Robustness evaluation of convolutional neural networks for malware classification

Vincenzo Carletti[1], Antonio Greco[1], Alessia Saggese[1] and Mario Vento[1]

[1] *Dept. of Computer Engineering, Electrical Engineering and Applied Mathematics, University of Salerno, Italy*

Abstract

In a world increasingly connected with smart devices, smartphones, tablets and servers in constant communication with each other, malware is a serious threat for the security of users and systems. Every day they are becoming more sophisticated and can rely on a growing attack surface. Traditional malware analysis techniques are becoming unable to deal with this growth; to this reason new approaches are arising. Among these, the most promising ones aim to exploit the disruptive accuracy and flexibility of convolutional neural networks (CNNs) to realize innovative techniques to detect and classify malware by using an intermediate image-based representation. However, several papers have highlighted the natural tendency of CNNs to be fooled by perturbations applied on the input. In this paper we benchmark four different CNNs widely used for images. To this purpose, we have specialized the CNNs, through transfer learning, to classify malware belonging to 9 different families. Then, we have evaluated their robustness against the obfuscation of the malware executable. All the CNNs achieved an impressive classification accuracy on both the original and the obfuscated datasets confirming their suitability for malware classification.

Keywords

Image-based malware analysis, Malware classification, Convolutional Neural Networks

1. Introduction

Any software intentionally designed to affect the integrity and the functionality of a digital system in order to cause harm to users or other systems is classified as a malicious software, namely a malware. Different families of harmful software lie under the definition of malware, divided according to their functionalities [1]: virus, adware, ransomware, backdoor, trojan are among them. Until a few years ago, most of the targets were mainly servers or personal computers, but nowadays the scenario is completely changed. In fact, as highlighted in a recent thread reports from Symantec [2] and Avira [3], the widespread diffusion of smart devices constantly connected to the network which communicate with personal computers and cloud services has enormously increased the opportunities to perform an attack. Therefore, effective and adaptive methods are required to deal with the incessant growing of malware variants.

Most of the state-of-the-art approaches are based on traditional machine learning techniques [4, 5], in which the features to distinguish a malware from a benign software or classify

ITASEC21 - ITALIAN CONFERENCE ON CYBERSECURITY

vcarletti@unisa.it (V. Carletti); agreco@unisa.it (A. Greco); asaggese@unisa.it (A. Saggese); mvento@unisa.it (M. Vento)

0000-0002-9130-5533 (V. Carletti); 0000-0002-5495-2432 (A. Greco); 0000-0003-4687-7994 (A. Saggese); 0000-0002-2948-741X (M. Vento)

CEUR Workshop Proceedings (CEUR-WS.org)

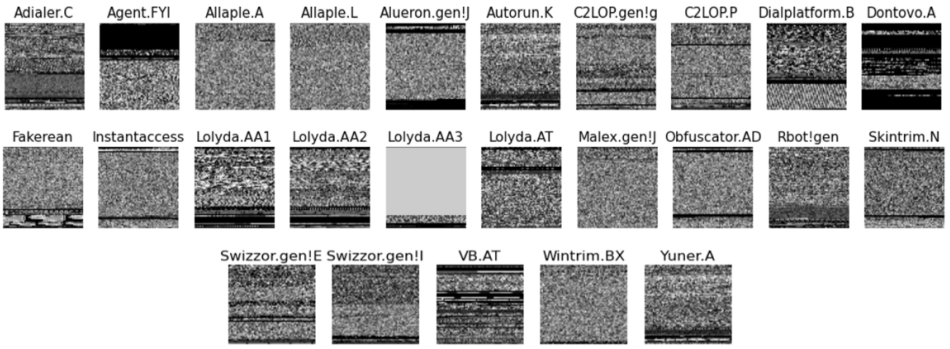

Figure 1: Example of images extracted from malware binary that are contained in the dataset MalImg [7]

the family it belongs to are manually selected by expert analysts. These methods usually rely on static or dynamic analysis of the malware. In the former case, the malware is analyzed considering the metadata of the executable, the assembly code instructions and binary data; the main drawback of the static approach is the necessity to disassemble the executable, which can be a complex and time consuming process. On the other side, the dynamic analysis requires to execute the malware in a sandbox, a virtual safe environment like a Petri dish where it can not damage the underling system but can be easily monitored. Also in this case there is a downside, since the setup of the virtual environment can be very complex and getting an outcome is time demanding.

Unfortunately, as discussed in [1, 6, 5], these traditional approaches are becoming unable to deal with the huge variety of malware. As a consequence, new approaches are coming into play [6, 5]. Among these, the Image-based Malware Analysis is the most promising one; the main idea of this approach is to represent the executable of a malware as a gray-scale image [7] or as a RGB image [8]. It is important to note that the real innovation of this approach lies in the fact that it does not require neither to disassemble the executable nor to configure complex sandboxes, and allows to exploit successful and accurate methods coming from the image analysis.

Once we have an image, different pattern recognition and machine learning methods can be applied to perform the classification task. In [7], the first paper proposing an image-based method, the authors extracted texture features from the image through a wavelet decomposition [9]; then, the classification is performed using a k-Nearest Neighbour. Tucher et al. in [10] propose to use local neighborhood binary patterns (LNBP) together with a principal component analysis (PCA) to select the features and a linear discriminant analysis (LDA) to classify the malware. However, these methods based on traditional pattern recognition techniques do not solve the problem of a hand-crafted feature selection that requires to have a deep experience about image analysis and malware. An attempt to face this problem is discussed in [11], by extracting hybrid features and then using a Support Vector Machine (SVM) for classification.

Except for a few papers, the trend is to exploit deep neural networks [6] that have been demonstrated to be extremely effective on image classification tasks without the need of per-

forming a feature analysis. The most immediate way, proposed in [12, 1, 13, 14, 8], is to use Convolutional Neural Networks (CNN) already available for other tasks such as ResNet50 [15], InceptionV3 [16], VGG16 [17] and MobileNet [18] and specialize them on malware images through transfer learning. In particular, Vasan et al. [8] propose a malware classification system, named IMCFN, that uses VGG16 to obtain an embedding of the malware image and two fully-connected layers to perform the classification. Another interesting solution, proposed in [19, 20], is to realize an ensemble of multiple CNNs and then combine the output of different networks to address the problem.

Although there are undeniable benefits in using CNNs to classify malware, they can be also very sensitive to perturbation of the input, as demonstrated by the possibility to generate adversarial examples [21, 22] able to properly force the outcome of the network. Therefore, together with the accuracy of the system, it is also essential to evaluate the robustness against techniques aimed at modifying the malware to fool the classification. It is worth to note that common methods which generate adversarial examples or distribution shifts through augmentation, like those used in [14], designed for standard images, are not meaningful in the case of malware images. In fact, they are not designed to generate an image that is still the representation of the same malware and a valid executable. For this reason, the perturbation must be applied not on the resulting image, but on the original executable using methods like the obfuscation.

In this paper we discuss a robustness analysis against obfuscation performed considering four CNNs, namely ResNet50 [15], InceptionV3 [16], VGG16 [17] and MobileNet [18], that are widely adopted on images and used as base to realize image-based malware classification systems. To this purpose, we have retrained the CNNs, through transfer learning, to classify malware belonging to 9 different families from the dataset BIG2015 [23]. The latter has been published by Microsoft during the Malware Classification Challenge and, differently from other datasets like MalImg [7], it also provides the binary code. We have extended the BIG2015 dataset by generating an obfuscated version of the samples it contains, in order to analyze the robustness of the considered CNNs.

In the following sections we describe the setup realize to perform the proposed analysis and the experimental results confirming the effectiveness of the image-based approaches.

2. System Setup

Assessing the robustness of a machine learning system requires two steps, that we address in this paper: i) the performance evaluation of the system; ii) the robustness evaluation against perturbations of the input. Thus, in this section we detail the considered malware classification system (see Subsections 2.1) and describe the obfuscation techniques we introduce for assessing the stability of the system (see Subsections 2.2).

2.1. Malware classification system

As mentioned before, we consider an image-based malware classification system in which the analysis of the image is performed through a CNN (see Figure 2). The system adopts an intermediate representation based on gray-scale image, firstly introduced in [7]. This is justified

File Size Range	Image Width
<10kB	32
10kB - 30kB	64
30kB - 60kB	128
60kB - 100kB	256
100kB - 200kB	384
200kB - 500kB	512
500kB - 1000kB	768
>1000kB	1024

Table 1
The table reports the correspondence considered for computing the width of the image which represents the malware. On the left we report the file size range, while on the right the corresponding image width.

by the observation that, as visually confirmed in Figure 1, malware samples belonging to the same family have a similar visual appearance while those of different families have not.

In more details, starting from the hexadecimal representation of an executable file, each byte is converted into an integer which can varies in a range between 0 and 255. Subsequently, each integer is inserted into an array, that is successively reshaped into a two-dimensional matrix. This matrix represents the grey scale image. In [7] the authors also experimentally evaluate how to fix the width of the matrix. Indeed, they propose to vary this parameter, depending on the whole image size, and in particular depending on the size of the file, as summarized in Table 1. Subsequently, the number of pixels composing the height is obtained by dividing the file size by the width.

Given the image, we consider four widely adopted CNN architectures, namely VGG16 [17], ResNet50 [24], Xception [25] and MobileNet [18]. This choice has been made so as to consider (i) networks of different dimensions, namely large (VGG16), medium sized (ResNet50, Xception) and small networks (MobileNet), thus characterized by different computational requirements and processing times; (ii) networks based on different concepts, from traditional convolutional layers (VGG16) to more modern blocks inspired by Network-In-Network architectures, respectively based on residual blocks (ResNet50) and on depthwise separable convolutional layers (Xception, MobileNet).

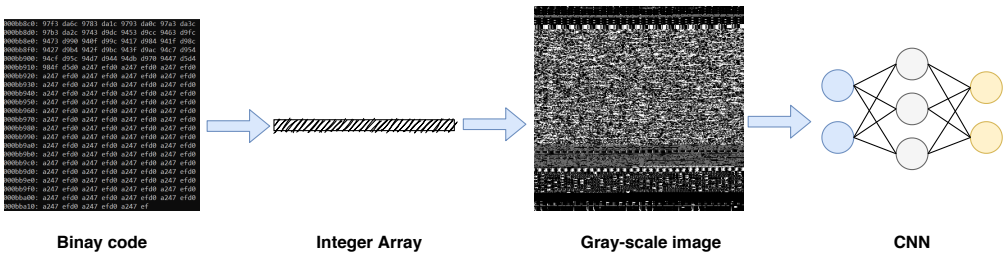

Figure 2: Overview of the considered malware classification system. The binary code is represented as an array of integers (values between 0 and 255), which is then arranged in a square gray-scale image, whose width depends on the original size of the malware. Finally, the image is fed to a CNN trained for malware classification.

For the sake of clearness, VGG16 is the biggest network we considered. It is composed by a stack of convolutional layers, followed by fully connected layers. The convolutional layers employ filters with a very small receptive field, namely 3×3, which is the smallest size to capture the notion of left/right, up/down, center.

ResNet is based on the concept of Residual Blocks; typically, in a deep convolutional neural network, several layers are stacked; the network learns low/middle/high level features at the end of each layer. In residual learning, the residuals are learnt instead than the features. Residual can be seen as the subtraction of feature learned from input of that layer.

Xception is a simplified version of the Inception network (Xception stands for *eXtreme Inception*). It is composed by depthwise separable convolutional layers structured into modules, all of which have linear residual connections around them, except for the first and last modules.

Finally, MobileNet is the smallest network (only 16 MB required for storing), designed for being efficient on mobile and embedded devices. Like Xception, it is based on depthwise separable convolutions; this is a form of factorized convolutions, able to factorize a standard convolution into a depthwise convolution and a pointwise convolution.

Similarly to [8], we removed the top layers of the original CNNs and added four new layers: two fully connected layers with 2048 neurons; a dropout layer for regularization purposes; a fully connected layer, responsible for the classification, with a softmax activation function and a number of neurons equal to the number of considered malware categories. For all the CNNs we performed transfer learning, by training only the weights of the four additional layers and freezing all the convolutional part of the networks with the weights pre-trained over ImageNet.

2.2. Obfuscation

As for their biological version, the first need of a malware is to extend its lifetime and be able to infect as much targets as possible. To this aim, a malware must be able to evade the defenses of the attacked system and hopefully perform its job without being detected and removed. Therefore, the camouflage is an essential characteristic for a malware to survive in the wild. There are four main stealth methodologies: encryption, oligomorphism, polymorphism, and metamorphism. in this paper, we focus our analysis on metamorphic techniques because these can be applied directly on the hexadecimal representation of a binary file. In more details, we adopt a dead code insertion. We ensure that the junk code instructions are inserted into the text section, which contains the instructions of the file itself. The adopted algorithm is reported in Algorithm 1, while the list of instructions considered is listed in Table 2.

As we can see, the algorithm works as follows: for each instruction in the *text section* of the binary file, and if the maximum number of allowed dead instructions (namely *max_insertions*) has not been reached, it adds an obfuscation dead code sequence with a uniform random probability (namely *insertion_probability*). Also, the dead code sequence has a random length, which varies between 1 and *sequence_max_len*. As constrain it has been chosen to insert a dead code instruction in a specific junk code sequence only one time.

Algorithm 1 Outline of the procedure used to obfuscate the binary file of a malware. The output of the procedure is a new binary file containing the instructions of the original malware with the random addition of junk instructions.

Input: *file*, *junk_instructions*, *insertion_probability*, *max_insertions*, *sequence_max_len*
Output: *obfuscated_file*

function Obfuscation(*file*, *junk_instructions*, *insertion_probability*, *max_insertions*, *sequence_max_len*)
    count_insertion ← 0
    for *row* in *file* **do**
        Insert *row* in *obfuscated_file*
        if *section* = 'text' and *count_insertion* < *max_insertions* **then**
            Take *p* uniformly chosen in the range [0,1]
            if *insertion_probability* >p **then**
                Choose a random integer *sequence_len* in the range [1,*sequence_max_len*]
                instructions_set ← []
                i ← 0
                for *i* < *sequence_len* **do**
                    Choose *instr* ∈ *junk_instructions* with *instr* ∉ *instructions_set*
                    Insert *istr* in *obfuscated_file*
                    instructions_set ← *instr*
                    count_insertion ← *count_insertion* + 1
                    i ← *i* + 1
    return *obfuscated_file*

3. Experiments

3.1. Dataset

We used BIG2015 dataset [23] for our malware classification experiments. It includes 10868 malware, belonging to 9 different families: Rammit, Lollipop, Kelihos_ver3, Vundo, Simbda, Tracur, Kelihos_ver1, Obfuscator.ACY, Gatak. Therefore, the dataset contains various types of Worm, Adware, Backdoor, Trojan, TrojanDownloader and Obfuscated malware. The detailed composition of the dataset is reported in Table 3. It points out that the dataset is strongly unbalanced; in fact, half of the dataset consists of Lollipop and Kelihos_ver3 samples, while there are only 42 Simbda samples.

For each malware, the dataset makes available the binary content in hex dump representation without the portable executable header and the disassembled file generated through the IDA Pro software. The latter is important since it allows to understand and analyze the workflow of the malware and to extract handcrafted features.

Table 2
Possible instructions that can be used as junk code to obfuscate the malware.

Assembly Instruction	Binary Instruction
nop	90
inc eax; dec eax	40;48
inc ebx; dec ebx	43;4B
inc ecx; dec ecx	41;49
inc edx; dec edx	42;4A
add eax,0	83 C0 00
add ebx,0	83 C3 00
add ecx,0	83 C1 00
add edx,0	83 C2 00
sub eax,0	8E E8 00
sub ebx,0	83 EB 00
sub ecx,0	83 E9 00
sub edx,0	83 EA 00

3.2. Results

In order to evaluate the robustness of the considered CNNs for malware classification, we applied them over the original dataset and on three different versions obfuscated with three growing levels of severity [0,1,2]. In particular, at severity 0 the maximum length for a junk code sequence is 2, at severity 1 is 4, while at severity 2 is 10. The adopted experimental protocol is a stratified 3-fold cross validation.

The results of our experiments are reported in Table 4. MobileNet achieves the best accuracy over the original dataset (99.25%), but it is also the most robust to obfuscations (95.42% with severity 2). Even Xception obtains good results on the original dataset (99.07%) and on samples obfuscated with severity 0 (95.69%), but it is less robust to stronger obfuscations (94.77% and 93.05% at severity 1 and 2). VGG16 achieves similar performance (98.51% on the original dataset), slightly worse in absolute but suffering less in percentage on the obfuscated samples.

Table 3
Composition of BIG2015 dataset

Family Name	Type	#Samples
Ramnit	Worm	1541
Lollipop	Adware	2478
Kelihos_ver3	Backdoor	2942
Vundo	Trojan	475
Simbda	Backdoor	42
Tracur	TrojanDownloader	751
Kelihos_ver1	Backdoor	398
Obfuscator.ACY	Any kind of obfuscated malware	1228
Gatak	Backdoor	1013

Finally, ResNet achieves substantially worse results on the original dataset (95.48%) and on the obfuscated ones (93.17%, 92.11% and 90.55%).

For the sake of comparison, we have also reported the results of XGBoost [26], a standard machine learning algorithm which is known for being the most efficient among the ones based on handcrafted features. Based on features obtained from the binary source code and from the disassembled malware, it achieves the best accuracy over the original dataset (99.43%) and over low and medium obfuscation levels (96.90% and 96.34% at severity 0 and 1), but it suffers strong obfuscations more than MobileNet (95.22% vs 95.42%).

However, it is worth mentioning that, in the worst case, methods based on CNNs require less than 5 seconds for obtaining the image from the malware and for performing the classification, while XGBoost can require up to 105 seconds for a single sample. The slight accuracy improvement is strongly payed in terms of processing time. Therefore, we can conclude that the method based on MobileNet is surely the best trade-off between accuracy and processing time.

Table 4
Accuracy achived on the original dataset and on the obfuscated dataset using different severities. The drop of accuracy on the obfuscated dataset is reported in brackets.

CNN	Original Dataset	Obfuscated Dataset		
		Severity 0	Severity 1	Severity 2
MobileNet	99.25%	96.62% (2.63%)	95.87% (3.38%)	**95.42%** (**3.83%**)
VGG16	98.51%	95.47% (3.04%)	94.25% (4.26%)	92.98% (5.53%)
Xception	99.07%	95.69% (3.38%)	94.77% (4.40%)	93.05% (6.02%)
ResNet50	95.48%	93.17% (**2.33%**)	92.11% (3.37%)	90.55% (4.93%)
XGBoost[26]	**99.43%**	**96.90%** (2.53%)	**96.34%** (**3.09%**)	95.22% (4.21%)

4. Conclusions

In this paper we have have evaluated the robustness of convolutional neural networks when used on image-based malware classification tasks. The analysis have considered four state-of-the-art CNNs: VGG16, ResNet50, MobileNet, Xception and a standard machine learning approach XGBoost. The CNNs have been tuned to classify malware belonging to 9 different families. The analysis required to realize an extended version of the original BIG2015 dataset, composed of more than 10.000 samples, to include obfuscated malware. The analysis have demonstrated that image-based approaches are able to achieve an impressive accuracy with a limited drop on obfuscated samples. In particular, MobileNet have shown a high accuracy and robustness together with a very short classification time. Therefore, although a more extensive analysis on larger datasets is required, we can conclude that CNNs are enough robust and accurate to be adopted on malware analysis systems.

References

[1] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, K. Sakurai, Lightweight classification of iot malware based on image recognition, in: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), volume 02, 2018, pp. 664–669. doi:10.1109/COMPSAC.2018.10315.

[2] S. Corporation, Symantec internet security threat report, 2020.

[3] A. O. GmbH, Malware threat report:q2 2020, 2020. URL: https://www.avira.com/en/blog/malware-threat-report-q2-2020-statistics-and-trends.

[4] D. Ucci, L. Aniello, Survey on the usage of machine learning techniques for malware analysis, Computers & Security 81 (2017). doi:10.1016/j.cose.2018.11.001.

[5] B. Yadav, S. Tokekar, Recent innovations and comparison of deep learning techniques in malware classification : A review, International Journal on Information Security Science 9 (2021) 230 – 247.

[6] Q. Le, O. Boydell, B. Mac Namee, M. Scanlon, Deep learning at the shallow end: Malware classification for non-domain experts, Digital Investigation 26 (2018) S118–S126. doi:https://doi.org/10.1016/j.diin.2018.04.024.

[7] L. Nataraj, S. Karthikeyan, G. Jacob, B. S. Manjunath, Malware images: Visualization and automatic classification, in: Proceedings of the 8th International Symposium on Visualization for Cyber Security, Association for Computing Machinery, 2011. doi:10.1145/2016904.2016908.

[8] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, Q. Zheng, Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture, Computer Networks 171 (2020) 107138. doi:10.1016/j.comnet.2020.107138.

[9] Torralba, Murphy, Freeman, Rubin, Context-based vision system for place and object recognition, in: Proceedings Ninth IEEE International Conference on Computer Vision, 2003, pp. 273–280 vol.1. doi:10.1109/ICCV.2003.1238354.

[10] T. Tuncer, F. Ertam, S. Dogan, Automated malware recognition method based on local neighborhood binary pattern, Multimedia Tools and Applications (2020) 1 – 18.

[11] H. Naeem, B. Guo, M. R. Naeem, A light-weight malware static visual analysis for iot infrastructure, in: 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), 2018, pp. 240–244. doi:10.1109/ICAIBD.2018.8396202.

[12] S. Yue, Imbalanced malware images classification: a cnn based approach, arXiv preprint arXiv:1708.08042 (2017).

[13] N. Bhodia., P. Prajapati., F. D. Troia., M. Stamp., Transfer learning for image-based malware classification, in: Proceedings of the 5th International Conference on Information Systems Security and Privacy - Volume 1: ForSE,, INSTICC, SciTePress, 2019, pp. 719–726. doi:10.5220/0007701407190726.

[14] M. Nisa, J. Shah, S. Kanwal, M. Raza, M. Khan, R. Damasevicius, T. Blazauskas, Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features, Applied Sciences 10 (2020). doi:10.3390/app10144966.

[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[17] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations, 2015.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).

[19] Y. Lin, X. Chang, Towards interpretable ensemble learning for image-based malware detection, ArXiv abs/2101.04889 (2021).

[20] A. Çayır, U. Ünal, H. Dağ, Random capsnet forest model for imbalanced malware type classification task, Computers & Security 102 (2021) 102133. URL: https://www.sciencedirect.com/science/article/pii/S0167404820304065. doi:https://doi.org/10.1016/j.cose.2020.102133.

[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: International Conference on Learning Representations, 2014. URL: http://arxiv.org/abs/1312.6199.

[22] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 39–57. doi:10.1109/SP.2017.49.

[23] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, M. Ahmadi, Microsoft malware classification challenge, arXiv preprint arXiv:1802.10135 (2018).

[24] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[25] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

[26] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, G. Giacinto, Novel feature extraction, selection and fusion for effective malware family classification, 2016. doi:10.1145/2857705.2857713.