Image-based Malware Family Detection: An Assessment between Feature Extraction and Classification Techniques

Giacomo Iadarola[1], Fabio Martinelli[1], Francesco Mercaldo[1,2] and Antonella Santone[2]
[1]*Institute for Informatics and Telematics, National Research Council of Italy (CNR), Pisa, Italy*
[2]*Department of Biosciences and Territory, University of Molise, Pesche (IS), Italy*

Keywords: Machine Learning, Mobile Security, Android, Malware Classification, Image Texture Analysis.

Abstract: The increasing number of malware in mobile environment follows the continuous growth of the app stores, which required constant research in new malware detection approaches, considering also the weaknesses of signature-based anti-malware software. Fortunately, most of the malware are composed of well-known pieces of code, thus can be grouped into families sharing the same malicious behaviour. One interesting approach, which makes use of Image Classification techniques, proposes to convert the malware binaries to images, extract feature vectors and classifying them with supervised machine learning models. Realizing that researchers usually evaluate their solutions on private datasets, it is difficult to establish whether a model can be generalized on another dataset, making it difficult to compare the performance of the various models. This paper presents a comparison between different combination of feature vector extraction methods and machine learning models. The methodology aimed to evaluate feature extractors and supervised machine learning algorithms, and it was tested on more than 20 thousand images of malware, grouped into 10 different malware families. The best classifier, a combination of GIST descriptors and Random Forest classifiers, achieved an accuracy of 0.97 on average.

1 INTRODUCTION

Nowadays, malware classification and detection are one of the biggest open problems in the cybersecurity research fields (Canfora et al., 2018). Our PCs and smartphones handle huge quantity of sensitive data every day, and many attackers are getting interested in stealing information and profit from them (Casolare et al., 2019). Smartphones, laptops and smart devices in general (the Internet-of-Things) brought all kind of technologies into our homes and makes them accessible to everyone (Cimino et al., 2020). These devices are powerful, and their usability and easy-to-use design allow non-tech people to use all their features without any knowledge of the inner-working: that is the knowledge gap where most of the attacks reside. Most of the malware trick the users, the weakest link in the security chain, to perform unsafe actions, like opening insecure links or sending sensitive information to criminals that act as reliable entities. The malware performs legitimate actions, but their sequence leads to a malicious outcome (Iadarola et al., 2019). Nonetheless, tech people are not safe as well. If an attacker can upload its malware on a trusted app store, that piece of software will be trusted by everyone, because no one has the time to examine all the software acquired on its own. There is a huge need for automatizing the malware detection problems and keep safe our smart devices.

Among all the Mobile Operating System, Android has the biggest share with almost 80% of the market (StatCounter, 2020), and also the most affected by malware. Most of the proposed approaches for malware detection are signature-based, thus able to identify signatures of a previously known attack. This approach is efficient because of the code-reuse (McLaughlin et al., 2017; Suarez-Tangil et al., 2017): it is very unlikely to find a completely new malware, since most of them reuse, at least part of, code from previous malware. Therefore, we can easily categorize the malware by groups of families coming from the same source code.

The number of new applications and software uploaded on the app stores is growing constantly, around 6100 apps are uploaded in the Google Play store every day (Statista platform, 2018). We cannot rely anymore on manual code inspection for debugging software, it is a time-consuming process which requires

Iadarola, G., Martinelli, F., Mercaldo, F. and Santone, A.
Image-based Malware Family Detection: An Assessment between Feature Extraction and Classification Techniques.
DOI: 10.5220/0009817804990506
In *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security (IoTBDS 2020)*, pages 499-506
ISBN: 978-989-758-426-8

great expertise and high level of concentration, that makes the process faulty. It follows that designing efficient classification techniques is the key to keep as much secure and safe as possible the application stores.

One good practice in research is to learn the lesson from other fields, to study recent and interesting findings in different areas, and applying those intuitions and ideas to address different problems. The computer science field has a wide domain on image classification. Many methods making use of Machine Learning and Deep Learning were proposed in the latest year, and they achieved great accuracy in classifying images following similar patterns. As we pointed out, also the malware in the mobile environment has a point in common with similar images: they are grouped by families that share similar code. By following this intuition, we can find works in the literature that apply image classification models to malware classification (see Section 2).

This paper introduces a study on some image classification techniques applied to a malware classification problem. We evaluate the effectiveness of a couple of models proposed in the literature on a collection of more than 20 thousands malware, split into 10 families (Argus Cyber Security Lab, 2020; Wei et al., 2017). The aim of the work is being a starting point for a deeper and complete analysis of all the state-of-the-art approaches that were adopted for addressing this classification task.

The paper proceeds as follows: the next section reports similar works that have a relation with ours; Section 3 presents the proposed methodology to classify malware families and the experiments performed to validate it. Section 4 reports the experimental results, while Section 5 provides insight discussions on the strengths and weaknesses of the proposed approach. Limitations, future works and conclusion are reported in the last two sections, respectively Section 6 and Section 7.

2 RELATED WORK

Malware classification using image processing is a technique widely adopted in the latest years. Malware binaries visualized as gray-scale images show that those belonging to the same malware family appear very similar in texture and layout, mainly because code reuse. The first efficient approach was presented by Nataraj L. et al. (Nataraj et al., 2011), applying Gabor filter on around 9000 malwares of 25 different family and then classifying them using k-nearest Neighbors approach with Euclidean distance for classification. They achieved 98% classification accuracy and demonstrated the feasibility of the methodology. Moreover, the dataset used in the experiment, called Malimg, was adopted in many next works as baseline for comparison. The paper by Ni S. et al. (Ni et al., 2018) proposes a malware classification algorithm that uses static features extracted by disassembling the malware and encode it by SimHash. Similar approaches regarding Deep Learning are presented by Kalash M. et al. (Kalash et al., 2018) and Kabanga EK. et al (Kabanga and Kim, 2017), that propose CNN models for malware image classification The paper by Agarap AF. and Pepito FJH (Agarap, 2017) presents an interesting comparison between three different Deep Leaning models (CNN, GRU and MLP toghether with an SVN) to classify malware binaries. They adopted the Malimg dataset and achieved the best result (around 85% accuracy) with the GRU model. The result presented by Akarsh S. et al. (Akarsh et al., 2019) improve the accuracy to 94% on the Malimg dataset, by applying an hybrid cost-sensitive network of one-dimensional CNN and a Long Short-Term memory model.

3 METHODOLOGY

The methodology proposed is straightforward and based on papers of similar research (Nataraj et al., 2011; Ni et al., 2018; Mercaldo and Santone,).

First of all, the malware are converted into images by reading malware binaries into matrixes, that can be seen as a grayscale image. The malware images analysed came from the AMD dataset (Argus Cyber Security Lab, 2020; Wei et al., 2017). Following this conversion, we processed the data by applying different filters for static feature extraction and generate a feature vector for each image. Finally, machine learning models took the feature vectors as input and perform the classification. Figure 1 shows the method steps, which are explained in the next subsections in details.

3.1 Dataset Preprocessing

Considering the scope of this work, to compare a couple of methods and discussing the result, we were not interested in a complete analysis of the database, but instead a reliable test on different approaches. Therefore, we preprocessed the database and selected only the families with more than 500 example, in order to have a strong base ground to train the models. By doing so, we focused the attention on the approaches instead of the data, and we were able to test different

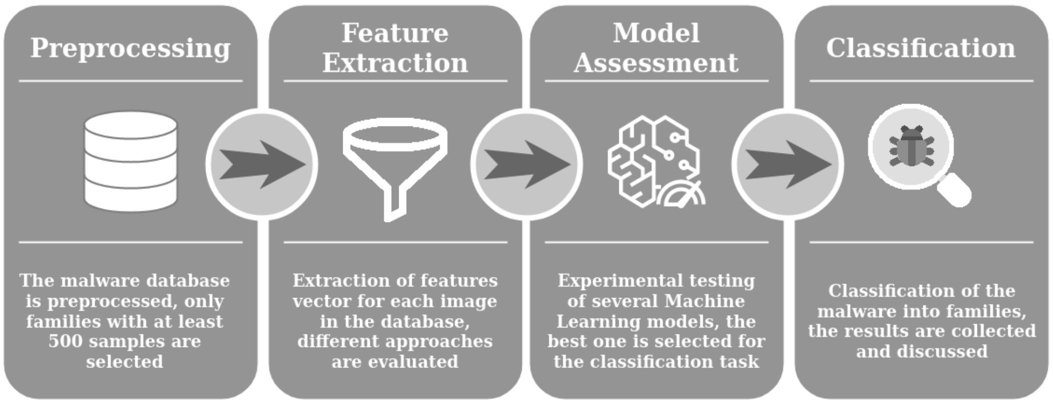
Preprocessing
The malware database is preprocessed, only families with at least 500 samples are selected
Feature Extraction
Extraction of features vector for each image in the database, different approaches are evaluated
Model Assessment
Experimental testing of several Machine Learning models, the best one is selected for the classification task
Classification
Classification of the malware into families, the results are collected and discussed

Figure 1: Steps of the proposed study.

methods with just a few different families but a big quantity of examples, the key for training Machine Learning (ML) models. The information regarding the families selected are shown in Table 1.

3.2 Feature Extraction

We applied 4 different Image filters to the database in order to extract features from each image and create feature vectors for the ML models. The next subsections report information regarding the Image filters.

It is worth noting that this feature extraction step converts all the images into fixed-size vectors, while the initial images in the database have different sizes and dimensions. Indeed, this process modifies the database, which is not technically composed of images anymore. The approaches in the literature (see Section 2) suggest to either resize the images to the same size, usually a very small one, or convert the feature vectors to matrices and tract them as images, in order to apply Deep Learning model on them. We discuss these solutions in Section 6.

3.2.1 Gabor Filter

The Gabor filters are linear filters used for texture analysis. They basically look for patterns of specific frequency content in the image in a localized region around a point of analysis. We link to reference for further details on the Gabor filters (Movellan, 2002).

3.2.2 Color Layout Filter

The images in the database did not contain figures or shapes but basically just different distribution of colours (grayscales) over different areas. Therefore, we tested a couple of filters focused on colours, the Color Layout Filter and the Autocolor Correlogram Filter (see next subsection), which extract colour layout descriptors capable of capturing the spatial distribution of colour in an image. The Color Layout filter extracts the MPEG7 features (Kasutani and Yamada, 2001; Cieplinski, 2001), the multimedia content description standard. The filter divides an image into several blocks and computes the average colour for each one, and then features are calculated from the averages.

3.2.3 Autocolor Correlogram Filter

The Autocolor Correlogram filter focuses on the colour correlation of an image and encodes the spatial similarities of colours in the image. Briefly, the colour correlogram merges statistics of amount for each colour (a traditional colour histogram), with spatial information on the distribution of the colour.

3.2.4 GIST Descriptor

We extract the GIST descriptor (Oliva and Torralba, 2001) for each image. Intuitively, the GIST procedure summarizes the gradient information (scales, orientations, edges) of specific areas of the image, to provide a rough description of the entire image.

3.3 Machine Learning Models

We use `Weka 3.8.3`[1] on a Linux environment to assessing different Machine Learning models on the feature vectors extracted from the malware database. We decided for standard ML models because we were interested to test different feature extraction approaches and combinations. We wanted to focus the attention on the feature vector generation, instead of the ML model. Moreover, one more work taking

[1]https://www.cs.waikato.ac.nz/~ml/weka/

into account directly the pictures and applying Deep Learning models is in progress (see Section 6).

We tried many ML models among the ones available in the Weka Framework, Table 3 shows the model assessment experimental results. In details, we perform experiments with K-nearest neighbours classifier (Aha et al., 1991) with Euclidean Distance as distance function, Decision table classifier (Kohavi, 1995), the C4.5 (J48) decision tree classifier (Quinlan, 2014), and random decision forest classifier (Breiman, 2001). We point to the references for further information regarding these ML models.

4 RESULT

Notations. We compute the metrics of accuracy (Acc), precision (PR), recall (RC) and F-measure (Fm) to estimate the performances of our approach. We considered one specific family at the time and defined the metrics as follows:

$$PR = \frac{TP}{TP+FP};\ RC = \frac{TP}{TP+FN};$$
$$Fm = \frac{2PR\,RC}{PR+RC};\ Acc = \frac{TP+TN}{TP+FN+FP+TN}$$

where TP is the number of malware correctly identified in that family (True Positives), TN is the number of malware correctly identified as not belonging to that family (True Negatives), FP is the number of malware incorrectly identified in that family (False Positives), and FN is the number of malware incorrectly identified as not belonging to that family (False Negatives).

As we pointed out in the previous section, we preprocessed the data and resize the database. The original dataset contained 24549 malware (Argus Cyber Security Lab, 2020; Wei et al., 2017), divided into 71 families. Each family is also split into several varieties, up to 8 varieties for the same family. Nevertheless, most of the families have just a few malware as samples. Accordingly, we took into account only 10 families and grouped under the same class all the varieties for each one. The information regarding the families selected are shown in Table 1. We removed 60 families up to 71 but this process narrows down the number of malware only to 20748, by reducing the size of the dataset just of the 15%. The authors are planning to take into account the entire dataset in future works, this point is discussed in Section 6. It is worth noting that some family have many more examples than the others, for instance, only the Airpush family counts for the 38% of the entire database. The aggregate result of the classification task shown in this Section refers to the weighted averange.

Table 1: Families selected into the malware database after preprocessing.

Family	No. of Malware
Airpush	7843
BankBot	648
Dowgin	3385
DroidKungFu	545
FakeInst	2172
Fusob	1275
Jisut	560
Kuguo	1199
Mecor	1820
Youmi	1301
Total	20748

We tried different filters and combination of feature vectors, Table 2 reports information regarding these databases of vectors, where “Correlogram“ refers to the Autocolor Correlogram filter (see Section 3.2.3), “Color“ to the Color Layout filter (see Section 3.2.2), the Number of Attributes refers to the size of the vectors, and the precision refers to the Random Forest classifiers.

Table 2: Result of experiments on different feature vectors, the precision refers to the Random Forest classifiers.

Feature Extraction	No. Attributes	Precision
GIST	960	91.27%
Correlogram + Color	1058	84.12%
Gabor + Color	93	82.97%
Correlogram	1024	79.93%
Gabor	60	72.26%

We evaluated 4 different ML models: k-nearest neighbours classifier called IBk in Weka (Instance-Based Learner), Decision Table classifier, J48 classifier and random forest classifier. The result shows in Table 3 demonstrates that the Random Forest classifier stood out among all the ML models examined. Therefore, the following experiments, reported in this Section, refers to the Random Forest classifiers.

Table 3: Model assessment.

ML model	Precision	Recall	F-Measure
Random Forest	0.91	0.91	0.91
IBk	0.88	0.88	0.88
J48	0.84	0.84	0.84
Decision Table	0.74	0.73	0.71

The confusion matrix for each family is reported in Table 5, while its graphical normalized representation is shown in Figure 2. Table 4 summarizes the experiment results achieved by the Random Forest model.

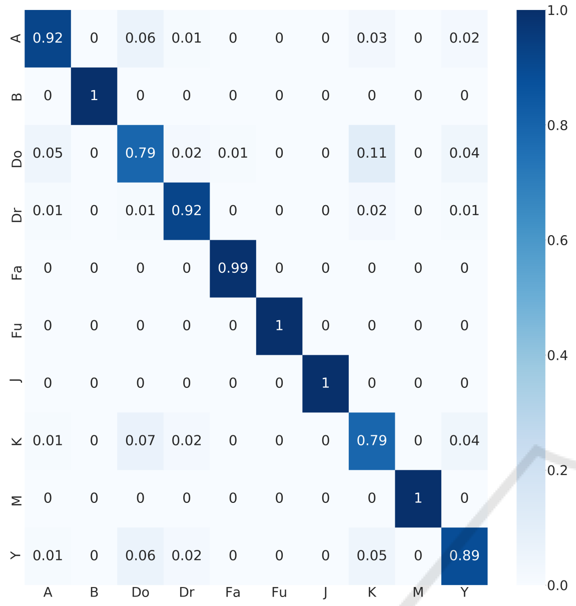

Figure 2: Normalized confusion Matrix with True label on the y axis and Predicted label on the x one.

5 DISCUSSION

The results in Table 2 and Table 3 demonstrate that the combination of GIST descriptor and Random Forest classifier achieves the best performance in the classification task.

The feature vectors created by the Color layout filter and Gabor filters may be too small in size for classifying the malware properly since they have less than 100 attributes each (93 and 60 respectively). Nonetheless, the Color Layout filters, which has just 33 attributes, improve considerably the performance of the classification; the Gabor filter only achieved a precision of 72%, while the Gabor and Color filters together jumped up to 83%. This improvement may be explained with the importance of the colour in this classification task since the images are essentially a colour (grayscale) distribution with no shapes, edges or elements as a standard image classification task could have.

Probably, the GIST filter outperforms the other feature extraction approaches because has a suitable size (960 attributes) to provide variety and richness information to classify the malware correctly. Also, it is the one that provides the broader representation of the image, because it summarizes information regarding orientation, scales and gradients.

As far as the ML models are concerned, the experimental results prove that the Random Forest classifier outperforms all the other models. By looking at Table 4 and the related heat map in Figure 2, the data confirms that the classifier achieves good performance, with accuracy values close to 1 for most of the families except Airpush, Dowgin, Kuguo and Youmi. In particular, the performance in the precision evaluation for Dowgin and Kuguo drop down the final result, indeed, they are the only two families with precision values under 0.89 (0.79 both of them). This is clearly displayed in Figure 2, where we can see the differences in performances between these two families and all the others. Moreover, it appears that many samples of Dowgin malware were wrongly classified as Kuguo and vice versa. These two families together count for the 19% of the total number of malware in the database, thus they considerably influence the final result. The anomaly may be explained by looking at the two family images. Figure 3 reports a comparison between two samples of the Dowgin and Kuguo.

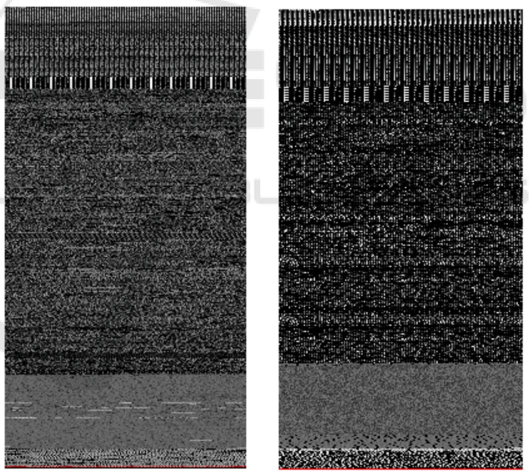

Figure 3: Example of malware belonging to Kuguo (left) and Dowgin (right) family.

The two images look similar, they have comparable patterns in colour disposition and distribution, which may have led the classifier to misclassified them.

6 LIMITATIONS AND FUTURE WORKS

This paper reports preliminary results and the authors plan to extend the research with further experiments in the malware classification and detection field.

Table 4: Summary of experiment result on the malware families.

TP Rate	FP Rate	Accuracy	Precision	Recall	F-Measure	Family
0.964	0.050	0.955	0.922	0.964	0.942	Airpush
0.989	0.000	1.000	0.997	0.989	0.993	BankBot
0.824	0.044	0.934	0.785	0.824	0.804	Dowgin
0.754	0.002	0.992	0.922	0.754	0.829	DroidKungFu
0.995	0.001	0.998	0.989	0.995	0.992	FakeInst
0.994	0.000	1.000	0.999	0.994	0.996	Fusob
0.975	0.000	0.999	0.996	0.975	0.986	Jisut
0.708	0.012	0.972	0.786	0.708	0.745	Kuguo
0.995	0.000	0.999	0.998	0.995	0.996	Mecor
0.693	0.006	0.976	0.894	0.693	0.781	Youmi
0.913	0.027	0.969	0.913	0.913	0.911	Weighted Avg.

Table 5: Summary about confusion matrix per family.

Family	TP	TN	FP	FN
Airpush	7561	12263	642	282
BankBot	641	20098	2	7
Dowgin	2788	16600	763	597
DroidKungFu	411	20168	35	134
FakeInst	2161	18552	24	11
Fusob	1267	19472	1	8
Jisut	546	20186	2	14
Kuguo	849	19318	231	350
Mecor	1811	18924	4	9
Youmi	902	19340	107	399

First of all, we plan to include all the families in the database. By removing the preprocessing step, we are interested in moving the analysis to a wider investigation, and studies the performances of ML models when trained with just a few examples.

Moreover, we are interested in applying Deep Learning models directly on the images. To do so, we need to adopt new approaches to resize the database. The images are greatly different in sizes and dimensions, and many of them are also notable big for grayscale images (more than 4 Mb). The Deep Learning analysis may be timing and computationally expensive. This approach needs to be carefully validated, in order to find the best trade-off between efficiency and accuracy. One solution could be to extract features and then convert the vectors to matrices and then treat them as images. It could be feasible in terms of time and computational power, but the analysis would be still on feature vectors instead of real images. On the other hand, it may be achievable to apply Deep Learning models on the images themself, without a significant efficiency loss, by performing standardization techniques and applying filters for compression.

We did not take into account the obfuscation, which is a serious complication in malware detection, since it may mislead the classification. However, we plan to test also the robustness of the approaches examined by applying adversarial learning approaches and evaluate whenever the models are still able to detect the malware or not.

7 CONCLUSION

The paper presents a brief analysis of different feature extraction techniques and machine learning models to address the problem of classifying malware into families. The experiments were performed on a dataset of more than 20 thousand malware (Argus Cyber Security Lab, 2020; Wei et al., 2017), divided into 10 different families. The malware were converted to grayscale images by reading the binaries into matrixes. We evaluated four different Feature extraction approaches (Gabor filter, GIST descriptor, Color Layout filter and Autocolor Correlogram filter) and several combinations between them. Also, we perform a model assessment with different machine learning models. In details, we tested K-nearest neighbours classifiers, Decision table classifiers, the C4.5 decision tree classifiers, and Random Decision Forest classifiers. Among all of them, the Random Forest classifier achieved the best result. It reached 0.97 as accuracy, 0.91 as precision, recall, and f-measure. The authors aim to extend the paper by adding more classifiers (Marulli and Visaggio, 2019; Pota et al., 2019), feature extraction techniques (Amato et al., 2018; Marulli and Mercaldo, 2017), and making the entire process robust from potential attacks against machine learning.

ACKNOWLEDGEMENTS

This work has been partially supported by MIUR - SecureOpenNets and EU SPARTA contract 830892, CyberSANE projects, and the EU project CyberSure 734815.

REFERENCES

Agarap, A. F. (2017). Towards building an intelligent anti-malware system: a deep learning approach using support vector machine (svm) for malware classification. *arXiv preprint arXiv:1801.00318*.

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine learning*, 6(1):37–66.

Akarsh, S., Simran, K., Poornachandran, P., Menon, V. K., and Soman, K. (2019). Deep learning framework and visualization for malware classification. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 1059–1063. IEEE.

Amato, F., Di Martino, B., Marulli, F., and Moscato, F. (2018). A federation of cognitive cloud services for trusting data sources. In *Conference on Complex, Intelligent, and Software Intensive Systems*, pages 1022–1031. Springer.

Argus Cyber Security Lab (2020). Android malware dataset. https://amd.arguslab.org. Accessed: 26-02-2020.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Canfora, G., Martinelli, F., Mercaldo, F., Nardone, V., Santone, A., and Visaggio, C. A. (2018). Leila: formal tool for identifying mobile malicious behaviour. *IEEE Transactions on Software Engineering*, 45(12):1230–1252.

Casolare, R., Martinelli, F., Mercaldo, F., and Santone, A. (2019). A model checking based proposal for mobile colluding attack detection. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5998–6000. IEEE.

Cieplinski, L. (2001). Mpeg-7 color descriptors and their applications. In *International Conference on Computer Analysis of Images and Patterns*, pages 11–20. Springer.

Cimino, M. G., De Francesco, N., Mercaldo, F., Santone, A., and Vaglini, G. (2020). Model checking for malicious family detection and phylogenetic analysis in mobile environment. *Computers & Security*, 90:101691.

Iadarola, G., Martinelli, F., Mercaldo, F., and Santone, A. (2019). Formal methods for android banking malware analysis and detection. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 331–336. IEEE.

Kabanga, E. K. and Kim, C. H. (2017). Malware images classification using convolutional neural network. *Journal of Computer and Communications*, 6(1):153–158.

Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., and Iqbal, F. (2018). Malware classification with deep convolutional neural networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE.

Kasutani, E. and Yamada, A. (2001). The mpeg-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 1, pages 674–677. IEEE.

Kohavi, R. (1995). The power of decision tables. In *European conference on machine learning*, pages 174–189. Springer.

Marulli, F. and Mercaldo, F. (2017). Let's gossip: Exploring malware zero-day time windows by social network analysis. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 704–709. IEEE.

Marulli, F. and Visaggio, C. A. (2019). Adversarial deep learning for energy management in buildings. In *Proceedings of the 2019 Summer Simulation Conference*, page 50. Society for Computer Simulation International.

McLaughlin, N., Martinez del Rincon, J., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z., Doupé, A., et al. (2017). Deep android malware detection. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 301–308.

Mercaldo, F. and Santone, A. Deep learning for image-based mobile malware detection. *Journal of Computer Virology and Hacking Techniques*, pages 1–15.

Movellan, J. R. (2002). Tutorial on gabor filters. *Open Source Document*.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. (2011). Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*, pages 1–7.

Ni, S., Qian, Q., and Zhang, R. (2018). Malware identification using visualization images and deep learning. *Computers & Security*, 77:871–885.

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175.

Pota, M., Marulli, F., Esposito, M., De Pietro, G., and Fujita, H. (2019). Multilingual pos tagging by a composite deep architecture based on character-level features and on-the-fly enriched word embeddings. *Knowledge-Based Systems*, 164:309–323.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

StatCounter (2020). Mobile operating system market share worldwide. https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed: 01-03-2020.

Statista platform (2018). Average number of new android app releases per day from 3rd quarter 2016 to 1st quarter 2018. https://www.statista.com/statistics/276703/android-app-releases-worldwide/. Accessed: 01-03-2020.

Suarez-Tangil, G., Dash, S. K., Ahmadi, M., Kinder, J., Giacinto, G., and Cavallaro, L. (2017). Droidsieve: Fast and accurate classification of obfuscated android malware. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 309–320.

Wei, F., Li, Y., Roy, S., Ou, X., and Zhou, W. (2017). Deep ground truth analysis of current android malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'17)*, pages 252–276, Bonn, Germany. Springer.