Title

MalwareInSight: Classifying Malware Images Using Convolutional Neural Networks

Abstract

Keywords

Highlights

1. The custom model performs with a ~99% accuracy with MalImg benchmark.
2. Custom architecture outperforms pretrained models VGG16 & VGG19.
3. Malware image classification is best integrated with different techniques.
4. Introduction

Each year, the world is becoming more and more dependent on the Internet. Consequently, its rapid rise has been synonymous with the growing spread of malware – software code or any program that was intentionally designed to infiltrate and gain unauthorized access to devices (IBM, n.d.). The scale of malware presence across the cyberspace is staggering, with over 1.16 billion unique signatures detected as of 2024 (AV-Atlas, n.d.). Proliferation has also been rapidly increasing. In the first half of 2024 alone, 50 million new signatures have been identified (AV-Atlas, n.d.).

The financial consequences of malware attacks are equally concerning. It is estimated that the cost of cybercrime is expected to balloon to USD 10.5 trillion by 2025 (Sausalito, 2023). These costs include damage to intellectual property, stolen information, identity theft, and the overall disruption of business operations.

In 2011, Nataraj et al. developed a unique approach to convert malware code into grayscale images. The discovery opened opportunities to turn traditional malware classification into an image classification problem. This study explored said methodology and built a deep learning model that can accurately classify different classes of malware.

* 1. Problem Statement

Due to the rapid pace of malware development, cyber defense specialists and vendors rely on automated malware analysis tools to distinguish benign from malicious programs. Most software available commercially utilize signature-based classification. This approach identifies unknown programs by comparing them with known programs often stored on a database.

Signatures are unique identifiers of a binary files and are created using static and dynamic methods. Both methods have advantages and disadvantages, and have been continuously been improved through the years (Shijo & Salim, 2015).

However, in the constant arms race against hackers and malicious players, exploring other ways to classify malware could prove worthwhile. Classification of malware images provide a unique approach that utilizes visual similarity that does not necessitate the assembly or execution of code. The original work by

Nataraj et al. (2011) utilized image processing techniques to generate promising results at 98% accuracy.

To this day, the work’s techniques and original datasets are being used as a benchmark for this type of malware classification. With recent advancements and liberalization of deep learning applications, an opportunity to apply these new techniques to this problem presents itself.

* 1. Objective

The study aimed to beat the original work’s accuracy score by using advanced deep learning techniques. The baseline is also set to also beat more recent attempts to use deep learning – primarily Convolutional Neural Network-based – in terms of performance to provide a valuable contribution to development.

Furthermore, the study explored applications of pretrained models, ones trained on millions of datapoints, to see if out-of-the-box solutions are better than custom built deep learning architectures.

1. Related Works
   1. Signature-based detection

Given the rate of malware proliferation, signature-based detection requires frequent updates on its database. This is the primary disadvantage of this method. In static analysis, the features of potential viruses are taken from binary code and are used to make models that are then used to distinguish between malware and benign. This approach extracts useful information about a malware’s behavior. The weakness is when virus coders use obfuscation techniques make detection difficult (Moser et al., 2007).

On the other hand, dynamic analysis runs potentially malicious programs in a sandbox environment. This allows dynamic analysis to be more resilient to obfuscation. However, the monitoring process is time consuming and creating a secure environment to execute these programs require surmountable resources (Shijo & Salim, 2015).

* 1. Malware Image Visualization

Converting malware files into images require the file binary to be read as a vector of 8-bit unsigned integers which are then sorted into a 2-dimensional array. This allows the array to be visualized as a greyscale image in the of 0 to 255 where black is the earlier and white is the latter.

Nataraj et al.’s (2011) original work also explored ways to classify these images into different malware classes. In the 2011 paper, the authors utilized image processing GIST (Torralba, 2003) to extract features from the image. K-nearest neighbors with Euclidean distance were then used for classification. Experiments yielded a promising 98% accuracy.

* 1. VGG 16 &19

Multiple approaches have been used for malware detection and classification, many of which have been explored in literature (Gandotra et al., 2014). Several deep neural networks are applicable to image classification. However, Convolutional Neural Networks (CNN), inspired by the Kunihiko Fukushima Neocognitron, are believed to be one of the best at the task (Pant & Bista, 2021).

Along with the development of neural networks was the liberalization of pretrained models – open-source models built on millions if not billions of datapoints. Among them, VGG 16 and VGG 19 by Simonyan & Zisserman (2015) have been two of the most impactful and used models today.

Simonyan & Zisserman (2015) were one of the first scientist to demonstrate the ability of deeper layers combined with smaller convolution filters to outperform prior configurations of the time. As the name suggests the VGG16 and 19 architectures were created with a depth of 16 and 19 layers respectively, with small (3x3) filters.

1. Data and Methods
   1. Data Set

This study utilized the MalImg dataset, the same dataset generated and tested by Nataraj et al. (2011). The copy, sourced from Kaggle, consisted of 25 different malware classes and 9,339 images. It is worth noting that there is no “benign” class, representing files that are not malicious. All images are in grayscale (Figure #), with varying lengths and widths. The full list of malware classes can be seen in Appendix 1.

* 1. Exploratory Analysis

Upon observation, the MalImg dataset has class imbalance. As shown in Figure #, select malware classes are better represented than other. Specifically, Allaple.A, Allaple.L, and Yuner.A. These imbalances could potentially affect performance and the validity of the results. Hence, are to be taken account for later.

* 1. Baselines
     1. Proportional Chance Criterion (PCC)

The PCC was calculated to establish a baseline in which the performance of the created models would be tested. This core provides an early indicator as to what is the accuracy if predictions were done at random, given the distribution of classes in the dataset.

The PCC was calculated at 14.67%. Multiplied 1.25 yields 18.34% suggesting this would be minimum score to beat for the model. However, considering this value to be too low, the paper will not be referencing this baseline when discussing the results.

* + 1. From Literature

As referenced, the MalImg dataset is a famous benchmark for researchers trying to advance malware image classification techniques. Hence, this work is not the first application of deep learning and CNNs to this problem.

An earlier work by Pant & Vista (2021), which utilized a custom CNN architecture, was able to yield 98.07% accuracy with 98% precision and 99% recall. A more recent work by Paardekooper, at al. (2022) which utilized Genetic Algorithm to optimize their CNN topology and hyperparameter tuning, yielded an astounding 98.5% accuracy. These accuracy scores will be the primary target of this study to generate meaningful contribution to the field.

* 1. Preprocessing

The images were resized to 256x256 pixels, rescaled with a factor of 1/255, with a batch size of 64. Training, validation, and test splits were done at 70-20-10 distribution respectively. Data augmentation techniques such as rotation, shear, zoom, dimension shift, and flips were tested, with fill mode set to nearest, but proved ineffective.

* 1. Modeling

The study implemented two pipelines in trying to build models that would beat the baseline scores. First would be utilizing pretrained CNN-based models, specifically VGG16 and VGG 19 by Simonyan & Zisserman (2015). These models were chosen given their popularity among image classification tasks and although have been outperformed by their successors, would make good baselines for future studies.

Second was creating a custom CNN architecture. Initially inspired by the published architecture of Pant & Vista (2021) the architecture (see Figure 2) was a result of several iterations from the original.

The custom architecture is comprised of two convolution blocks, each with a max pooling layer at the end. The blocks comprise of a single CNN layer with a 3x3 filter and a rectified linear unit (ReLU) activation function. The blocks were then flattened and passed through two fully connected layers followed by a dropout layer.

The output layer was activated by a SoftMax function while optimizer Adam, loss Categorical Cross entropy, and evaluation metric accuracy, were used in model compilation.

Training was done with a decreasing learning rate that lowers on plateau, set to a minimum of 5e-5. Checkpoints were also implemented such that the weights of the model with the best validation accuracy are saved at each epoch.

1. Results and Discussion

Table of results

* 1. Model Performance

For this study, accuracy was the primary metric of success, given that the dataset contained only malware classes and the main goal of the study is its classification. Observing Table #, it is apparent that the MalwareInSight (custom) model performed best in terms of accuracy with 98.94%, followed by 97.25% precision and 97.13% recall.

Looking the confusion matrix (Appendix #), it is apparent that the imbalance prior in the dataset did not greatly affect the ability of the custom model to classify malware images. This is especially true considering that the misclassified items were not from the least represented classes of the dataset.

The pretrained models, VGG 16 and VGG 19 also performed admirably at 97.89% and 96.72% accuracy scores respectively. It is worth noting that these pretrained models were used out-of-the-box. No unfreezing and retraining of layers were conducted. The only additional layers added before output was flattening and a fully connected layer to cater to hardware limitations.

* 1. Data Augmentation

As referred to in preprocessing, data augmentation techniques proved non-beneficial in achieving better scores in training and test. The likely reason behind this phenomenon would be the nature of the malware images.

The malware images are binaries turned image. Hence, as seen in Figure#(same as dataset), the images resemble textures rather than object. This makes typical augmentation techniques futile as they would change the binary representation of each image. The same observation was also noted in the findings of Pant & Vista (2021).

* 1. Custom vs. Pretrained Architecture

An interesting finding of the study was how a custom was able to outperform a pretrained model. Intuitively, more complex models trained on higher resources should perform better than simple architectures. However, this was not the case.

A likely cause is how VGG 16 and 19, as well as other well-known pretrained models are trained on. These models are often trained on large datasets of real images (e.g. cats, dogs, cars). Hence, these models are not designed to capture textural differences present in the MalImg dataset.

Another reason would be the complexity of the models. As noted by Simonyan & Zisserman (2015), their models are deep layers, comprised of several blocks of convolution. This type of architecture performed well in classifying objects found in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) but might be too complex when optimizing performance for malware images. Nonetheless, the performance as of these models out of the box for a type of classification they were not initially designed for is noteworthy.

* 1. Custom vs. Baseline

In their published work, Pant & Vista (2021) utilized a more complex CNN architecture to classify malware images. Their model, as shown in *Image-based Malware Classification using Deep Convolutional Neural Network and Transfer Learning*, utilized three convolution blocks and five fully connected layers prior to the output layer. It was also explicitly mentioned that dropouts did not yield better accuracy during their experiments.

This architecture differs from the custom MalwareInSight architecture in two key aspects. First, the high number of fully connected layers led to overfitting based on the iterations done by the study. Second, a dropout layer, combined with given architecture, allowed for greater generalizability in the model. These changes allowed the custom model to outperform Pant & Vista’s (2021) by almost one percent in accuracy.

1. Conclusion

In conclusion, the study was able to develop a custom CNN architecture that outperforms models from recent literature and the original work of Nataraj et al. (2011) at 98.94% test accuracy. The study also demonstrated how deeper models, including pretrained models, do not necessarily result in better performance for malware image classification.

The study also explored potential reasons why data augmentation techniques are not applicable to this type of classification. Moreover, the findings indicate that the data wherein pretrained models VGG 16 and 19 were trained on might influence how these models performed on textured (non-object) images.

These results build a strong case to commercialize malware image classification. However, both the authors of this work and the original creator of the methodology Nataraj et al. (2011) believe this is not a replacement for signature-based detection. Rather, this is best integrated with current techniques to create more robust cyber defense systems.

1. Recommendations

Based on the findings of this study, the following recommendations are presented to enhance the performance, relevance, and feasibility for commercial deployment of future studies in the field of malware image classification:

* 1. Explore pretrained for Texture classification

The study revealed that custom-made architectures surpassed pre-trained models like VGG 16 and 19. However, this does not negate the potential of pre-trained models entirely. The authors suggest exploring pre-trained models specifically designed for texture classification tasks.

Malware images often exhibit unique textural patterns that differentiate them from benign files. Pre-trained models adept at recognizing textures such as Zhang et al.’s (2017) Deep TEN, could potentially be fine-tuned for superior performance in malware classification.

* 1. Include more malware & benign files

The study acknowledges the potential limitations of the MalImg dataset used. Given the dataset was created in 2011, more diverse and complex malware types have been created since. Expanding the dataset scope can significantly improve the generalizability and robustness of future models.

This includes incorporating a wider variety of malware families and including benign file types. Including more recent malware samples is crucial to stay ahead of hackers. By diversifying the dataset, future models can learn a more comprehensive representation of malicious and benign software, leading to more accurate classifications.

* 1. Unsupervised learning

The study primarily focused on supervised learning techniques, where labeled data is used to train the model. However, acquiring a vast amount of labeled malware data can be challenging and time-consuming.

As also suggested by Nataraj, et al (2011), exploring unsupervised learning techniques could open new avenues for research. Unsupervised learning such as clustering could allow models to identify patterns and relationships within unlabeled data. This could potentially be used to automatically cluster malware images based on inherent similarities, aiding in the identification of new malware variants.

References

AV-TEST-The Independent IT-Security Institute. (n.d.). AV-ATLAS - The Threat Intelligence Platform by AV-TEST. AV-ATLAS - The Threat Intelligence Platform by AV-TEST. Retrieved June 7, 2024, from https://av-atlas.org/

Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware analysis and classification: A survey. Journal of Information Security, 2014.

Moser, A., Kruegel, C., & Kirda, E. (2007, December). Limits of static analysis for malware detection. In Twenty-third annual computer security applications conference (ACSAC 2007) (pp. 421-430). IEEE.

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

Paardekooper, C., Noman, N., Chiong, R., & Varadharajan, V. (2022, July). Designing Deep Convolutional Neural Networks using a Genetic Algorithm for Image-based Malware Classification. In 2022 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Pant, D., & Bista, R. (2021). Image-based malware classification using deep convolutional neural network and transfer learning. 2021 3rd International Conference on Advanced Information Science and System (AISS 2021), 1–6.

Sausalito, C. (2023, December 13) Boardroom Cybersecurity Report 2023. Cybercrime Magazine. https://cybersecurityventures.com/cybersecurity-boardroom-report-2023/

Shijo, P. V., & Salim, A. J. P. C. S. (2015). Integrated static and dynamic analysis for malware detection. *Procedia Computer Science*, *46*, 804-811.

Torralba. (2003, October). Context-based vision system for place and object recognition. In Proceedings Ninth IEEE International Conference on Computer Vision (pp. 273-280). IEEE.

Zhang, H., Xue, J., & Dana, K. (2017). Deep ten: Texture encoding network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 708-717).

1. Appendix
   1. Malware count
   2. Image dimension graphs
   3. VGG16 & 19 architecture
   4. Confusion Matrix of MalwareInSight