor.gen!I	132
Trojan Downloader	Swizzor.gen!E	128
Worm	VB.AT	408
Rogue	Fakerean	381
Trojan	Alueron.gen!J	198
Trojan	Malex.gen!J	136
PWS	Lolyda.AT	159
Dialer	Adialer.C	125
Trojan Downloader	Wintrim.BX	97
Dialer	Dialplatform.B	177
Trojan Downloader	Dontovo.A	162
Trojan Downloader	Obfuscator.AD	142
Backdoor	Agent.FYI	116
Worm:AutoIT	Autorun.K	106
Backdoor	Rbot!gen	158
Trojan	Skintrim.N	80

Table 1. MalImg: Distribution of Samples

After analysing the number of samples of this dataset, we can notice that the MALIMG datatest is quite unbalanced: more than 30% of the images belong to class: Allaple.A and 17% to class : Allaple.L!

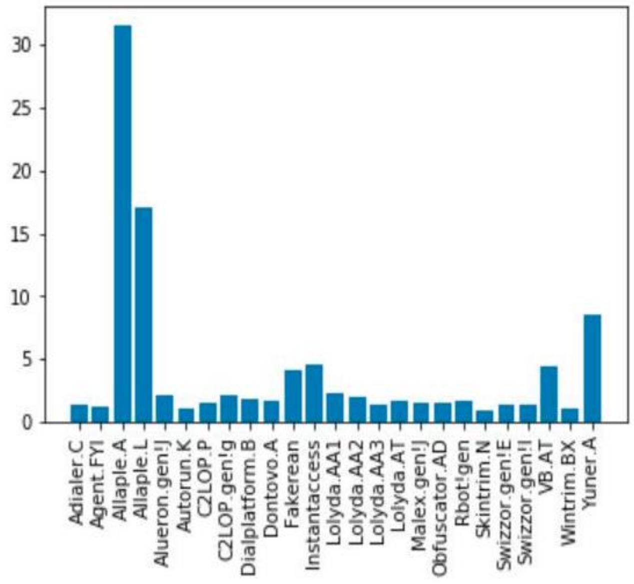


Figure 3. MalImg: Unbalanced dataset

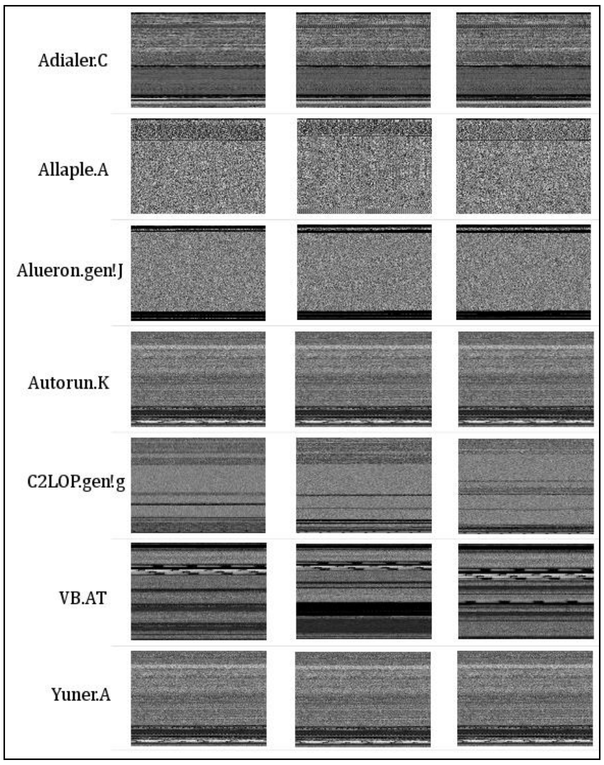


Figure 4. shows the representation of the malware samples, belonging to twenty-five different families as gray-scale images. It can be observed that the images of malware belonging to the same family are very similar, and they are different from other families.

3.3 Transfer Learning for Malware Classification

The general structure of a CNN is the combination of two components: The feature extractor in the first stage and the classifier:

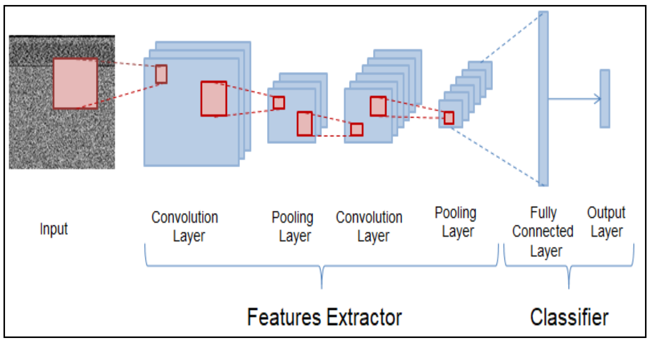


Figure 5. CNN Architecture

The transfer learning is to replace the Classifier component of the pre-trained model, VGG16 in our case, from VGG family (Visual Geometry Group at University of Oxford) with a customized classifier to resolve our classification problem.

In practice we replace the last layer of the VGG16 (Figure 6), which takes a probability for each of the 1000 classes in the ImageNet (Krizhevsky, Sutskever, and Hinton 2012) and replaces it with a Fully Connected layer that takes 25 probabilities corresponding to 25 families of malwares. This way, we use all the knowledge that VGG16 has trained on the ImageNet dataset and apply it to our malware classification problem.

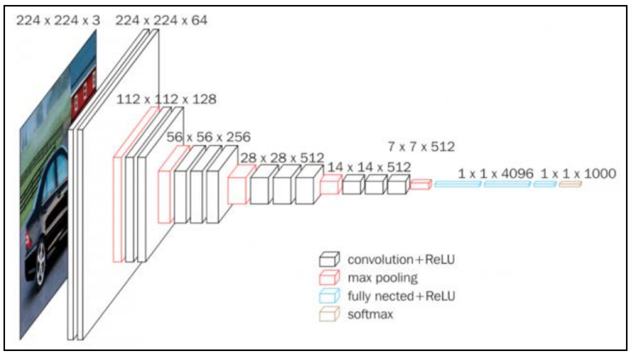


Figure 6. VGG16 Architecture (Simonyan and Zisserman 2015)

The VGG16 network architecture (Simonyan and Zisserman 2015) is shown in figure 6.

The input layer is an RGB image of fixed size 224 × 224, then the image is passed through a stack of convolutional layers, where the size of the filters used is 3 x 3 with a stride 1, and it always uses the same padding and maxpool layer of 2x2 filter of stride 2.

Finally for classification, it has 2 fully connected layers followed by a softmax for output.

The 16 in VGG16 refers to it has 16 layers in total:
- 5 convolutional layers,
- 5 max pooling layers,
- 3 fully-connected layers
- output layer (softmax)

3.4 Our proposed models

As explained above, we propose a CNN model for malware classification based on the pre-trained model VGG16, using transfer learning (Figure 8). And we make a performance comparison with a second CNN model (Figure 7) trained from scratch.

The input of our network is a malicious program represented as a grayscale image, and the output is the predicted class of the malware sample.

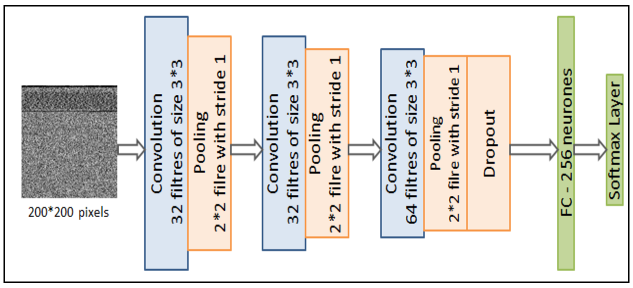


Figure 7. Our proposed CNN model for classification of malware represented as grayscale images.

For our first proposed architecture (Figure 7), we used three convolutional layers where the size of the filters used is 3x3 that scans the whole images and create a feature map to predict the class probabilities for each feature.

After each convolution layer we used a max pooling layer of 2x2 filters to scales down the amount of information generated for each feature and maintains only the most essential information.

At the end, the generated feature maps are flattened and combined to be used as input of the following fully connected layer composed of 256 neurons. Lastly, the output of the fully connected layer passes to a Softmax layer to classify the binary malware into its corresponding family.

To prevent overfitting during the training phase, we employed one dropout layer (Srivastava et al. 2014) to ignoring units of certain set of neurons which is chosen at random.

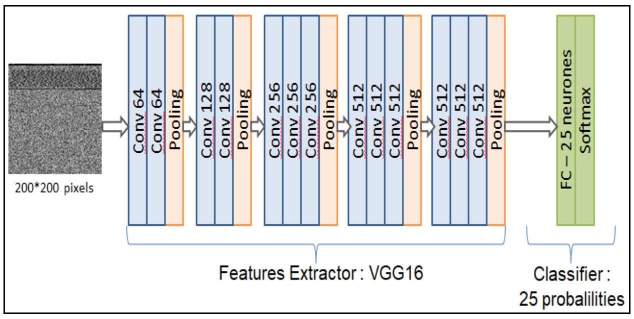


Figure 8. Our proposed CNN model for classification of malware represented as grayscale images using VGG16 as features extractor connected with optimised classifier for 25 families of malwares.

In this second model we customize the VGG16 architecture to our classification problem by adding a fully-connected layer containing 25 neurones corresponding to 25 malware families, instead of final fully connected layer (intended for 1000 classes).

The objective of this architecture is to employ the initial weights of pre-trained CNN of natural images (ImageNet dataset) to classify the binary malwares.

3.5 K-fold cross validation

To evaluate the generalization performance of our models we used K-fold cross validation. The dataset is divided into K equal size folds. Of the K subsamples, a single subsample is retained as the validation data for testing the model and the remaining subsamples are used as training data. This procedure is repeated as many times as there are folds, with each of the K folds used exactly once as the validation data.

3.6 The performance metric

To train our two models we will use an unbalanced dataset (Malimg). Furthermore, the 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 (T_P) divided by all true positive predictions (T_P) plus the number of false positives (F_P).

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

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

$$R=\frac{T_p}{T_p+F_n}$$

F1 Score: the weighted average of precision and recall.

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

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

$$macro\_F1=\frac{1}{q}\sum_{i=1}^{q} F_1^i$$

Where q = number of classes in the dataset
F_1^i = F_1 score of classe i

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

4. RESULTS AND DISCUSSION

We performed two different experiments and we made a comparative study of the obtained results. We present in this section the performed experiments and we discuss the results.

After several experiments we have optimized the hyper-parameters for the two proposed models (batch-size, epochs, and number of folds) to achieve the best performance.

4.1 Experiment 1

To train our first CNN model (Figure 7) using Malimg dataset we used the Cross-Validation algorithm (defined above) with 10 Folds and 40 epochs, and we downsampled the images to a fixed size. The size of the new images was set to 200*200 pixels.

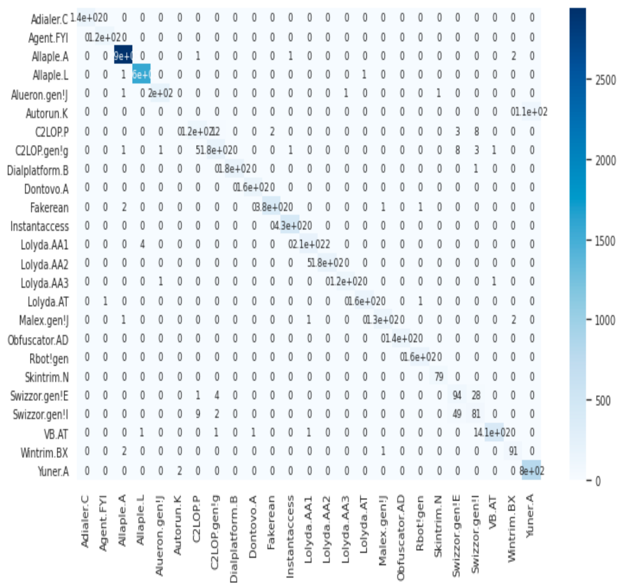
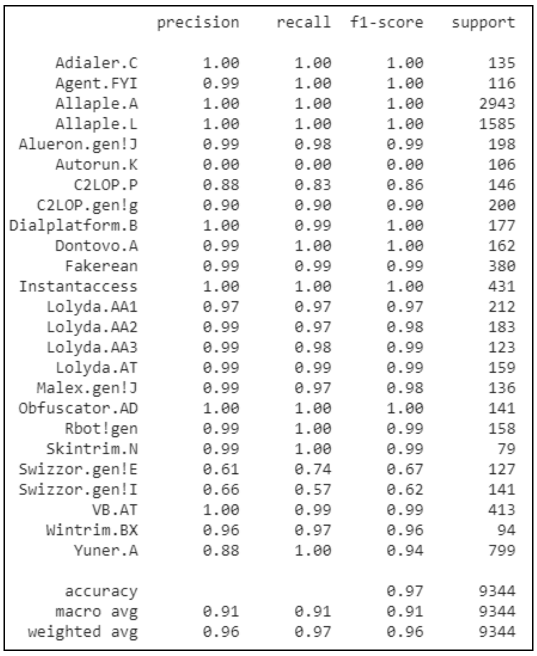


Figure 9. Confusion matrix for 10-fold cross validation of CNN model with 3 convolutional layers connected to one fully connected layer.


```
                precision    recall  f1-score   support

     Adialer.C       1.00      1.00      1.00       135
     Agent.FYI       0.99      1.00      1.00       116
     Allaple.A       1.00      1.00      1.00      2943
     Allaple.L       1.00      1.00      1.00      1585
 Alueron.gen!J       0.99      0.98      0.99       198
     Autorun.K       0.00      0.00      0.00       106
       C2LOP.P       0.88      0.83      0.86       146
   C2LOP.gen!g       0.90      0.90      0.90       200
Dialplatform.B       1.00      0.99      1.00       177
     Dontovo.A       0.99      1.00      1.00       162
      Fakerean       0.99      0.99      0.99       380
 Instantaccess       1.00      1.00      1.00       431
     Lolyda.AA1      0.97      0.97      0.97       212
     Lolyda.AA2      0.99      0.97      0.98       183
     Lolyda.AA3      0.99      0.98      0.99       123
     Lolyda.AT       0.99      0.99      0.99       159
   Malex.gen!J       0.99      0.97      0.98       136
 Obfuscator.AD       1.00      1.00      1.00       141
      Rbot!gen       0.99      1.00      0.99       158
     Skintrim.N      0.99      1.00      0.99        79
  Swizzor.gen!E      0.61      0.74      0.67       127
  Swizzor.gen!I      0.66      0.57      0.62       141
        VB.AT        1.00      0.99      0.99       413
    Wintrim.BX       0.96      0.97      0.96        94
       Yuner.A       0.88      1.00      0.94       799

      accuracy                           0.97      9344
     macro avg       0.91      0.91      0.91      9344
  weighted avg       0.96      0.97      0.96      9344
```

Figure 10. Performance metrics for CNN model with 2 convolutional layers connected to one fully connected layer.

According to the obtained results, this first model is very powerful for all malwares given in input except the following families:
The Autorun.K family is classified incorrectly as Yunner.A, as you can see in the (figure 10), the precision of the Autorun.K family is 0. That is because these two families are very similar and are indistinguishable by the human eye (Figure 11).

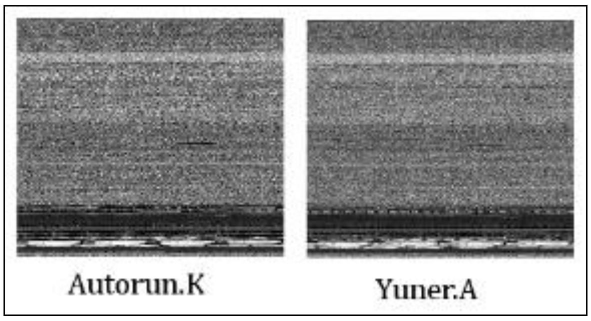


Figure 11. Autorun.K and Yuner.A samples

Also, the model can't distinguish correctly the samples belonging to the same family: Swizzer.genE and Switzer.gen! I. (precision 0.61 and 0.66).

4.2 Experiment 2

To train our second model (Figure 8) (based on the transfer learning) using Malimg dataset, we used 5 Folds cross validation and 10 epochs, and we downsampled the images to a fixed size. The size of the new images was set to 200*200 pixels. This model has proven the best performance by using only 90% of dataset.

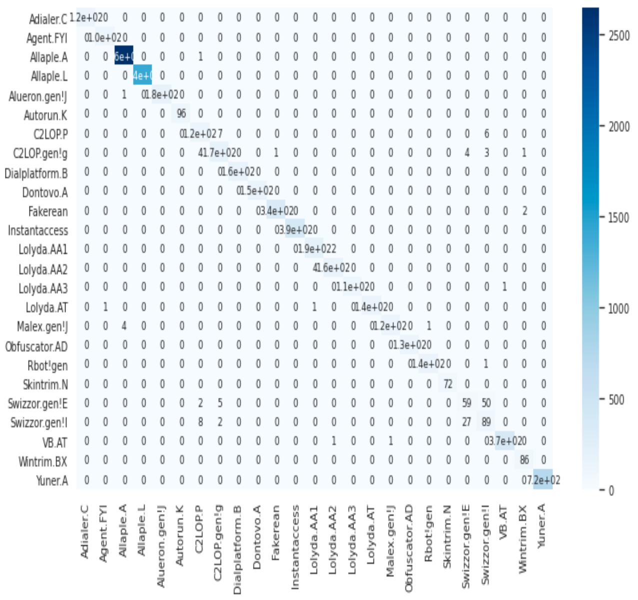
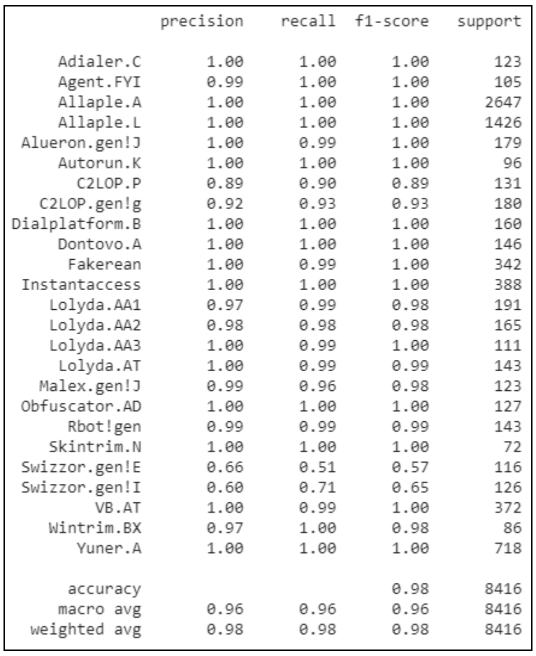


Figure 12. Confusion matrix for CNN model for malware classification using VGG16 as features extractor connected with optimised classifier for 25 families of malwares.


```
                precision    recall  f1-score   support

     Adialer.C       1.00      1.00      1.00       123
     Agent.FYI       0.99      1.00      1.00       105
     Allaple.A       1.00      1.00      1.00      2647
     Allaple.L       1.00      1.00      1.00      1426
 Alueron.gen!J       1.00      0.99      1.00       179
     Autorun.K       1.00      1.00      1.00        96
       C2LOP.P       0.89      0.90      0.89       131
   C2LOP.gen!g       0.92      0.93      0.93       180
Dialplatform.B       1.00      1.00      1.00       160
     Dontovo.A       1.00      1.00      1.00       146
      Fakerean       1.00      0.99      1.00       342
 Instantaccess       1.00      1.00      1.00       388
    Lolyda.AA1       0.97      0.99      0.98       191
    Lolyda.AA2       0.98      0.98      0.98       165
    Lolyda.AA3       1.00      0.99      1.00       111
     Lolyda.AT       1.00      0.99      0.99       143
   Malex.gen!J       0.99      0.96      0.98       123
 Obfuscator.AD       1.00      1.00      1.00       127
      Rbot!gen       0.99      0.99      0.99       143
    Skintrim.N       1.00      1.00      1.00        72
  Swizzor.gen!E      0.66      0.51      0.57       116
  Swizzor.gen!I      0.60      0.71      0.65       126
         VB.AT       1.00      0.99      1.00       372
    Wintrim.BX       0.97      1.00      0.98        86
       Yuner.A       1.00      1.00      1.00       718

      accuracy                           0.98      8416
     macro avg       0.96      0.96      0.96      8416
  weighted avg       0.98      0.98      0.98      8416
```

Figure 13. Performance metrics for CNN model for malware classification using transfer learning.

Compared to the first model, this CNN model based on the VGG16 architecture classified correctly 96 samples of Autorun.K with a precision of 1 (as you can see on the figure 13) and in the confusion matrix (Figure 12).

Concerning the samples belonging to the same family Swizzer.genE and Switzer.gen! I, this model is also not precise (precision 0.48 and 0.53).

4.3 Comparison of models performance:

To compare the performance of our two models, we will use the following metrics already explained above. However, we obtain an overall classification accuracy of 97% for the CNN model with the simple architecture and trained from scratch, which represents a significant decline from the VGG16 model accuracy of 98%.

The others performance metrics are summarized in the following table.

Method	Accuracy	Precision	Recall	F1 score
CNN	0.97	0.91	0.91	0.91
Using VGG16	0.98	0.95	0.95	0.95

Table 2. Comparison of performance metrics for our models

The following table present a comparison of accuracy performance of our two models with the literature.

Method	Technique	Accuracy
(Nataraj et al. 2011)	GIST + KNN	96.97%
(Gibert et al. 2019)	CNN	97.5%
(Yue 2017)	Fine-tuning VGG19	97.32%
Our model 1	CNN	97%
Our model 2	Transfer learning using VGG16	98%

Table 3. Comparison of accuracy performance

CONCLUSION

In this paper, we propose an image-based malware classification system, using a pre-trained deep learning image recognition model. We compared these image-based deep learning (DL) results to a simpler convolutional neural network (CNN) approach trained from scratch. We carried out two experiments using the same dataset with the same image sizes.

Our experiments, has proven that the model based on the transfer learning results are particularly impressive with high accuracy. So we can deduce that the transfer learning technique can be used for the classification of malwares.

This study can be considered as an introduction to many new experiments in the field of using transfer learning for malware classification.

REFERENCES

Abou-Assaleh, Tony, Nick Cercone, Vlado Kesˇelj, and Ray Sweidan, 2004. Detection of New Malicious Code Using N-Grams Signatures. Second Annual Conference on Privacy, Security and Trust.

Ahmadi, Mansour, Dmitry Ulyanov, Stanislav Semenov, Mikhail Trofimov, and Giorgio Giacinto, 2016. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. *In* Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy - CODASPY ’16 Pp. 183–194. New Orleans, Louisiana, USA: ACM Press. http://dl.acm.org/citation.cfm?doid=2857705.2857713.

Gibert, Daniel, Carles Mateu, Jordi Planes, and Ramon Vicens 2019. Using Convolutional Neural Networks for Classification of Malware Represented as Images. Journal of Computer Virology and Hacking Techniques 15(1): 15–28.

Kong, Deguang, and Guanhua Yan, 2013. Discriminant Malware Distance Learning on Structural Information for Automated Malware Classification. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 9.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton 2012 ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM 60(6): 84–90.

McAfee Labs Covid-19 threats report, juillet 2020. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-july-2020.pdf.

Nataraj, L., S. Karthikeyan, G. Jacob, and B. S. Manjunath, 2011. Malware Images: Visualization and Automatic Classification.ACMPress. http://dl.acm.org/citation.cfm?doid=2016904.2016908.

Ranveer, Smita, and Swapnaja Hiray, 2015. Comparative Analysis of Feature Extraction Methods of Malware Detection. International Journal of Computer Applications 120(5): 1–7.

Sibi Chakkaravarthy, S., D. Sangeetha, and V. Vaidehi 2019. A Survey on Malware Analysis and Mitigation Techniques. Computer Science Review 32: 1–23.

Simonyan, Karen, and Andrew Zisserman 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv:1409.1556 [Cs]. http://arxiv.org/abs/1409.1556, accessed September 13, 2020.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting.

Talukder, Sajedul, 2020. Tools and Techniques for Malware Detection and Analysis. ArXiv:2002.06819 [Cs]. http://arxiv.org/abs/2002.06819, accessed July 16, 2020.

Xiao, Guoqing, Jingning Li, Yuedan Chen, and Kenli Li 2020. MalFCS: An Effective Malware Classification Framework with Automated Feature Extraction Based on Deep Convolutional Neural Networks. Journal of Parallel and Distributed Computing 141: 49–58.

Yue, Songqing, 2017. Imbalanced Malware Images Classification: A CNN Based Approach. ArXiv:1708.08042 [Cs, Stat]. http://arxiv.org/abs/1708.08042.