information

Article

Malware Detection Based on Code Visualization and Two-Level Classification

Vassilios Moussas [1,†] and Antonios Andreatos [2,*,†]

[1] School of Engineering, University of West Attica, Aigaleo, Attica 12210, Greece; vmouss@uniwa.gr
[2] Division of Computer Engineering and Information Science, Hellenic Air Force Academy, Dekeleia, Attica 13671, Greece
* Correspondence: antonios.andreatos@hafa.haf.gr
† These authors contributed equally to this work.

Abstract: Malware creators generate new malicious software samples by making minor changes in previously generated code, in order to reuse malicious code, as well as to go unnoticed from signature-based antivirus software. As a result, various families of variations of the same initial code exist today. Visualization of compiled executables for malware analysis has been proposed several years ago. Visualization can greatly assist malware classification and requires neither disassembly nor code execution. Moreover, new variations of known malware families are instantly detected, in contrast to traditional signature-based antivirus software. This paper addresses the problem of identifying variations of existing malware visualized as images. A new malware detection system based on a two-level Artificial Neural Network (ANN) is proposed. The classification is based on file and image features. The proposed system is tested on the 'Malimg' dataset consisting of the visual representation of well-known malware families. From this set some important image features are extracted. Based on these features, the ANN is trained. Then, this ANN is used to detect and classify other samples of the dataset. Malware families creating a confusion are classified by a second level of ANNs. The proposed two-level ANN method excels in simplicity, accuracy, and speed; it is easy to implement and fast to run, thus it can be applied to antivirus software, smart firewalls, web applications, etc.

Keywords: malware classification; malware visualization; malimg dataset; artificial neural network; image features; ensemble; confusion matrix; Matlab

Citation: Moussas, V.; Andreatos, A. Malware Detection Based on Code Visualization and Two-Level Classification. *Information* **2021**, *12*, 118. https://doi.org/10.3390/info12030118

Academic Editor: Avinash Srinivasan and Arkaitz Zubiaga

Received: 31 December 2020
Accepted: 4 March 2021
Published: 11 March 2021

1. Introduction

Malicious software or Malware has become a global industry worth millions of euros and is growing every year with increasing dynamics. Depending on its functionality, malware is divided into several categories, namely Viruses, Worms, Trojans, Backdoors, etc. Due to the rapid proliferation and production of malware, there is an exponential increase in the number of new signatures released every year [1].

McAfee Labs Threats Reports reveal that 100,000,000 new malware samples were discovered during Q1 + Q2 of 2020, whereas Total Malware for the same period exceeded 1,200,000,000 samples [2].

Thus, it is essential to detect and prevent malware attempting to damage information systems, as well as, single users' computers. Malware classification is a common task which can be accomplished by machine learning models quite efficiently [3].

Reverse engineering compiled malware executables is a task with a steep learning curve, meaning that it is difficult to learn and that expending a lot of effort does not increase proficiency by much. Typical approaches of malware analysis and classification include static and dynamic code analysis [1]. These techniques require either disassembly or execution of malware code.

Like any executable binary file, a malware executable is represented as a string of zeros and ones. A string is also a vector of hexadecimal values and as such, it can be reshaped into a matrix and viewed as an image. Once the malware is converted into grayscale images, malware detection can be reduced to an image recognition problem. Malware samples belonging to the same family present significant visual similarities when converted to images. This is due to code re-use when creating new variants [1].

Visualization of compiled malware executables does not require code analysis but still shows significant performance. Furthermore, it is resilient to popular obfuscation techniques such as section encryption, packing and polymorphism. When malware samples belonging to the same family are packed with the same packer, it is possible that the images of packed malware look similar [1]. Donahue et al. proposed a method for packed malware visualisation [4].

Malware detection and classification through visualization is significantly faster, as well as more accurate than traditional code analysis methods [1,5].

Various classification approaches for classifying malware programs after visualizing them as images using only compiled files have appeared [6].

Objective

The objective of this work is to propose a new method for classifying malware based on the visualization of executables. For this, a new set of image and file features is proposed. The visual representation of malware is used to feed and train an Artificial Neural Network (ANN). Once trained, the ANN can easily and successfully identify new variations of known malware families. The low complexity of the proposed method achieves fast response and limited computational power compared to other methods proposed in the bibliography.

2. Related Work

Visualization of compiled malware executables is not a new approach. The first efforts of visualizing binary files for computer security purposes were reported back in 2008 [7].

In 2009 Quist and Liebrock presented a method using dynamic analysis of program execution to visually represent the overall flow of a program [8].

Conti et al. [9] introduced several interesting visualization tools into an environment resembling a hex editor. The three visualizations work simultaneously to improve the workflow of the analysts. The 'Byteview' visualization provides an at-a-glance view of an entire file, where each byte is represented as a pixel. This is feasible because both code bytes and image pixels range from 00 to FF in hexadecimal. The intensity of each pixel dependents on the hex value of the corresponding byte. Hence, similar code sequences produce similar images.

The 'Byte Presence' display works side-by-side with the 'Byteview' display. Each row of pixels in the 'Byte Presence' display summarizes the existence of bytes in the 'Byteview' display. 'Dot Plot' visualization is borrowed from biology, where a dot plot is used to align genome sequences. In this instance, the dot plot is used to compare two files. Such a visualization shows the presence of similar byte sequences between files. However, this visualization can only be used on a subset of each file due to memory and display issues.

In the past 12 years, various classification approaches for classifying malware programs after visualizing them as images using the compiled files have appeared [5,6].

In 2011 a milestone paper was published by Nataraj et al. [1]. The authors proposed a method for visualizing and classifying malware using image processing techniques.

Malware binaries were visualized as gray-scale images. Images of different malware families appear visually similar and distinct from those of different families. Based on this observation, a classification method using image texture analysis was proposed. GIST features were extracted from malware images to be used for classification via the K-Nearest Neighbor technique (k-NN). A 'gist' is an abstract representation of an image which

spontaneously activates its memory representation and category, and is obtained using a wavelet decomposition [10].

Neither disassembly nor code execution is required for classification based on visualization. Experimental results on a malware database of 9339 samples organized in 25 families achieved 98% classification accuracy (under specific conditions).

If the families which create confusion are combined together as one, the recomputed accuracy increases to 0.992. A result of the work of Nataraj et al. was the 'Malimg' dataset, which was later made available to the public.

In the years to come, various approaches for classifying gray scale images of malware executables based on texture similarity appeared, using Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Artificial Neural Networks (ANNs), autoencoders, etc. [5,11,12].

Makandar and Patrot extracted image features using Gabor wavelet transform and GIST. They used an ANN for the classification of 3131 binary samples comprising 24 unique malware families of the Mahenhur dataset, achieving an accuracy of 96.35% [13].

Vasan et al. discerned three major tools in malware identification based on visualization: (a) statistical similarity measurements; (b) machine learning and (c) deep learning [5].

Mallet in a nice tutorial presented an interesting approach to Malware Classification using Convolutional Neural Networks and Keras and reached a final accuracy of 95% [3]. This performance can be improved by creating a larger dataset using a preprocessing step described in the article. What is of utmost importance for our research is the remark that, "although most of the Malwares were well classified, Autorun.K is always mistaken for Yuner.A. This is probably because we have very few samples of Autorun.K in our dataset and that both are part of a close Worm type. Moreover, Swizzor.gen!E is often mistaken with Swizzor.gen!I, which can be explained by the fact that they come from really close kinds of families and types and thus could have similarities in their code" [3].

Several researchers, in order to increase malware detection accuracy even more, have used Machine Learning approaches, often in combination with traditional malware analysis methods, tree-maps, thread graphs, etc. [5].

The most commonly used Machine Learning approaches are Convolutional Neural Networks (CNN) and deep learning, because they greatly facilitate feature extraction [14–19].

Narayanan and Davuluru [6] used an ensemble approach with Support Vector Machine for the BIG 2015 dataset. This dataset contains an assembly file and a compiled file for each malware program. Compiled files are visualized as images and are classified using Convolutional Neural Networks (CNNs). Assembly files consist of machine language opcodes that are distinguished among classes using Long Short-Term Memory (LSTM) networks after converting them into sequences. In addition, features are extracted from these architectures (CNNs and LSTM) and are classified using a support vector machine or logistic regression, achieving an accuracy of 99.8%.

Vasan et al. [5] proposed a hybrid deep learning model (called 'IMCFN') based on visualization, which uses a fine-tuned CNN architecture for malware detection and classification. Data augmentation, as well as conversion of malware binaries into color images are used to optimize the performance of the IMCFN algorithm and to cope with imbalanced datasets. Their method achieved the best results in terms of accuracy (98.82%). Vasan et al. [5] also presented an up-to-date comparative summary of Multi-class Malware Family Classification Techniques, all using the 'Malimg' Dataset, in their Table 8. All these approaches resolve code obfuscation issues; the main challenge that they face however, is the relatively high computational power for complex texture feature extraction using methods such as GIST, DSIFT, SURF, LBP or GLCM [16]. Another drawback is that these feature extraction techniques are less efficient when applied to large datasets [5].

3. Methodology

In this work we focus on malware classification using only the visualised images of compiled malware executables. The Malimg Dataset, a real-life malware database for the Windows operating system, most popular among researchers, will be used [1,5]. Hence the

problem of malware classification has been reduced to an image recognition problem using specific criteria. Therefore in this work, in contrast to other approaches presented in the bibliography, Artificial Neural Networks (ANNs) are used, in order to reduce processing time. Matlab was used for processing the visualized malware images and simulating the classification methods.

3.1. About the Malimg Dataset

The Malimg Dataset contains 9339 malware images, organized in 25 families [3]. Figure 1 shows representative images from six malware families: Adialer.C, Agent.FYI, Rbot!gen, Lolyda.AA1, Fakerean and Swizzor.gen!E. Information regarding the families of the dataset is given in Table 1. Our objective is to devise an ANN classifying visualized malware.

The Malimg Dataset is quite unbalanced (Figure 2). More than 30% of the images belong to class 2: Allaple.A and 17% to class 3: Allaple.L! [1,3]. This is an important issue which affects the design and optimization of the ANN. In order to cope with this issue, either a restricted number of samples from these two families should could be used, or, a 'padding' of the other families with artificially created data. In this work we kept unaltered the original malimg samples, for compatibility and comparison with other works.

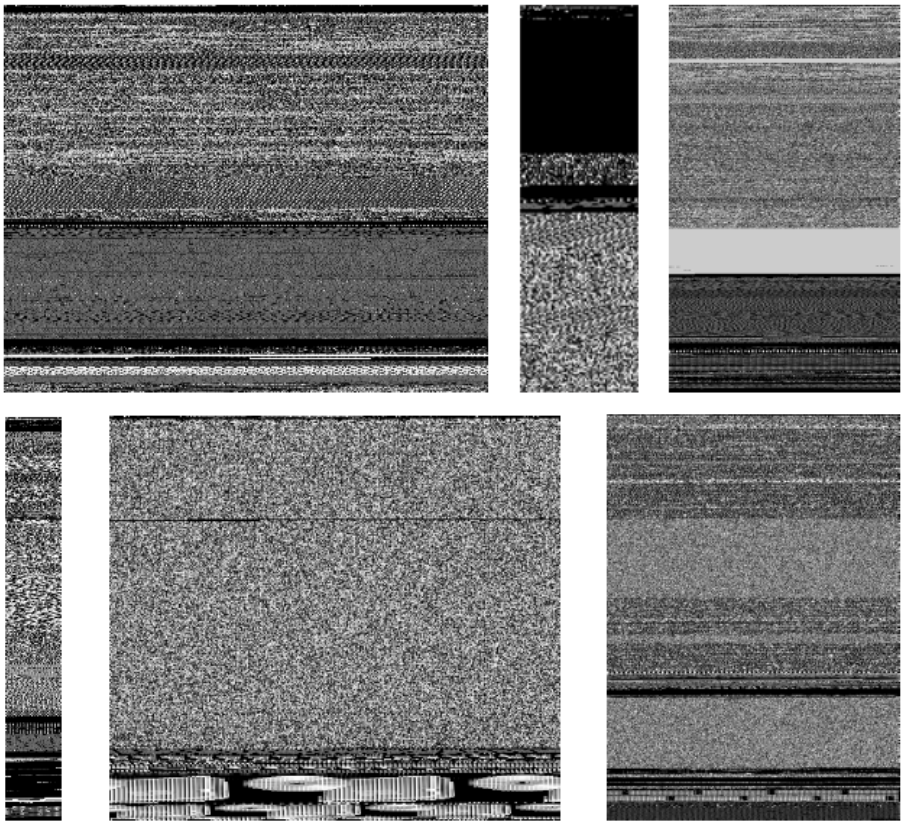

Figure 1. Representative visualized malware images.

Table 1. Information about the Malimg Dataset Families.

No.	Family Name	Malware Type
1	Adialer.C	Dialer
2	Agent.FYI	Backdoor
3	Allaple.A	Worm
4	Allaple.L	Worm
5	Alueron.gen!J	Worm
6	Autorun.K	Worm:AutoIT
7	C2LOP.P	Trojan
8	C2LOP.gen!g	Trojan
9	Dialplatform.B	Dialer
10	Dontovo.A	Trojan Downloader
11	Fakerean	Rogue
12	Instantaccess	Dialer
13	Lolyda.AA1	PWS
14	Lolyda.AA2	PWS
15	Lolyda.AA3	PWS
16	Lolyda.AT	PWS
17	Malex.gen!J	Trojan
18	Obfuscator.AD	Trojan Downloader
19	Rbot!gen	Backdoor
20	Skintrim.N	Trojan
21	Swizzor.gen!E	Trojan Downloader
22	Swizzor.gen!I	Trojan Downloader
23	VB.AT	Worm
24	Wintrim.BX	Trojan Downloader
25	Yuner.A	Worm

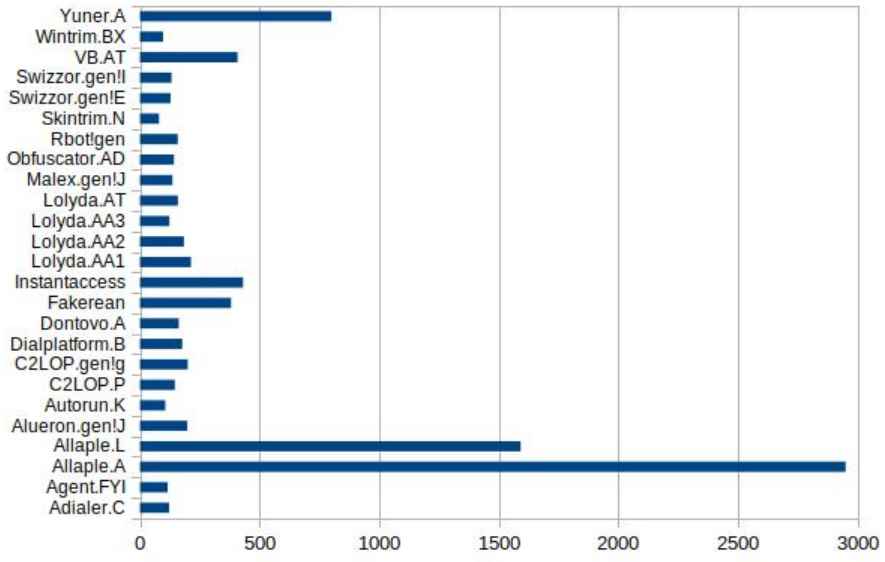

Figure 2. Number of samples per family.

3.2. Preprocessing

To increase matching accuracy, two types of features have been considered: image features (such as geometry, height, width, entropy, contrast, correlation, energy, homogeneity, mean image intensity, histogram, etc.) and file features (such as size, type, etc). A Matlab script was written to extract specific features from each image. After removing some redundant characteristics regarding image geometry which were correlated with the file size, the following features were finally selected.

1. File size: File size is characterizing each family, since all members have similar sizes.
2. Entropy: Entropy is a statistical measure of randomness used to characterise the texture of the input image.
3. Contrast: Contrast is the difference in luminance that makes an object in an image distinguishable.
4. Correlation: The correlation coefficient between an image and the same image processed with a median filter.
5. Energy: Grayscale images have gray levels, and gray levels are units of energy.
6. Homogeneity: The distribution of gray values within an image.
7. Mean Image Intensity: Every pixel of a grayscale image has an intensity (value) in the range [0, 255]. Mean Image Intensity is the mean of the intensity of all pixels.

Figure 3 shows the correlation matrix of the final selected features after removing the more correlated ones.

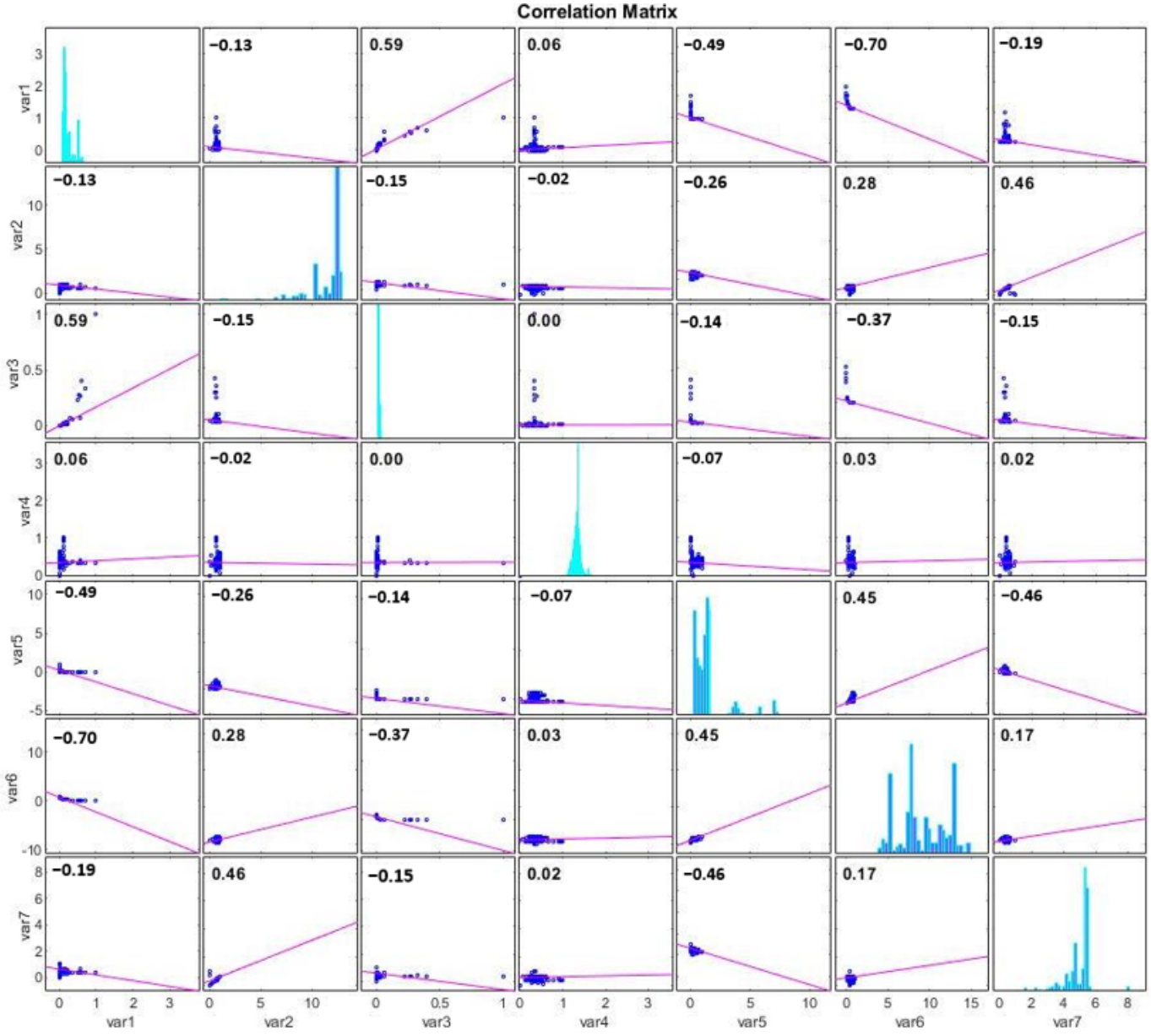

Figure 3. Correlation matrix of the final selected features (variables 1 to 7 as shown in the list above).

3.3. Training the ANN

Once the features for each sample were extracted, a function to randomly split the Dataset in training data and test data, following the popular 70%–30% ratio was used (e.g., [5]). The 9339 samples of the Malimg Dataset were split in two datasets: a Training set of 6537 samples, and a Testing set of 2802 samples that is also considered as 'unseen'.

The test dataset was further divided into two subsets, a 15% 'validation' subset to monitor overfitting during the training phase and a 15% pure 'test' subset for the final test.

3.4. Defining the ANN Architecture

A Pattern Recognition feed-forward ANN is implemented to process the database and recognize the virus categories. The ANN architecture comprises at least 3 layers. An input layer, an output layer and one or more hidden layers. The number of nodes in the Input layer is equal to the number of features used, the number of nodes in the output layer is equal to the number of Virus categories studied, and the number of nodes in the hidden layer is subject to investigation in order to achieve better performance.

Several ANN configurations were tested. The full size version uses 7 nodes to input the 7 selected features and 25 outputs for the corresponding dataset categories. One to three hidden layers were tested and the hidden layer size ranged from 2 to 256 nodes. The best results were obtained for the single hidden layer using 64 nodes for the double hidden layer ANN again with 64 nodes per layer and for the 3 hidden layers ANN using 128 nodes per layer (Figure 4).

The negligible improvement in accuracy offered by the 2- and 3-hidden layer ANNs comes at an extra cost of complexity and a possible loss of generalization; therefore, the simplest configuration is finally selected for the ANN with one hidden layer of 64 nodes (Figure 5).

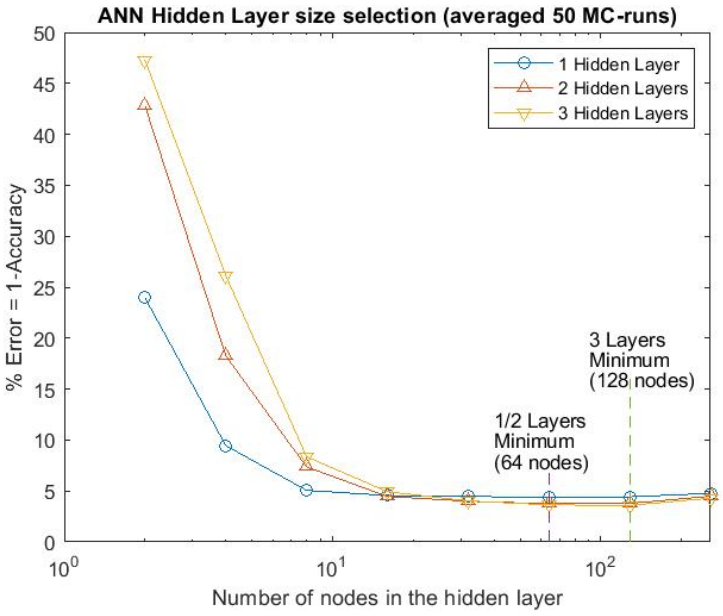

Figure 4. ANN node number selection.

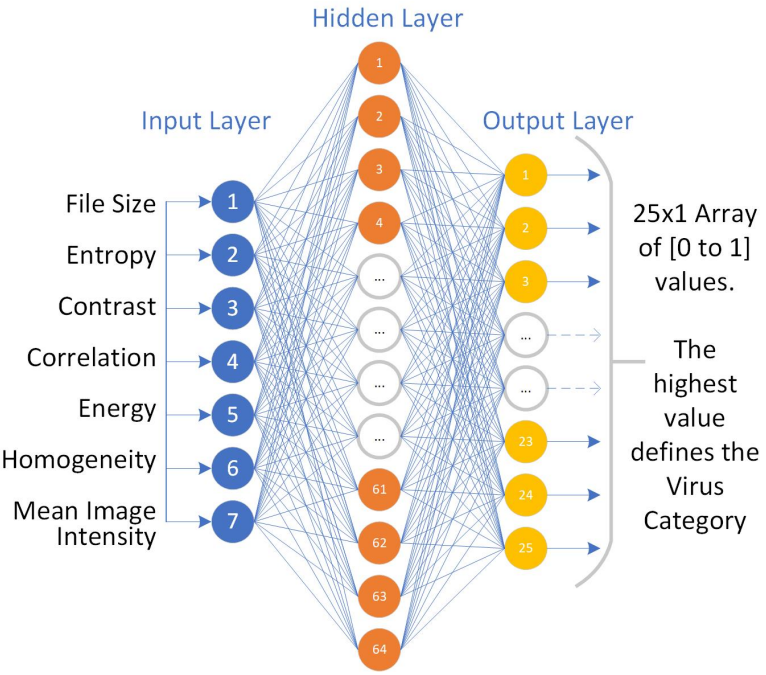

Figure 5. Single ANN architecture.

Figure 6 shows the confusion matrix produced by a single ANN, with one hidden layer of 64 nodes; the accuracy is 96%.

Preliminary classification results indicate that some malware families with similar characteristics confuse the classification process and limit precision and accuracy. The greatest confusion is created from the similarity between Autorun.K and Yuner.A (families no. 6 and 25), as well as Allaple.A and Malex.gen!J (families no. 3 and 17). In particular, all Autorun.K images were classified as members of Yuner.A, whereas most of family 17 images were classified as family 3 members. These pairs of families are marked with light blue (families no. 6 and 25) and yellow (families no. 3 and 17) in Figure 6. A smaller confusion between families 7, 8, 21, and 22 is also observed: more than half of family 7 images were classified in other categories (8, 21, 22).

This problem was initially faced by Nataraj et al. [1] where a separation of the dataset in 22 families was suggested. Other works using the malimg dataset have also encountered the same problem [3,5,18].

To cope with this issue, during preprocessing families Autorun.K and Yuner.A were merged into a new family called group 1 (G1); similarly, families Allaple.A and Malex.gen!J formed group 2 (G2). Hence, a variation of the original dataset, containing the same number of samples grouped in 23 families was created. We call this 'improved dataset' or '23-families dataset'.

The proposed architecture is a two-level ANN. The first level performs a coarse classification. The input to the first level is the original dataset but organized in 23 families as described above. The second level performs a fine classification only for those groups. When a sample belonging to groups G1 or G2 is encountered, it is forwarded to a second level of processing, in order to decide the exact original family. Hence, the proposed architecture has just two ANNs at the second level: one for G1 (which decides between families Autorun.K and Yuner.A) and another for G2 (which decides between families Allaple.A and Malex.gen!J).

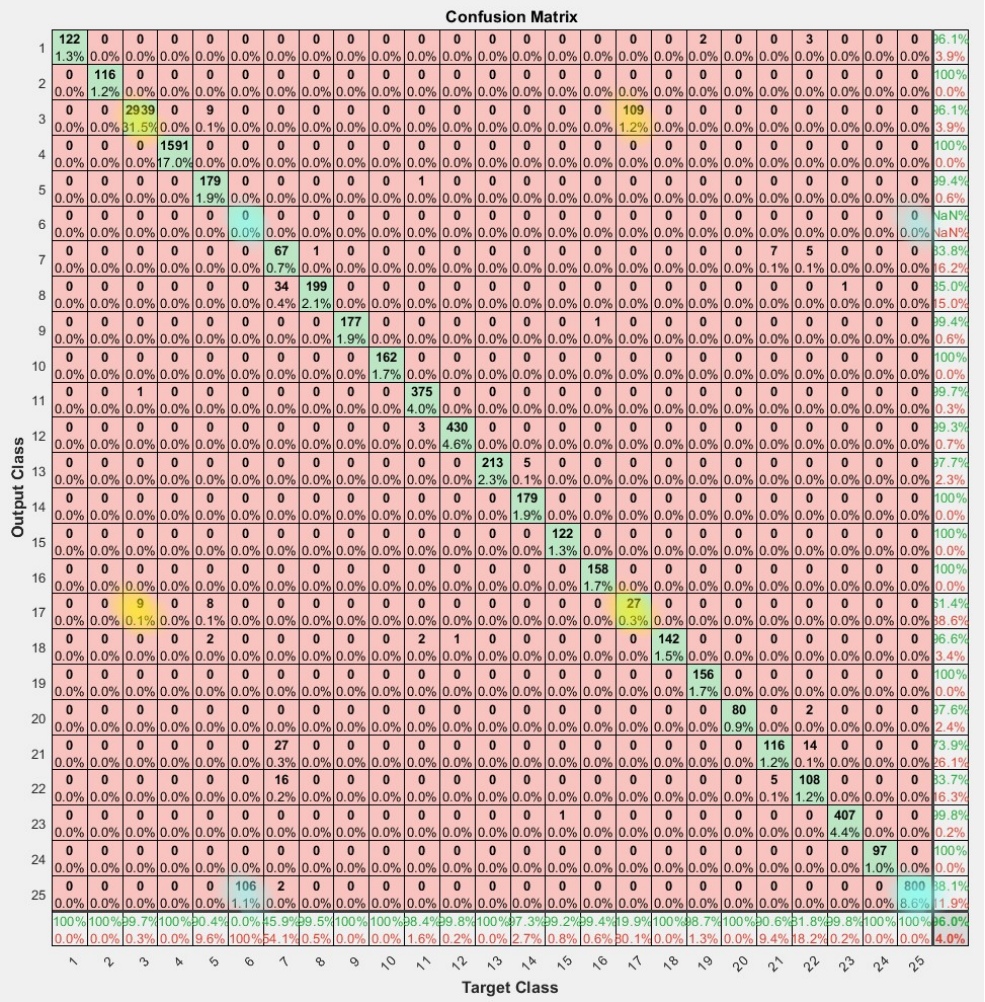

Figure 6. Confusion matrix produced by a simple ANN.

As one can see in Figure 7, a simple ANN succeeds to successfully classify the 23-families dataset. The process is over for 23 of the 25 families with a fast, simple and cost-effective architecture; two additional simple ANNs, one for each group of confused families, continue the classification if needed. The final accuracy rate is satisfactory while the complexity and run-time are very low.

Again, several different configurations were tested for the first classification step (1st level), and the best results were obtained with an ANN using one hidden layer of 64 nodes. 50 × 50 Monte Carlo runs, i.e., 50 random ANN initializations and 50 random data splits (70–30%) were tested, in order to suppress any circumstantial result.

Much simpler ANNs with only one hidden layer and a few nodes ranging from 2 to 10 –depending on the specific group– were used at the 2nd level performing the fine classification. Specifically, for the first group (G1) of merged families (6 and 25) an ANN with 2 nodes was proven sufficient, while for the second group G2 (families 3 and 17) an ANN with 16 nodes was used.

Figure 7. Confusion Matrix produced by a simple ANN and the improved dataset.

Only two ANNs are needed at the 2nd level; hence the extra complexity is small. The architecture of the two-level ANN is presented in Figure 8.

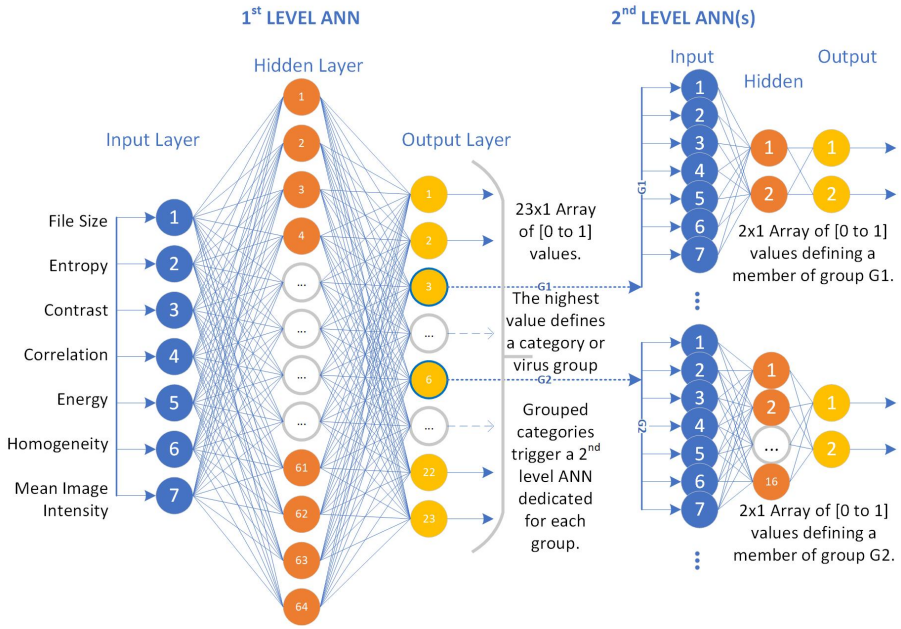

Figure 8. Architecture of the two-level ANN.

3.5. Testing Other Classification Tools

Several typical Machine Learning tools for Classification were also tested for comparison. We focused on two popular categories of classification tools, the Nearest Neighbor and the Ensemble methods, as they are commonly used by other researchers [5]. From the various Ensemble methods (Trees, Subspace, etc.) the Ensemble Bagged Trees method (EnsembleBT), as well as, from the various Nearest Neighbor methods (various k values, cosine, cubic, etc.) the k-Nearest Neighbor method with k = 1 (FineKNN), performed better than the rest in their category, and therefore, they are used for comparison with the proposed ANN.

The above classifications methods were again tested using the malimg dataset of 9339 samples split in 70% 'seen' data for training and 30% 'unseen' data for testing. To avoid any circumstantial values, due to the random separation of the dataset, the procedure (permute data - split seen/unseen - build classifier from seen—apply to unseen) was repeated several times and the results were averaged.

4. Results

4.1. Performance of the Two-Level ANN

After training and testing, the 1st of the two-level ANN reached an accuracy of 98.83%. This concerns the 5339 of the total 9339 images. The rest 4000 images were forwarded for further classification to the 2nd level, where G1 (900 images) achieved an accuracy of 100% and G2 (3100 images) achieved an accuracy of 99.41%.

Hence, the average accuracy achieved by the two-level ANN is $(5339/9339) * 98.83 + (900/9339) * 100 + (3100/9339) * 99.41 = 99.135\%$.

4.2. Performance of Other Classification Methods

The above two-level classification can also be used with other classifiers. In this research we have tested two other approaches: the k-NN classification method, as well as the Ensemble Bagged Trees classification method (EnsembleBT).

Both the ANN and Ensemble tools were implemented as standalone functions and tried over the entire dataset to measure their relative execution speed.

After 500 Monte Carlo runs the average accuracies were: for the EnsembleBT tool 98.39% and for the FineKNN tool 96.74%.

The two methods are tested also with the improved dataset and their performance was slightly improved. After 500 Monte Carlo runs the average accuracies were: for the EnsembleBT tool 98.70% and for the FineKNN tool 97.69%.

Two-level classification can be applied to these two methods as well, giving an average accuracy of 98.938% for the EnsembleBT tool and 98.288% for the FineKNN tool.

The ANN performed the 9339 recognitions in 0.024 sec (averaged after 500 MC-runs), the Fine k-NN performed the 9339 recognitions in 0.032 sec, and the Ensemble Bagged trees performed the 9339 recognitions in 0.345 sec, on a common Windows 10 laptop with 8GB RAM and AMD Ryzen 3 3200U processor with 2 CPU cores and 3 GPU cores, normal clock frequency 2.6 GHz and max clock frequency 3.5GHz. All these results are displayed in Table 2.

It is clear that any improvement offered by the Ensemble tool comes with a penalty of 10+ times longer execution time, making this approach less attractive for real time systems.

Table 2. Comparative summary of classification algorithms using the Malimg dataset.

Method	k-NN	EnsembleBT	ANN
Average time (s)	0.032	0.345	0.024
Average Accuracy @L1	97.69%	98.70%	98.83%
Average Accuracy @L2,G1	100%	100%	100%
Average Accuracy @L2,G2	98.82%	99.04%	99.09%
Overall Average Accuracy	98.288%	98.938%	99.135%

4.3. Comparison with Other Approaches

Comparison with other approaches is possible in terms of reported criteria such as precision and accuracy [5]. It is not easy to compare run-times because each researcher uses different hardware and software. Our approach can perform 9339 recognitions in 0.032 sec, much less than 0.81 sec per sample needed by the approach of [5] (which is one of the fastest). This is due to its simplicity (shallow NN vs. deep NN), as well as the avoidance of time-consuming feature extraction methods such as GIST.

The following Table 3 presents a comparative summary of classification algorithms using the Malimg dataset.

Table 3. Comparative summary of classification algorithms using the Malimg dataset.

Year	Researchers	Methods	Technique	Accuracy (%)
2011	Nataraj et al.	GIST	Machine Learning	98
2017	S. Yue	CNN	Deep Learning	97.32
2017	Makandar and Patrot	Gabor wavelet-kNN	Machine Learning	89.11
2018	Yajamanam et al.	GIST+kNN+SVM	Machine Learning	97
2018	Cui, Xue, et al.	GIST+SVM [15]	Deep Learning	92.20
2018	Cui, Xue, et al.	GIST+kNN	Deep Learning	91.90
2018	Cui, Xue, et al.	GLCM+SVM	Deep Learning	93.20
2018	Cui, Xue, et al.	GLCM+kNN	Deep Learning	92.50
2018	Cui, Xue, et al.	IDA+DRBA	Deep Learning	94.50
2019	Cui, Du, et al.	CNN, NSGA-II	Deep Learning	97.6
2020	Mallet	CNN, Keras	Deep Learning	95.15
2020	Vasan et al.	IMCFN, Color images	Deep Learning	98.82
2021	Two-level ANN	Image and file features, ANN	Two-level ANN	99.13

5. Discussion and Future Work

Based on our experiments, we believe that hierarchical detection of new malware variants has several advantages:

By applying a hierarchical taxonomy on the malware families we can reduce the vast search space required to cover all virus instances, and our tools can focus on a smaller number of classes. We can further reduce the number of features required at each level of the hierarchy, and consequently, end up with a much simpler and faster tool to implement.

The number of levels in the hierarchy is not an issue. The proposed ANN runs at a fraction of time required by the decision trees or other more complex deep learning methods.

From our results, it seems that the hierarchical detection is more profitable for the tools that build generalized models to identify a class (such as Neural Nets, Decision Trees), than for those which just compare the input with an existing bank of known cases (e.g., Nearest Neighbor). The latter are also prone to the continuous increase of available knowledge, as this greatly affects their complexity and execution time.

With the exponentially increasing number of malware today, we believe that future research should investigate the varying performance of the detection tools, under the changing load of the continuously increasing number of virus samples and families.

Future Work

- A future task is to test our ANN with additional datasets such as the BIG 2015 dataset.
- The performance of various classification methods depends on the size of the dataset; for example, k-NN performance decreases with the number of inputs. It would be interesting to compare the accuracy of the various classification methods with large datasets.
- Finally, it would be interesting to test hybrid schemes with combinations of methods, that is, one method at the 1st level and a different method at the second level, in an effort to combine their advantages. For instance, an ANN at the 1st level (which can

cope with big amounts of input data) to perform the coarse classification and a k-NN at the 2nd level (where the input data will be limited) to perform the fine classification.

6. Conclusions

In this paper, a set of image features for malware classification based visual representation was first proposed. Then, based on this set, several classification algorithms were examined: the Ensemble Bagged Trees method (Ensemble BT), the Fine k-Nearest Neighbor method (k-NN with k = 1) and an Artificial Neural Network (ANN), all with one and two levels of processing.

All of these algorithms demonstrate superior performance in terms of accuracy and run-time, while maintaining low complexity.

Image-based malware classification does not demand any domain expert knowledge, such as reverse engineering, binary disassembly, static and dynamic code analysis. At the same time, our architectures can detect obfuscated malware contained in the malimg dataset.

Our approach excels in speed compared to other complex methods presented in the bibliography. The final accuracy for the malimg dataset consisting of 9339 images is close to 99%.

Moreover, the ANN runs up to 30 times faster than the Ensemble classification method and its accuracy will increase with larger and better prepared datasets. Simpler ANNs still present satisfactory performance and are proposed for implementations with hardware limitations.

Automatic analysis, detection and classification of malware based on its visual representation has several advantages over traditional signature-based antivirus software. One of the advantages of this method is that new variations of known malware families can be instantly detected. Thus, this method could prove valuable for antivirus companies and security researchers who receive hundreds of malware everyday. It can be applied to antivirus software, smart firewalls, web application firewalls, etc.

Author Contributions: Conceptualization, A.A.; Data preprocessing & feature extraction, A.A.; Investigation and literature review, V.M. and A.A.; Methodology, V.M. and A.A.; Machine learning design and simulation, V.M. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the University of West Attica.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: We render our sincere thanks to L. Nataraj, S. Karthikeyan, G. Jacob and B.S. Manjunath for making the 'Malimg' dataset publicly available.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 July 2011; pp. 1–7.
2. McAfee Labs Threats Reports. November 2020. Available online: https://www.mcafee.com/enterprise/en-us/threat-center/mcafee-labs/reports.html (accessed on 23 December 2020).
3. Mallet, H. Malware Classification Using Convolutional Neural Networks—Step by Step Tutorial. A Quick and Easy Tutorial about an Interesting Approach to Malware Classification. 27 May 2020. Available online: https://towardsdatascience.com/malware-classification-using-convolutional-neural-networks-step-by-step-tutorial-a3e8d97122f (accessed on 18 July 2020).
4. Donahue, J.; Paturi, A.; Mukkamala, S. Visualization Techniques for Efficient Malware Detection. RiskSense Technical White Paper Series. Available online: https://www.risksense.com/wp-content/uploads/2018/05/Visualization-Techniques-for-Efficient-Malware-Detection.pdf (accessed on 28 December 2020).
5. Vasan, D.; Alazab, M.; Wassan, S.; Naeem, H.; Safaei, B.; Zheng, Q. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Comput. Netw.* **2020**, *171*, 107138. [CrossRef]

6. Narayanan, B.N.; Davuluru, V.S.P. Ensemble Malware Classification System using Deep Neural Networks. *Electronics* **2020**, *9*, 721. [CrossRef]
7. Conti, G.; Dean, E.; Sinda, M.; Sangster, B. Visual reverse engineering of binary and data files. In *Lecture Notes in Computer Science, Proceedings of the 5th International Workshop on Visualization for Computer Security, VizSec '08, Cambridge, MA, USA, 15 September 2008*; Springer: Berlin/Heidelberg,Germany, 2018; pp. 1–17.
8. Quist, D.A.; Liebrock, L.M. Visualizing compiled executables for malware analysis. In Proceedings of the 6th International Workshop on Visualization for Cyber Security (VizSec), Atlantic City, NJ, USA, 11 October 2009; pp. 27–32.
9. Conti, G.; Bratus, S.; Shubina, A.; Lichtenberg, A.; Ragsdale, R.; Perez-Alemany, R.; Sangster, B.; Supan, M.A. *Visual Study of Binary Fragment Types*; Black Hat: San Francisco, CA, USA, 2010.
10. Oliva, A.; Torralba, A. Modeling the shape of a scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **2001**, *42*, 145–175. [CrossRef]
11. Kancherla, K.S.; Mukkamala, S. Image visualization based malware detection. In Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, 16–19 April 2013; pp. 40–44.
12. Narayanan, B.N.; Djaneye-Boundjou, O.; Kebede, T.M. Performance analysis of machine learning and pattern recognition algorithms for malware classification. In Proceedings of the 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS), Dayton, OH, USA, 25–29 July 2016; pp. 338–342.
13. Makandar, A.; Patrot, A. Malware Analysis and Classification using Artificial Neural Network. In Proceedings of the 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15), Bangalore, India, 21–22 December 2015.
14. Yue, S. Imbalanced Malware Images Classification: A CNN based Approach. Submitted on 27 August 2017. Available online: https://arxiv.org/abs/1708.08042 (accessed on 30 December 2020).
15. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.; Chen, J. Detection of Malicious Code Variants Based on Deep Learning. *J. IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
16. Cui, Z.; Du, L.; Wang, P.; Cai, X.; Zhang, W. Malicious code detection based on CNNs and multi-objective algorithm. *J. Parallel Distrib. Comput.* **2019**, *129*, 50–58. [CrossRef]
17. Ni, S.; Qian, Q.; Zhanga, R. Malware identification using visualization images and deep learning. *Comput. Secur.* **2018**, *77*, 871–885. [CrossRef]
18. Yajamanam, S.; Selvin, V.R.S.; Di Troia, F.; Stamp, M. Deep Learning versus Gist Descriptors for Image-based Malware Classification. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018), Funchal, Portugal; 22–24 January 2018, pp. 553–561.
19. Makandar, A.; Patrot, A. Malware class recognition using image processing techniques. In Proceedings of the 2017 International Conference on Data Management, Analytics and Innovation (ICDMAI), Pune, India, 24–26 February 2017.