

Improve malware classifiers performance using cost-sensitive learning for imbalanced dataset

Ikram Ben Abdel Ouahab, Lotfi Elaachak, Mohammed Bouhorma

Computer Science, Systems and Telecommunication Laboratory (LIST), Faculty of Sciences and Techniques,
University Abdelmalek Essaadi, Tangier, Morocco

Article Info

Article history:

Received Sep 5, 2022

Revised Jan 21, 2023

Accepted Mar 10, 2023

Keywords:

Convolutional neural network

Cost-sensitive

Cybersecurity

Deep learning

Malware classification

ABSTRACT

In recent times, malware visualization has become very popular for malware classification in cybersecurity. Existing malware features can easily identify known malware that have been already detected, but they cannot identify new and infrequent malwares accurately. Moreover, deep learning algorithms show their power in term of malware classification topic. However, we found the use of imbalanced data; the Maling database which contains 25 malware families don't have same or near number of images per class. To address these issues, this paper proposes an effective malware classifier, based on cost-sensitive deep learning. When performing classification on imbalanced data, some classes get less accuracy than others. Cost-sensitive is meant to solve this issue, however in our case of 25 classes, classical cost-sensitive weights wasn't effective is giving equal attention to all classes. The proposed approach improves the performance of malware classification, and we demonstrate this improvement using two Convolutional Neural Network models using functional and subclassing programming techniques, based on loss, accuracy, recall and precision.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Ikram Ben Abdel Ouahab

Computer science, systems and telecommunication laboratory (LIST), Faculty of Sciences and Techniques

University Abdelmalek Essaadi

Tangier, Morocco

Email: ibenabdelouahab@uae.ac.ma

1. INTRODUCTION

Malware is malicious code designed to install covertly on a target system. The malicious intention could be: destroying data, installing additional malicious programs, exfiltrating data, or encrypting data to get a ransom [1]. Malware compromise the confidentiality, integrity, and availability of the user's data. The landscape of malware is constantly evolving. In the past, malware was typically created to be fast and easily detectable, often carrying out destructive actions shortly after infecting a system [2], [3]. Older types of malware had specific procedures for dealing with different types of infections. However, today's malware is designed to be stealthy and difficult to detect. It spreads slowly over time, gathering information over a longer period before exfiltrating it. Modern-day malware tends to utilize a single set of procedures, as most attacks are blended and incorporate multiple methods [4], [5].

In cybersecurity, the use of artificial intelligence (AI) is being necessary [6]–[8]. Many works are focusing on solving the imbalanced data issue in literature [9], [10]. Since, most algorithms are designed to work well with balanced databases. Recently, most researcher work with malware visualization technique. This method deals indirectly with the malicious code. The main idea is to visualize a malicious binary executable as a grayscale or colored image. These images are presented as arrays in the range of (0, 255). This technique was initiated the very first time by Nataraj in 2011 [11], where they deliver the Maling database which contains

directly 9,369 malware images in 25 classes. Researchers use many methodologies [12], [13], where the common thing is the very first step of malware visualization (dealing with images). A malware detection method was proposed by Di Wu [14], which utilized cascading extreme gradient boosting (XGBoost) and cost-sensitive techniques to handle unbalanced data. The method used extracted application programming interface calls (API) from portable executable (PE) files as features, and adopted a three-tier cascading XGBoost approach for data balancing and model training. Di Wu used a database that contained two classes for malicious and benign API calls, achieving a high accuracy of 99% with this method. In a separate study, Roland Burks [15] incorporated generative adversarial to generate synthetic training data for malware detection. Two models were utilized - the generative adversarial network (GAN) and variational autoencoder (VAE)-with the goal of improving the performance of the residual network (ResNet-18) classifier. The addition of synthetic malware samples to the training data resulted in a 2% accuracy improvement for ResNet-18 using VAE, and a 6% accuracy improvement using GAN.

In this paper, we perform malware classification into 25 malware families. To deal with imbalanced data we proposed a new approach to calculate weights as part of the cost-sensitive learning application. Then, we evaluated the proposed approach using two different convolutional neural networks (CNN) models that we developed from scratch using functional and subclassing Keras API. We compare the proposed weights approach with classical approach such as weights calculated using sklearn and random weights value. The overall goal of this work is to increase the performance of the classifier while working with imbalanced data. Finally, we reach our goal and our proposed weights approach performs better than the other techniques and better than without using any cost-sensitive learning approach. *This manuscript* is structures: First, an introduction to malware classification challenges, and how researchers deal with imbalanced data. Second, the proposal description. Third, we defined methods and materials used in the whole approach, then, experimentations and obtained results. Finally, we discuss these results, compare them with others in literature and conclude with future perspectives.

2. PROPOSAL

This article's contribution is to propose a weights approach for cost sensitive to deal with imbalanced data in general and malware image data in particular as shown in Figure 1. We demonstrate that the classical used weight is not effective in the case of too many classes, as we have 25 classes. Then, we evaluated our approach using two CNN models; with functional and subclassing APIs. All the experiments have a common goal to detect and classify malware variants effectively into their corresponding families. Then, we could see clearly the improvement between classical weights and the proposed weights for 25 classes as a use case, in term of classification metrics. So, our main contribution includes,

- Proposing a customized weight for Cost sensitive to deal with Maling imbalanced database.
- Evaluate the cost sensitive approach, using two CNN models and compare with classical approach.

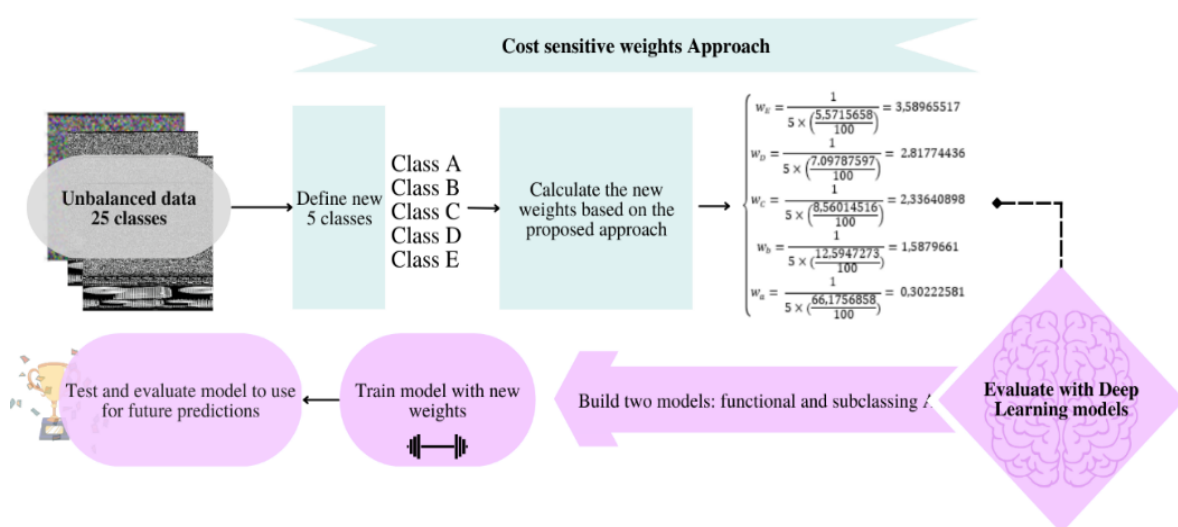


Figure 1. Workflow of the proposed solution

3. METHODS

3.1. Image representation of a malware

Malware visualization is an area focused on detecting, classifying, and presenting malware features in the form of visual cues that can be used to convey more data about a specific malware type. Visualization techniques can use to display static data, monitor network traffic, or manage networks. In [16], the visualization technique is used to discover and visualize malware behavior. Recently, researchers focus on the development of orthogonal methods motivated by signal and image processing to deal with malware variants. They took advantage of the fact that most malware variants have a similar structure, since new malware are simply a variant of existing one's in most cases. So, a malware is treated as digital signals and apply Signal and Image Processing techniques. These techniques are proved to be effective in malware classification and detection in many researches [17]. The traditional way to view and edit malware binaries is by using Hex editors, which show us the byte by byte of the binary file in a hexadecimal format. In [11], authors proposed a new method to view binary files as grayscale image or signal. A malware binary is read as a vector of 8bits unsigned integers as shown in Figure 2. These integers are then organizers to be presented as 2D array. So that, it can be viewed as grayscale image in the range of [0-255]. After converting a malware binary to grayscale image, the image itself keep a significant structure as described in an older work [18]. The binary fragments of a malware show special image textures, and that allow as to classify malware images effectively since years.



Figure 2. Malware visualization process

3.2. Database: Maling

Maling stands for malware images. It's a wide used database [19], in malware classification contexts. Most works cover malware images classification using machine learning and deep learning models. In our case, we have already large data (Total of 9,369 and 25 families); however, the problem is that these data are not balanced as shown in Figure 3. Some classes having a lot of samples; more than 1,200, while others having less than 100 sample. This difference creates an imbalanced dataset. One of the rules in machine learning and deep learning is to balance out the data set or at least get it close to balance. The main reason for this is to give equal priority to each class in laymen terms.

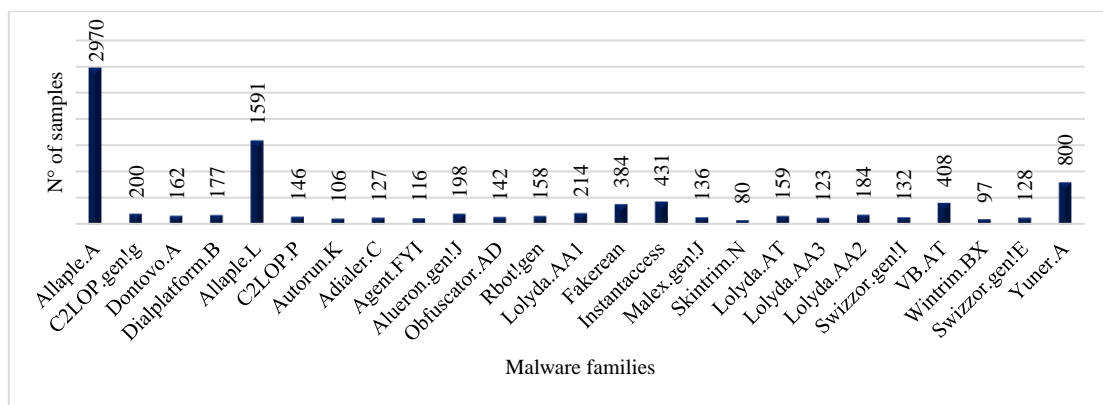


Figure 3. Maling data distribution

3.3. Cost-sensitive learning approach

Methods for addressing class imbalance can be divided into three main categories. **Data level preprocessing:** that operate on the training dataset and change its class distribution using resampling techniques. These methods aim to alter datasets in order to make standard machine learning algorithms work. **Cost-sensitive learning:** here we keep the training dataset unchanged and assign different penalties to the misclassification of samples. Therefore, this will cause the machine learning algorithm to pay more attention

to samples from the minority class. **Ensemble learning**: combines multiple techniques from one or both categories (data level preprocessing and/or cost-sensitive learning). Hence, this method is broadly referred to as ensemble learning and it can be viewed as a wrapper to other methods [20]–[23].

In general, the goal of a machine learning algorithm is to minimize the cost function of loss function (1). In cost-sensitive learning, we modify this cost function to take into account that the cost of a false positive and a false negative may not be the same. We have below the standard cost function for the logistic regression classifier also known as binary cross entropy loss. In logistic regression, we call the positive class 1 and the negative class 0. These values are just for convenience, and doesn't really matter what numerical values we give to each class since we're using two different numbers.

$$\text{cost} = \frac{1}{n} \sum_{i=1}^n -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i) \quad (1)$$

Where:

n is the size of training samples

y_i is the actual labels

\hat{y}_i is the predicted probability

$-y_i \log(\hat{y}_i)$ present the cost for $y_i = 1$ (minority)

$(1 - y_i) \log(1 - \hat{y}_i)$ present the cost for $y_i = 0$ (majority)

For the modified cost function (2), we define two class rates w_1 and w_0 to incorporate the significance of each class in the cost function. In general, w_j is defined as the total number of samples over the number of classes times the number of samples in each class j .

$$\text{cost}_{\text{modified}} = \frac{1}{n} \sum_{i=1}^n -w_1 y_i \log(\hat{y}_i) - w_0 (1 - y_i) \log(1 - \hat{y}_i) \quad (2)$$

Where,

$$w_j = \frac{n}{\text{classes} \times n_j}$$

n is the total number of samples

classes is the number of classes in the dataset

n_j is the number of samples in class j

We take as example a binary classification. To see the difference between the original cost and the modified cost, let's first look at the case of balanced dataset. In this case, we have $n_1 = n_0 = \frac{n}{2}$. If we plug in these two values into the previous equation then,

$$\left. \begin{aligned} w_1 &= \frac{n}{2 \times \frac{n}{2}} = 1 \\ w_0 &= \frac{n}{2 \times \frac{n}{2}} = 1 \end{aligned} \right\} \rightarrow \boxed{w_1 = w_0} \quad (3)$$

here, w_1 and w_0 are identical, which means that we are putting the same weight for making mistakes in terms of false positive and false negative. However, if these two classes are imbalanced, that's mean the minority class has for example 10% of total number of samples. And the majority class has 90%. Then we can plug in these values again into the equation for the weight parameter: $n_1 = \frac{n}{10}$ (minority), and, $n_0 = \frac{9n}{10}$ (majority). So,

$$\left. \begin{aligned} w_1 &= \frac{n}{2 \times \frac{n}{10}} = 5 \\ w_0 &= \frac{n}{2 \times \frac{9n}{10}} = \frac{10}{18} < 1 \end{aligned} \right\} \rightarrow \boxed{w_0 < w_1} \quad (4)$$

We got here that the weight of minority class is greater than the weight of majority class. Based on the cost function that we have before, that's mean we are paying more attention to the minority class. Moving to weights calculation. First, we use the sklearn function in order to compute weights. This function is an implementation of the previous formulas, so there is no need to redo it. As presented in Table 1, these weights are tiny in the range of 10-6. After, that we use random values of 1 and 2 for all the 25 classes. These two weights methods are not effective in our case. That lead us thinking of a new way to compute weights.

Table 1. Compute classical weights

<i>Compute weights with sklearn: Calculate weights from compute_sample_weights [24] function of sklearn.</i>	
Sklearn_weights = [1.919583075689459e-07, 1.919583075689459e-07, 4.089868507531758e-06, 1.919583075689459e-07, 5.578086324574372e-06, 1.919583075689459e-07, 4.314199846567821e-07, 1.919583075689459e-07, 1.8309806929611888e-06, 5.11477323058941e-06, 1.8309806929611888e-06, 1.9391917413870327e-06, 1.9391917413870327e-06, 1.919583075689459e-07, 4.314199846567821e-07, 1.919583075689459e-07, 1.919583075689459e-07, 1.919583075689459e-07, 6.1790973660723525e-06, 1.919583075689459e-07, 1.919583075689459e-07, 9.455790754281687e-07, 4.585617728202409e-06, 7.716454450658827e-06, 4.314199846567821e-07]	
Random_weights = [1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0]	

Above all, we proposed a new approach to calculate weights for multiclass databases. The reason the previous weights were too small is that we divide by 25 classes based on the weight's formula. So, we decided to redistribute classes in order to increase these weights to be more relevant. In our approach, we first arranged classes ascending, then we divide database into 5 classes: class A, class B, class C, class D and class E. In other words, class A contains the majority classes and the class E contains the minority 5 classes. After that, we calculate summary of samples in our new classes, and the percentage of each class over the whole database. Then, we calculated the new weights. But this time we have only 5 classes, so the formula of weight will be. In general, we have the simplified formula of weight for class i (5). As planned, our weights respect the ordering of classing. Based on the new weights given in Table 2, we can say that the model will give more attention with high weight (class E), and it will give less attention to classes with low weights (class A).

$$w_i = \frac{1}{\text{classes} \times N_i} \quad (5)$$

Where,

classes is the number of classes

N_i is the percentage of class i over database

Table 2. The proposed weights approach

Label	Family	N°	New classes	sum	%	weights
16	Skintrim.N	80	class E	522	5.5715658	$w_E = \frac{1}{5 \times \left(\frac{5,5715658}{100}\right)} = 3.58965517$
22	Wintrim.BX	97				
6	Autorun.K	106				
8	Agent.FYI	116				
18	Lolyda.AA3	123				
7	Adialer.C	127	class D	665	7.09787597	$w_D = \frac{1}{5 \times \left(\frac{7,09787597}{100}\right)} = 2.81774436$
23	Swizzor.gen!E	128				
20	Swizzor.gen!I	132				
15	Malex.gen!J	136				
10	Obfuscator.AD	142				
5	C2LOP.P	146	class C	802	8.56014516	$w_C = \frac{1}{5 \times \left(\frac{8,56014516}{100}\right)} = 2.33640898$
11	Rbot!gen	158				
17	Lolyda.AT	159				
2	Dontovo.A	162				
3	Dialplatform.B	177				
19	Lolyda.AA2	184	class B	1,180	12.5947273	$w_B = \frac{1}{5 \times \left(\frac{12,5947273}{100}\right)} = 1.5879661$
9	Alueron.gen!J	198				
1	C2LOP.gen!g	200				
12	Lolyda.AA1	214				
13	Fakerean	384				
21	VB.AT	408	class A	6,200	66.1756858	$w_A = \frac{1}{5 \times \left(\frac{66,1756858}{100}\right)} = 0.30222581$
14	Instantaccess	431				
24	Yuner.A	800				
4	Allaple.L	1591				
0	Allaple.A	2970				

3.4. Convolutional neural network

TensorFlow provides 3 methods for building deep learning models: Sequential API, functional API, and model subclassing. Model subclassing is a high-level API style using pure oriented object programming concept used rarely. This method gives it user the change to customize everything. In contrary of sequential and functional APIs, the model subclassing provides a full control over every nuance of the network and the training process. The first CNN model is composed of different layers types including: Con2D, MaxPooling2D, ZeroPadding2D, dropout, flatten and dense. The construction is given in Figure 4. We train and evaluate the

model without cost sensitive. Here we use Keras TensorFlow functional API. The model architecture is simple with known layers as shown in Figure 4. The second model architecture is given in figure below. First, we build the CNNBlock. Second, we create the ResBlock base on the previous block. Third, we perform the global malware detection model which contain the previous blocks in addition to other wide known layers as MaxPooling, flatten, and dense. We train, and evaluate the model without cost sensitive and using default then the proposed weights. Here, we have more flexibility and options to customized in term of coding as shown in Figure 5.



Figure 4. CNN model 1 architecture (functional)

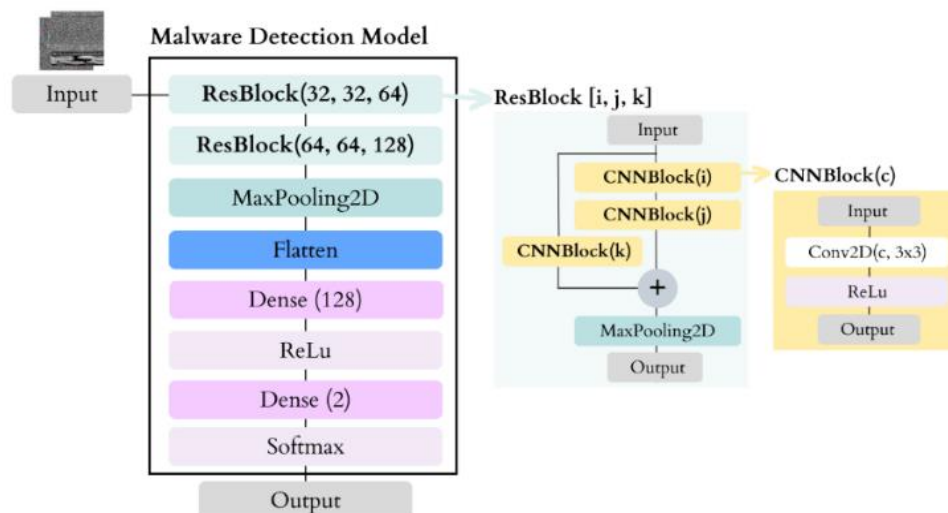


Figure 5. Malware detection model using CNN subclassing

3.4. Tools

Powerful hardware is essential for image processing, and in our laboratory, we use the NVIDIA Quadro T1000 with Max-Q graphics processing unit (GPU) workstation due to its high compute capability of 7.5, which allows us to process images quickly compared to other devices. For deep learning using TensorFlow, we found that installing compute unified device architecture (CUDA) and CUDA deep neural network (cuDNN) on the GPU environment was necessary. We also installed additional required Python packages. While attempting to use a simple Conda command for installation, we encountered numerous errors, leading us to recommend a manual installation and configuration to save time and ensure successful installation. A manual installation guide for TensorFlow can be found in reference [25].

4. RESULTS AND DISCUSSION

As a result of using cost sensitive, we demonstrate that effectively cost sensitive technique allows us to improve the performance of malware classifier. We compare four evaluation metrics using 2 deep learning models, a functional CNN model and a subclassing CNN model. Then, for both models we used different cost-sensitive methods: *our proposed weights*, *random weights*, *sklearn function-based weights*, and *without cost-sensitive*. Each time, we computed classical evaluation metrics: loss, accuracy, precision and recall. Hence, the best performances go to the subclassing CNN model with cost sensitive using our proposed weights approach.

The loss is 1%, the accuracy is 98.46%, the precision is 98.5%, and the recall is 98.42%. These values retain to be the best over several experimentation tests. In addition, cost sensitive using out proposed weights approach gives also best results comparing to the other methods for the function CNN model. So, we proved the efficacy of this approach in the case of many classes, 25 classes in our case. However, classical weights calculated from sklearn function are the worst with the first model, and average with the second model. Results are presented in Figure 6 and Table 3.

Table 3. Results of cost-sensitive implementation

	<i>Functional CNN</i>			
	<i>Without cost sensitive</i>	<i>Our proposed weights approach</i>	<i>Cost sensitive using random weights</i>	<i>Cost sensitive using computed weights (sklearn)</i>
Loss	0.1229	0.0885	0.0964	1.5832
Accuracy	0.9790	0.9798	0.9717	0.6355
Precision	0.9814	0.9806	0.9734	0.7022
Recall	0.9764	0.9790	0.9717	0.6231
<i>Subclassing CNN</i>				
Loss	0.0114	0.0101	0.0107	0.0079
Accuracy	0.9747	0.9846	0.9769	0.9794
Precision	0.9751	0.9850	0.9769	0.9799
Recall	0.9747	0.9842	0.9769	0.9794

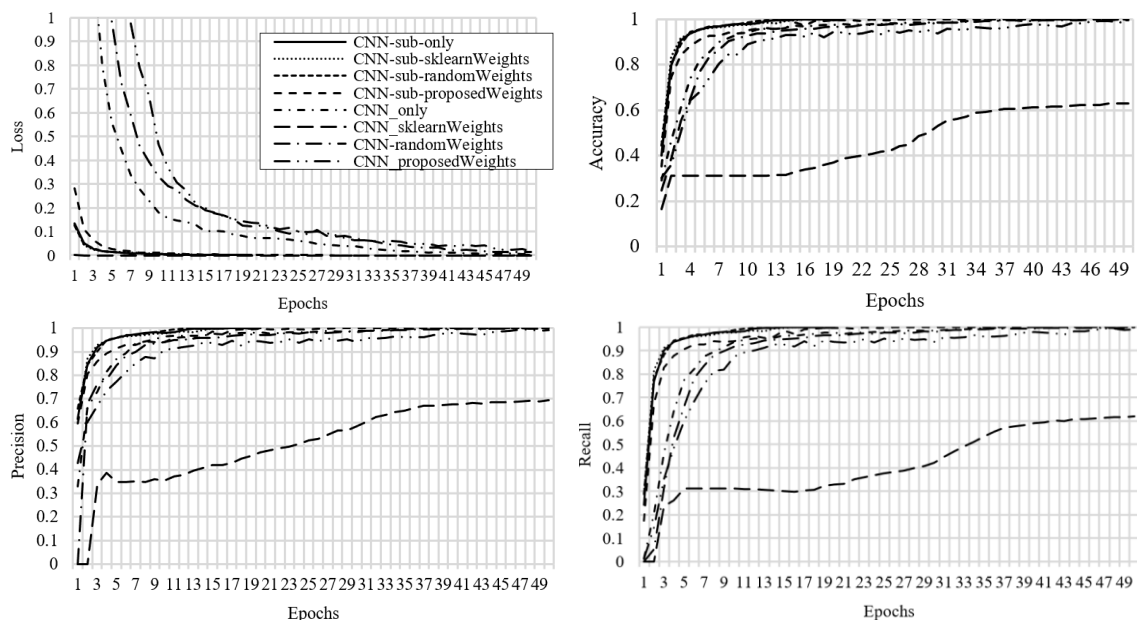


Figure 6. Loss, accuracy, precision, and recall curves using various techniques

So, customized weights are effective with Maling imbalanced database. In general, we can apply the same approach to deal with any imbalanced data. The point here is to not use default weights especially, when we are working with multiclass database, we can rebuild classes to calculate weights. The proposed weights approach here gives a details calculation for 25 classes. In other context the same idea could be applied. In literature, most of works using cost-sensitive in different domains and application shows the improvement while using this technique to deal with imbalanced data [26], [27]. For instance, in [14] the use of cost-sensitive was implemented for binary classification, and the obtained result reach 99%. Then, for Maling imbalanced database, researchers in [15] used GAN which also give acceptable results (90%). In our paper, we proposed the cost-sensitive weights approach to deal with Maling imbalance data and the given result is 98%. The most important thing while doing this is that all classes have same attention and weights, so, even if a class has few samples, we gave it a good weight and the classifier was able to recognize this class more effectively than before. The final performance of the overall model without cost-sensitive or any other technique that deal with imbalance data, could be very high, but when we give in details, we found that the model lack to recognize effectively or with high accuracy some classes (mainly those with less data) [28], [29].

5. CONCLUSION

Summing up, in this work, we investigate cost-sensitive learning for advancing the classification of imbalanced data. A new cost-sensitive weights computation was proposed and evaluated using 2 CNN models along with evaluation metrics. The main goal is to improve the performance of malware classification into their corresponding families. So, we proposed a new approach for cost sensitive using customized weights approach to deal with unbalanced database. We order the classes by the number of samples, then we make subclasses where each new class englobe 5 of the malware classes. Then we compute weights, here the new weights will be given to all malware families belonging to the new subclass. The idea is to give more attention to classes having few samples. After that, we compare the proposed weights to the classical computed weights. When applying the proposed weights, the model performance improved clearly using both CNN models one by one. As a conclusion, we recommend to use customized weights in the case of many classes e.g. 25 classes, in order to improve the performance overall, and especially the performance withing minority classes. The best results in this paper is related to the customized approach of cost sensitive with CNN subclassing model where we have improved the accuracy with +0.1% (and so with other metrics). As future work, we aim to develop a framework based one our methods to defend again malwares using malware images and deep learning. Also, we are looking forward to use GAN as data augmentation technique and compare it to actual findings. Moreover, 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.

ACKNOWLEDGMENTS

This work was supported by the “Centre National pour la Recherche Scientifique et Technique”, CNRST, Morocco.




REFERENCES

- [1] P. W. Singer and A. Friedman, “Cybersecurity: What everyone needs to know,” *Oxford University Press*, no. September, pp. 1–7, 2018, [Online]. Available: https://www.researchgate.net/profile/Sushma-Rao-4/publication/354907006_Cybersecurity_What_Everyone_needs_to_know_Cybersecurity_What_Everyone_needs_to_know/links/6153a90b14d6fd7c0fb7a705/Cybersecurity-What-Everyone-needs-to-know-Cybersecurity-What-Everyone-2018.pdf.
- [2] M. Macas, C. Wu, and W. Fuertes, “A survey on deep learning for cybersecurity: Progress, challenges, and opportunities,” *Computer Networks*, vol. 212, 2022, doi: 10.1016/j.comnet.2022.109032.
- [3] A. Gaurav, B. B. Gupta, and P. K. Panigrahi, “A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system,” *Enterprise Information Systems*, vol. 17, no. 3, 2023, doi: 10.1080/17517575.2021.2023764.
- [4] F. L. Barsha and H. Shahriar, “Mitigation of malware using artificial intelligence techniques,” *Security Engineering for Embedded and Cyber-Physical Systems*, pp. 221–234, 2022, doi: 10.1201/9781003278207-13.
- [5] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, and J. F. Connolly, “Cybersecurity threats and their mitigation approaches using machine learning- A review,” *Journal of Cybersecurity and Privacy*, vol. 2, no. 3, pp. 527–555, 2022, doi: 10.3390/jcp2030027.
- [6] S. E. Donaldson, S. G. Siegel, C. K. Williams, and A. Aslam, “Enterprise cybersecurity study guide,” *Enterprise Cybersecurity Study Guide*, 2018, doi: 10.1007/978-1-4842-3258-3.
- [7] S. MahdaviFar and A. A. Ghorbani, “Application of deep learning to cybersecurity: A survey,” *Neurocomputing*, vol. 347, pp. 149–176, 2019, doi: 10.1016/j.neucom.2019.02.056.
- [8] L. F. Sikos, “AI in cybersecurity,” *Springer*, 2018, doi: 10.1007/978-3-319-98842-9.
- [9] J. H. Lee and K. H. Park, “GAN-based imbalanced data intrusion detection system,” *Personal and Ubiquitous Computing*, vol. 25, no. 1, pp. 121–128, 2021, doi: 10.1007/s00779-019-01332-y.
- [10] Y. Fu, Y. Du, Z. Cao, Q. Li, and W. Xiang, “A deep learning model for network intrusion detection with imbalanced data,” *Electronics (Switzerland)*, vol. 11, no. 6, 2022, doi: 10.3390/electronics11060898.
- [11] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” *ACM International Conference Proceeding Series*, 2011, doi: 10.1145/2016904.2016908.
- [12] I. Obaidat, M. Sridhar, K. M. Pham, and P. H. Phung, “Jadeite: A novel image-behavior-based approach for Java malware detection using deep learning,” *Computers and Security*, vol. 113, 2022, doi: 10.1016/j.cose.2021.102547.
- [13] T. Sree Lakshmi, M. Govindarajan, and A. Sreenivasulu, “Malware visual resemblance analysis with minimum losses using Siamese neural networks,” *Theoretical Computer Science*, vol. 943, pp. 219–229, 2023, doi: 10.1016/j.tcs.2022.07.018.
- [14] D. Wu, P. Guo, and P. Wang, “Malware detection based on cascading XGboost and cost sensitive,” *Proceedings - 2020 International Conference on Computer Communication and Network Security, CCNS 2020*, pp. 201–205, 2020, doi: 10.1109/CCNS50731.2020.00051.
- [15] R. Burks, K. A. Islam, Y. Lu, and J. Li, “Data augmentation with generative models for improved malware detection: A comparative study,” *2019 IEEE 10th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2019*, pp. 0660–0665, 2019, doi: 10.1109/UEMCON47517.2019.8993085.
- [16] K. Han, J. H. Lim, and E. G. Im, “Malware analysis method using visualization of binary files,” *Proceedings of the 2013 Research in Adaptive and Convergent Systems, RACS 2013*, pp. 317–321, 2013, doi: 10.1145/2513228.2513294.
- [17] I. B. A. Ouahab, M. Bouhorma, L. El Achak, and A. A. Boudhir, “Towards a new cyberdefense generation: Proposition of an intelligent cybersecurity framework for malware attacks,” *Recent Advances in Computer Science and Communications*, vol. 15, no. 8, pp. 1026–1042, 2020, doi: 10.2174/2666255813999201117093512.
- [18] G. Conti et al., “A visual study of primitive binary fragment types,” *Black Hat USA*, pp. 1–17, 2010.




- [19] “Maling (Original),” [Online]. Available: <https://www.kaggle.com/dataset/bd61bcb9c0540c6a7453a5547f136402d5c59e8ad7021c25efbc5f167d26c597>.
- [20] T. Vanderschueren, T. Verdonck, B. Baesens, and W. Verbeke, “Predict-then-optimize or predict-and-optimize? An empirical evaluation of cost-sensitive learning strategies,” *Information Sciences*, vol. 594, pp. 400–415, 2022, doi: 10.1016/j.ins.2022.02.021.
- [21] G. Petrides, D. Moldovan, L. Coenen, T. Guns, and W. Verbeke, “Cost-sensitive learning for profit-driven credit scoring,” *Journal of the Operational Research Society*, vol. 73, no. 2, pp. 338–350, 2022, doi: 10.1080/01605682.2020.1843975.
- [22] Y. Ding, M. Jia, J. Zhuang, and P. Ding, “Deep imbalanced regression using cost-sensitive learning and deep feature transfer for bearing remaining useful life estimation,” *Applied Soft Computing*, vol. 127, 2022, doi: 10.1016/j.asoc.2022.109271.
- [23] W. Liu, H. Fan, M. Xia, and M. Xia, “A focal-aware cost-sensitive boosted tree for imbalanced credit scoring,” *Expert Systems with Applications*, vol. 208, 2022, doi: 10.1016/j.eswa.2022.118158.
- [24] Scikit-learn.org, “Sklearn.Utils.Class_Weight.Compute_Sample_Weight,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_sample_weight.html?highlight=sample#sklearn.utils.class_weight.compute_sample_weight.
- [25] I. B. A. Ouahab, “This is a step by step guide to install latest version of TensorFlow on GPU,” [Online]. Available: https://github.com/ikrambenabdelouahab/Manual_tf_GPU.
- [26] F. Liu and Q. Qian, “Cost-sensitive variational autoencoding classifier for imbalanced data classification,” *Algorithms*, vol. 15, no. 5, 2022, doi: 10.3390/a15050139.
- [27] N. Gupta, V. Jindal, and P. Bedi, “CSE-IDS: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems,” *Computers and Security*, vol. 112, 2022, doi: 10.1016/j.cose.2021.102499.
- [28] G. Aguiar, B. Krawczyk, and A. Cano, “A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework,” Apr. 2022, [Online]. Available: <http://arxiv.org/abs/2204.03719>.
- [29] S. J. Basha, S. R. Madala, K. Vivek, E. S. Kumar, and T. Ammannamma, “A review on imbalanced data classification techniques,” *2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022*, 2022, doi: 10.1109/ICACTA54488.2022.9753392.

BIOGRAPHIES OF AUTHORS






Ikram Ben Abdel Ouahab    received her master degree in Computer Systems and Networks from Faculty of Sciences and Techniques of Tangier, Morocco. She is currently working toward her PhD degree in LIST Laboratory of FSTT, University Abdelmalek Essaadi, Tangier, Morocco. Her main research interests include cybersecurity, malware analysis, artificial intelligence, and IoT. She participated in many international conferences, and have published more than 10 scientific papers in 3 years with LIST Laboratory team. In August 2022, she graduates from CPITS (Certification Program in IT Security) program provided by Trend Micro, an intensive 9 weeks training in cybersecurity technologies and industry trends, including certification in industry leading solutions for endpoint, cloud, and network security. She is a fresh AWS (Amazon Web Services) Solution Architect certified. She can be contacted at email: ibenabdelouahab@uae.ac.ma.



Prof. Dr. Lotfi Elaachak    is an Assistant Professor, Doctor at the Faculty of Sciences and Technologies, University Abdelmalek Essaadi, Tangier. His recent research and policy interests concentrate broadly in the area of serious game, augmented reality, e-learning, machine learning/deep learning, and nlp for education. He can be contacted at email: lelaachak@uae.ac.ma.



Prof. Dr. Mohammed Bouhorma    is an experienced academic who has more than 25 years of teaching and tutoring experience in the areas of Information Security, Security Protocols, AI, Big Data and Digital Forensics at Abdelmalek Essaadi University. He received his M.S. and Ph.D. degrees in Electronic and Telecommunications from INPT in France. He has held a Visiting Professor position at many Universities (France, Spain, Egypt and Saudi Arabia). His research interests include, Cybersecurity, IoT, Big Data Analytics, AI, Smart Cities technology and serious games. He is an editorial board member for over a dozen of international journal and has published more than 100 research papers in journals and conferences. He can be contacted at email: mbouhorma@uae.ac.ma.