RESEARCH

MalwareInSight: Classifying Malware Images Using Convolutional Neural Networks

Jason Catacutan*, Lorenzo de la Paz, Joshua San Juan and Kyle Uy

*Correspondence:
jcatacutan.msds2024@aim.edu
Full list of author information is available at the end of the article

Abstract

Traditional signature-based malware detection methods, while effective, are often outpaced by the rapid evolution of malware. This study builds on the innovative approach by Nataraj et al. (2011) to convert malware code into grayscale images, leveraging deep learning techniques to enhance malware classification. Utilizing the MalImg dataset, this research explores the efficacy of custom Convolutional Neural Network (CNN) architectures versus pre-trained models like VGG16 and VGG19.

The custom CNN, named MalwareInSight, achieved a superior 99.05% accuracy, surpassing previous benchmarks. The study highlights the limitations of data augmentation techniques and complex architectures for malware images. The findings advocate for the commercial viability of malware image classification as a complementary tool to existing methods. Recommendations include expanding datasets, integrating texture-based pre-trained models, and exploring unsupervised learning for future studies.

Keywords: malware; cyber defense; image classification; deep learning; Convolutional Neural Network

Highlights

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

1 Introduction

Our reliance on the internet has become a double-edged sword. While it fuels global connectivity and progress, it also creates fertile ground for malicious actors. Malicious software, or malware, has become a rampant threat, infiltrating devices and compromising security. As of 2024, a staggering 1.16 billion unique malware signatures lurk in cyberspace, according to AV-Atlas. This number is far from static - the first half of 2024 alone saw a staggering 50 million new malware signatures identified, highlighting the alarming rate of proliferation.

The financial repercussions of these attacks are equally concerning. Cyber defense firm Cybersecurity Ventures predicts that global cybercrime costs will skyrocket to a jaw-dropping $10.5 trillion by 2025

(Sausalito, 2023). These costs encompass a wide range of damages, including theft of intellectual property, sensitive information, and disruption of critical business operations

A significant breakthrough in malware detection occurred in 2011. Nataraj et al. (2011) developed a novel approach that converted malware code into grayscale images. This ingenious method opened a new avenue for tackling malware classification - by transforming it into an image classification problem. This study delves into this methodology, exploring the construction of a deep learning model capable of accurately classifying different malware types.

1.1 Problem Statement

Due to the rapid pace of malware development, automated malware analysis tools are crucial in distinguishing benign programs from malicious ones. Traditionally, commercially-available software relies on signature-based classification. This approach identifies unknown programs by comparing them to known malware signatures stored in a database.

Signatures act as unique identifiers for binary files, generated through static and dynamic analysis methods. Both methods have their merits and drawbacks, and researchers have continuously improved them over the years (Shijo & Salim, 2015).

However, in the constant arms race against malicious actors, exploring other ways to classify malware holds immense value. Classification of malware images provide a unique approach that utilizes visual similarity that does not necessitate the disassembly 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 dataset, MalImg, 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.

1.2 Objectives

This study aims to push the boundaries of malware classification by leveraging advanced deep learning techniques. The goal is not just to surpass the accuracy achieved by the pioneering work of Nataraj et al. (2011), but also to outperform more recent attempts that have utilized deep learning, particularly Convolutional Neural Networks (CNNs). By exceeding these benchmarks, the study aspires to make a significant contribution to the ongoing fight against malware.

Furthermore, the study delves into the potential of pre-trained models. These models, trained on massive datasets containing millions of data points, are already adept at recognizing complex patterns. The research investigates whether these "out-of-the-box" solutions can outperform custom-built deep learning architectures specifically designed for malware classification.

2 Related Works

2.1 Signature-based Detection

The relentless pace of malware creation exposes a critical weakness in signature-based detection: the constant need for database updates. This approach relies on comparing unknown programs to a library of known malware signatures. However, new malware strains emerge faster than updates can be deployed, leaving a window of vulnerability (Moser et al., 2007).

Static analysis works by extracting features, like patterns in the binary code, to create models for identifying malicious programs. This approach can glean valuable insights into a malware's potential behavior. However, its effectiveness is hampered by obfuscation techniques employed by virus coders. By deliberately obscuring the code, attackers can make detection significantly more challenging (Moser et al., 2007).

Dynamic analysis offers a potential countermeasure. It involves running suspicious programs in a controlled sandbox environment to observe their behavior. This approach boasts greater resilience against obfuscation tactics. However, the trade-off lies in the time-consuming nature of monitoring and the substantial resources required to create and maintain a secure sandbox environment (Shijo & Salim, 2015).

2.2 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 of pixels valued from 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.

2.3 VGG16 & 19

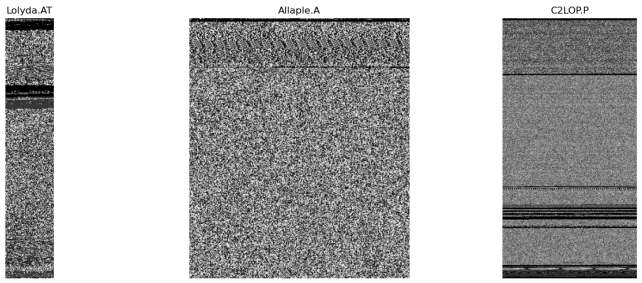
Multiple approaches have been used for malware detection and classification, many of which have been explored in literature (Gandotra et al., 2014). Several deep neural networks are applicable to image classification. However, Convolutional Neural Networks (CNN), inspired by the Kunihiko Fukushima Neocognitron, 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 pre-trained models – open-source models built on millions if not billions of datapoints. Among them, VGG16 and VGG19 by Simonyan & Zisserman (2015) have been two of the most impactful and used models today.

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

3 Data and Methods

Figure 1 Sample Malware Images. (a) Lolyda.AT (b) Allaple.A (c) C2LOP.P



3.1 Dataset

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 (Fig. 1), with varying lengths and widths (See Appendix Figure 3). The full list of malware classes can be seen in Appendix Table 3.

3.2 Exploratory Data Analysis

Upon observation, the MalImg dataset has class imbalance. As shown in Figure 2, select malware classes are better represented than others, 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.

3.3 Preprocessing

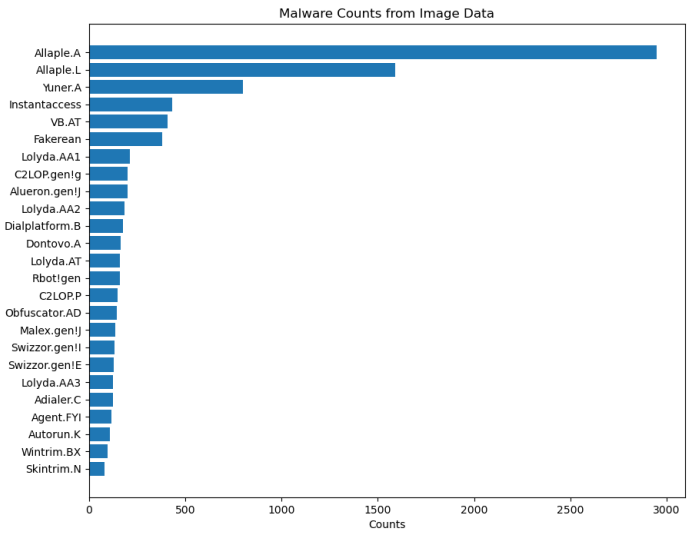
The images were resized to 256x256 pixels, rescaled with a factor of 1/255, with a batch size of 64. Training, validation, and test splits were done at 70-20-10 distribution respectively. Data augmentation techniques such as rotation, shear, zoom, dimensional shifts, and flips were tested, with fill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

3.4 Baseline

Proportional Chance Criterion(PCC)

The PCC was calculated to establish a baseline in which the performance of the created models would be tested. This score provides an

Figure 2 Malware Distribution. Shows the distribution of Malware Classes in the Dataset



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 by 1.25 yields 18.34%, suggesting this would be minimum score to beat for the model. However, considering the value to be too low, the paper will not be referencing this baseline when discussing the results.

Previous Work

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.

Modeling

The study implemented two pipelines in trying to build models that would beat the baseline scores. First would be utilizing pre-trained CNN-based models, specifically VGG16 and VGG19 (see Appendix Figures 4 and 5) 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.

Table 1 Model: MalwareInSight Custom CNN Model

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 256, 256, 3)	0
conv2d (Conv2D)	(None, 256, 256, 64)	1,792
max_pooling2d (MaxPooling2D)	(None, 85, 85, 64)	0
conv2d_1 (Conv2D)	(None, 85, 85, 32)	18,464
max_pooling2d_1 (MaxPooling2D)	(None, 28, 28, 32)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 512)	12,845,568
dense_1 (Dense)	(None, 64)	32,832
dropout (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 25)	1,625

Second was creating a custom CNN architecture. Initially inspired by the published architecture of Pant & Vista (2021) the architecture (see Table 1) 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 in 40 epochs, 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.

4 Results and Discussions

Table 2 Performance Metrics of Malware Classification Models

Model	Accuracy (%)	Precision (%)	Recall (%)
MalwareInSight Custom CNN	99.05	97.90	97.53
VGG16	98.20	95.65	95.46
VGG19	97.78	94.79	94.64
Pant & Bista (2021)	98.07	98	99
Paardekooper, et al. (2022)	98.5	-	-

4.1 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 2, it is apparent that the MalwareInSight (custom) model performed best in terms of accuracy with 99.05%, followed by 97.90% precision and 97.53% recall.

Looking the confusion matrix (Appendix Fig. 4), 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 pre-trained models, VGG16 and VGG19 also performed admirably at 98.20% and 97.78% accuracy scores respectively. It is worth

noting that these pre-trained 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.

4.2 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 1, 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).

4.3 Custom vs. Pre-trained Architecture

An interesting finding of the study was how a custom-designed model outperformed pre-trained models. Intuitively, more complex models trained with more resources should perform better than simple architectures. However, this was not the case.

A likely cause is where VGG16 and 19, as well as other well-known pre-trained models are trained on. Pre-trained models are typically trained on vast datasets of everyday objects like cats, dogs, and cars. This training focuses on recognizing general image patterns, making them less suited to identifying the specific textural variations within the MalImg dataset.

Another reason would be the architectural complexity of the pre-trained models. As noted by Simonyan & Zisserman (2015), their models have 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) which might be overly complex for optimizing performance on malware images. Nonetheless, the performance of these models out of the box, for a type of classification they were not initially designed for, is noteworthy.

4.4 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 one percent in accuracy.

5 Conclusions

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 99.05% test accuracy. The study also demonstrated how deeper models, including pre-trained 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 pre-trained models VGG16 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.

6 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 pre-trained Models for Texture Classification
The study revealed that custom-made architectures surpassed pre-trained models like VGG16 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.

2 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.

3 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.

7 References

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

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

3 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.

4 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).

5 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.

6 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.

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

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

9 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.

10 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).

Appendix

Table 3 Malware Distribution

Malware	Count
Allaple.A	2949
Allaple.L	1591
Yuner.A	800
Instantaccess	431
VB.AT	408
Fakerean	381
Lolyda.AA1	213
C2LOP.gen!g	200
Alueron.gen!J	198
Lolyda.AA2	184
Dialplatform.B	177
Dontovo.A	162
Lolyda.AT	159
Rbot!gen	158
C2LOP.P	146
Obfuscator.AD	142
Malex.gen!J	136
Swizzor.gen!I	132
Swizzor.gen!E	128
Lolyda.AA3	123
Adialer.C	122
Agent.FYI	116
Autorun.K	106
Wintrim.BX	97
Skintrim.N	80

Table 4 Model: VGG16

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, None, None, 3)	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1,792
block1_conv2 (Conv2D)	(None, None, None, 64)	36,928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73,856
block2_conv2 (Conv2D)	(None, None, None, 128)	147,584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295,168
block3_conv2 (Conv2D)	(None, None, None, 256)	590,080
block3_conv3 (Conv2D)	(None, None, None, 256)	590,080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1,180,160
block4_conv2 (Conv2D)	(None, None, None, 512)	2,359,808
block4_conv3 (Conv2D)	(None, None, None, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2,359,808
block5_conv2 (Conv2D)	(None, None, None, 512)	2,359,808
block5_conv3 (Conv2D)	(None, None, None, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

Table 5 Model: VGG19

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, None, None, 3)	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1,792
block1_conv2 (Conv2D)	(None, None, None, 64)	36,928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73,856
block2_conv2 (Conv2D)	(None, None, None, 128)	147,584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295,168
block3_conv2 (Conv2D)	(None, None, None, 256)	590,080
block3_conv3 (Conv2D)	(None, None, None, 256)	590,080
block3_conv4 (Conv2D)	(None, None, None, 256)	590,080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1,180,160
block4_conv2 (Conv2D)	(None, None, None, 512)	2,359,808
block4_conv3 (Conv2D)	(None, None, None, 512)	2,359,808
block4_conv4 (Conv2D)	(None, None, None, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2,359,808
block5_conv2 (Conv2D)	(None, None, None, 512)	2,359,808
block5_conv3 (Conv2D)	(None, None, None, 512)	2,359,808
block5_conv4 (Conv2D)	(None, None, None, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

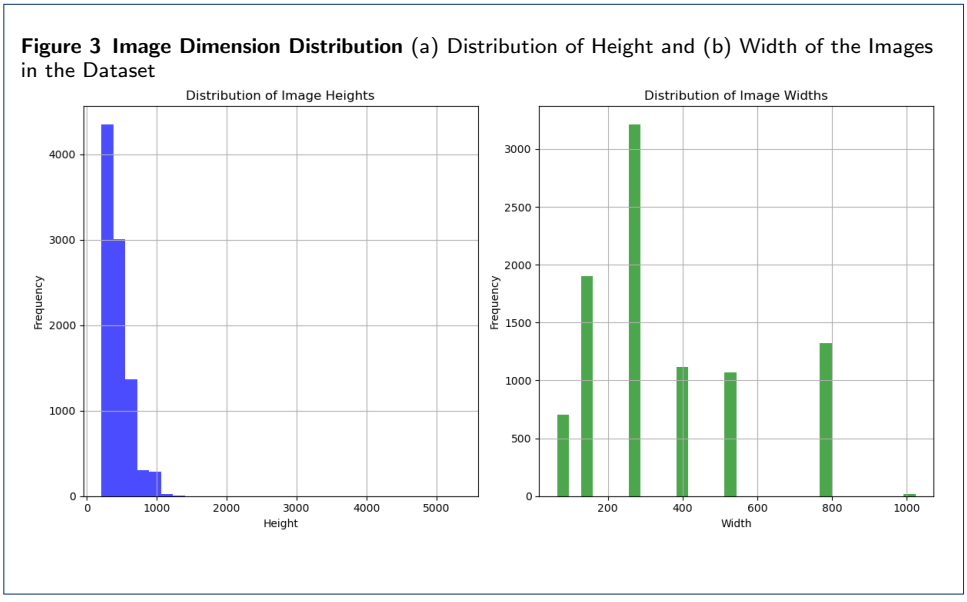


Figure 3 Image Dimension Distribution (a) Distribution of Height and (b) Width of the Images in the Dataset

Figure 4 Confusion Matrix Based on the Predictions of the MalwareInSight Custom CNN Model