Enhancing Malware Family Classification in the Microsoft Challenge Dataset via Transfer Learning

Marcelo Invert Palma Salas
marcelopalma@ic.unicamp.br
University of Campinas
Cidade Universitária, Campinas, São Paulo, Brazil

Paulo Lício de Geus
pgeus@unicamp.br
University of Campinas
Cidade Universitária, Campinas, São Paulo, Brazil

Marcus Felipe Botacin
botacin@tamu.edu
Texas A&M University
College Station, Texas, USA

ABSTRACT

In recent years, malware developers have introduced new and advanced protection techniques against conventional signature-based and heuristic-based malware analysis techniques to avoid detection and removal by conventional antivirus. With the progress of deep learning, techniques such as Convolutional Neural Networks (CNN) are useful to detect the global structure of the code and to be able to decipher the patterns in the binary code datasets converted to RGB or grayscale images. This article takes advantage of the spatial structure of imaged malware, using a series of pre-trained Imagenet convolutions to generate feature maps that learn how to recognize and group malicious code into malware families. This research added a customized neural network on top of eight pre-trained networks (Xception, VGG16, VVG19, ResNet50, InceptionV3, MobileNet, MobileNetV2, and DenseNet169) to classify 10868 malware samples from the Microsoft Malware Classification Challenge dataset, achieving results close to 99% through the use of parameter adjustments and increasing the size of the dataset in order to generalize the model and reduce the risk of overfitting for malware that uses evasion techniques against classification.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; • **Computing methodologies → Neural networks**.

KEYWORDS

malware classification, Convolutional Neural Networks, transfer learning, Xception, VGG16, VVG19, ResNet50, InceptionV3, MobileNet, MobileNetV2, DenseNet169

ACM Reference Format:
Marcelo Invert Palma Salas, Paulo Lício de Geus, and Marcus Felipe Botacin. 2023. Enhancing Malware Family Classification in the Microsoft Challenge Dataset via Transfer Learning. In *Proceedings of 12th LatinAmerican Symposium on Dependable Computing (LADC 2023).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

LADC 2023, October 16–20, 2023, La Paz, Bolivia

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In the history of computing, the emergence and evolution of malware have been topics of great interest and study. From the first incidents of malware [6], [24] to the development of techniques to combat it [22], the term malware has been used to describe any type of software designed to harm or compromise computer systems without the consent of the user.

The term "malware" is a combination of the words "malicious" and "software" (malicious software). It was coined by Yisrael Radai on July 4, 1990, in a public posting in which he wrote[16]: *"Trojans constitute only a very small percentage of malware (a word I just coined for trojans, viruses, worms, etc.)"*. Since then, the term has become widespread and is widely used to refer to all types of malicious software, such as viruses, worms, Trojan horses, ransomware, spyware, botnets, adware, and others.

The exponential growth of malware [1] has posed significant challenges for researchers. As attackers use obfuscation techniques, polymorphism, metamorphism, and encrypted communication to modify and camouflage the structure of these malicious codes, conventional techniques such as signature-based and heuristic-based analysis become less effective. Given this situation, there is a need for more sophisticated analysis methods that can decipher the patterns and characteristics of the malware.

With the progress of deep learning, techniques such as Convolutional Neural Networks [14] (CNN) have proven to be powerful tools for deciphering complex patterns in large image data sets. A CNN is a type of neural network that is characterized by its ability to perform convolutional operations, i.e., mathematical operations applied to images, with the main objective of learning filters that detect specific patterns in images. As the Convolutional Neural Network processes an image, it generates a set of feature maps that indicate where the feature sought by each filter has been detected.

The advancement of deep learning has made it possible to use CNN to classify malware using binary code converted to RGB or grayscale images. This research uses the capacity of CNN networks to learn the global structure of the code and decipher the patterns present in the datasets generated from these images.

We also used pre-trained network architectures (Xception, VGG16, VVG19, ResNet50, InceptionV3, MobileNet, MobileNetV2, and DenseNet169) that participated in the ImageNet Large Scale Visual Recognition Challenge [21] to train modified models for malware classification. The transfer learning technique allowed us to take advantage of the knowledge learned by the models previously trained in Imagenet and adjust these models to the task of classifying malware families, improving efficiency and accuracy even against the evasion techniques used by attackers and significant imbalances in

the classes of malware families, i.e., improve the classification of minority classes through transfer learning.

To analyze our approach, we used the Microsoft malware classification challenge [20] dataset, consisting of 10868 binaries/samples. After conversion from byte files to PNG files representative of the malware samples, a set of pre-trained CNN network models were selected to classify the dataset into nine malware families. The results show that the use of the MobileNet network architecture has an accuracy of 98.41% with a log loss of 0.08078.

The rest of the article continues: Section 2 describes the state of the art in malware classification through convolutional networks; Section 3 describes the Convolutional Neural Networks, the transfer learnings technique and the pre-trained CNN network architectures used. The methodology used and results of the research are exposed in Section 4. The main contributions and future work are described in Section 5.

2 RELATED WORK

Malware classification is a critical task in computer security since it allows the identification and mitigation of potential cybersecurity threats. Convolutional Neural Networks (CNN) have proven to be an effective tool in malware detection, especially when combined with the transfer learning technique. Gibert et. al. [7] , [8] propose a deep learning approach for malware classification into families based on a set of patterns extracted from their visualization as images using the benchmark Microsoft Malware Classification Challenge dataset (BIG 2015). The results obtained in both approaches, 98.28% and 97.50% respectively, under validation by 5-fold cross-validation and 10-fold cross-validation respectively, demonstrate the effectiveness of CNNs in malware classification and offer interesting prospects for improvement of detection techniques and classification of computer threats.

A new approach was proposed by Hemalatha et. al. [10] through the use of a reweighted class-balanced loss function in the final classification layer in a model based on the DenseNet architecture to achieve performance improvements in classifying malware by handling imbalanced data issues. The approach achieved 98.46% for the Microsoft BIG 2015 dataset. A similar approach was used by Wang et. al. [28], using a set of layers as the Depthwise Efficient Attention Module (DEAM) combined with a DenseNet architecture with grayscale images transformed from malware, achieving 98.5% in the Microsoft BIG 2015 dataset.

The authors in [12] convert malware binary into grayscale images and use two datasets to validate their proposal: Maling and Microsoft Big 2015. The experiments were run on two architectures. The implementation of a GIST feature extractor with an SVM classifier and the implementation of an M-CNN neural network. The authors hardly described the training results, obtaining 93.23% (GIST+SVM), 98.52% (M-CNN) for the Maling dataset, and 98.99% in M-CNN. All results were not tested on a test dataset.

The authors in [13] used CNN-BiLSTM (CNN+RNN) with the class balance sampling technique in order to classify malware from the Microsoft Big 2015 dataset. This combination of techniques allowed them to address the classification problem without the need for an extensive feature engineering process, getting 98.20%.

In [28], the authors propose the use of the DenseNet architecture with the DEAM service module. The DEAM module manages to reduce the number of parameters and improve the robustness of the model. The results obtained in Microsoft Big 2015 was 97.30%.

In recent years, research has focused on the use of data-based augmentation methods such as CycleGan. Tekerek et. al. [27] proposed a process for transforming binary files into RGB and grayscale images called B2IMG. In order to reduce the class imbalance in malware classification, the authors used CycloGan on top of the DenseNet architecture and achieved 99.86% for RGB images and 99.76% for grayscale images on the Microsoft BIG 2015 dataset.

In [4] the authors implement a model on the DenseNet architecture with the bicubic interpolation algorithm to reconstruct the generated malware images to solve the problem of image size imbalance and use the CycleGAN model for data augmentation to balance the number of samples among malware families and build. Experimental results show that the system reached 99.76% and 99.62% accuracy for RGB and grayscale, respectively, for the Microsoft BIG 2015 dataset.

Class imbalance in malware classification is a common challenge, where some malware families may be underrepresented compared to others. The use of imaging techniques in the context of malware classification does not accurately represent malicious code transformed into RGB or grayscale images. In addition, synthetic images are introduced that may not capture the essential characteristics of the malware and could even add noise or irrelevant information, thus generating overfitting. Additionally, it is perceived that some research does not use test datasets and presents its results based on the training dataset, leading to the implementation of invalid experiments.

The contributions of this research are the following:

(1) The PNG (Portable Network Graphics) format was used instead of JPG (Joint Photographic Experts Group) because it uses lossless compression, i.e., preserving as much visual information as possible to capture features relevant to classification problems.
(2) To capture richer and more complex visual characteristics, our approach used the three-channel RGB (Red, green, and blue) format.
(3) In order to identify the best architecture for malware classification, we implemented a set of models based on the Keras architecture library with pre-trained weights from ImageNet, and we got accuracy, precision, recall, and F1-score for each model.
(4) Additional custom layers were used after the pretrained base layer to allow the model to be trained on the visual malware classification task for nine families.
(5) The number of epochs is limited to 50 to reduce overfitting. Mantener el número de epochs en un val The number of epochs was kept at 50 in order to achieve a more generalizable model (avoid overfitting), efficient (due to limited computational resources), and effective when reaching values close to 99% through the MobileNet model.
(6) None of the present models make use of data augmentation methods (CycloGAN) to avoid introducing synthetic images that can add noise to the classification.

(7) Also, **the code of this research can be accessed at** https://github.com/ecram/malware_classification_cnn

The results show that the lightest architectures, such as MobileNet (98.41%), allow obtaining equal or acceptable results to heavy architectures such as DenseNet, which allows its use in environments with limited computational resources.

3 BACKGROUND

3.1 Convolutional Neural Networks

Convolutional Neural Networks [15] (CNN) are a type of deep learning architecture originally designed to process image data and have proven to be very effective in object recognition, feature detection, segmentation, and image classification tasks, among others.

These networks use convolutional layers to apply input filters to the images and thus detect local features such as edges, textures, and visual patterns, allowing spatially relevant information to be captured in the images. They are also used with pooling layers to reduce the dimensionality of the extracted features while preserving the most representative information in the image. Pooling, like max pooling, reduces the size of features by selecting the maximum value in a neighborhood of the image. In addition, one or more fully connected (dense) layers are often added to perform the final classification or regression. These layers are made up of neurons fully connected to each other and allow the network to learn more abstract representations and make predictions based on these representations [14].

These Convolutional Neural Networks are used in the area of cybersecurity for facial recognition identification, malware detection and classification, behavior analysis, and network traffic, among others.

3.2 Transfer Learning in Malware Classification

Transfer learning [14] is a technique that consists of using the knowledge acquired in one task to improve performance in another related task. In the context of Convolutional Neural Networks (CNNs), transfer learning involves taking a network pre-trained on a large data set and applying it to a different task using previously learned weights and features.

Pre-trained CNN networks, such as VGG, ResNet, Inception, and MobileNet, among others, have been trained on massive data sets, such as ImageNet [21], which contains millions of images of different classes. These networks have learned to extract relevant visual features from images, allowing them to capture complex and subtle patterns.

The use of transfer learning with CNN for malware classification is beneficial due to the insufficiency of data that exists in certain families, as shown in Fig. 1, where class 5 or the Simda family represents only 0.4% of the dataset or 42 malware samples. On the other hand, class 3 or Kelihos_ver3 family represents 27.1% or 2942 samples of the total.

Other benefits of using transfer learning are the ability to generalize features, the reduction of training time, the prevention of overfitting, and the best performance and accuracy in classification.

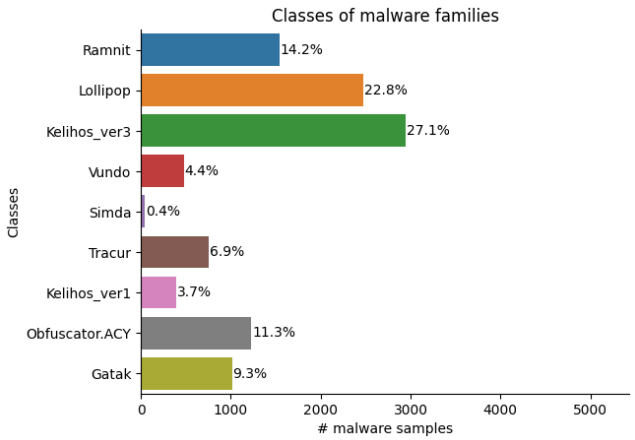


Figure 1: Microsoft malware families samples.

3.3 CNN Architectures for Malware Classification

Below, we describe the main architectures used in this research to classify malware:

3.3.1 VGG-16 & VGG-19. Both architectures [23] use small size filters (3x3) on all layers, followed by layer pooling. The VGG-19 version adds three additional convolutional layers to the end of the VGG-16 architecture, allowing a deeper representation of image features.

3.3.2 ResNet50. Developed by Kaiming et. al. [9], this 50-layer architecture uses residual connections that allow for jumping between layers and help mitigate the problem of gradient fading during training.

3.3.3 InceptionV3. This architecture [25] is an improvement on its predecessor, GoogLeNet. It makes use of convolutional blocks designed to capture features at multiple scales and levels of abstraction. It also uses dropout regularization and uses auxiliary layers to help train and propagate gradients.

3.3.4 MobileNet & MobileNetv2. It is an architecture [11] designed for mobile devices and applications with limited computing resources. It uses a technique called separable depthwise convolution to reduce the number of computational operations and the number of model parameters. MobileNetv2 uses a combination of linear and nonlinear convolution blocks and uses a spreading layer followed by a projection layer to capture features at different scales, achieving better results.

3.3.5 Xception. CNN relies on depth-separable convolutions and residual connections to extract features efficiently and improve the performance of the Convolutional Neural Network. Its innovative design has proven effective in image classification and object detection tasks [5].

3.3.6 EfficientNet. It is a family of CNN architectures proposed by Tan et al. [26] in 2019. These architectures are designed with the

goal of achieving an optimal balance between performance and computational efficiency. EfficientNet uses an automatic search strategy to find the optimal set of hyperparameters that improves model performance without greatly increasing computational complexity.

4 THE PROPOSED METHOD

4.1 Experiment settings

The runtime environment of the experiment includes (1) ASUS nv580vd with an Intel® Core™ i7 7700HQ 2,8GHz processor, 16 GB SDRAM, NVIDIA GeForce GTX 1050, 4GB GDDR5 VRAM, Ubuntu 20.04 LTS (64bit). (2) i440fx-xenial. Intel Core i7 9xx (Nehalem Core i7, IBRS update), 8GB, Ubuntu 22.04 LTS. The PIL 9.4.0 library was used for the conversion of binary files to the PNG format. TensorFlow and Keras 2.12.0 libraries over Python 3.8.10, Pandas 1.5.3, and Numpy 1.23.5 were used to implement the convolutional network models.

4.2 Dataset Description

For the experiment we used the well-known Microsoft Malware Classification Challenge dataset [20] or BIG 2015. The dataset is almost 200GB, consisting of a set of 10868 known malware bytes files representing a mix of 9 different families, as shown in Table 1. Each malicious file has a 20 character hash value uniquely identify, and a class label (1 to 9) representing 1 of the 9 family names.

Table 1: Malware Samples in the Dataset

Family Name	# Train Samples	Type
Ramnit	1541	Worm
Lollipop	2478	Adware
Kelihos_ver3	2942	Backdoor
Vundo	475	Trojan
Simda	42	Backdoor
Tracur	751	TrojanDownloader
Kelihos_ver1	398	Backdoor
Obfuscator.ACY	1228	Obfuscated malware
Gatak	1013	Backdoor

The malicious fileset (10868) is made up of raw data with a hexadecimal representation of the file's binary content, without the header, i.e., to ensure sterility.

The data set can be downloaded from the competition website [1].

4.3 Evaluation Metrics

The evaluation metrics used in the CNN malware classification allowed us to provide a quantitative measure of the performance and quality of the models developed in this task. Below, we present the metrics used in this research.

4.3.1 Accuracy. This basic metric measures the proportion of malware classified correctly by the model out of the total number of examples. It is calculated using the following equation (1):

$$Accuracy = \frac{TP}{TP + FP} \tag{1}$$

Where TP is the number of malware samples correctly classified into their respective families and FP is the number of malware samples that were incorrectly classified into an incorrect family.

4.3.2 Logloss. Or log loss is a metric that measures the quality of the classification probabilities generated by a model. It is calculated using the following equation (2):

$$Logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) \tag{2}$$

Where N is the total number of examples. (y_i) is the actual label of the example (i) (0 or 1) and (p_i) is the predicted probability of the example (i) of belonging to the positive class.

4.3.3 Precision, Recall and F1. The precision is calculated as the proportion of samples correctly classified in a given family with respect to the total number of samples classified in that family. In the context of malware classification, it is calculated as follows in equation (3):

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

Where TP represents the number of samples correctly classified in their respective families and FP represents the number of samples that were incorrectly classified.

Recall, also known as sensitivity, is calculated as the proportion of correctly classified samples in a certain family with respect to the total number of real samples in that family. In this context, it is calculated as follows in equation (4):

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Where TP represents the number of samples correctly classified in its respective family and FN represents the number of samples that were incorrectly classified as belonging to other families.

F1-score provides a balanced measure of model performance by taking both precision and recall into account. Its formula is as follows in equation (5).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

4.4 Visualizing Malware as an Image

This area was introduced by Nataraj et. al. [17] who performed the conversion from a byte file to an image, interpreting each byte as a pixel. In this research, the PIL (Python Imaging Library) library [2] was used to convert byte files into an image in PNG format, as shown in Fig. 3, which scales the values of an image to a matrix from 0 to 255, converting to the 'uint8' type. This conversion allows creating images in RGB format.

Once each byte has been assigned an RGB value, the resulting set of values is organized into a three-dimensional array. Each array represents the image in one set color and has dimensions that depend on the size and structure of the original byte file. One of the

One of the benefits of using RGB was that it allowed more information to be captured over three channels (red, green, and blue)

[1] https://www.kaggle.com/competitions/malware-classification/

[2] https://pillow.readthedocs.io/en/stable/

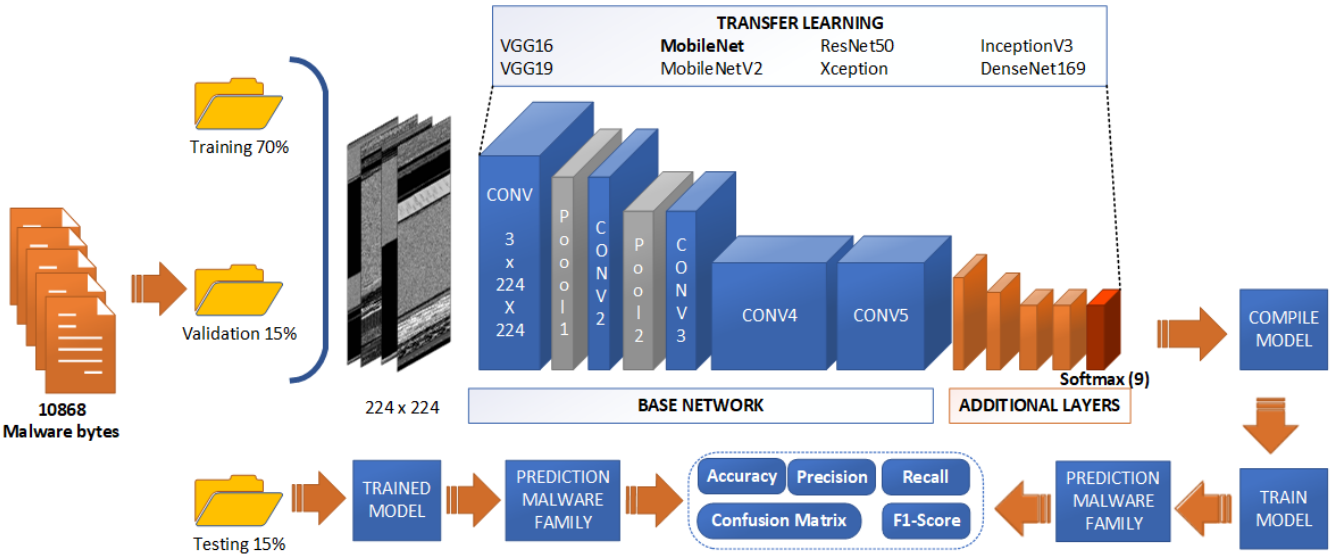


Figure 2: Implementation of transfer learning models for malware classification.

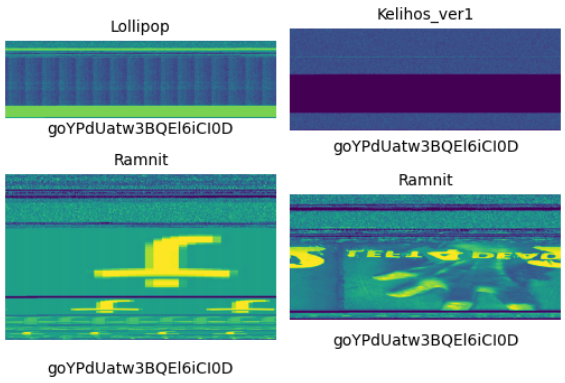


Figure 3: Malware bytes conversion to a PNG image. The family name (upper) and the file name (lower) are described.

and made it easier to use pre-trained ImageNet architectures such as VGG16, ResNet, and MobileNet.

From the transformation process of 10868 malware bytes, the same number of images was obtained to avoid compression loss, improving the quality and contrast of the images in PNG format. By using the train_test_split Scikit-learn library [3], we split the samples into three sets, 70% (7607) for model training, 15% (1630) for validation of the model, and 15% (1631) for tests as well as obtaining evaluation measures. Each set was classified into nine malware families. Next, the samples were moved into their corresponding directory through an algorithm developed in order to facilitate the training, validation, and testing of the models.

[3]https://scikit-learn.org/stable/

4.5 Proposed Models

In the development of the models, the transfer learning technique was used (see Section 3.2) through the Keras library [4]. For the selection of the architectures, we have considered aspects [5] such as the size and complexity of the data set, the size of the data sample, the computational requirements, performance and evaluation metrics, as well as the research and literature on malware analysis and classification described in Section 3.3.

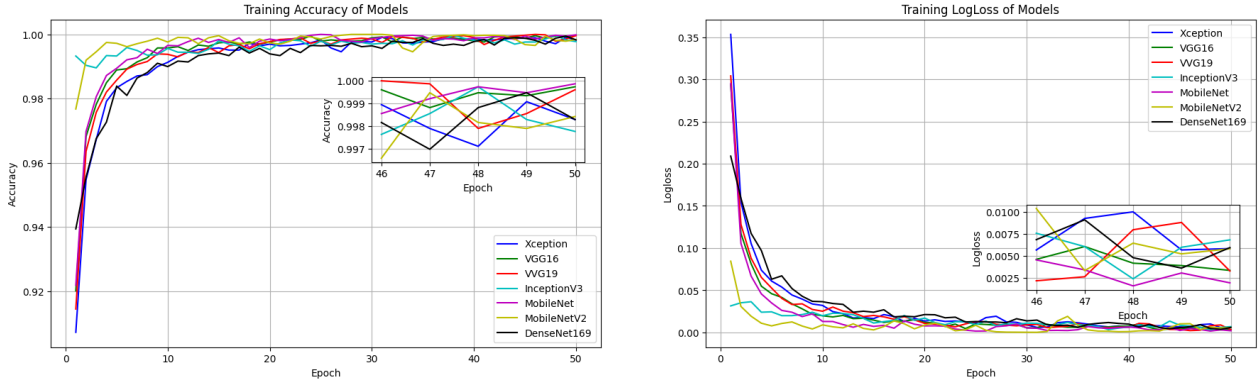
The selection of the Microsoft BIG 2015 [20] dataset benchmark, split into three parts: 70% training, 15% validation, and 15% testing, allows the model to be correctly evaluated through the use of the data set test, avoiding hyperparameter overfitting or "information leaking". This error was noticed by the team in many of the reviewed research papers. In our research, the test data was not used during the training phase. All the experiments used a total of 50 epochs because it was found that the models stabilized at 30 epochs and we wanted to avoid the problem of overtraining.

The development and implementation of the supervised models do not allow the use of data augmentation due to one-to-one correspondence problems [2]. Thus, we do not use any method of data augmentation.

Next, we describe the development process of the models based on the transfer learning technique, as shown in Fig. 2:

- Initially, the pre-trained models were loaded with the ImageNet model weights, excluding the top classification layer of the model to add a new classification layer specific to the Microsoft BIG 2025 malware classification problem.
- The convolutional layers of the pretrained model were frozen.
- The output of the pretrained model is taken, and a series of custom layers are applied to it to perform the specific malware classification. Layers used include:

[4]https://www.tensorflow.org/api_docs/python library /tf/keras
[5]https://keras.io/api/applications/



(a) Accuracy of the models in the training in 50 epochs.

(b) Logloss of the models in the training in 50 epochs.

Figure 4: Accuracy and Logloss resultas. ResNet50 is not graphed because it has a value below the results average.

- Flatten
- Dense(512, ReLU activation)
- BatchNormalization
- Dropout
- Dense(9, softmax)

- For the compilation we use the Adam optimizer with a low learning rate and a loss function in "categorical_crossentropy".
- For the training of the model, 50 epochs are specified per stage of train and validation.
- Finally, the model was tested through the evaluation metrics in the test dataset.

The general scheme used to generate the models can be observed in Fig. 2. In this figure, we use the MobileNet architecture to implement our model, which achieved 98.41% in the test dataset with the training parameters described in Table 2. Also, **the code can be accessed at** https://github.com/ecram/malware_classification_cnn.

Table 2: Training parameters for Malware Classifications models.

Parameter	Value
Learning Rate	0.001
Epochs	50
Optimizer	Adam
Loss Function	Sparse categorical cross entropy

4.6 Experimental result and analysis

The results are described in Table 3 for the classification of malware into nine families using eight models based on architectures described in Section 3.3 for the Microsoft BIG 2015 benchmark. Each model is evaluated in terms of precision, logarithmic loss, or Logloss, precision, recall, and F1-Score. We describe the main results below:

- Xception: This model reached an accuracy of 98.22% with a Logloss of 0.08117, which indicates that it has a correct prediction and a good data fit. The precision (98%) and recall (96%) indicate that the model can correctly identify most instances.
- VGG19: The VGG19 model has an accuracy of 98.10%, slightly higher than the VGG16 model. The Logloss value is 0.07824, indicating a good fit. The accuracy and recall are 96% and 95% respectively, and the F1 score is 95%.
- MobileNet: This model has the best performance in terms of accuracy, with an impressive 98.41%. The Logloss value is 0.08078, indicating a good fit. Although the accuracy is 94%, the recall is high at 98%, which suggests that the model is very good at identifying positive instances. The F1 score is 96%.
- MobileNetV2: This model has an accuracy of 98.22%, similar to Xception. The Logloss value is 0.09172. The accuracy is 95% and the recall is 94%, with an F1 score of 94%.
- In general, the models present a solid performance in the classification of malware in the 9 families evaluated. Most of them have an accuracy above 97%, with the exception of ResNet50 (96.01%), which indicates that they are capable of correctly classifying the vast majority of malware samples.

Table 3: Results of the prediction on the test set for the classification of malware

Models	Accuracy	Logloss	Precision	Recall	F1-Score
Xception	98.22%	0.08117	98%	96%	97%
VGG16	97.79%	0.08509	96%	94%	95%
VGG19	98.10%	0.07824	96%	95%	95%
ResNet50	96.01%	0.15323	96%	92%	94%
InceptionV3	97.11%	0.15323	94%	95%	94%
MobileNet	**98.41%**	**0.08078**	94%	98%	96%
MobileNetV2	98.22%	0.09172	95%	94%	94%
DenseNet169	97.98%	0.09244	96%	96%	96%

In Fig. 4 we can see the accuracy and Logloss curves during training, with a rapid progression from the first epoch, and a smooth progression from epoch 10. In the case of Fig. 4a for accuracy, both InceptionV3 and MobilNet require only 5 epochs to learn how to

Table 4: Comparison of the proposed model in the literature using the Microsoft BIG 2015 dataset

Authors	Models	Dataset	Validation	Accuracy	Precision	Recall	F1-Score
Gibert et. al. [7]	CNN	Bytes	5-fold cross-validation	98.28%	-	-	96.36%
Gibert et. al. [8]	CNN	Bytes (Gray)	10-fold cross-validation	97.50%	-	-	94.00%
Kalash et. al. [12]	CNN (25 epochs)	Bytes (Gray)	train:90%-test:10%	98.99%	-	-	-
Le et. al. [13]	CNN (100 epochs)	Bytes	5-fold cross-validation	98.20%	-	-	96.05
Hemalatha et. al. [10]	DenseNet (100 ep.)	Bytes (Gray)	train:70%-test:30%	98.46%	98.58%	97.84%	98.21%
Wang et. al. [28]	DenseNet+DEAM	Bytes (Gray)	train:60%-test:20%-test:20%	97.3%	95.3%	95/4%	95.4%
Proposed model	MobileNet	Bytes (RGB)	**train:70%-valid:15%-test:15%**	**98.41%**	**94.44%**	**98.25%**	**95.96%**
Tekerek et. al. [27]	DenseNet+Cycle-Gan	Bytes (Gray)	train:80%-test:20%	99.76%	95.00%	96.00%	96.00%
Tekerek et. al. [27]	DenseNet+Cycle-Gan	Bytes (RGB)	train:80%-test:20%	99.86%	98.00%	97.00%	97.00%
Chen et al. [4]	DenseNet+Cycle-Gan	Bytes (Gray)	train:80%-test:20%	99.62%	97.62%	97.72%	97.47%
Chen et al. [4]	DenseNet+Cycle-Gan	Bytes (RGB)	train:80%-test:20%	99.76%	98.90%	97.92%	98.39%

classify malware. All the models, with the exception of ResNet50, converge to values close to 98% precision. Regarding Logloss, as shown in Fig. 4b, as the epochs increase, the models fit the data better and the results begin to be similar, highlighting InceptionV3 and MobileNet for starting with a Logloos minor.

Based on the results presented and considering the combination of high accuracy (98.41%), low Logloss (0.08078), high recall (98%) and computational efficiency, i.e., it stands out for its efficient design in terms of the use of computational resources. and size of the model, MobileNet stands out as a strong and promising option for malware classification in all nine families tested. As we can see in Fig. 5 of the Confusion Matrix of the MobileNet model, the model manages to correctly classify all families with the exception of family 4 Objuscartor.ACY, which are confused in classifying on some samples.

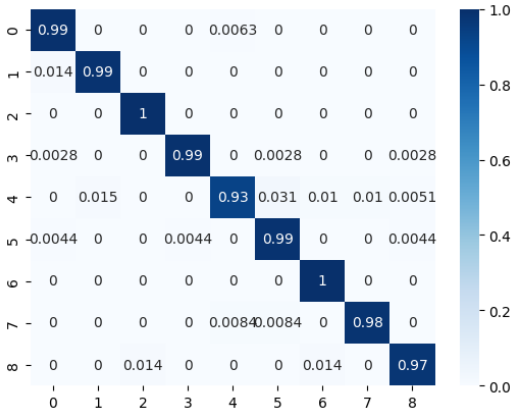


Figure 5: Confusion matrix of the MobileNet.

In Fig. 6 we perform the prediction of the classification of malware examples in a random way, obtaining very good results.

4.7 Limitations

The results described in Table 4 show that the proposal to use the transfer learning technique in the MobileNet model has satisfactory

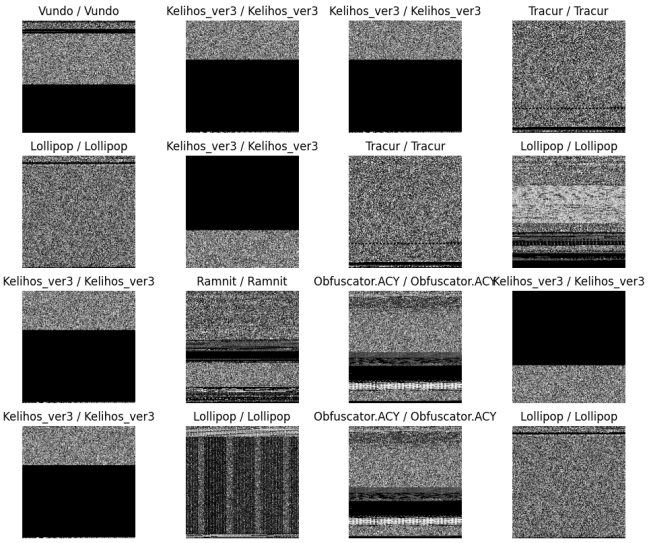


Figure 6: Malware prediction through the MobileNet model.

results regarding accuracy (98.41%), recall (98.25%) and F1- Score (95.96%). In addition, it was shown that the analysis methodology of the eight known CNN architectures allowed for the selection of MobileNet as a lightweight architecture for environments with limited computational resources, obtaining very promising results.

Our research differs from other studies that obtained similar results by employing the CycloGAN technique for data augmentation in image-to-image translation tasks. Unlike these studies, which relied on synthetic data to address unsupervised problems with no direct correspondence between input and output images, our approach did not involve any form of artificial data augmentation.

By limiting it to 50 epochs, we avoid overfitting compared to other research [18], [19], that got excellent results but, when applied to many epochs, had a tendency to overfit. Furthermore, [3] concluded that texture analysis presents many challenges, such as scalability and class imbalance, especially in real conditions with larger datasets than the experiment presented. Also, some naive premises associated with the selection of samples in the datasets

caused the introduction of biases that, in the end, produced non-reproducible results.

In our research, the use of a training set division scheme (70%-15%-15%) with the problem of class imbalance reduces the efficiency of classifying families based on the model. However, given the results, they show that MobileNet can be an effective and efficient model for malware classification with high computational performance.

5 CONCLUSIONS AND FUTURE WORK

Based on the comparative analysis of the results of the proposed model, under the approach of using transfer learning in eight architectures (Xception, VGG16, VVG19, ResNet50, InceptionV3, MobileNet, MobileNetV2, and DenseNet169) in the Microsoft BIG 2015 benchmark for the classification of malware, we can conclude that:

The performance of the proposed MobileNet model was satisfactory in terms of precision (98.41%), recall (98.25%), and F1-Score (95.96%), indicating a remarkable ability to classify malware samples correctly. Furthermore, the F1-Score shows a good balance between precision and recovery.

Compared with the results of previous studies, the proposed model has achieved competitive and, in some cases, remarkable results without the use of data augmentation (CycloGAN), which is not inherent in supervised problems. With a correct division of the data set to avoid hyperparameter overfitting or "information filtering" and limiting epochs to 50 to avoid overfitting.

As an additional contribution to the research on the classification of malware families, we share the **code:** https://github.com/ecram/malware_classification_cnn.

Based on these conclusions, we can propose the following future work with the aim of improving the results of CNNs in the classification of malware in the wild through their generalization and avoiding overfitting:

- Use new ways of extracting information from bytes in image files in order to obtain a better representation, reduce false positives, and improve the accuracy of the model under the malware detection and classification approach.
- Use techniques to expand the number of corresponding malware samples, avoiding hyperparameter overfitting and improving performance estimation.
- Evaluate different malware datasets in order to generalize and improve the performance of the classification system by adding malware in the wild.

REFERENCES

[1] AV-TEST GmbH. [n. d.]. AV-TEST Malware Statistics. https://www.av-test.org/en/statistics/malware/. Accessed: May 20, 2023.

[2] Ghazal Bargshady, Xujuan Zhou, Prabal Datta Barua, Raj Gururajan, Yuefeng Li, and U Rajendra Acharya. 2022. Application of CycleGAN and transfer learning techniques for automated detection of COVID-19 using X-ray images. *Pattern Recognition Letters* 153 (2022), 67–74.

[3] Tamy Beppler, Marcus Botacin, Fabrício JO Ceschin, Luiz ES Oliveira, and André Grégio. 2019. L (a) ying in (Test) Bed: How Biased Datasets Produce Impractical Results for Actual Malware Families' Classification. In *Information Security: 22nd International Conference, ISC 2019, New York City, NY, USA, September 16–18, 2019, Proceedings 22*. Springer, 381–401.

[4] Zhiguo Chen, Shuangshuang Xing, and Xuanyu Ren. 2023. Efficient Windows malware identification and classification scheme for plant protection information systems. *Frontiers in Plant Science* 14 (2023), 345.

[5] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1251–1258.

[6] Fred Cohen. 1987. Computer viruses: theory and experiments. *Computers & security* 6, 1 (1987), 22–35.

[7] Daniel Gibert, Carles Mateu, Jordi Planes, and Ramon Vicens. 2018. Classification of malware by using structural entropy on convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[8] Daniel Gibert, 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 (2019), 15–28.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[10] Jeyaprakash Hemalatha, S Abijah Roseline, Subbiah Geetha, Seifedine Kadry, and Robertas Damaševičius. 2021. An efficient densenet-based deep learning model for malware detection. *Entropy* 23, 3 (2021), 344.

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[12] Mahmoud Kalash, Mrigank Rochan, Noman Mohammed, Neil DB Bruce, Yang Wang, and Farkhund Iqbal. 2018. Malware classification with deep convolutional neural networks. In *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*. IEEE, 1–5.

[13] Quan Le, Oisín Boydell, Brian Mac Namee, and Mark Scanlon. 2018. Deep learning at the shallow end: Malware classification for non-domain experts. *Digital Investigation* 26 (2018), S118–S126.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[16] Ellen Messmer. 2008. Tech talk: Where'd it come from, anyway? *Pc World: Business Cen* (2008).

[17] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and Bangalore S Manjunath. 2011. Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*. 1–7.

[18] Edmar Rezende, Guilherme Ruppert, Tiago Carvalho, Fabio Ramos, and Paulo De Geus. 2017. Malicious software classification using transfer learning of resnet-50 deep neural network. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 1011–1014.

[19] Edmar Rezende, Guilherme Ruppert, Tiago Carvalho, Antonio Theophilo, Fabio Ramos, and Paulo de Geus. 2018. Malicious software classification using VGG16 deep neural network's bottleneck features. In *Information Technology-New Generations: 15th International Conference on Information Technology*. Springer, 51–59.

[20] Royi Ronen, Marian Radu, Corina Feuerstein, Elad Yom-Tov, and Mansour Ahmadi. 2018. Microsoft malware classification challenge. *arXiv preprint arXiv:1802.10135* (2018).

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115 (2015), 211–252.

[22] Michael Sikorski and Andrew Honig. 2012. *Practical malware analysis: the hands-on guide to dissecting malicious software*. no starch press.

[23] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[24] Eugene H Spafford. 1989. The Internet worm program: An analysis. *ACM SIGCOMM Computer Communication Review* 19, 1 (1989), 17–57.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[26] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[27] Adem Tekerek and Muhammed Mutlu Yapici. 2022. A novel malware classification and augmentation model based on convolutional neural network. *Computers & Security* 112 (2022), 102515.

[28] Changguang Wang, Ziqiu Zhao, Fangwei Wang, and Qingru Li. 2021. A novel malware detection and family classification scheme for IoT based on DEAM and DenseNet. *Security and Communication Networks* 2021 (2021), 1–16.

Received 30 May 2023; revised 12 June 2023; accepted 24 July 2023