Journal of Theoretical and Applied Information Technology
28th February 2023. Vol.101. No 4

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MALIMG2022: DATA AUGMENTATION AND TRANSFER LEARNING TO SOLVE IMBALANCED TRAINING DATA FOR MALWARE CLASSIFICATION

IKRAM BEN ABDEL OUAHAB[1], LOTFI ELAACHAK[2], YASSER A. ALLUHAIDAN[3], MOHAMMED BOUHORMA[3]

[1,2,4]Computer science, systems, and telecommunication laboratory (LIST),

University Abdelmalek Essaadi, FSTT, Tangier, Morocco

[3]Cybersecurity Architect at Saudi Data and Artificial Intelligence Authority

E-mail: [1]ibenabdelouahab@uae.ac.ma, [2]lelaachak@uae.ac.ma, [3]y.alluhaidan@gmail.com, [4]mbouhorma@uae.ac.ma

ABSTRACT

Data augmentation is creating new images by transforming old ones. It's used to solve imbalanced image classification problems in many domains. Usually, data augmentation is used when we are unable to get more data for underrepresented classes. So, data augmentation techniques help us to increase the size of training data in order to avoid any bias in the classifier. This paper's main contribution is to develop a balancing tool for any imbalanced multiclass database. Then, we use this approach in application to the Malimg database to improve its effectiveness and speed to solve imbalanced data problems. As a result, we generated 2 versions of the Malimg database namely Malimg2022 (Large and XXLarge), also we make the "Balance Me" application that can balance any database using augmentation techniques. These new versions are balanced, having the same number of samples per class using data augmentation with different transformation techniques. From a technical point of view, Zero-day malwares are none than old ones with few modifications, so data augmentation could be seen as a simulation of new malware variants that should be detected effectively. Finally, the new balanced data were evaluated using transfer learning models. So, the generated databases didn't show an overall improvement in the whole classification task, however, we have found some improvements related to some malware classes. Here, we can say that the use of data augmentation to balance data isn't always a good choice.

Keywords: *Cybersecurity, Image augmentation, Malware classification, Transfer Learning, Deep Learning.*

1. INTRODUCTION

Malware is designed for stealth and it's been difficult to notice nowadays. It spreads slowly over time and gathers information over an extended period before exfiltration. Current-state malware use one set of procedures as most attacks are blended and use multiple methods. Malware is changing. Historically, the malware was designed for speed and easy to notice. It does quick actions by destroying data or some other malicious activity soon after infection. Older malware has distinct procedures for handling various categories of infection. Large and small companies and businesses are constant targets. Also, smaller victims can play a part in larger attacks as an entry door to more important targets. In a company, malware leads to loss of data, intellectual property, competitive advantage, overall consumer confidence. Companies should care about malware because of the risk legal action if negligent or not properly protected. The cost of malware infections can be unbelievable. In 2014, $491 billion was spent on recovery from malware infections. $25 billion spent by consumers as a result of security threats. 1.2 billion hours dealing with the after-effects. $315 billion was the result of criminal organizations [1], [2], [3].

Major forms of malware are: viruses, worms, trojan horses, mobile attacks, and blended attacks:

- Viruses are self-replicating and installing themselves into the user data, typically triggered via some action. Two main categories of viruses: compiled and interpreted viruses. The compiled virus is executed by the OS and can be an infected file or a boot sector virus. The interpreted virus is executed by an application for example macro and script viruses.

- Worms are self-replicating programs that are usually self-contained and can execute and spread without any user interaction. Two main types of worms: network service and mass mailing worms. The network service worms exploit network vulnerability to propagate and infect others. The mass mailing worms exploit email systems to spread and infect others.
- Trojan horse programs are non-replicating malicious program that has a hidden payload. It is often disguised as friendly programs or applications. It replaces existing files with malicious ones and/or deliver additional attacker tools to the host.
- Malicious mobile code is a code delivered remotely that runs on a local host, it typically runs without host intervention. Java, ActiveX and VBScript are examples of popular languages used for infection.
- Blended Attacks use multiple methods of transmission, combines propagation characteristics viruses and worms, can be delivered via various methods including via Trojan Horses.

Recently, hackers use many ways and methods of transmission, including: direct access to a host system via an infected disk or usb, social engineering, phishing as spear-phishing or whale-phishing, visiting a malicious website, and others. It is important to know that no one is immune, and everybody could be infected from experts to end users. The main goals of malware analysis are: preparing and training, identifying, containing, eradicating/mitigating, recovering, and documenting/lessons learned. Malware analysis is all about learning how infection was done in order to protect effectively in the future. Two main types of malware detection: Manual and Automated Detection.

- Manual detection is time consuming and labor intensive. It doesn't scale well for large companies. It isn't able to contain quickly enough for rapidly spreading malware. Also, it could be difficult to identify physical location of host. This method may be required only for highly customized malware.
- Automated detection is quick and scalable. It allows for rapid containment. It may miss highly customized malware or never before seen malware (zero-day malware). Using automated method, we can make remediation efforts more effective. It also allows for remote management of incidents.

Classically, malware analysis is divided into two main categories: Static malware analysis and Dynamic malware analysis [4].

- Static malware analysis entails inspecting any delivered malware sample without running or executing the malicious code. This is generally done by determining the signature of the malware binary; the signature is an original identification for the binary file. This method is limited to known malware and fails to recognize newborn or zero-day malware.
- Dynamic malware analysis involves running the malware sample and observing its behavior on the system in order to remove the infection or stop it from spreading into other systems. The system is setup in a closed, isolated virtual environment so that the malware sample can be studied thoroughly without the risk of damage the physical device itself. While this method is time consuming but still effective in the case of new malware variants. However, it required high expertise in malware analysis skills.
- Malware analyst combine these two analysis methods and they perform a hybrid malware analysis. It is simply done by taking advantage of both static and dynamic analysis reports. Their combination gives more information about the malicious software.

Artificial intelligence is that the science of constructing computers smarter. Complicated algorithms permit computers to resolve issues that were antecedently solely soluble by humans. AI will learn to finish a selected task by researching large amounts of information independently, a method that's known as machine learning and deep learning. Today, every domain is been related with Artificial Intelligence. Everyone is trying to get benefit from the power of AI to gain performance, and time.

While doing classification using either deep learning or machine learning, we always start with data processing. It's taken more attention in the global application process. A database could be balanced or unbalanced. Balanced databases are those who have a similar amount of data per class. However, unbalanced data have literally unbalanced classes. That's mean it has lots of samples in some classes, and few samples in other ones. When building and training a deep learning model using unbalanced data, the model could be biased towards the majority class. Most algorithms are designed to work well with balanced databases.

Journal of Theoretical and Applied Information Technology
28th February 2023. Vol.101. No 4

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Better we know the malware type, better we protect our systems. In this paper, we perform data augmentation to deal with imbalanced malware images database, namely, Malimg. The contribution of this paper include new balanced database Malimg2022-Large and Malimg2022-XXLarge. As a generalization of the augmentation approach, we developed a friendly GUI application called "Balance Me". This application can be used by anyone, without touching any technical code, such as for biological researchers in case of image classification tasks. New balanced databases are evaluated using transfer learning. We adopted : Xception, ResNet50 and InceptionV3, with the original database Malimg and the balanced generated databases.

2. CONTRIBUTION

Our main contribution in this paper include:

- Generating new balanced versions of Malimg database by doing Data augmentation using Keras.
- Evaluate new database using transfer learning: Xception, InceptionV3 and ResNet50.
- A GUI Application to augment any imbalanced data.

As a generalization to the concept of Image data augmentation of a multiclass image database, we built an application that can transform any imbalanced data to a balance database using data augmentation techniques, called "Balance Me" (Figure 1). As input, you give the original database path and the desired number of samples you want to be in the final database. The workflow of the proposed application passes by several steps to discover the given database (figure 2), count missing values, remove extra data, generate new data and a final check to insure a balanced database output.

Among our contribution, we deliver 2 new versions of Malimg databases having balanced data. Augmentation 1 (Malimg-Large) [5]: we augmented database with the goal of 530 samples per class for training and 70 for testing per class. So, we have a total of 13250 images for training and 1750 images for testing. And the new Malimg-Large version have 15000 images. Here we deleted some samples from some classes. Database size: 945 Mo. Augmentation 2 (Malimg-XXLarge) [6]: we augment data based on the majority class (3000 samples). So, the value goal is 3000 per class for training and 70 per class for testing. Here, we don't delete any data. So, the Malimg-XXLarge have 76750 images in total. But the testing dataset present only 2% of the global database. Database size: 1,73 Go.

The choice of data augmentation is directly related to the processing power available and storage. In our case we tried to fit our hardware capabilities in order to build the maximum images per class. So, there isn't a mathematical relation we have based on to generate 3000 images per class. It's true that, in data augmentation, we try to reach the class that have the maximum number of images, but here the gap was too large to go with this rule. That's why we tried to make a logical augmentation, taking into consideration the large class and the very tiny one. The only limitation, as mentioned before, was the processing power of the hardware and storage available for images.

We hope that researchers take advantages on doing further research on them. We making it available publicly in GitHub.

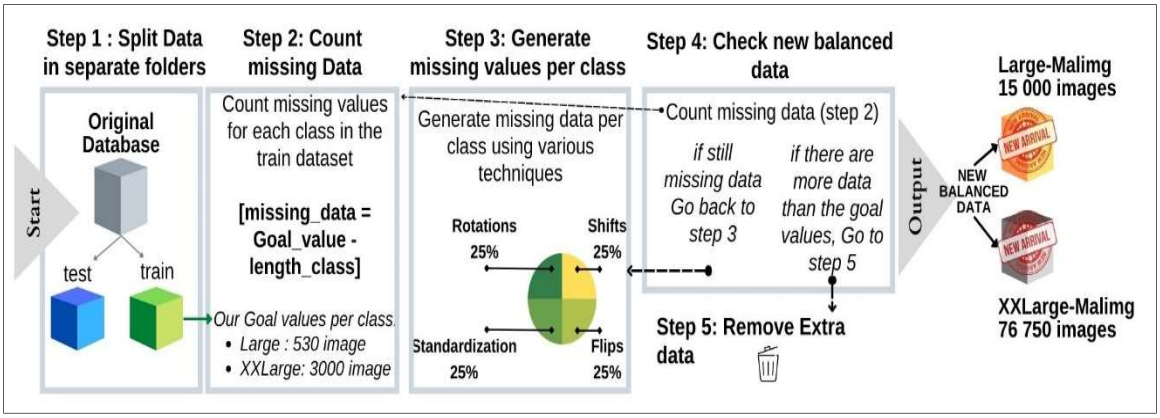


Figure 1 Data augmentation Application workflow

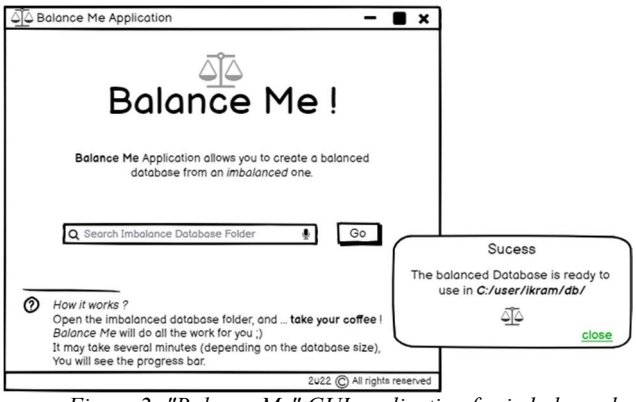


Figure 2 :"Balance Me" GUI application for imbalanced databases

3. RELATED WORKS

The creator of Malimg database, Nataraj [7] used cross-validation to boost their performances. They proposed a KNN model using GIST features. The cross-validation technique success to deal with small or imbalanced data. As a result, the accuracy is 98% with Malimg database.

Di Wu [8] proposed a malware detection method using cascading XGBoost and cost sensitive in order to deal with unbalanced data. They used extracted API calls from PE files as features, then they adopt three-tier cascading XGBoost for data balancing and model training. Di Wu used a database with 2 classes for malicious and benign API calls. The results obtained using this method are good, and reach 99% of accuracy.

Niccolò Marastoni [9] used obfuscation as data augmentation tool. They made a comparative study on OBF data and Malimg using both CNN and LSTM models. The given result is an accuracy of 92.3% and 92.6% for CNN and LSTM models respectively with OBF database. However, Malimg database reach higher accuracy of 98.3% and 98.5% for CNN and LSTM models respectively. As conclusion of the study, LSTM is shown to be best classifier in their case, with Malimg database.

Ferhat Ozgur Catak [10] applied data augmentation to malware samples using Gaussian, Poisson, and Laplace noises. They use a combination of noises and RGB malware images to augment the input database. Then, five deep CNN models were built for detecting malware families especially in a metamorphic malware environment. As a result, the best accuracy is 0.98 for Laplace/Gaussian.

Roland Burks [11] adopt generative models in order to generate synthetic training data. They used two models: the Generative Adversarial Network (GAN) and Variational Autoencoder (VAE). Their main contribution aims to increase the performance of Residual Network (ResNet-18) classifier for malware detection. So, after adding synthetic malware samples to the training data the accuracy improved for ResNet-18 by 2% using VAE, and by 6% using GAN.

Adem Tekerek [12] proposed the B2IMG approach for the transformation of bytes file into gray and RGB image formats. To deal with imbalanced data, they employ CycleGAN-based data augmentation method. The proposed approach was applied to BIG2015 and DumpWare10 datasets. So, the performance of model increased after data augmentation. The accuracy is 99.86% and 99.60% for BIG2015 and DumpWare10, respectively.

Yi-Wei Ma [13] evolves an intelligent framework namely AI@nti-Malware. The proposed framework combined AI algotihms, data balancing, feature evaluation techniques in order to build a malware classification model to defend against malware attacks. To deal with imbalanced data, they used SMOTEENN algorithm to generate training data for minority classes. To evaluate their approach, authors applied machine learning algorithm XGBoost and deep learning backpropagation algorithm on CTU-13 dataset. The results they got are impressive. The accuracy result reach 99.98% for XGBoost and 98.88% for backpropagation model.

Table 1 Comparative study of related works

Ref	Data	Methods	Result
[8]	Malicious and benign API calls	Cost sensitive Cascading XGBoost	Acc. 99%
[9]	OBF data	Obfuscation CNN LSTM	▪ Acc. 92.3% ▪ Acc. 92.6%
	Malimg	CNN LSTM	▪ Acc. 98.3% ▪ Acc. 98.5%
[10]	RGB images for 8 families including malicious and benign	Gaussian, Poisson, and Laplace noises + five deep CNN models	98%
[11]	Malimg (25 families)	VAE GAN Without aug ResNet-18	▪ +2% (85%) ▪ +6% (90%) ▪ 83%
[7]	Malimg (25 families)	Cross-validation KNN	98%
[12]	BIG2015 DumpWare 10	CycleGAN CNN	▪ 99.86% ▪ 99.60%
[13]	CTU-13	SMOTEENN XGBoost Backpropagation	▪ 99.98% ▪ 98.88%

4. METHODS

4.1 Malware visualization technique

The use of byte plot visualization for automatic malware classification was initially investigated by Nataraj et al. [31]. They collected texture-based information from the malware picture after converting every malware sample to grey-scale byte plot representations. They computed texture information from photos using the GIST abstract representation approach. 9,458 malware samples from 25 distinct classifications were obtained from the Anubis [3] system for their collection. They classified malware samples into their appropriate classes using a K-Nearest Neighbor model trained on the global image-based characteristics, using Euclidean distance as the distance metric, and achieved an accuracy of 97.18%. The outcomes demonstrated that malware classification methods based on image processing may categorize malware more quickly than dynamic approaches currently in use.

4.2 Data Augmentation with Keras

Image data augmentation is a technique used to extend a database for 2 mains reasons: first, when we have small database, and second, when we have an imbalanced database where the mount of images in each class is nor equal nor nearly equal. When we use more data to train a deep learning model, we expect a skillful model. So, by using augmentation techniques we create more images that can improve the quality of the model. The Keras deep learning library provides the ImageDataGenerator class which allow us to implement flexibly various augmentation techniques by specifying the class parameters.

In literature, we found that the performance of deep learning models often improves after using data augmentation to have more data or to balance the database [14], [15], [16], [17]. Data augmentation is applied only on the training database part. It creates transformed versions of images in the training data part for each class. These transformations are some operations from image manipulation field, like: shifts, flips, zoom, standardization, etc. The main goal of data augmentation is to expand the training data part with reasonable examples. For example, the new image generated with data augmentation technique should still be recognize as the original one and refereed to the same object.

Image Augmentation is the process of expanding the image training data, by using transformations such as random rotations, shear transforms, shifts zooms and flips, on available image data. We use image augmentation when we don't have enough training data to train our model. In such situations, we can create new images out of the existing images, by applying transformations to them. Though these images look similar, they are considered as entirely new images by Deep learning models (as CNN). This will help us to create a larger training dataset and consequently will enable our model to converge more efficiently.

Here, we are describing four augmentation techniques, that we'll be using to augment the whole database. To see how this will be done, we give in the figure 3, the 2 images as example: one from Malimg database and the other random picture. Then, we present the output using each data augmentation for each input.

Method 1: Feature standardization technique scale pixel values to have a zero mean and unit variance. It is supported at two levels: either per-image (called sample-wise) or per-dataset (called feature-wise). Specifically, the mean and/or mean and standard deviation statistics required to standardize pixel values can be calculated from the pixel values in each image only (sample-wise) or across the entire training dataset (feature-wise). In our implementation will be using the feature-wise technique. We applied standardization using ImageDataGenerator class from Keras API.

Method 2: Random Rotation. Image rotation is one of the widely used augmentation techniques and allows the model to become invariant to the orientation of the object. ImageDataGenerator class allows you to randomly rotate images through any degree between 0 and 360 by providing an integer value in the rotation_range argument. When the image is rotated, some pixels will move outside the image and leave an empty area that needs to be filled in. You can fill this in different ways like a constant value or nearest pixel values, etc.

Method 3: Random shifts. When the object isn't located in the center of the image, we use random shifts. This technique will shift the pixels of the image either horizontally or vertically; this is done by adding a certain constant value to all the pixels.

Method 4: Random flips. Flipping images makes sense to use with a lot of different objects. ImageDataGenerator class has parameters horizontal_flip and vertical_flip for flipping along the vertical or the horizontal axis.

Journal of Theoretical and Applied Information Technology
28th February 2023. Vol.101. No 4

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

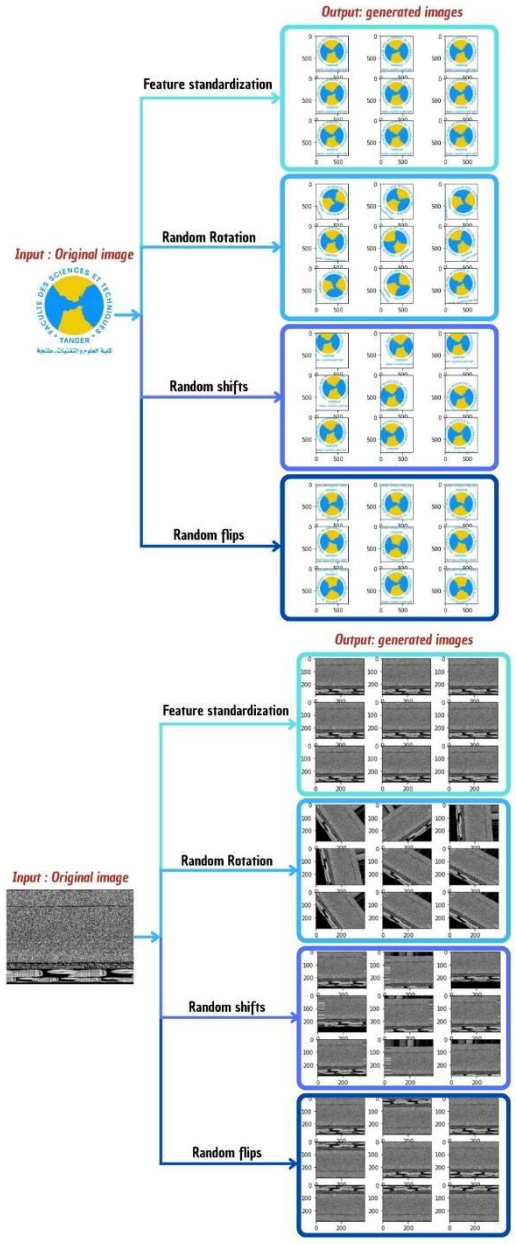


Figure 3 Example of data augmentation techniques

The first step is to split data into train, test and evaluation part before doing the balancing techniques. We are going to use library: split-folders from Pypi [18]. This library does exactly what we need, it split folders with files (images in our case) into train, validation and test folders, class by class. At the beginning we had a folder that contains 25 subfolders which represent malware families (or classes). Then we need to separate train from test database parts. Now, we have a train folder with 25 subfolders, and a test folder with 25 subfolders. When needed, we can also split the test part into test and evaluation part at the end.

The split-folders proposes two methods of split, depending on the situation. The first one allows user to split data using ratio, for example 80% for training and 20% for testing. This technique id used when data are already balanced. Otherwise, the second method allows us to split data using fixed values, and this what we need here. We'll explain why. To understand the difference, in our case, we give an example of imbalanced data with 2 classes. We conclude that using fixed values to separate test data is most appropriate in our case. Because training part will be balanced after that. And the total database will also be balanced.

Table 2 A demo of splitting data using ratio and fixed options

DB	***Classes***	***N° of samples / class***	***Train***	***Test***	***Result***
Using Ratio .8 for train and .2 for test					
DB 1	Class 1	100	80	20	⊗*Testing part imbalanced!*
	Class 2	200	160	40	
Using fixed value = 70 samples/family in test part					
DB 2	Class 1	100	30	70	✓ **Testing data balanced.**
	Class 2	200	130	70	

Returning to the Malimg database, the smallest class contains 80 samples. From there, we decided to take 70 samples for testing and 10 for training. These 10 samples will be used with different data augmentation techniques to generates more and more data. In general, for a given goalValue the application will generate missing data using 4 augmentation techniques equally.

$$to\_be\_augmented_i = goalValue - actualValue_i$$

Where:

- $to\_be\_augmented_i$ are data that we are going to generate using augmentation techniques for class i
- $goalValue$ is the number of images per class we want in the final balanced dataset
- $actualValue_i$ is the number of images in the imbalanced dataset for class i

For each, data augmentation technique, we generate 25% of $to\_be\_augmented_i$ value.

4.3 Transfer learning for malware classification

Transfer learning is a deep learning technique that enables us to profit from the knowledge gained from a previously utilized deep learning models for a lookalike mission. It helps creating new models based on previous ones, fast and effectively. First, we train a model which so many data, and then we can use this old trained model and retrained it again with new data in order to have a new model with double knowledge. In most of times, transfer learning gives the best results.

Xception Model is proposed by Francois Chollet[19]. Xception replaced the Inception modules with a depthwise Separable Convolutions, so Xception model is an extension of the inception architecture. It's used in classification tasks and gives very impressive results in many domains.

Resnet50 is composed by 50 layers deep as it name mentioned. ResNet stands for "short for Residual Networks" is a classic neural network used as a backbone for lots of computer vision problems. It was introduced by Kaiming He [20] especially for visual recognition tasks. It's a well known model, used after that in many other applications that needs images processing and manipulation.

InceptionV3 is another CNN pretrained model, used for images recognition with a deep architecture.

4.4 Tools

Image processing needs powerful hardware, in our laboratory we use the NVIDIA Quadro T1000 with Max-Q GPU workstation, the hardware specifications are given in the table 3. It has a high compute capability of 7.5, and allow us processing big databases in a short timing as compared to other devices.

TensorFlow for deep learning is like water for plants. In our experimentation, we first set up the GPU environment by installing CUDA and cuDNN, then we installed the other required python packages. After many tries to get things done using a simple conda command, it failed and gives a ton of errors. So, we suggest doing the installation and configuration manually to save time and to make sure everything is well installed. As describe here's a manual installation guide of TensorFlow to avoid any kind of errors [21].

Table 3 Hardware specifications

Operating System	Windows 10 Professional 64-bit
System Model	HP ZBook Fury 17 G7 Mobile Workstation
Processor	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz (12 CPUs)
Memory	16384MB RAM
GPU	NVIDIA Quadro T1000 with Max-Q Design
Display Memory	12040 MB
Hard disk	243.1 GB, NTFS
DirectX 12	24 Frames/Sec
GPU Compute	2554 Ops/Sec
Compute capability	7.5

5. RESULTS AND DISCUSSION

Data augmentation techniques lead to the generation of two balanced databases. Here, we evaluate the new database using various deep learning architectures. First, transfer learning based on VGG-16 with the Malimg2022-Large version. The obtained results are bad. The challenge here is the computation power and the insufficiency of RAM memory. To deal with this challenge we decrease the deep of the models in general. Also, we decrease the size of image which is 32x32 so that the input tensor fit well with existing hardware specifications. The second model is subclassing CNN model. The results are not good neither, for the same raisons. The thirst model architecture was tested with the very big database version Malimg2022-XXLarge. Here, we face much more challenges related to hardware insufficiency. After dealing with those challenges, we were able to train a model however the results wasn't acceptable.

Regarding these augmented data, there are many others possibilities to be tested. For example, increasing the input image size from 32x32 to 64x64. Which lead automatically to the need of more powerful hardware, especially more RAM memory to deal with the huge tensors and calculations. Another way is that we suggest building another deep models in the future to improve the accuracy.

Overall, data augmentation work gives the birth of new databases that could be used in future works by researchers' community. In literature, we couldn't find yet any similar works that aims to create new balanced version of Malimg dataset. So, we will compare with various works as cited in the related works section. These works used different databases and techniques to deal with imbalanced data. As a resume, the accuracy found in literature is between 90 and 99% [8] [9] [10] [11] [7] [12] [13].

In our paper, the best result yet reach 98.46% which is accurate with the literature review.

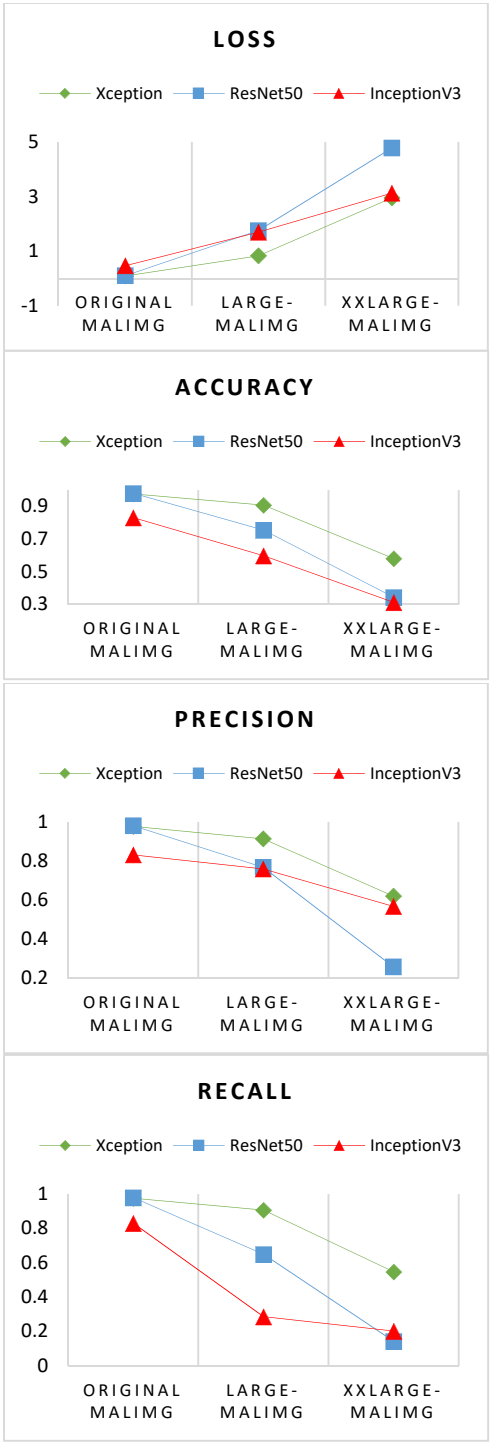


Figure 3 Evaluation metrics curves (loss, accuracy, precision and recall), comparison of models by database

Data augmentation aims to generates new samples from existing ones. We applied this technique to Malimg database. Our very first goal is to balance data since the original database have unbalanced classes. After applying various techniques, we generates two new versions namely Malimg2022 Large and XXLarge. Hopefully, we are looking forward to improve the performance of classification models for the minority classes. In table below, we present advantages and disadvantages of the three databases.

Working with balanced data is the first rule to respect while doing deep learning or machine learning training. During our experimentations, training using balanced data doesn't improve the performances yet. However, we believe that using other models or algorithms could give better results. So, we make the new databases versions available for future research works.

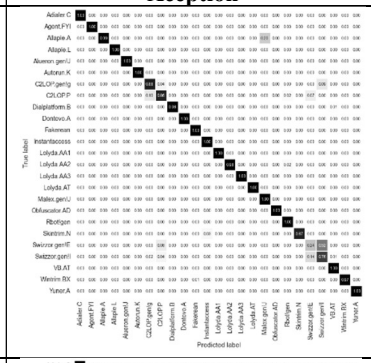
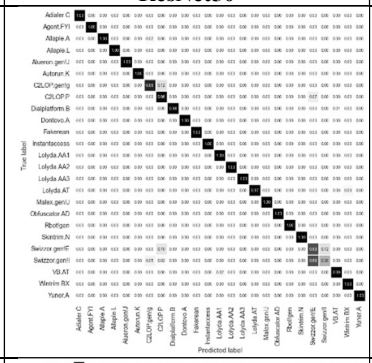
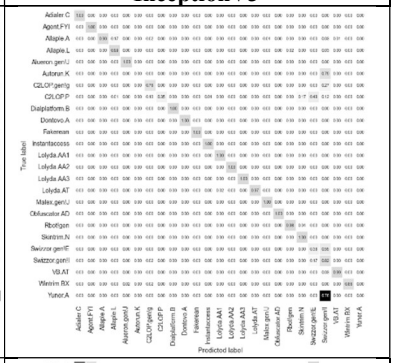
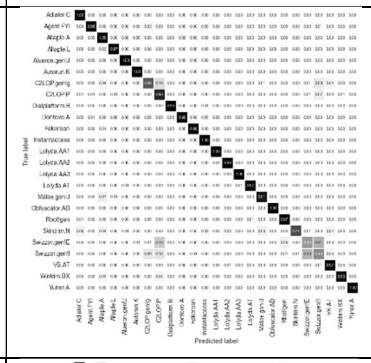
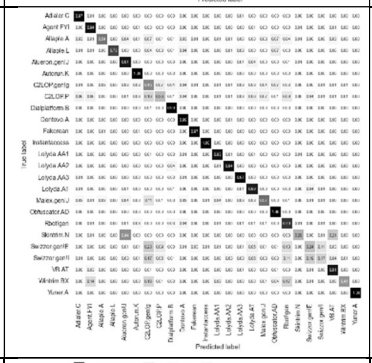
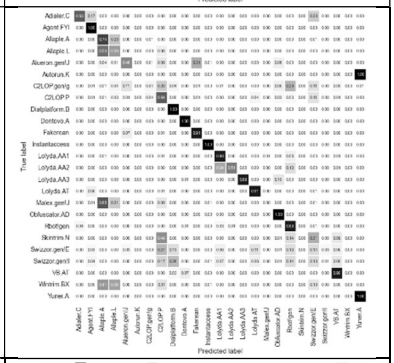
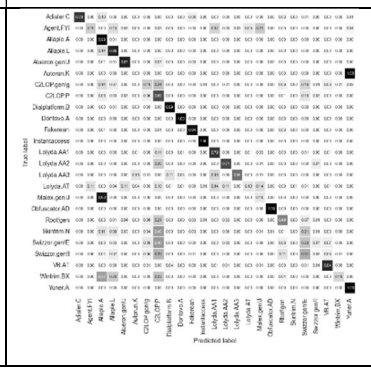
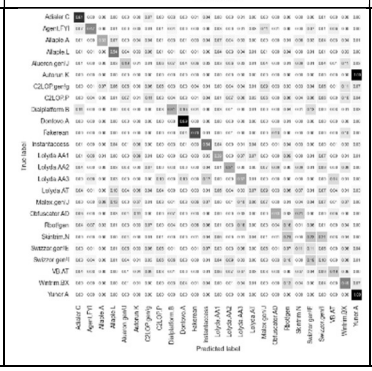
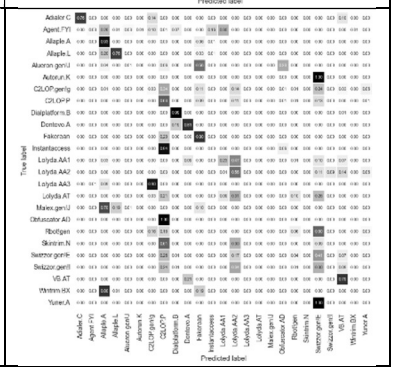
The accuracy obtained using Malimg2022-XXLarge is very low. It could be justified by the fact of having only 2% of testing data which is a very small amount of data as compared to the training data 98%. In general, the rule to split data into train/test parts is 70%-30% or 80%-20% or in between the two. For Malimg2022-Large database, the split of train and test data is acceptable. However, the XXLarge version don't respect this rule in any way. So, this issue may be corrected in a future work.

As we can see in confusion matrix (Table 4) of databases and the models, Large DB is quite near to the original for the Xception ttransfer learning case.

Journal of Theoretical and Applied Information Technology
28th February 2023. Vol.101. No 4

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 4 Confusion matrix

	Xception	ResNet50	InceptionV3
Original			
Large			
XXLarge			

6. CONLUSION

Otherwise, transfer learning and CNN models are both tested with malware images databases. We developed and tested many architectures based on VGG16 model, CNN from scratch using functional and subclassing API. As conclusion, we found that the very right way to do CNN models is using subclassing API. It gives the developer lots of possibilities to customize literally everything. We are looking forward to dive in deeper in this context and propose a customized layers and functions.

Finally, in this paper we do malware classification into their corresponding families. We generate new versions of an existing database to ensure data balancing characteristic using data augmentation techniques. The balanced databases didn't give us an improvement of the classification quality. So, there are two possibilities, either the augmentation techniques or parameters wasn't good enough. Or, the database itself doesn't require the balancing for some technical reasons. In general, to solve imbalanced database problem in any database, researchers can freely use our "Balance Me" application to have a balanced database easily. To, evaluate this approach, we build, train and test CNN models using various architectures and API.

AVAILABILITY OF DATA AND MATERIALS

The datasets supporting the conclusions of this article are available in the Large-Malimg and XXLarge-Malimg repository,

- *http://www.kaggle.com/dataset/1db817273501cb4217d5376edbb612ea671d03be55cd8cbcd887779208d2d214*
- *http://www.kaggle.com/dataset/5ebeed2d0a1255183c9205b9f2bf8275a07f746d455a605920dc9256468ef225*

REFERENCES:

[1] « 2019 Cost of Cybercrime Study | 9th Annual | Accenture », 9 avril 2020. https://www.accenture.com/us-en/insights/security/cost-cybercrime-study (consulté le 9 avril 2020).

[2] « Malware Statistics in 2022: Frequency, impact, cost & more », *Comparitech*. https://www.comparitech.com/antivirus/malware-statistics-facts/ (consulté le 9 mai 2022).

[3] « Malware Statistics & Trends Report | AV-TEST », 8 juin 2021. /en/statistics/malware/ (consulté le 8 juin 2021).

[4] *Malware Analysis and Detection Engineering*. Consulté le: 9 juin 2022. [En ligne]. Disponible sur: https://link.springer.com/book/10.1007/978-1-4842-6193-4

[5] B. A. O. Ikram, *Large-Malimg*. [En ligne]. Disponible sur: www.kaggle.com/dataset/1db817273501cb4217d5376edbb612ea671d03be55cd8cbcd887779208d2d214

[6] B. A. O. Ikram, *XXLarge-Malimg*. [En ligne]. Disponible sur: www.kaggle.com/dataset/5ebeed2d0a1255183c9205b9f2bf8275a07f746d455a605920dc9256468ef225

[7] L. Nataraj, S. Karthikeyan, G. Jacob, et B. S. Manjunath, « Malware images: visualization and automatic classification », in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, New York, NY, USA, juill. 2011, p. 1-7. doi: 10.1145/2016904.2016908.

[8] D. Wu, P. Guo, et P. Wang, « Malware Detection based on Cascading XGBoost and Cost Sensitive », in *2020 International Conference on Computer Communication and Network Security (CCNS)*, août 2020, p. 201-205. doi: 10.1109/CCNS50731.2020.00051.

[9] N. Marastoni, R. Giacobazzi, et M. Dalla Preda, « Data augmentation and transfer learning to classify malware images in a deep learning context », *J. Comput. Virol. Hacking Tech.*, vol. 17, nº 4, p. 279-297, déc. 2021, doi: 10.1007/s11416-021-00381-3.

[10] F. O. Catak, J. Ahmed, K. Sahinbas, et Z. H. Khand, « Data augmentation based malware detection using convolutional neural networks », *PeerJ Comput. Sci.*, vol. 7, p. e346, 2021, doi: 10.7717/peerj-cs.346.

[11] R. Burks, K. A. Islam, Y. Lu, et J. Li, « Data Augmentation with Generative Models for Improved Malware Detection: A Comparative Study », in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, oct. 2019, p. 0660-0665. doi: 10.1109/UEMCON47517.2019.8993085.

[12] A. Tekerek et M. M. Yapici, « A novel malware classification and augmentation model based on convolutional neural network », *Comput. Secur.*, vol. 112, p. 102515, janv. 2022, doi: 10.1016/j.cose.2021.102515.

[13] Y.-W. Ma, J.-L. Chen, W.-H. Kuo, et Y.-C. Chen, « AI@nti-Malware: An intelligent framework for defending against malware attacks », *J. Inf. Secur. Appl.*, vol. 65, p. 103092, mars 2022, doi: 10.1016/j.jisa.2021.103092.

[14] C. Shorten et T. M. Khoshgoftaar, « A survey on Image Data Augmentation for Deep Learning », *J. Big Data*, vol. 6, nº 1, p. 60, juill. 2019, doi: 10.1186/s40537-019-0197-0.

[15] P. Chlap, H. Min, N. Vandenberg, J. Dowling, L. Holloway, et A. Haworth, « A review of medical image data augmentation techniques for deep learning applications », *J. Med. Imaging Radiat. Oncol.*, vol. 65, nº 5, p. 545-563, 2021, doi: 10.1111/1754-9485.13261.

[16] F. Farahanipad, M. Rezaei, M. S. Nasr, F. Kamangar, et V. Athitsos, « A Survey on GAN-Based Data Augmentation for Hand Pose Estimation Problem », *Technologies*, vol. 10, nº 2, Art. nº 2, avr. 2022, doi: 10.3390/technologies10020043.

[17] L. Nanni, M. Paci, S. Brahnam, et A. Lumini, « Feature transforms for image data augmentation », arXiv, arXiv:2201.09700, janv. 2022. doi: 10.48550/arXiv.2201.09700.

[18] *split-folders · PyPI*. Consulté le: 9 juin 2022. [En ligne]. Disponible sur: https://pypi.org/project/split-folders/

[19] F. Chollet, « Xception: Deep Learning With Depthwise Separable Convolutions », présenté à Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, p. 1251-1258. Consulté le: 14 novembre 2022. [En ligne]. Disponible sur: https://openaccess.thecvf.com/content_cvpr_2017/html/Chollet_Xception_Deep_Learning_CVPR_2017_paper.html

[20] K. He, X. Zhang, S. Ren, et J. Sun, « Deep Residual Learning for Image Recognition ». arXiv, 10 décembre 2015. Consulté le: 9 septembre 2022. [En ligne]. Disponible sur: http://arxiv.org/abs/1512.03385

[21] I. B. A. OUAHAB, *This is a step by step guide to install latest version of TensorFlow on GPU*. 2022. Consulté le: 27 mars 2022. [En ligne]. Disponible sur: https://github.com/ikrambenabdelouahab/Manual_tf_GPU