L(a)ying in (Test)Bed

How biased datasets produce impractical results for actual malware families' classification

Tamy Beppler[1], Marcus Botacin[1], Fabrício J. O. Ceschin[1], Luiz E. S. Oliveira[1], and André Grégio[1]

Federal University of Paraná (UFPR), Curitiba, PR – Brazil {tebeppler, mfbotacin, fjoceschin, lesoliveira, gregio}@inf.ufpr.br

Abstract. The number of malware variants released daily turned manual analysis into an impractical task. Although potentially faster, automated analysis techniques (e.g., static and dynamic) have shortcomings that are exploited by malware authors to thwart each of them, i.e., prevent malicious software from being detected or classified accordingly. Researchers then invested in traditional machine learning algorithms to try to produce efficient, effective classification methods. The produced models are also prone to errors and attacks. Novel representations of the "subject" were proposed to overcome previous limitations, such as malware textures. In this paper, our initial proposal was to evaluate the application of texture analysis for malware classification using samples collected in-the-wild in order to compare them with state-of-the-art results. During our tests, we discovered that texture analysis may be unfeasible for the task at hand, if we use the same malware representation employed by other authors. Furthermore, we also discovered that naive premises associated to the selection of samples in the datasets caused the introduction of biases that, in the end, produced unreal results. Finally, our tests with a broader unfiltered dataset show that texture analysis may be impractical for correct malware classification in a real world scenario, in which there is a great variety of families and some of them make use of quite sophisticate obfuscation techniques.

Keywords: Malware classification · Texture analysis · Malware visualization.

1 Introduction

Malware is one of the biggest threats to networked systems. Despite the myriad of defensive tools and techniques, malware writers constantly evolve their code to prevent detection. A multitude of variants emerge from automated malware creation toolkits that, even resulting in samples with few differences from each

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

other, are capable of accomplishing undetectable artifacts [2]. Due to that, security researchers are stuck in an eternal arms race with malware developers, so as to discover new ways for protection whereas understanding incoming menaces.

Classifying a file as malicious requires knowing its features. Analysis techniques are used to extract features that can be used to further malware categorization (by intent, functionality, damage etc.) and better understanding of the threats they pose. However, current malware detection techniques are slow to react to new attacks and threats, since they are based on overwhelming static and/or dynamic analysis to obtain and match malicious patterns against files. To gain speed, Nataraj et al. [20] proposed the use of texture analysis—fast, accurate, and resilient to obfuscation techniques, according to the authors—to change the malware classification problem into an image recognition one.

Image processing techniques are largely used in many fields because they accomplish high precision for pattern recognition [37] tasks. By converting a binary file into a gray-scale texture, we can recognize a pattern for each family of files, thus potentially allowing for malware classification. The goal of this paper is to evaluate state-of-the-art texture analysis techniques for malware classification in a real-life, larger dataset, as well as to validate the claim of resilience against obfuscation. Our main contributions are threefold: (i) we analyze the literature on texture-based malware classification; (ii) we evaluate texture analysis techniques from literature in a publicly available malware dataset, and then in a private, broader, imbalanced dataset composed of malware samples collected in the wild; (iii) we compare these literature texture-based techniques for malware classification regarding the dataset used and the setup of the experiment, in order to discuss our discovery of biased results due to improper testing.

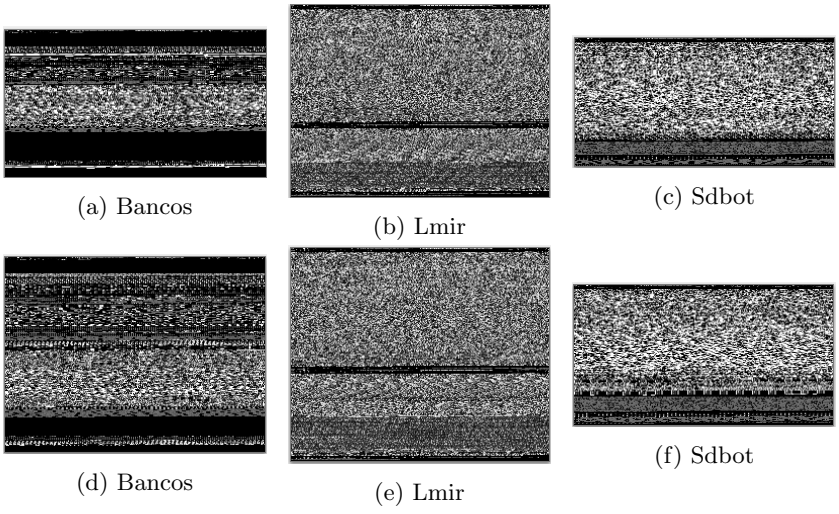
The remainder of this paper is organized as follows. In Section 2 we present the background and related work on texture analysis and malware classification. In Section 3, we show the methodology adopted for our experiments. In Section 4, we describe our tests and analyze the obtained results. In Section 5, we discuss our findings. Finally, in Section 6, we present the concluding remarks.

2 Background and Related Work

There are several ways to represent malicious software. Representation enables the extraction of distinctive features, which allow further classification. It is expected that samples from the same family have similar features. One representation may be the source code or instructions' flow, from which features can be obtained by static analysis. Another representation possible is the malicious program execution trace or the set/sequence of functions or system calls launched in runtime, from which features are extracted by dynamic analysis. The former prevents infection and evasion because the analysis is performed without execution and, according to [20], offers more complete coverage. However, static analysis may fail under binary obfuscation, such as the packing present in approximately 80% of malware samples [19]. Accomplishing wider code coverage may also be difficult if the malware developer uses opaque constants [18]. Both anti-analysis

techniques increase the processing and memory costs, making static analysis unfeasible. The latter actually runs the sample in a controlled environment so as to obtain the execution behavior. Thus, dynamic analysis does not suffer from the same obfuscation techniques (packing, compressing, and encryption of original binary) than static, becoming more effective in certain cases [20]. However, it suffers from other evasion techniques, such as anti-emulation/virtualization, stalling, sleeping, and others, being slower and higly resource consuming. Combined, hybrid approaches were proposed to address the limitations of static and dynamic analysis, but the achieved results were similar in quality and worse in computational cost [6]. Therefore, a different way to extract malware features should be used to avoid evasion and obfuscation techniques, and, in an ideal case, be fast and cost less computing resources.

Malware Texture Analysis. Many applications use features extracted from textures to improve classification and recognition problems [3] [5] [31]. Conti et al. [4] mapped large binary objects by classifying regions using texture analysis and claimed that it can be used for malware identification. Based on that, Nataraj et al. [20] proposed an orthogonal approach using texture representations of malware for classification purposes. Similar to static analysis, texture analysis examines the binary without execution, i.e., it does not suffer with infection, evasion techniques, and obfuscation techniques, according to the claims of [20]. The texture representation preserves similar features even in obfuscated binary and, according to Zhang et al. [37], image recognition methods may improve malware detection. An image texture is a block of pixels with repeated patterns. In a malware texture representation, each byte of the binary file is converted to its correspondent value in a gray-scale image with fixed width and the binary's size divided by the fixed width as height. In Figure 1, we show that samples from the same family may be similar among themselves, but distinctive from other families.

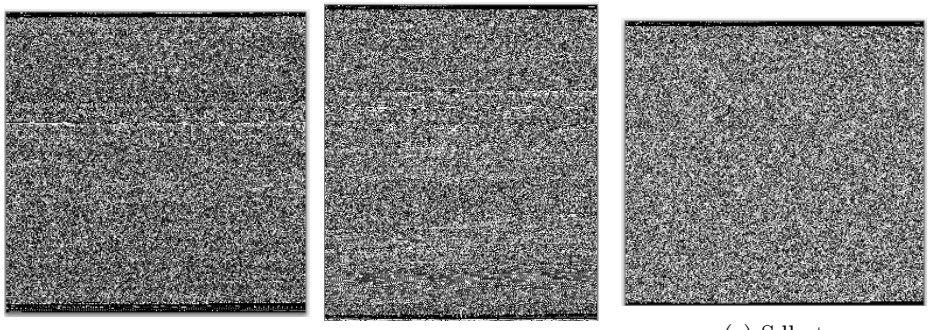


(a) Bancos (b) Lmir (c) Sdbot

(d) Bancos (e) Lmir (f) Sdbot

Fig. 1: Malware variant textures of different families from our dataset.

Although textures seem promising, since many authors use their analysis considering only similarities within families, not all samples produce textures that allow their comparison with other family members. Indeed, many samples neither are related to their families, nor distinguishable from other families' samples. It clearly difficults correct malware classification, as we can observe in Figure 2, which illustrates selected samples harder to distinguish from the same dataset of Figure 1. This observation makes us realize that the overwhelmingly great results presented in literature might have issues.



(a) Bancos (b) Lmir (c) Sdbot

Fig. 2: Other textures of variants from already seen families from our dataset.

Classification Techniques. Nataraj et al. [20] first proposed texture analysis for malware classification—they showed that it is possible to identify patterns among samples from the same family after converting their binaries into gray-scale textures. Using GIST descriptor and K-nearest neighbor (KNN) with euclidean distance, achieve 98% of accuracy. The authors affirm that it is a fast approach and agnostic to platform (while usually malware analysis techniques are made for a specific operational system). The possible disadvantages are binary section reallocation or insertion of amount redundant data. After that, the authors used Host-Rx, Malheur, and VX Heaven datasets to compare texture and dynamic analysis [22]. Texture analysis achieves similar accuracy 4,000 times faster than dynamic analysis. Additionally, this approach is said to be resilient to obfuscation techniques. Following the same strategy, Makandar and Patrot [15] use GIST and Gabor Wavelet Transform (GWT) descriptors to select features, and Artificial Neural Network (ANN) for malware classification with 96.35% accuracy rate on Malheur dataset. Makandar et al. [12] use Support Vector Machine (SVM) on the same dataset with accuracy of 89.68%.

Nataraj [19] proposed texture and signal analysis to try to discover unknown attacks due to the combination of orthogonal methods, since it is more difficult/expensive to build malware that evade all detection schemes. Applying GIST descriptor and KNN classifier to different plataform datasets, the accuracy rate are 97.4%, 98.37%, 83.27% e 84.55% for Malimg (Windows), Malheur

(Windows), VxShare (Linux) and Malgenome (Android) respectively. Kosmidis and Kalloniatis [8] followed the same approach with different machine learning classifiers (Decision Trees - DT, Nearest Centroid - NC, Stochastic Gradient Descent - SGD, Perceptron, Multilayer Perceptron - MLP and Random Forest - RF) to find the best solution for texture analysis. All of them achieve more than 87% on Malimg Dataset (the best was Random Forest; accuracy: 91.6%).

Luo and Lo [11] compared GIST to LBP with different classifiers (SVM, KNN and TensorFlow, a library for machine learning). LBP provided better accuracy in all cases, achieving 93.17% with TensorFlow. Discrete Wavelet Transform descriptor is used by Makandar and Patrot [14] with KNN and SVM classifiers. On Malimg Dataset it produced 98.8% accuracy rate with SVM and 98.84% with KNN. The authors changed the size of textures in [16] and with SVM classifier achieve accuracy of 98.53% on Malimg Dataset and 91.05% on Malheur dataset.

Convolutional Neural Networks (CNN) were used for malware texture classification by Yue [36] and it results in 98.63% accuracy rate on Malimg Dataset. Singh [30] used the Residual Networks (ResNet) architecture with 152 layers on his own dataset achieving 98.21% accuracy rate and, on Malimg Dataset, 96.08%. Rezende et al. [27] utilize CNN with Visual Geometry Group with 16 layers (VGG-16) architecture pre-trained on the ImageNet Dataset to extract features and, to classification, SVM that results in 92.97% of accuracy on VirusSign Dataset. SVM was also used in Makandar and Patrot [17], but the features were extracted with GWT on Malimg Dataset, the accuracy rate was 75.11%, and 89.11% when using KNN at the same conditions. CNN was also used by Yakura et al. [34] [35] on VX Heavens dataset, resulting on 49.03% of accuracy. The same classifier was used by Kabanga and Kim [7] but they achieve 98% of accuracy on Malimg Dataset. The authors affirm that a small change on the image, even that invisible to a human, could cause a misclassification of image.

We observed that the descriptor affects accuracy rate [11]: the same dataset and classifiers used in [20] and [14] with different descriptors resulted in distinct accuracy rate. Although it is not possible to properly compare all research due to differences in datasets and resizing, we show the accuracy rate per classifier using GIST (Figure 3) and other descriptors (Figure 4). KNN performed better with other descriptors than LBP and GWT, followed by RF (only tested with GIST). SVM presented the worst results (except when applied in a reduced dataset) and may be considered inadequate to classify textures of these datasets.

Furthermore, as mentioned earlier, some classifiers select features automatically, i.e. do not need texture descriptors. Figure 5 displays the accuracy rate of CNN used on literature in different datasets. CNN with more layers (increasing depths) presents better accuracy, but exploding/vanishing of gradients and degradation problem may appear in deeper CNN [30]. Before select features, many authors resize the initial texture. Each malware binary has different size, which request resizing to use some descriptors. This size of input texture could interfere on classification results. Nataraj et al. (2013) [21] opt to set 64x64 as resizing value because a lesser value do not result in a robust signature and larger, increase computational complexity. However, this value is attributed empirically

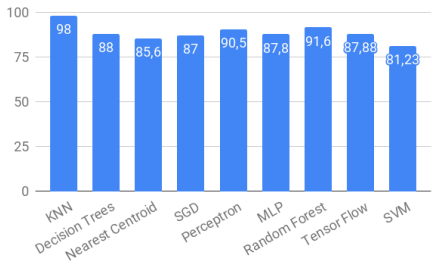


Fig. 3: Classification accuracy using GIST descriptor by classifier.

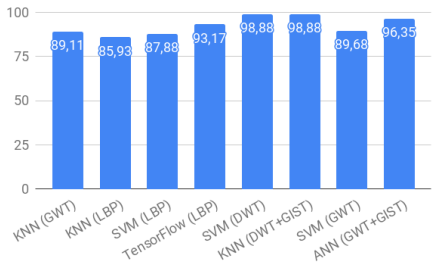


Fig. 4: Classification accuracy using different descriptors by classifier.

and there are none consent about the best one. The differences between worst and best accuracy rate achieved with set values from literature are shown in Figure 6. This graphic show us that, despite the better accuracy uses square of 64x64, it would be more prudent consider the resizing of 224x224 because it has lower variation and maintain a high accuracy rate in all cases.

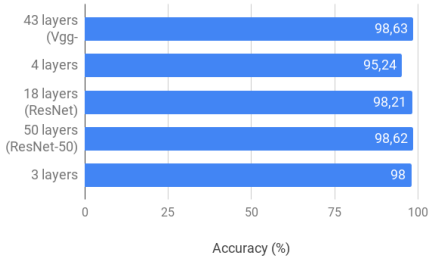


Fig. 5: Accuracy rate from different CNN architectures.

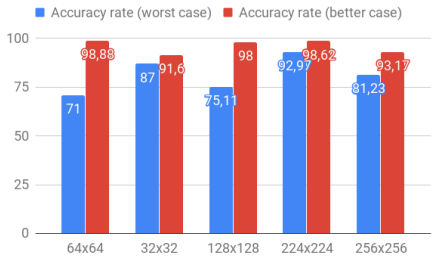


Fig. 6: Best and worst accuracy by resizing value.

For fair comparison, all studies should use the same dataset. We found eight different datasets used to malware texture classification. The most used is Malimg [20] (9,342 samples converted to textures distributed in 25 malware families). Table 1 exhibits basic information of each dataset and their achieved accuracy, and shows that the dataset with more families has a decrease of accuracy rate. It is worth to emphasize that each work applied different techniques, descriptors, and classifiers, which interfere with obtained results.

Considering only the classifier: KNN was used in [20,22,19,13,11,14,17], achieving the best result of literature in [13]; SVM was used in [12,11,14,16,17], with the same result as KNN in [14]; a comparison of many classifiers is in [8]; ANN is used in recent research [15,30,26,35,7]. Table 2 shows the state-of-the-art in malware texture classification and respective techniques and results.

Table 1: Datasets from literature used for malware families classification.

Dataset Name	Samples	Families	Accuracy (%)	Dataset Name	Samples	Families	Accuracy (%)
Host-Rx	393	6	95.14	MalGenome	1094	13	97.40
Malheur	3131	24	89.68 a 98.37	VXShare	568	8	83.27
VX Heaven	63002	531	72.80	Singh 2017	44945	20	95.24 a 98.21
Malimg	9342	25	75.11 a 98.88	VirusSign	10136	20	92.97

3 Methodology

To evaluate texture-based techniques, we follow Nataraj et al. methodology [9], illustrated in Figure 7. The methodology steps are as folows.

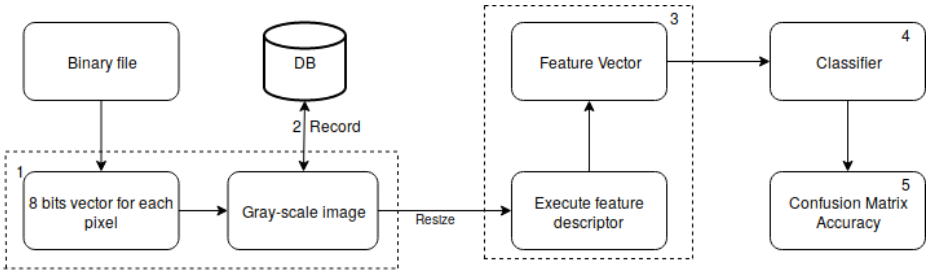


Fig. 7: Steps for malware texture classification.

Step 1: **Conversion of binary files into digital images**: each byte from binary file represents one gray-scale pixel (0, black; 255, white), following a defined width[19], while the heights vary according to the file size. This step is really fast, for instance to convert our dataset took 0.0163 seconds by sample.

Step 2: **Dataset organization**: before classification, we divide the samples into families according to the labels from VirusTotal [32]. We select the labels for our dataset with AVClass [29], similar to **??**, in which the selected label is a common label between six antivirus (AV) vendors (AVClass uses all AV vendors).

Step 3: **Extraction of feature vector**: we compute the features from all samples after re-sizing their produced images, as suggested in [19]. To do so, we compared two descriptors (GIST and LBP) and performed a resize of 128x128, since it presented better results in our preliminary experiments when compared to the resize scales used in literature.

Step 4: **Supervised classification**: after the extraction of the feature vectors, we can classify the produced textures. In this paper, we used the same traditional classifiers of literature, and for the CNN algorithm we used the ResNet architecture from [26].

Step 5: **Results visualization**: we used common classification metrics to show the obtained results. We opt to use accuracy rate, widely utilized in the literature, and confusion matrix, to better visualization of accuracy and error for each family.

Table 2: Techniques used for malware classification based on texture analysis (*n/i*: no information available; ✗: no descriptor).

Ref.	Scale	Descriptor	Classifier	Accuracy (%)	Dataset
[20]	*n/i*	GIST	KNN	98.00	Malimg
[22]	64x64	GIST	KNN	95.14	Host-Rx
				97.57	Malheur
				72.80	VX Heaven
[15]	64x64	GWT + GIST	ANN	96.35	Malheur
[12]	*n/i*	GWT	SVM	89.68	Malheur
[19]	64x64	GIST	KNN	97.40	Malimg
				98.37	Malheur
				97.40	MalGenome
				83.27	VXShare
[13]	*n/i*	DWT, GIST	KNN	98.88	Malimg
[8]	32x32	GIST	DT	88.00	Malimg
			NC	85.60	
			SGD	87.00	
			Perceptron	90.50	
			MLP	87.80	
			RF	91.60	
[36]	*n/i*	✗	CNN	98.63	Malimg
[11]	*n/i*	LBP	CNN	93.17	Malimg
			SVM	87.88	
			KNN	85.93	
		GIST	CNN	87.88	
			SVM	81.23	
			KNN	82.83	
[14]	64x64	DWT	KNN	98.84	Malimg
			SVM	98.88	
[16]	256x256	DWT	SVM	91.05	Malheur
				92.53	Malimg
[30]	32x32	✗	CNN	95.24	Malshare + VirusShare + VirusTotal.
			CNN (ResNet)	98.21	
			CNN (ResNet)	96.08	Malimg
[26]	224x224	✗	CNN (ResNet)	98.62	Malimg
[27]	224x224	✗	SVM	92.97	VirusSign
[35]	64x64	✗	CNN	49.03	VX Heaven
[17]	128x128	GWT	KNN	89.11	Malimg
			SVM	75.11	
[7]	128x128	✗	CNN	98.00	Malimg

Experimental setup. We performed all experiments, but CNN, on a desktop computer (Ubuntu 16.04 LTS, Intel Core i5-650, 3.GHz 4MB Cache and 4GB RAM). CNN ran on a server (Ubuntu 14.04.3 LTS, Intel Xeon E5620, 2.4GHz, 12MB Cache, 32GB RAM). An image descriptor is applied for extraction of meaning features only. In this research, we use a global (GIST [24]) and a local (LBP [23]) descriptor. The first is the most used in literature for malware texture classification, but global descriptors may lose information and classify incorrectly when redundant data is added or there is relocation of sections [20]. To address that, a local descriptor is used, LBP, which is widely used and well-known local descriptors due to its easy implementation and good results for texture-based classification. Both descriptors are used to extract features in a feature vector of textures with resizing of 128x128. We used `pyleargist` library to implement our GIST descriptor with 20 filters and 320 as dimension, and `scikit-image` [33] to our LBP descriptor, which is divided into 3x3 blocks (thus evaluating 8 neighbors), and applies the *uniform* method (rotation invariant and gray-scale).

Regarding the classification algorithms used here, we set the same parameters used in literature or, if not mentioned, the default from `scikit-learn` [25]. For the CNN classifier, we opt to use the technique proposed by Rezende et al. [26], since it achieves one of the highest accuracy rates and presents enough information to replicate the experiment. Our classification metrics were implemented with the `sklearn.metrics` module—accuracy rate and confusion matrices came from the `accuracy_score` and `confusion_matrix` functions, respectively. We selected data for training and testing with 10-fold cross-validation.

Datasets. We use two datasets, a publicly available one (Malimg) and a private, locally collected in our lab along the years. We do that because, for fair evaluation, we must have reliable, unbiased, not manipulated, correctly labeled datasets. The only way to maximize our chances of meeting these requirements is to use a public dataset already scrutinized and composed of few families to validate our preliminary premises and, after that, confirm if the hypothesis still holds when we use a private, bigger, unfiltered dataset closer to a real scenario. For our dataset (collected along the years), we follow Rossow et al. recommended practices for security experiments [28] and Li et al. advices on watching for biases in data selection [10]. Malimg [20] consists of 9,342 textures distributed into 25 malware families, and the dataset description is shown in Table 3. We found thirteen related work that used this public dataset in their experiments. Our local dataset consists of 19,979 unique malware distributed into 5,715 families, which we re-labeled with AVClass [29] on VirusTotal [32] labels. These samples were collected by a partner CSIRT and shared with our lab[1] since 2007. In Table 4 we show an excerpt of our dataset containing the most representative families (with at least 50 samples each). In our experiments, we used the whole dataset as well as some of its subgroups.

[1] Additional information about samples will be available after acceptance to do not violate the conference blindness requirement.

Table 3: Malimg dataset. Notice that variants may be in distinct families and that the two Allaple families correspond to almost half of the dataset.

	Class	**Family**	**Variants**	**%**
1	Worm	Allaple.L	1591	17.03
2	Worm	Allaple.A	2949	31.57
3	Worm	Yuner.A	800	08.56
4	PWS	Lolyda.AA 1	213	02.28
5	PWS	Lolyda.AA 2	184	01.97
6	PWS	Lolyda.AA 3	123	01.32
7	Trojan	C2Lop.P	146	01.56
8	Trojan	C2Lop.gen!G	200	02.14
9	Dialer	Instantaccess	431	04.61
10	Trojan Downloader	Swizzor.gen!I	132	01.41
11	Trojan Downloader	Swizzor.gen!E	128	01.37
12	Worm	VB.AT	408	04.37
13	Rogue	Fakerean	381	04.09
14	Trojan	Alueron.gen!J	198	02.12
15	Trojan	Malex.gen!J	136	01.46
16	PWS	Lolyda.AT	159	01.70
17	Dialer	Adialer.C	125	01.34
18	Trojan Downloader	Wintrim.BX	97	01.04
19	Dialer	Dialplatform.B	177	01.89
20	Trojan Downloader	Dontovo.A	162	01.73
21	Trojan Downloader	Obfuscator.AD	142	01.52
22	Backdoor	Agent.FYI	116	01.24
23	Worm:AutoIT	Autorun.K	106	01.13
24	Backdoor	Rbot!gen	158	01.69
25	Trojan	Skintrim.N	80	00.86

Table 4: Description of an excerpt of our dataset (families with samples >= 50).

	Class	**Family**	**Variants**	**%**
1	Backdoor	Agobot	202	02,86
2	Backdoor	Aimbot	73	01,03
3	Worm	Bagle	138	01,96
4	Trojan Banker	Banbra	75	01,06
5	Trojan	Bancos	327	04,63
6	Trojan Downloader	Banload	215	03,05
7	Backdoor	Bifrose	128	01,81
8	Trojan	Constructor	110	01,56
9	Trojan Downloader	Dadobra	145	02,05
10	Trojan	Delf	1267	17,96
11	Trojan Downloader	Dyfuca	84	01,19
12	Trojan	Goldun	58	00,82
13	Trojan	Harnig	58	00,82
14	Virus	Hllc	61	00,86
15	Virus	Hllo	60	00,85
16	Backdoor	Hupigon	332	04,71
17	Trojan Downloader	Inservice	115	01,63
18	Backdoor	Ircbot	151	02,14
19	Trojan Downloader	Istbar	199	02,82
20	Worm	Kelvir	82	01,16
21	Trojan PSW	Ldpinch	123	01,74
22	Trojan PSW	Lineage	101	01,43
23	Trojan	Lmir	410	05,81
24	Trojan	Lowzones	60	00,85
25	Worm	Mytob	86	01,22
26	Trojan	Nsanti	50	00,71
27	Backdoor	Pcclient	79	01,12
28	Trojan PSW	Qqpass	123	01,74
29	Trojan	Qqrob	58	00,82
30	Backdoor	Ranky	72	01,02
31	Backdoor	Rbot	909	12,88
32	Virus	Score	120	01,70
33	Backdoor	Sdbot	685	09,71
34	Worm	Spybot	122	01,73
35	Trojan Downloader	Swizzor	56	00,79
36	Backdoor	Wootbot	63	00,89
37	Trojan	Zlob	59	00,84

4 Experiments and Results

In this section, we reproduce literature experiments using both publicized datasets and our malware collection and evaluate classifiers resistance to obfuscation.

Reproducing Literature Results. To validate reported literature results, we first reproduced their experiments using the same dataset (Malimg) and parameters described in the considered papers (GIST and LBP descriptors and 32x32, 64x64 and 128x128 (standard) texture's scales). The first experiment consists on identifying the best algorithm and feature extractors for texture classification. Table 5 shows the accuracy rate for all algorithms and feature extractors applied to the complete Malimg dataset. The higher accuracy rates (above 90%) for all classifiers are achieved using the GIST descriptor. The CNN classifier without a descriptor also achieves high accuracy.

Table 5: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP in most classifiers. ([1]: direct mapping without texture descriptor; [2]: resize from literature; [3]: standardized scale).

Classifier	CNN[1]	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[2]	N/A	96.97	93.06	95.23	91.33	95.23	91.44	90.03	95.99
LBP[2]	N/A	85.59	69.34	78.66	47.02	53.41	37.38	30.88	71.07
GIST[3]	N/A	97.94	96.32	97.83	96.42	95.88	94.58	91.33	96.42
LBP[3]	N/A	95.77	92.96	95.67	84.72	51.25	40.41	66.41	83.31
N/A	98.57	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Understanding Classifier's Errors. While some combinations of feature extractors and classifiers achieve high accuracy results in the complete Malimg dataset, most combinations do not present accuracy results high enough to be considered practical in actual contexts, due to either the low TP rates or high FP rates, which contradicts published literature results.

To understand classifier's errors, we performed an in-depth investigation on how they classified each malware sample. We discovered that all classifiers errors occur on malware families that are very similar among themselves, thus producing very similar texture patterns. Table 6 shows that both RF and CNN classifier confuse the families `Swizzor.gen!E` (Figure 8) and `Swizzor.gen!I` (Figure 9).

Table 6: RF & CNN Confusion Matrices: Classifiers mix samples from `Swizzor.gen!E` and `Swizzor.gen!I` families.

Samples	S.gen!E	S.gen!I	Allaple
S.gen!E	**99**	48	0
S.gen!I	**48**	38	0
Allaple	0	0	**100**

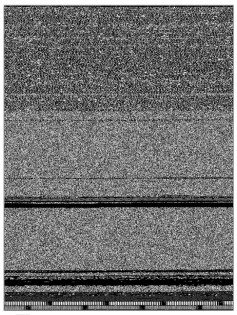


Fig. 8: `Swizzor.gen!E`

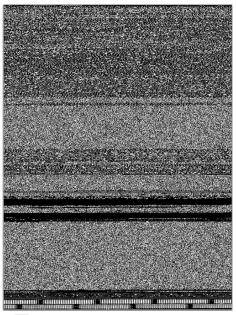


Fig. 9: `Swizzor.gen!I`

The classifier's confusion between similar families suggests that current texture-based approaches are only able to detect completely distinguishable families, as already pointed out by studies in other scenarios [10]. Therefore, we repeated our experiments, but now limiting the dataset to the samples used in [13], with no conflict between similar families. Table 7 shows accuracy rates for distinct classifiers and feature extractors combinations. The GIST descriptor keeps overperforming LBP for most scenarios, as in the complete dataset, but now achieving an accuracy of 100%. LBP classifiers also increased their rates, showing that similar families is an issue for all types of classifiers and feature extractors.

Table 7: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP for most classifiers: 100% of accuracy happens due to the selection of dissimilar families for the experiment. ([1]direct mapping without texture descriptor; [2]literature resizing; [3]standard scale).

Classifier	CNN[1]	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[2]	N/A	100.00	99.81	99.81	97.91	99.81	99.43	91.63	99.81
LBP[2]	N/A	99.24	96.77	99.05	71.10	78.90	73.76	86.50	90.87
GIST[3]	N/A	100.00	99.81	100.00	99.05	100.00	100.00	100.00	100.00
LBP[3]	N/A	99.43	98.67	99.43	94.68	78.52	78.33	87.26	91.44
N/A	99.87	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

These experiments results show that current texture-based approaches present best results when the considered malware families are clearly different, which is not acknowledge in the literature, thus highlighting the need for more granular and real-world considerations when developing and evaluating malware solutions.

Extending evaluations to other datasets. Given the previous findings that the ability of existing texture-based classifiers is strongly tied to datasets having no similar families, we hypothesized that classifiers would not perform well when working with a real-world dataset, which is very imbalanced, thus presenting some families with multiple samples and other with a very limited number of samples. To test this hypothesis, we considered a dataset of Brazilian malware samples daily collected from a CSIRT institution and submited all samples to classification procedures by the same classifiers and feature extractors from the previous tests. Table 8 shows accuracy results for all combinations of feature extractors and classifiers and, as hypothesized, all of them resulted in very low accuracy rates in this scenario closer to reality.

Table 8: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP in most classifiers. Accuracy rate decreases significantly when using a scenario closer to reality. ([1]: resize from literature; [2]: standardized scale).

Classifier	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[1]	34.36	14.46	18.87	04.10	27.90	11.08	10.05	29.85
LBP[1]	13.13	08.00	10.46	01.13	16.31	06.46	00.00	16.00
GIST[2]	35.08	19.49	23.18	03.69	28.51	13.33	04.10	29.54
LBP[2]	13.64	11.59	15.49	01.44	16.61	01.03	00.51	21.44

Since the experiments supported our hypothesis on the limits of applying texture-based classification in real scenarios, we investigated further to under-

stand the practical factors limiting classification performance. We first discovered that classification rates were underscored due to families' clusters having too few samples. To mitigate this effect, we limited our real-world dataset to a set of 37 families that have at least 50 different samples each. We highlight that such number of distinct families is still higher than the ones from the publicized datasets. Table 9 shows classification accuracy for all algorithms and feature extractors using this new dataset. As expected, accuracy increased for all classifiers, but it is still not enough for operating in an actual scenario.

Table 9: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP in most classifiers. Accuracy rate unsatisfying even when discarding small families. ([1]: direct mapping without texture descriptor; [2]: resize from literature; [3]: standardized scale).

Classifier	CNN[1]	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[2]	N/A	49.13	26.09	38.84	10.72	32.32	12.03	17.39	38.98
LBP[2]	N/A	23.77	15.51	23.77	07.10	20.43	13.19	03.48	22.46
GIST[3]	N/A	48.84	29.13	43.04	15.36	35.36	26.09	20.87	41.01
LBP[3]	N/A	30.43	22.90	31.16	12.46	21.88	10.29	08.26	26.67
N/A	45.52	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

We also discovered that the number of families was also a factor for classifiers' rates underscoring. Therefore, we limited our dataset to the 10 most prevalent malware families, randomly choosing 100 samples from each one of them. Table 10 shows accuracy for all combinations of classifiers and texture extractors. In this new, limited dataset, classifiers achieved almost 80% of accuracy, which is a significant accuracy rate (but limited in number of families).

Table 10: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP in most classifiers. Accuracy rate increases with balanced dataset and fewer families. ([1]: direct mapping without texture descriptor; [2]: resize from literature; [3]: standardized scale).

Classifier	CNN[1]	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[2]	N/A	78.00	59.00	70.00	54.00	69.00	62.00	58.00	72.00
LBP[2]	N/A	63.00	37.00	46.00	43.00	43.00	11.00	13.00	41.00
GIST[3]	N/A	78.00	73.00	70.00	63.00	61.00	57.00	52.00	66.00
LBP[3]	N/A	67.00	52.00	64.00	32.00	34.00	12.00	10.00	43.00
N/A	76.50	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Finally, to highlight that current texture-based malware classifier's effectiveness is limited to few malware families, we again limited our dataset, now to only the top 8 most prevalent malware families. Table 11 presents accuracy results for distinct combinations of classifiers and feature descriptors. As hypothesized, accuracy keeps increasing, now reaching scores greater than 84%.

Overall, our experiments indicate that: (i) KNN and CNN are the best malware texture classifiers; (ii) a 128x128 scale produces the greatest accuracy re-

Table 11: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP in most classifiers. Accuracy rate increases with fewer families. ([1]: direct mapping without texture descriptor; [2]: resize from literature; [3]: standardized scale).

Classifier	CNN[1]	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[2]	N/A	87.50	66.25	81.25	67.50	66.25	68.75	57.50	77.50
LBP[2]	N/A	62.50	42.50	50.00	37.50	53.75	23.75	27.50	47.50
GIST[3]	N/A	88.75	75.00	86.25	77.50	68.75	70.00	51.25	81.25
LBP[3]	N/A	76.25	67.50	77.50	47.50	46.25	35.00	66.25	52.50
N/A	84.75	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

sults; and (iii) the use of GIST results in greatest accuracy rates. On the one hand, our findings support the experiments conclusions from literature. On the other hand, we highlighted that these results do not hold true for actual scenarios, since current approaches are not fully capable of handling imbalanced dataset and a large number of families. Improving texture-based classifiers to handle these cases and make them practical is currently an open research question not acknowledge by the existing literature.

Evaluating Obfuscation Resistance. In addition to dataset imbalances, approaches tackling real-world scenarios will also face challenges to classify malware samples due to sample's characteristics themselves. For instance, statistics [1] report that more than 80% of malware files use some obfuscation techniques to avoid malware detection. The literature in texture-based malware classification reports that these approaches are resilient to obfuscation techniques [20] (in [19], UPX is used to demonstrate this resilience), but since they use static binary information for classification, it is doubtful that these approaches are able to correctly disambiguate obfuscated code. To test this hypothesis, we repeated all experiments now considering obfuscated binary versions. Based in the previous findings, we limited the dataset to the top 8 most prevalent malware families, a malware family range which previous experiments showed that texture-based malware detection is effective.

Evaluating Compression Resistance. In our first experiment we compressed all binaries with the popular `ZIP` and `TAR.GZ` tools, since compression reduces all redundancy that might allow family characterization. In addition, distributing compressed files via mail attachments is a popular strategy leveraged by attackers during their phishing campaings, thus it constitute a real-world scenario. Table 12 presents accuracy results for `ZIP`-compressed files and Table 13 presents accuracy results for `TAR.GZ`-compressed files.

All classifier were affected by binary compression and reduced their accuracy rates. The best classifier score lowered from 88% in the previous evaluation to 65% in the compressed dataset. The CNN classifier achieved only 45,75% in the `ZIP` dataset and 42,13% in the `TAR.GZ` dataset. These results are even more impacting when considered that we limited our evaluation to only the 8

Table 12: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP in most classifiers. Accuracy rate decreases with `ZIP`. ([1]: direct mapping without texture descriptor; [2]: resize from literature; [3]: standardized scale).

Classifier	CNN[1]	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[2]	N/A	65.00	42.50	52.50	40.00	53.75	31.25	23.75	51.25
LBP[2]	N/A	28.75	23.75	32.25	30.00	18.75	16.25	12.50	30.00
GIST[3]	N/A	60.00	42.50	57.50	45.00	43.75	30.00	31.25	56.25
LBP[3]	N/A	30.00	35.00	35.00	30.00	26.25	13.75	12.50	20.00
N/A	45.75	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 13: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP in most classifiers. Accuracy rate decreases with `TAR.GZ`. ([1]: direct mapping without texture descriptor; [2]: resize from literature; [3]: standardized scale).

Classifier	CNN[1]	KNN	DT	RF	NC	SVM	SGD	Perceptron	MLP
GIST[2]	N/A	66.25	40.00	53.75	40.00	50.00	25.00	30.00	58.75
LBP[2]	N/A	27.50	23.75	31.25	31.25	22.50	12.50	12.50	37.50
GIST[3]	N/A	66.25	48.75	56.25	55.00	48.75	16.25	31.25	53.75
LBP[3]	N/A	31.25	36.25	40.00	30.00	25.00	12.50	12.50	21.25
N/A	42.13	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

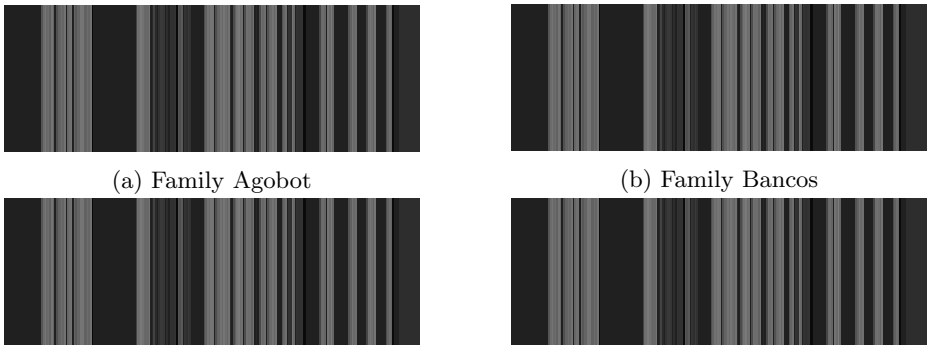
most well-classified families. Therefore, these results suggests that texture-based classifiers are unable to distinguish armored samples, as following discussed.

Evaluating Packing Resistance. More than compression, malware samples can be distributed in completely modified versions, which can be done, for instance, by using a packer. To evaluate the impact of packing in texture-based malware classifiers, we packed all samples usind The Ultimate Packer for eXecutables (UPX), a popular and open-source packing solution. The compression provided by UPX is greater than `GZIP` and it adds a decompression module to the executable [1], which makes samples to look like even more similar.

Table 14 shows classification accuracy for the distinct combinations of classifiers and feature descriptors. As hypothesized, classification accuracy is significantly reduced, as the textures now represents the packing structure and not the packed code. More specifically, the classification accuracy is almost the same for all classifiers. Our exploratory analysis identified that it happens because all samples are grouped to the same family, as shown in Figure 10. Therefore, the percentage of correctly labeled samples refers to the only ones that truly belong to the attributed family. In practice, malware samples can leverage packing solutions even more sophisticated than UPX, thus highlighting the need of developing more obfuscation-resistant texture-based malware classifiers.

Table 14: **Texture Descriptors & Classifiers Accuracy Evaluation.** GIST descriptor achieves higher detection rates than LBP in most classifiers. It is not possible to classify malware compressed with `UPX`. ([1]: direct mapping without texture descriptor; [2]: resize from literature; [3]: standardized scale).

Classifier	**CNN**	**KNN**	**DT**	**RF**	**NC**	**SVM**	**SGD**	**Perceptron**	**MLP**
GIST[1]	N/A	12.99	12.99	12.99	12.99	12.99	12.99	12.99	12.99
LBP[1]	N/A	12.99	12.99	12.99	12.99	12.99	12.99	12.99	12.99
GIST[2]	N/A	12.99	12.99	12.99	12.99	12.99	12.99	12.99	12.99
LBP[2]	N/A	12.99	12.99	12.99	12.99	12.99	12.99	12.99	11.69
N/A	12.50	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A



(a) Family Agobot (b) Family Bancos

(c) Family Rbot (d) Family Sdbot

Fig. 10: Variants texture from different families of compressed with `UPX`.

5 Discussion

In this section, we summarize our main findings and point future directions for the development of texture-based malware classifiers.

Representing a file as a texture is challenging. Although the creation of a new representation may ease some tasks, it creates new challenges. Representing a binary file as a texture requires defining: (i) an appropriate scale, which directly affects classification results; (ii) a dimensional representation (1D, 2D, 3D), which may imply in information loss if the binary file is too large; and (iii) padding texture, in the case where a binary file is too small. Therefore, future developments in texture classification should also focus in advancing representations and pre-processing steps in addition to providing new texture descriptors.

Biased datasets cannot be generalized. Whereas we were able to reproduce literature results using the publicized datasets, we were not able to achieve high accuracy in a dataset of malware samples collected in-the-wild, despite the generalization claims made in literature reports. Further investigations revealed that the described texture descriptors were able to classify only some malware families. These families were present in the publicized datasets but not in our real-world collection, which highlights the need of breaking down detection accuracy by malware families when reporting experiments results, otherwise reported results cannot be generalized to other contexts, as here demonstrated.

Handling Obfuscation is still a challenge. Despite claims of approaches' resistance to obfuscation, our experiments revealed that obfuscating malware binaries is still an effective measure to bypass detection mechanisms. Even samples previously classified correctly were mistakenly classified after being packed with popular packers, such as UPX. The major reason for classifiers wrongly labelling samples is that packing solutions produce similar code patterns regardless of the embedded payload, thus leading to the same textures. Therefore, future work should consider packed code in their developments and evaluations, otherwise the developed solutions might be impractical in actual scenarios.

6 Conclusion

Texture-based malware classification can be performed leveraging multiple approaches, with multiple scales, descriptors and classifiers. In this paper we presented an evaluation of the effectiveness of these approaches to identify the usage scenarios for which they are most suited. Our experiments considered both global (GIST) and local (LBP) descriptors and a multitude of classifiers (e.g., KNN, DT, RF, CNN). We were able to reproduce literature experiments using their publicized datasets and identified that the 128x128 scale produces higher results than other scales and that the GIST detector achieves higher detection rates than LBP. However, we were not able to achieve similar detection rates when evaluating a real dataset of malware samples collected in-the-wild. Further investigations revealed that the reported literature results hold true for only some malware families, which are prevalent in publicized datasets but not on our real

collection, thus reinforcing the need of presenting detection results broken down by malware families. Moreover, we notice that despite literature claims, obfuscation is still an effective technique to bypass detection. As a future work, we plan to develop new malware representation strategies, such as considering 3D textures instead of 2D ones to reduce information loss in conversion procedures, towards making texture-based malware classifiers practical in actual scenarios.

References

1. Al-Anezi, M.M.K.: Generic packing detection using several complexity analysis for accurate malware detection. Int. Journal Adv. Comp. Science **5**(1) (2014)
2. Awad, R.A., Sayre, K.D.: Automatic clustering of malware variants. In: Intel. and Sec. Informatics (ISI). pp. 298–303. IEEE (2016)
3. Bertolini, D., Oliveira, L.S., Justino, E., Sabourin, R.: Texture-based descriptors for writer identification and verification. Expert Systems with Applications (2013)
4. Conti, G., Bratus, S., Shubina, A., Sangster, B., Ragsdale, R., Supan, M., Lichtenberg, A., Perez-Alemany, R.: Automated mapping of large binary objects using primitive fragment type classification. digital investigation (2010)
5. Costa, Y.M., Oliveira, L., Koerich, A.L., Gouyon, F., Martins, J.: Music genre classification using lbp textural features. Signal Processing (2012)
6. Damodaran, A., Di Troia, F., Visaggio, C.A., Austin, T.H., Stamp, M.: A comparison of static, dynamic, and hybrid analysis for malware detection. Journal of Comp. Vir. and Hack. Tech. (2017)
7. Kabanga, E.K., Kim, C.H.: Malware images classification using convolutional neural network. Journal of Comp. and Communications (2017)
8. Kosmidis, K., Kalloniatis, C.: Machine learning and images for malware detection and classification. In: Pan-Hellenic Conf. on Inf. ACM (2017)
9. Laks: Sarvam blog. `http://sarvamblog.blogspot.com.br` (2014)
10. Li, P., Liu, L., Gao, D., Reiter, M.K.: On challenges in evaluating malware clustering. In: Jha, S., Sommer, R., Kreibich, C. (eds.) RAID. Springer (2010)
11. Luo, J.S., Lo, D.C.T.: Binary malware image classification using machine learning with local binary pattern. In: IEEE Big Data (2017)
12. Makandar, A., Patrot, A.: Malware analysis and classification using artificial neural network. In: I-TACT (2015)
13. Makandar, A., Patrot, A.: An approach to analysis of malware using supervised learning classification. In: Int. Conf. on Rec. Trends in Eng., Science Tech. (2016)
14. Makandar, A., Patrot, A.: Malware class recognition using image processing techniques. In: ICDMAI (2017)
15. Makandar, A., Patrot, A.: Malware image analysis and classification using support vector machine. Int. Journal of Trends in CS and Eng. (2015)
16. Makandar, A., Patrot, A.: Wavelet statistical feature based malware class recognition and classification using supervised learning classifier. Oriental Journal of CS and Tech. (2017)
17. Makandar, A., Patrot, A.: Trojan malware image pattern classification. In: Guru, D.S., Vasudev, T., Chethan, H., Kumar, Y.S. (eds.) Int. Conf. on Cogn. and Recogn. Springer (2018)
18. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: 23rd Annual Computer Security Applications Conference (2007)
19. Nataraj, L.: A Signal Processing Approach To Malware Analysis. UCSB (2015)

20. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.: Malware images: visualization and automatic classification. In: Int. Symp. on vis. for cyber sec. ACM (2011)
21. Nataraj, L., Kirat, D., Manjunath, B., Vigna, G.: Sarvam: Search and retrieval of malware. In: ACSAC NGMAD (2013)
22. Nataraj, L., Yegneswaran, V., Porras, P., Zhang, J.: A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In: Workshop on Sec. and AI. ACM (2011)
23. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. Trans. on Pattern Analysis and Machine Intelligence (2002)
24. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. Int. Journal of Comp. Vision (2001)
25. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of ML Research (2011)
26. Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., de Geus, P.: Malicious software classification using transfer learning of resnet-50 deep neural network. In: ICMLA (2017)
27. Rezende, E., Ruppert, G., Carvalho, T., Theophilo, A., Ramos, F., Geus, P.d.: Malicious software classification using vgg16 deep neural network's bottleneck features. In: Latifi, S. (ed.) Inf. Tech. - New Gen. Springer (2018)
28. Rossow, C., Dietrich, C.J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., Steen, M.v.: Prudent practices for designing malware experiments: Status quo and outlook. In: SP. IEEE (2012)
29. Sebastián, M., Rivera, R., Kotzias, P., Caballero, J.: Avclass: A tool for massive malware labeling. In: RAID. Springer (2016)
30. Singh, A.: Malware Classification using Image Representation. Master's thesis, Indian Institute of Technology Kanpur (2017)
31. Thakare, V.S., Patil, N.N., Sonawane, J.S.: Survey on image texture classification techniques. Int. Journal of Adv. in Tech. (2013)
32. VirusTotal: Virustotal. `https://www.virustotal.com/#/home/upload` (2017)
33. van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T., the scikit-image contributors: scikit-image: image processing in Python. PeerJ (2014)
34. Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y., Sakuma, J.: Malware analysis of imaged binary samples by convolutional neural network with attention mechanism. In: Conf. on Data and App. Sec. and Priv. ACM (2018)
35. Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y., Sakuma, J.: Malware analysis of imaged binary samples by convolutional neural network with attention mechanism. In: CODASPY. CODASPY '18, ACM (2018)
36. Yue, S.: Imbalanced malware images classification: a CNN based approach. CoRR (2017), `http://arxiv.org/abs/1708.08042`
37. Zhang, J., Qin, Z., Yin, H., Ou, L., Xiao, S., Hu, Y.: Malware variant detection using opcode image recognition with small training sets. In: ICCCN. IEEE (2016)