

Spring 5-10-2018

Malware Image Classification using Machine Learning with Local Binary Pattern

Jhu-Sin Luo
Kennesaw State University

Dan Lo
Kennesaw State University

Follow this and additional works at: https://digitalcommons.kennesaw.edu/cs_etd



Part of the [Information Security Commons](#)

Recommended Citation

Luo, Jhu-Sin and Lo, Dan, "Malware Image Classification using Machine Learning with Local Binary Pattern" (2018). *Master of Science in Computer Science Theses*. 16.

https://digitalcommons.kennesaw.edu/cs_etd/16

This Thesis is brought to you for free and open access by the Department of Computer Science at DigitalCommons@Kennesaw State University. It has been accepted for inclusion in Master of Science in Computer Science Theses by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact digitalcommons@kennesaw.edu.

Malware Image Classification using Machine Learning with Local Binary Pattern

A Thesis Presented to

The Faculty of the Computer Science Department

by

Jhu-Sin Luo

In Partial Fulfillment

of Requirements for the Degree

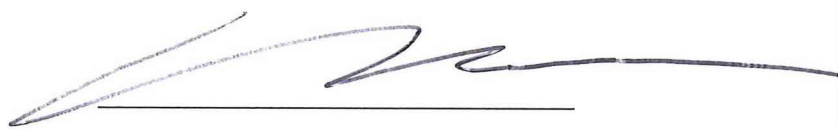
Master of Science, Computer Science

Kennesaw State University

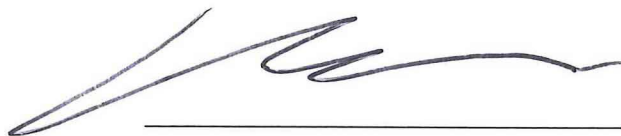
May 2018

Malware Image Classification using Machine Learning with Local Binary Pattern

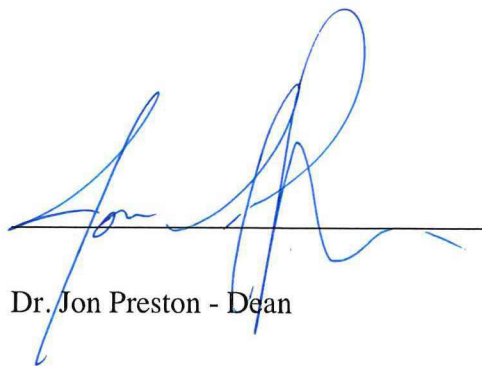
Approved:

A handwritten signature in black ink, consisting of a series of fluid, connected strokes, positioned above a horizontal line.

Dr. Dan Chia-Tien Lo - Advisor

A handwritten signature in black ink, similar to the one above, positioned above a horizontal line.

Dr. Dan Chia-Tien Lo - Department Chair

A handwritten signature in blue ink, featuring a large, prominent loop, positioned above a horizontal line.

Dr. Jon Preston - Dean

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Kennesaw State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of, this thesis which involves potential financial gain will not be allowed without written permission.

Jhu-Sin Luo

Notice To Borrowers

Unpublished theses deposited in the Library of Kennesaw State University must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this thesis is:

Jhu-Sin
Luo

1100 S Marietta PKWY,
Marietta, GA 30060

The director of this thesis is:

Dr. Dan Chia-Tien Lo

1100 S Marietta PKWY,
Marietta, GA 30060

Users of this thesis not regularly enrolled as students at Kennesaw State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their patrons are required to see that each user records here the information requested.

Malware Image Classification using Machine Learning with Local Binary Pattern

An Abstract of
A Thesis Presented to

The Faculty of the Computer Science Department

by

Jhu-Sin Luo
Bachelor of Science, National Kaohsiung First University of Science and Technology,
2014

In Partial Fulfillment
of Requirements for the Degree
Master of Science, Computer Science

Kennesaw State University

May 2018

Abstract

Malware classification is a critical part of the cybersecurity. Traditional methodologies for the malware classification typically use static analysis and dynamic analysis to identify malware. In this paper, a malware classification methodology based on its binary image and extracting local binary pattern (LBP) features is proposed. First, malware images are reorganized into 3 by 3 grids which are mainly used to extract LBP feature. Second, the LBP is implemented on the malware images to extract features in that it is useful in pattern or texture classification. Finally, Tensorflow, a library for machine learning, is applied to classify malware images with the LBP feature. Performance comparison results among different classifiers with different image descriptors such as GIST, a spatial envelope, and the LBP demonstrate that our proposed approach outperforms others.

Malware Image Classification using Machine Learning with Local Binary Pattern

A Thesis Presented to
The Faculty of the Computer Science Department

by

Jhu-Sin Luo

In Partial Fulfillment
of Requirements for the Degree
Master of Science, Computer Science

Advisor: Dr. Dan Chia-Tien Lo

Kennesaw State University

May 2018

ACKNOWLEDGEMENTS

First, I would like to thank God for giving me knowledge, strength and support all I needs. Second, I would like to express my very great appreciation to my advisor, Dr. Dan Lo, for his guidance, support and encouragement throughout this entire research. Third, I would like to express my gratitude to my parents and brother who always encourage and support me continually. Finally, this material is based in part upon work supported by the National Science Foundation under Grant Numbers 1623724, 1438858, 1244697, and 1241651. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

I. Introduction	14
1.1 Malware Classification and Detection Methodologies.....	15
1.2 Proposed Approach Overview.....	15
II. Related Works	17
2.1 Malware Classification.....	17
2.2 Malware Detection.....	20
III. Research Methodology	22
3.1 Dataset.....	22
3.2 Image Visualization.....	25
3.2.1 Convert to Grayscale.....	25
3.2.2 Convert to RGBA Color Space.....	25
3.3 Image Reorganization.....	26
3.4 Local Binary Pattern.....	27
3.5 GIST Descriptor.....	27
3.6 Convolutional Neural Network.....	28
3.6.1 Convolutional Layer.....	28
3.6.2 Pooling Layer.....	29
3.6.3 Fully Connected Layer.....	30

3.6.4 Rectified Linear Unit	31
3.6.5 CNN Architecture.....	31
3.7 Support Vector Machine.....	32
3.7.1 One-versus-Rest.....	33
3.7.2 One-versus-One.....	34
3.8 K-Nearest Neighbor.....	34
IV. EXPERIMENT, Result and Comparisons.....	36
4.1 Model Selection.....	36
4.2 First Dataset Experimental Results.....	37
4.3 Second Dataset Experiment Results.....	39
4.4 Pros and Cons.....	40
V. CONCLUSION AND FUTURE WORK.....	41
5.1 Future Work.....	41
5.2 Conclusion.....	41
REFERENCE.....	43

LIST OF TABLES

Table 1. Comparison of relative malware classification researches.....	19
Table 2. Comparison of relative malware detection researches.....	21
Table 3. Samples of Malware Image.....	22
Table 4. Malware Family.....	23
Table 5. Distribution of Second Malware Dataset.....	24
Table 6. Experiment Result of First Malware Dataset.....	37
Table 7. Confusion Matrix of CNN using LBP feature.....	38
Table 8. Confusion Matrix of KNN using LBP feature.....	38
Table 9. Confusion Matrix of SVM using LBP feature.....	38
Table 10. Experiment Result of Second Malware Dataset.....	39

LIST OF FIGURES

Figure 1. Statistic of total malware over past decade.....	14
Figure 2. Overview of Entire System.....	16
Figure 3. Hexadecimal Sample of Second Malware Dataset.....	24
Figure 4 Malware convert to Grayscale Image.....	25
Figure 5. Malware convert to RGBA Color Image.....	25
Figure 6. Reorganize Malware Image.....	26
Figure 7. Local Binary Pattern Operator.....	27
Figure 8. Convolution.....	28
Figure 9. Padding.....	29
Figure 10. Pooling.....	30
Figure 11. Fully Connected Layer.....	30
Figure 12. Rectified Linear Unit (ReLU)	31
Figure 13. CNN Architecture.....	32
Figure 14. Support Vector Machine.....	32
Figure 15. One-versus-Rest.....	33
Figure 16. One-versus-One.....	34
Figure 17. KNN Example/.....	35
Figure 18. Accuracy on Different CNN Models.....	36

Figure 19. Average Execution Time.....40

CHAPTER I

INTRODUCTION

Over the past few years, the Internet usage had experienced an exponential growth. It has become an important part of our daily lives. The cybersecurity is also playing a role in that the online financial activities such as the online payment and online money transaction become widespread [1]. The users of the Internet face threats from the malware which causes detriment to users of computer and the Internet. AV-TEST, an IT security Institute, registers over 583 million the malware in 2017[2] and based on their reports, the amount of the malware dramatically increases every year (Figure 1).

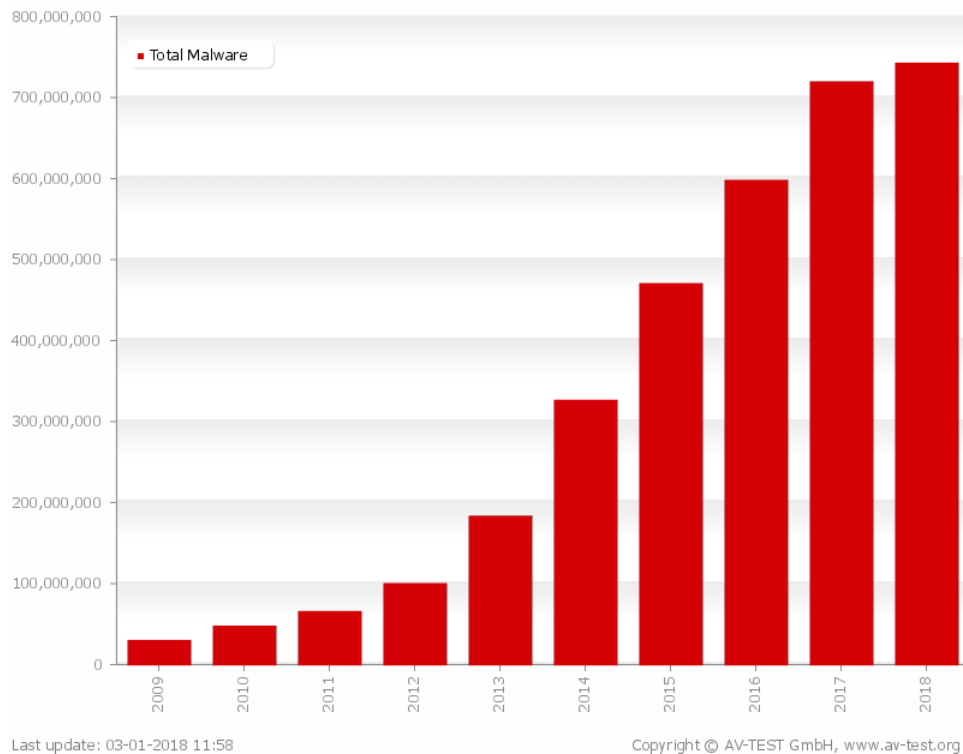


Figure 1. Statistic of total malware over past decade

1.1 Malware Classification and Detection Methodologies

Traditional methodologies for the malware [38] classification or detection mainly use static analysis and dynamic analysis to identify type of the malware and behavior of the malware. Both methodologies have their advantages and disadvantages.

Static analysis examines the executable file without actually executing. It extracts the binary code or disassemble instruction from the file to generate the patterns or features which could be used to identify whether the file is the malware or not. The advantages of static method are that binary code usually includes information about the malicious behavior and less resource intensive. The static analysis is ineffective against different code obfuscation and packing technique [3].

On the other hand, dynamic analysis verifies the file by executing on the secure environment or virtual environment. By executing file, the behaviors of the malware are able to observe. Its advantages are that it can against code obfuscation and packing. Nonetheless, dynamic analysis still exists disadvantages. The malware might have different behaviors in two different environments or some behaviors may need to be triggered on specific circumstances.

1.2 Proposed Approach Overview

Recently the deep machine learning is widely used and also obtains outperformed result in image classification [43, 46, 47]. Therefore, in this paper, a malware classification approach based on image processing and convolutional neural network is

proposed. First, as Figure 1 demonstrates, each pixel in the malware images are reorganized. The pixels of the original malware images are constructed by line by line. We rearrange each pixel of images by 3 by 3 grids. Second, the LBP is applied on the malware image to extract features. Finally, malware images are classified by TensorFlow and the result would be compared with other classifiers.

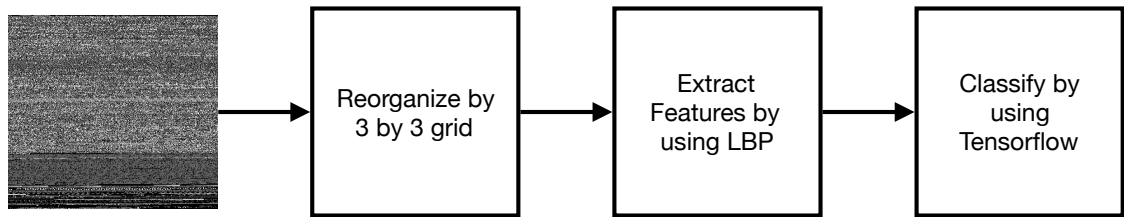


Figure 2. Overview of Entire System

CHAPTER II

RELATED WORK

This chapter mainly separate into two parts: Malware Classification and Malware Detection. We would summarize the previous works and researches regarding this two domains, which included the methodologies, dataset and approaches of feature extraction.

2.1 Malware Classification

In [1], L.Nataraj, S.Karthikeyan, G. Jacob and B. S. Manjunath visualize malware dataset which consist of 25 malware families and 9458 malware into grayscale. First they read each malware in binary and read as a vector of 8 bit unsigned integers. Each vector would be organized into a two-dimension array in the range between 0 and 255, which would be one grayscale image. They applied GIST descriptor on the malware images. The GIST descriptor is a computational model of the recognition of real world scene [9]. After obtaining the GIST images, the K-nearest neighbor is utilized to classify malware images, got an 97.18% accuracy over 25 families.

In [5], Aziz Makandar and Anita Patrot they mainly classify malware images, 3131 malware images over 24 malware types, based on applying Discrete Wavelet Transformation (DWT) to extract features. Their proposed methodology consists of three

phrases, pre-processing, feature extraction and classification. In pre-processing phrase, grayscale malware images are normalized into 256x256 by applying wavelet to de-noise. In feature extraction phrase, DWT is utilized to decompose malware images into four level. In classification phrase. they used Support Vector Machine (SVM) to discriminating the malware classes with static features which are extracted from level 4 decomposition of DWT and SVM gives 92.52% accuracy for 24 malware types.

In [14], Aziz Makandar and Anita Patrot they convert malware binary into grayscale and resize into 64x64. They also obtain the global features of the malware images by using gabor wavelet transform and GIST. This experiment is implemented on Mahenur dataset which include 3131 binary sample comprising 24 malware families. Finally, feed forward Artificial neural network (ANN) is used to train and classify malware images with 96.35% accuracy.

In [23], Seonhee Soek and Howon Kim, they build the convolutional neural network with three layers to classify malware. They examine their method on two dataset. One is Microsoft malware dataset which is consisted of 21741 samples for 9 malware families. Second dataset is VXHeaven which is consisted of 27 malware families. Those two dataset are feed into CNN model and get 96.2% and 82.9% respectively.

In [28], A. Makandar and A. Patrot, they proposed the multiclass malware classification from image processing perspective. They use Gabor wavelet, GIST and discrete wavelet transform to build effective texture feature vector. The reason they use

wavelet transform is that it reduce the dimension of feature vector and also reduce the complexity. Their proposed approach experiment on Mailing dataset which has 12470 samples. They randomly select 1610 training data and 1710 testing data from 8 malware families. Finally, the SVM gives 98.88% accuracy and KNN gives 98.84% accuracy.

In [32], B. N. Narayanan, O. Djaneye-Boundjou and T. M. Kebede, they visualize malware into image as they capture minor changes while retaining a global structure. Second, the feature is extracted by using Principle Component Analysis. Based on the PCA, they study the performance on different classifiers such as Artificial Neural Network, K-Nearest Neighbor, and Support Vector Machine to identify malware image into their corresponding classes. Finally, the KNN give the 96.6% accuracy over 10868 samples from 9 malware families.

Reference	Year	#Dataset	#Malware Families	Features	Classifiers	Accuracy
[1]	2011	9458	25	GIST	KNN	97.18%
[5]	2017	3131	24	DWT	SVM	92.52%
[14]	2015	3131	24	GIST	ANN	96.35%
[23]	2016	21741	9	N/A	CNN	96.2%
[23]	2016	N/A	27	N/A	CNN	86.9%
[28]	2017	12470	25	Gabor GIST DWT	SVM	98.88%
[28]	2017	12470	25	Gabor GIST DWT	KNN	98.84%
[32]	2016	10868	9	PCA	KNN	96.6%

Table 1. Comparison of relative malware classification researches

2.1 Malware Detection

In [24], S. Choi, S. Jang, Y. Kim and J. Kim, they build deep learning model which has three convolutional layers followed by a pooling layers respectively and two fully connected layers. The dataset they used is consisted of 2000 malware and 10000 normal files. They convert the files into grayscale and feed into the model without extracting feature. Finally, they get 95.66% accuracy.

In [25], K. Kancharla and S. Mukkamala, first, the executable is converted into grayscale image which they call byteplot. Their dataset is consisted of 25000 malware and 12000 benign. Second, they extract features using intensity, wavelet and Gabor. Finally, in this work they use Support Vector Machine and obtain 95.95% accuracy using combined feature set.

In [31], X. Zhou, J. Pang and G. Liang, they visualize malware into grayscale and extract image feature by using Gabor filter. Their dataset is consisted of 15781 samples, which includes 8759 malware and 7022 benign. The approach they proposed is used Extremely randomized tree with 10-fold cross validation as their classifier and they also study the performance of various classifier such as Gradient Boost Decision Tree, K-Nearest Random Forest. Extremely randomized tree is applied for detection and give 97.51% accuracy.

Reference	Year	#Malware	#Benign	Features	Classifiers	Accuracy
[24]	2017	2000	10000	N/A	CNN	95.66%
[25]	2013	25000	12000	Wavelet Gabor	SVM	95.95%
[31]	2017	8759	7022	Gabor	ET	97.51%

Table 2. Comparison of relative malware detection researches

CHAPTER III

RESEARCH METHODOLOGY

In this chapter, we demonstrate our approach step by step. First, we illustrate malware visualization and reorganization. Second, we introduce the LBP and how to apply the LBP on our images. Finally, we build convolutional neural network architecture which we utilize to train and classify. In addition, we also feed our data to different classifier such as Support Vector Machine and K-nearest neighbor.

3.1 Dataset

In this research, we use two malware datasets. The first dataset we use is provided by [1, 4]. This dataset consists of 32 families and around 12000 malware images with grayscale (table 3). The types of malwares mainly belong to trojan, password stealer and virus. The training dataset which consists of 80% of each malware family in dataset for training and the testing dataset consists of 20% of each malware family. Malware image samples display in following (Table 2).




		
Agent.FYI	Swizzor.gen	Lolyda.AA1

Table 3. Samples of Malware Image

Malware Family	Type of malware	Amount of malware
Adialer.C.UPX	Adialer	188
Agent.FYI	Backdoor	116
Aliser.7825	Trojan	256
Allaple.A	Worm	4540
Alueron_Gen_J	Trojan	198
Autorun.A	Worm	106
Azero.A	Trojan	121
Backdoor.Agent.AsPack	Backdoor	180
C2Lop	Trojan	692
Dialplatform.B	Dialer	177
Dontovo.A	TrojanDownloader	162
Fakerean	Rogue	381
Farfli.l	Backdoor	94
Instantaccess	Dialer	431
Lolyda.AA1	PasswordSteeler	213
Lolyda.AA2	PasswordSteeler	184
Lolyda.AA3	PasswordSteeler	123
Lolyda.AT	PasswordSteeler	159
Luder.B	Virus	509
Malex.gen!J	Trojan	136
Nuwar.A	Virus	51
Obfuscator.AD	TrojanDownloader	142
Rbot.gen	Backdoor	158
Sality.AM	Virus	127
Skintrim.N	Trojan	80
Swizzor.gen	TrojanDownloader	520
VB.AT	Worm	408
Virut.A	Virus	133
Virut.AC	Virus	269
Virut.AK	Virus	571
Wintrim.BX	TrojanDownloader	97
Yuner.A	Worm	800

Table 4. Distribution of First Malware Dataset

The second malware dataset we use is provide by Kaggle for Microsoft malware Classification Challenge [37]. This dataset consists of two sets: training dataset and testing dataset. Each raw data contains a hexadecimal representation of the file’s binary content and a corresponding assembly file which contains information extracted from the binary. In our research, we would only use hexadecimal file as input. The training dataset consisted of 10868 labeled sample for 9 categories. Table 5 demonstrates the distribution of each malware category. The testing dataset consisted of 10873 samples. Nonetheless, the label of testing data is not publicly available. Therefore, we would use training dataset in our research.

Malware Family	Number of Malware
Ramnit	1541
Lollipop	2478
Kelihos_ver3	2942
Vundo	475
Simda	42
Tracur	751
Kelihos ver1	398
Obfuscator.ACY	1228
Gatak	1013

Table 5. Distribution of Second Malware Dataset

```

00401000 56 8D 44 24 08 50 8B F1 E8 1C 1B 00 00 C7 06 08
00401010 BB 42 00 8B C6 5E C2 04 00 CC CC CC CC CC CC CC
00401020 C7 01 08 BB 42 00 E9 26 1C 00 00 CC CC CC CC CC
00401030 56 8B F1 C7 06 08 BB 42 00 E8 13 1C 00 00 F6 44
00401040 24 08 01 74 09 56 E8 6C 1E 00 00 83 C4 04 8B C6
00401050 5E C2 04 00 CC CC CC CC CC CC CC CC CC CC CC
00401060 8B 44 24 08 8A 08 8B 54 24 04 88 0A C3 CC CC CC
00401070 8B 44 24 04 8D 50 01 8A 08 40 84 C9 75 F9 2B C2

```

Figure 3. Hexadecimal Sample of Second Malware Dataset

3.2 Image Visualization

3.2.1 Convert to Grayscale

In [1], L.Nataraj, S.Karthikeyan, G. Jacob and B. S. Manjunath visualized the malware into grayscale image in the range [0, 255]. The width of image is fixed and the height is allowed to vary. In [14], Aziz Makandar and Anita Patrot also convert malware into grayscale in the range [0, 255]. In [5], the malware is also visualized into grayscale image and normalized into 256*256 dimension.

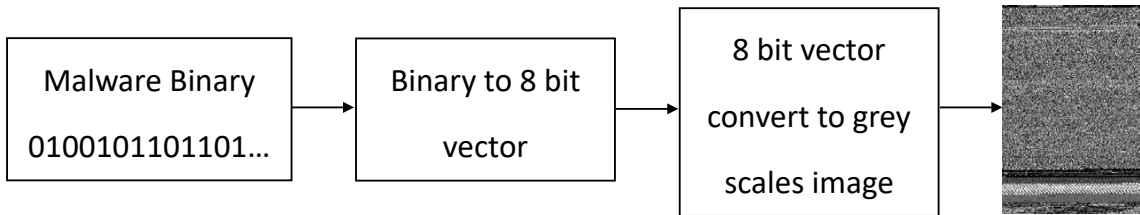


Figure 4. Malware convert to Grayscale Image

3.2.2 Convert to RGBA Color Space

The reason we convert malware to RGBA [40] color space is that RGBA can be represented as hexadecimal (#00ff0080) and the x86 instructions usually are longer than 8-bit binary. Therefore, if we convert more than 8-bit binary to one pixel, it can retain the relationship between instruction and pixel. This approach mainly focuses on second dataset in that second dataset is presented in hexadecimal. Each 8-bit value in hexadecimal would be as a pixel.



Figure 5. Malware convert to RGBA Image

3.3 Image Reorganize

Our methodology is that we reorganized the grayscale malware images which are provided by [1, 4] L.Nataraj, S.Karthikeyan, G. Jacob and B. S. Manjunath. They convert the malware into images with grayscale. The malware images with grayscale are obtained by reading malware in binary. A Malware binary is read as a vector of 8 bit unsigned integers and then arranged into 2D array (Figure 4). We rearrange each pixel in the malware images into 3 by 3 grid (Figure 6). We convert malware images into 3 by 3 grid in that it is suitable for extracting Local binary pattern descriptor.

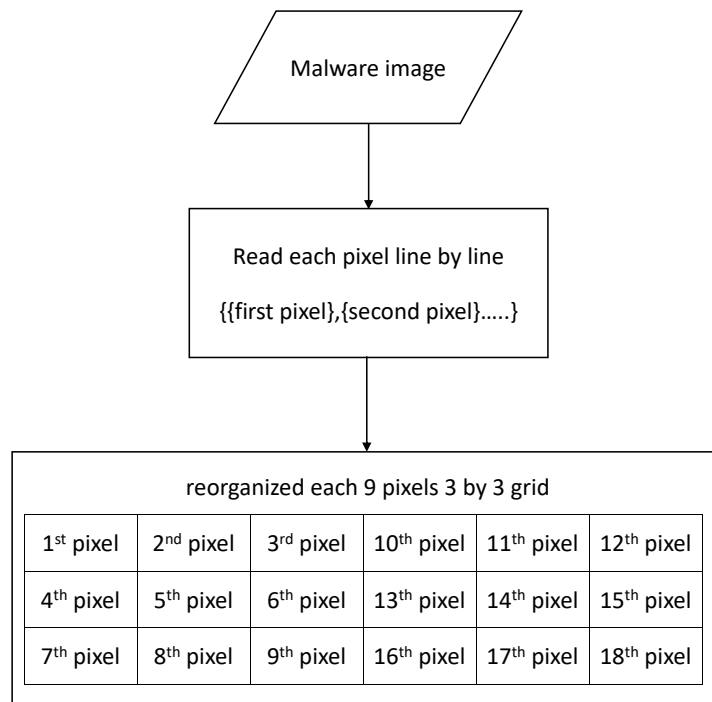


Figure 6. Reorganize Malware Image

3.4 Local Binary Pattern

Local Binary Pattern (LBP), a visual descriptor, is useful for texture analysis and texture classification [6, 7, 8, 12]. As Figure 7 demonstrates, the value of central pixel is

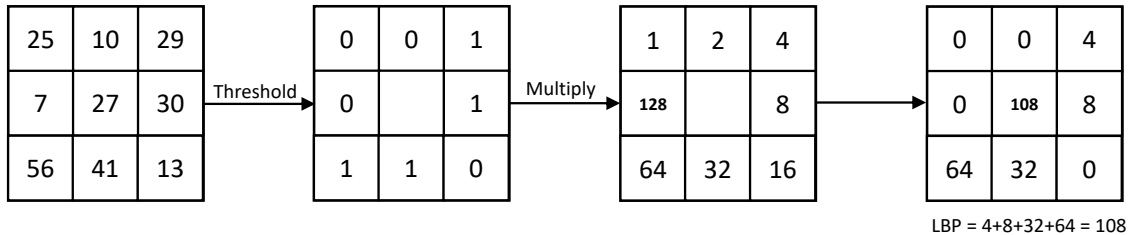


Figure 7. Local Binary Pattern Operator

threshold. The 8 neighbors around a pixel are compared with the central pixel. If a neighbor's value is greater than central pixel, the value of the neighbor is written '1'. The value of neighbor which less then threshold is written '0'. The threshold results are multiplied with weights which are given by power of two. The central value is the sum of the multiplying results. For each pixel in the image do the same process. The final LBP descriptor can be obtained by calculating the histogram of the image.

3.5 GIST Descriptor

We also use another image descriptor to extract feature from images. The GIST descriptor [10, 11, 16, 17] is originally used to compute the global feature vector and recognize real world scenes, which provide the holistic representation of an image. Given an image is computed by convolving the image with 32 Gabor filters at 8 orientations and 4 scales, which generate 32 feature maps of the same resolution as the given image. Each feature map is divide into 16 regions by 4x4 grid, and the average value of each region would be calculated. In the end, we obtain a 512-dimension GIST

descriptor by concatenating the 16 averaged value for each region. Therefore, the GIST provide a descriptor by summarizing the gradient information for each part of the image.

3.6 Convolutional Neural Network

We build the convolutional neural network (CNN) using TensorFlow [18, 19, 20] for training our data and classification. CNN is a mathematical model to solve optimization problem, which is comprised of one or more layers. Each layer is consisted of neurons. Each neuron would take an input and multiply weight and add bias on input. In order to obtain output, the input would be put in a non-linear function, activation function.

3.6.1 Convolutional Layer

Convolution is a mathematical operation, which is used to find the pattern in inputs or filter out the features. For example, we have an 5x5 input image and the filter size is 3x3. We pick the 3x3 sized chunk from the input image and do the convolution (dot product) with the filter as shown in Figure 8. In this example, each time we move the filter 1 pixel, this number is called stride.

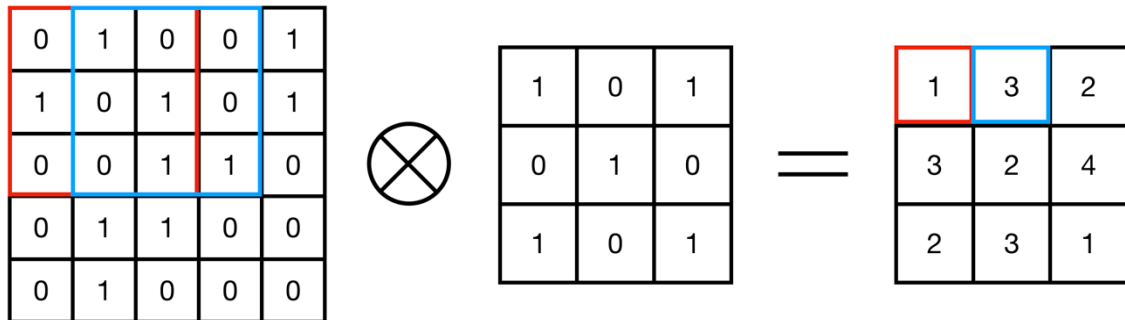


Figure 8. Convolution

As Figure 8 shows, we can observe that the dimension of input image decrease after applying convolution on it. If we keep applying convolution on the input image, the dimension of input image would decrease faster than we want. In order to preserve as much as information, we can add zeros on boundary of the input image after convolution operator so that we can maintain the dimension would be as same as origin. This process is called padding (Figure 9).

0	0	0	0	0
0	1	3	2	0
0	3	2	4	0
0	2	3	1	0
0	0	0	0	0

Figure 9. Padding

3.6.2 Pooling Layer

Pooling layer typically would be used after convolutional layer, which is an approach for decreasing the dimension while preserve the information. Max pooling is the most popular form of pooling. For example, we have a pooling filter with size 2x2 and stride is as same as width. When we apply this filter on the input image, each 2x2 sized chunk from the image would output the maximum value of that 2x2 area. In addition, applying

pooling not only preserve the information but also reduce the computation and avoid overfitting [48].

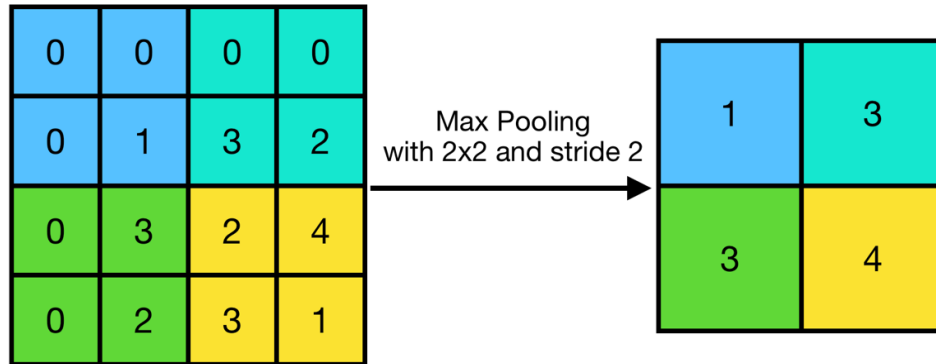


Figure 10. Pooling

3.6.3 Fully Connected Layer

Fully connected layer would receive all the input from neurons of previous layers and the output is value of certain predicted class. The value of output would do the matrix multiplication with weights and bias.

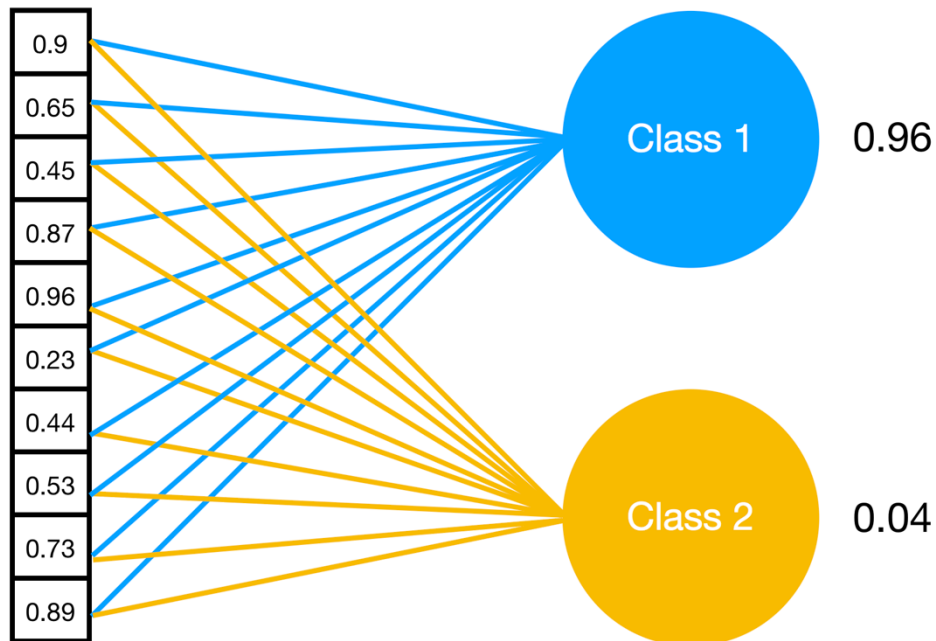


Figure 11. Fully Connected Layer

3.6.4 Rectified Linear Unit

The rectified linear unit (ReLU) [45] is the most common activation function in neural network. Activation function, simply put, is used to calculate weights and bias function. The ReLU function would return 0 if input is negative. On the other hand, this function will return the value back if input is positive.

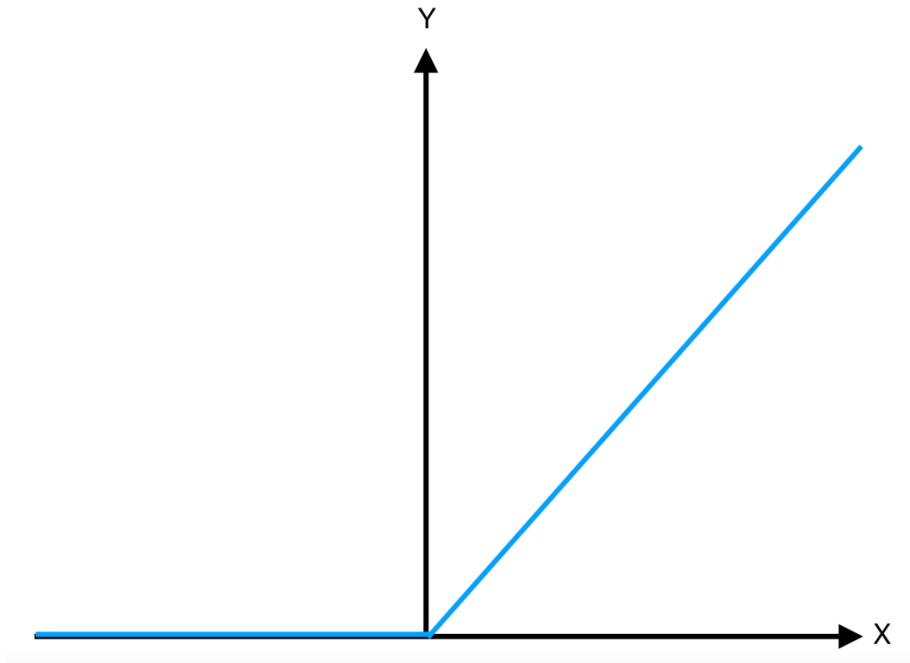


Figure 12. Rectified Linear Unit (ReLU)

3.6.5 CNN Architecture

We use TensorFlow [13, 15] to build CNN for training and testing. We build three CNN models with different number of layers, which have 5, 8, 11 layers respectively in that it is still an issue to determine number of layers we should build. Thereby, we alternatively build several models to evaluate the performance. Figure 13 shows one of our model which have 5 layers. There are five layer in our architecture. The first three layers consist of convolutional layer, max pooling layer and ReLU activation function. The last two layer are flatten layer and fully connected layer respectively. The second

CNN model possesses 6 convolutional layers, 1 flatten layer, and 2 fully connected layer. Last our CNN model is equipped with 9 convolutional layers, 1 flatten layer, and 2 fully connected layer.

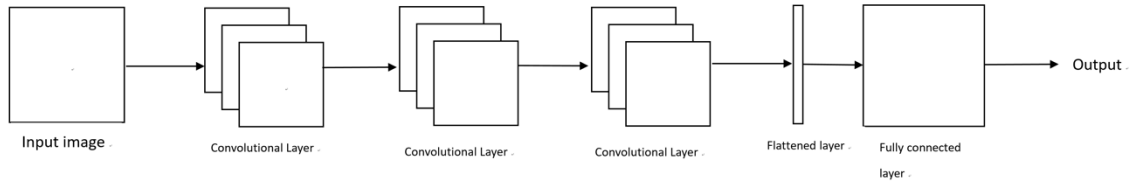


Figure 13. CNN Architecture

3.7 Support Vector Machine

Support Vector Machine (SVM) [21, 22, 29] is a supervised classifier in machine learning, which is used for classification and regression analysis. SVM would project the data into high-dimension space, find the most optimal Hyperplane, and to separate two of the classes.

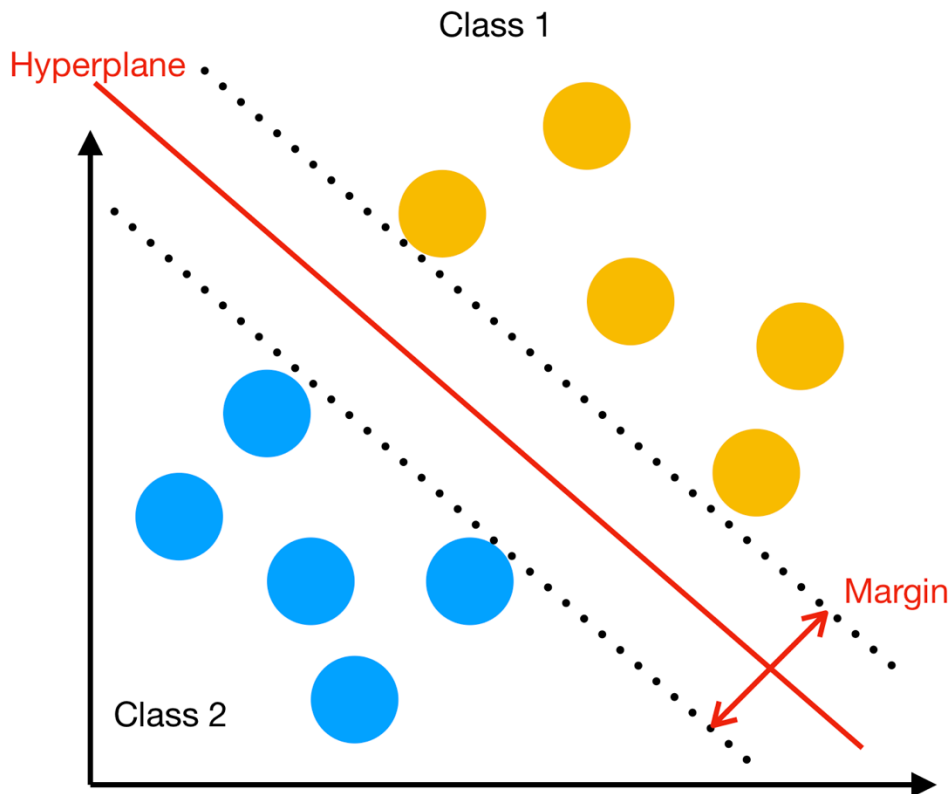


Figure 14. Support Vector Machine

As Figure 14 demonstrates, SVM wishes to find the hyperplane [41] between Class 1 and Class 2 with equidistant margin as far as possible for both side so that we can identify data into corresponding class clearly.

Basically SVM is a binary classifier. Nonetheless, in real situation, the number of class is larger than two. For example, there have 32 classes in our dataset. Therefore, there are two strategies [44] which could make SVM deal with multiclass issues.

3.7.1 One-versus-Rest

We assume that there are K classes where K is a constant and larger than two. Thereby, we can treat one of class in K as class A, and rest classes in K as class B so that we can classify the data to class A through K SVM classifiers.

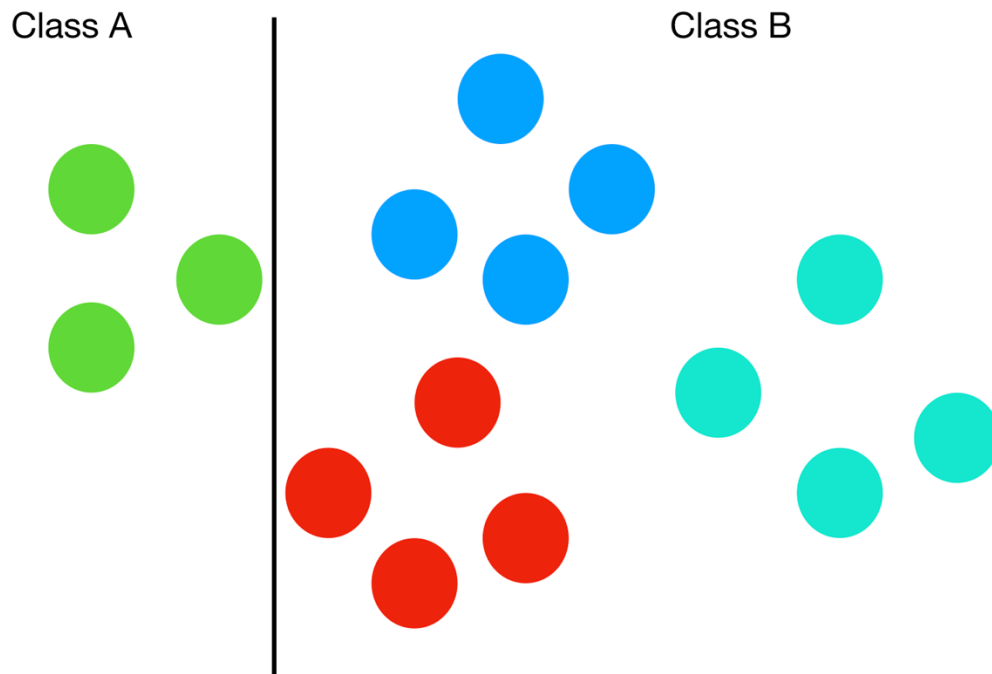


Figure 15. One-versus-Rest

3.7.2 One-versus-One

We can use the concept of binary tree. We train separate SVM classifiers for each pair of classes. In total, there would be $K(K-1)/2$ SVM classifier. We classify given data from the bottom of the tree. The top of the tree is the classification result.

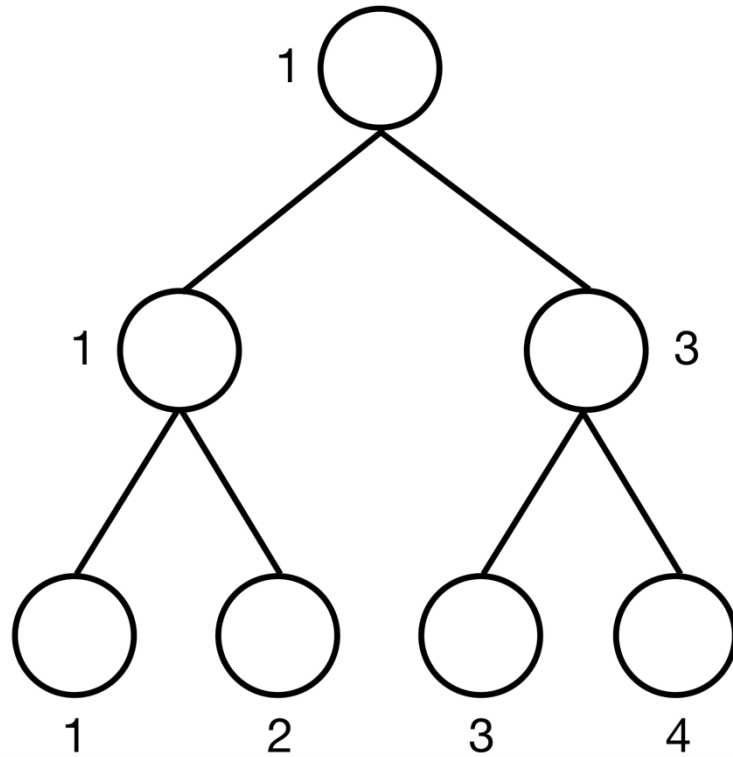


Figure 16. One-versus-One

3.8 K-Nearest Neighbor

K-Nearest Neighbor (KNN) [33, 34, 35] is the supervised learning and also a non-parametric learning algorithm, which is used for classification and regression analysis. The KNN algorithm calculate the distance between testing data and set of training data. The most common class between testing data's k nearest neighbor around it would be assigned to testing data. In our research, we use Euclidean to measure the distance. The

Euclidean distance [36] between two points p and q is the length of the line segment connecting them.

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Figure 17 demonstrates the KNN algorithm for two classes. The center white point is testing data. The inner circle includes 3 nearest neighbors for the testing data. The majority of inner circle is class 2. Therefore, the testing data would be assigned to class 2. The outer circle contains 5 nearest neighbor for the testing data. The majority of neighbors in outer circle is class 1. Thereby, the testing data would be classified as class 1.

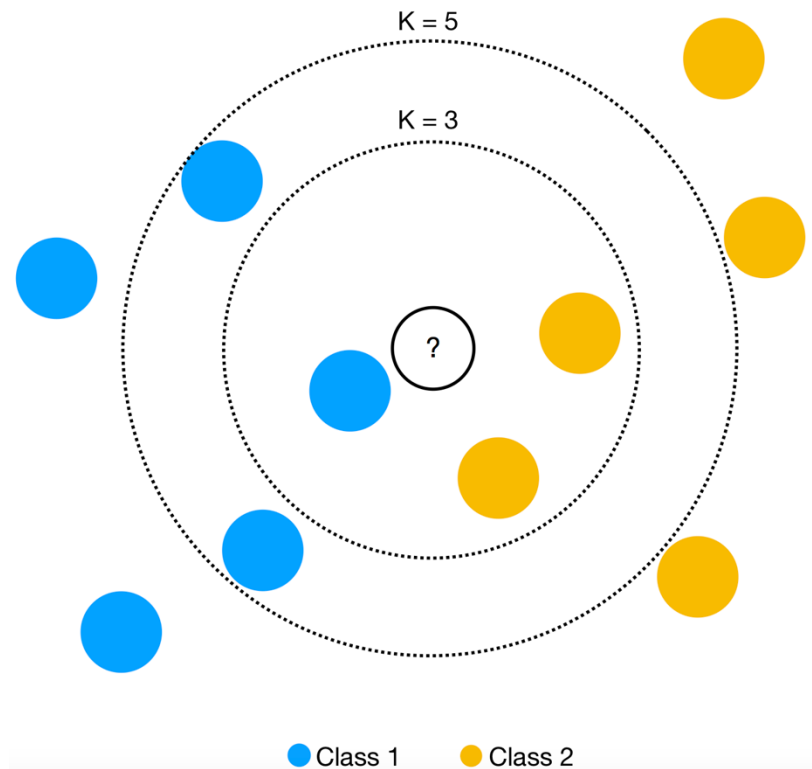


Figure 17. KNN Example

CHAPTER IV

Experiment, Results and Comparisons

4.1 Model Selection

We feed the training images to three different CNN model with several epochs after applying LBP, and classify the test data to evaluate which model is fit our dataset. As Figure 18 indicates, all of our model can reach 90% accuracy after 60 epochs, and the CNN model which is equipped 6 layers have better results than others which have 94% accuracy. Therefore, we would use CNN model with 6 layers to compare malware dataset with different image descriptors and different classifiers.

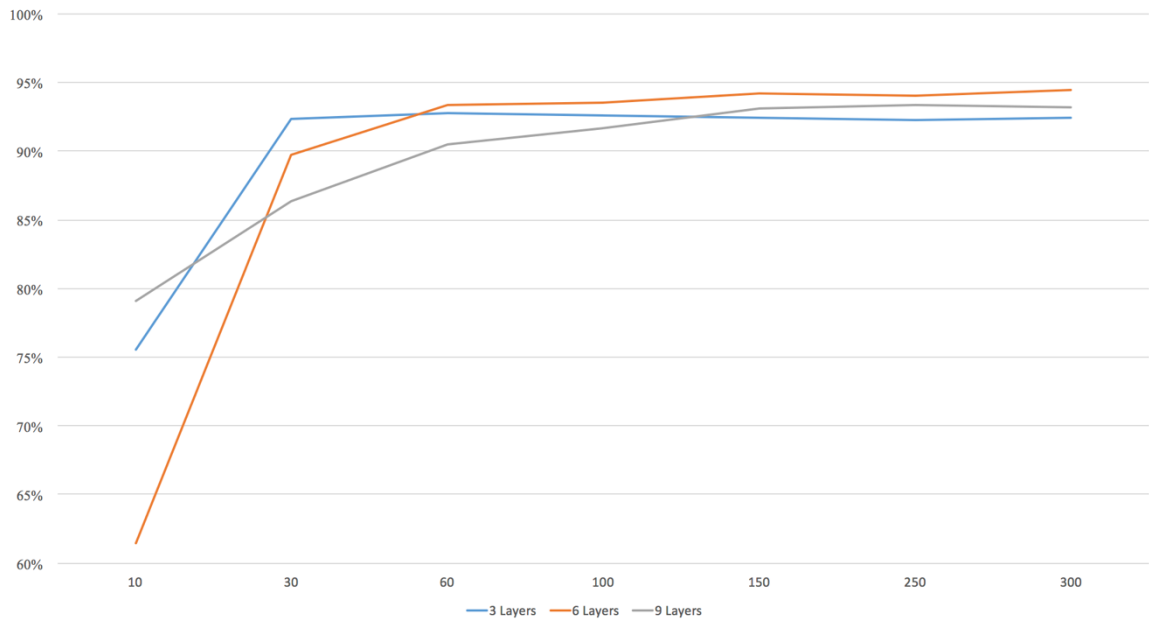


Figure 18. CNN Architecture

4.2 First Dataset Experimental Results

We evaluate CNN for the LBP features classification, and use LBP features for training Support Vector Machine (SVM) classifier and k-nearest neighbor (KNN) classifier. We also implement GIST [9, 10, 11] features with CNN, KNN and SVM. Table 6 displays the accuracy of different methodologies over 32 malware families. Table 7, table 8 and table 9 are the confusion matrices of CNN, KNN and SVM using LBP feature. According to the confusion matrices, we discover that the malware belongs to family 28, 29 and 30 which are Virut.A, Virut.AC and Virut.AT respectively are easy to get confused. As seen in table 5, CNN can differentiate these three with higher accuracy than others.

Classifier	#Dataset	#Family	Feature Descriptor	AVG. Accuracy
CNN	12348	32	LBP	93.92%
SVM			LBP	87.84%
KNN			LBP	85.93%
CNN			GIST	87.88%
SVM			GIST	81.23%
KNN			GIST	82.83%

Table 6. Experiment Result of First Malware Dataset

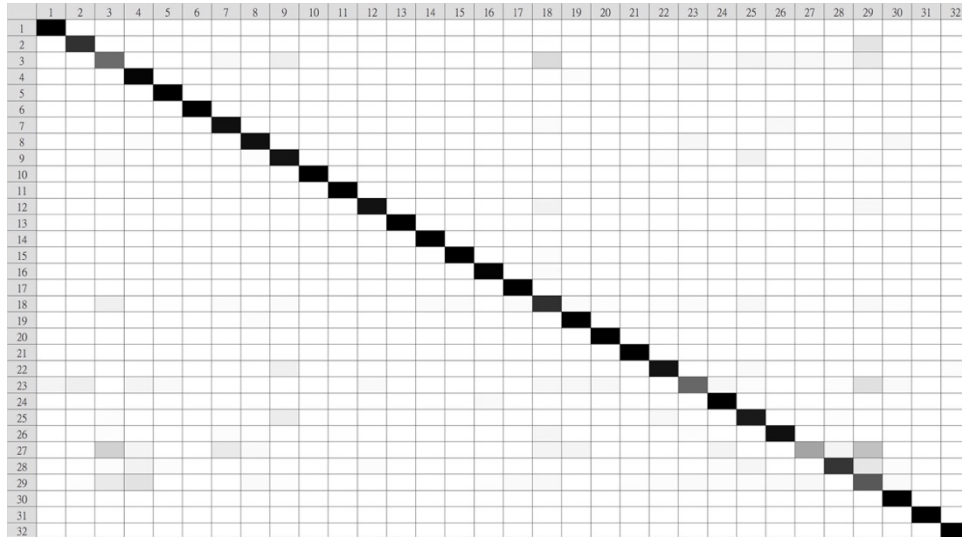


Table 7. Confusion Matrix of CNN using LBP feature

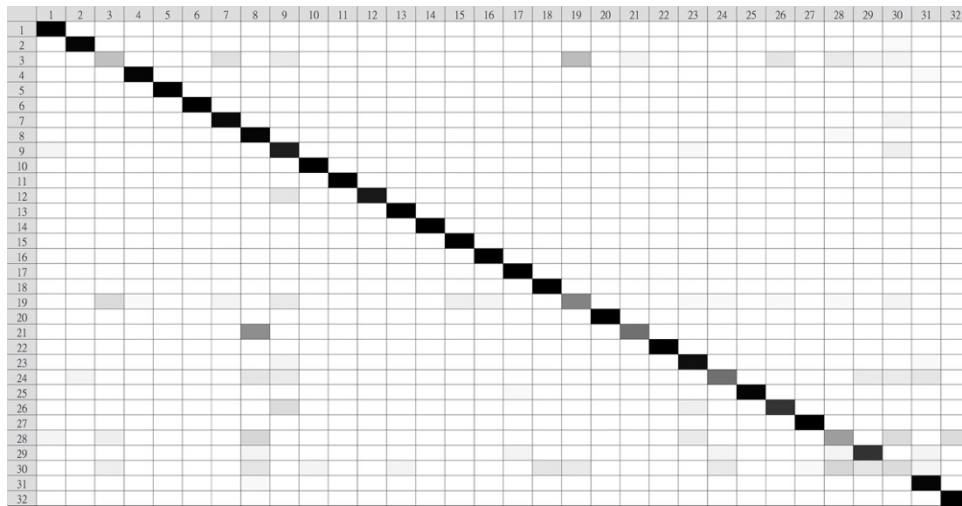


Table 8. Confusion Matrix of KNN using LBP feature

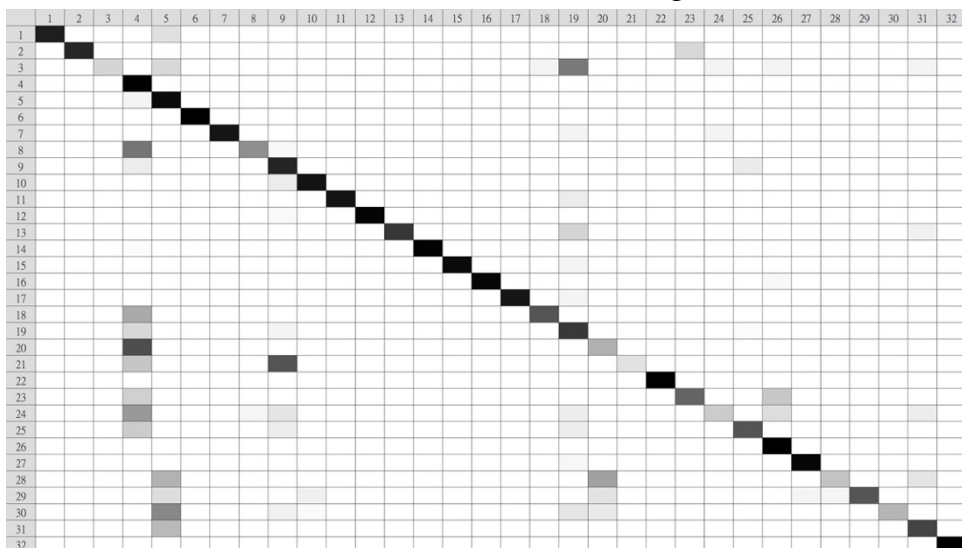


Table 9. Confusion Matrix of SVM using LBP feature

4.3 Second Dataset Experimental Results

In the second experimental, we focus on analysis the performance between grayscale image and RGBA image. We apply LBP both on gray and color image and use CNN to training and classify the data. The result demonstrates that using grayscale image is 4% higher than color image. The reason why using grayscale image is better than color image is that when we covert the malware to image, the grayscale image and color image have different structure. Converting color image might let the image lost original features.

Classifier	#Dataset	#Family	Color Space	AVG. Accuracy
CNN	10868	9	Grayscale	93.57%
CNN			RGBA	89.18%

Table 10. Experiment Result of Second Malware Dataset

4.4 Pros and Cons

As Figure 19 demonstrates, the execution time of CNN is better than other classifiers in that our approach run with GPU, which is significantly shorter the execution time. In [42] ,T. Ishii, R. Nakamura, H. Nakada, Y. Mochizuki and H. Ishikawa, they also obtain the similar result of execution time. Moreover, this method doesn't have to run on a virtual machine or virtual environment to observe the behavior of malware. Additionally, because our approach is based on image processing, we can apply other image descriptors to do the voting to achieve higher classification accuracy.

Although malware images can be analyzed with our approach based on local binary pattern and machine learning, there still have countermeasures. Because our approach converts the malware into binary and reorganizes. Therefore, if a rival who rewrites whole the program in other way or uses other instructions instead of original one result in changing whole the pattern of malware image, our approach may fail.

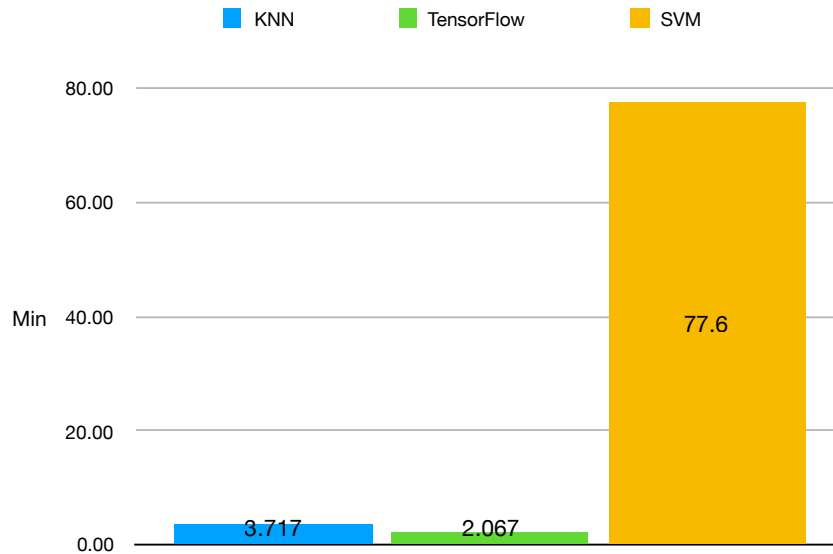


Figure 19. Average Execution Time

CHAPTER V

Conclusion and Future Work

5.1 Future Work

While our experimental results demonstrate that the accuracy using LBP as feature is slightly higher than other methodologies, there are ways of how the experiment could be improved. The first priority would be to extend the malware family, which means that increases the size and classes of dataset. At the meantime, applying other image processing approach instead of LBP to the malware image is one possible future work. Additionally, we plan to design a different architecture of Tensorflow and examine more different classifier such as Decision Tree, Fandom Forest, and Naïve Bayes to increase the accuracy and reduce time consumption. Furthermore, converting to HSV [39] color space is one option in that we can apply LBP only one time on Hue channel in stead applying LBP three times on RGB channels.

5.2 Conclusion

An experimental result shows that the accuracy based on our approach is 93.92%. The experiment is performed to classify malware images over 32 families around 12000 malware images. We reorganize malware images and utilize Local Binary Pattern as descriptor to extract features and classify the results with TensorFlow library. The

comparison over different classifiers and features demonstrates that using LBP with TensorFlow obtains higher accuracy than others approaches.

Furthermore, extending dataset of malware, converting malware to HSV color space, designing different architectures of TensorFlow and testing more image descriptors is our future works, which may improves the research and obtains more comprehensive methodology.

References

- [1] Nataraj L., Karthikeyan S., Jacob G., Manjunath B. S., “The malware Images: Visualization and Automatic Classification,” International Symposium on Visualization for Cyber Security (VizSec) , July 20, 2011, Pittsburg, PA, USA.
- [2] Malware statistic ,<https://www.av-test.org/en/statistics/the-malware/> , Accessed in 2017
- [3] A. Moser, C. Kruegel and E. Kirda, “Limits of Static Analysis for Malware Detection,” Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), Miami Beach, FL, 2007, pp. 421-430.
- [4] Malware Images, <http://vision.ece.ucsb.edu/~lakshman/malwareimages/album/>, Accessed in 2017
- [5] Aziz Makandar and Anita Patrot, “Wavelet Statistical Feature Based Malware Class Recognition and Classification using Supervised Learning Classifier,” Oriental Journal of Computer Science and Technology, ISSN: 0974-6471, June 2017, Vol. 10, No. (2): Pgs. 400-406
- [6] T. Ojala, M. Pietikainen, and D. Harwood, “A Comparative Study of Texture Measures with Classification Based on Feature Distributions,” Pattern Recognition, vol. 29, pp. 51-59, 1996.
- [7] Chao Zhu, Charles-Edmond Bichot and Liming Chen, “Multiscale Color Local Binary Patterns for Visual Object Classes Recognition,” 2010 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 3065-3068.

- [8] Chao Zhu, Charles-Edmond Bichot and Liming Chen, "Image region description using orthogonal combination of local binary patterns enhanced with color information," *Pattern Recognition*, Volume 46, Issue 7, 2013, Pages 1949-1963, ISSN 0031-3203
- [9] Aude Oliva, Antonio Torralba, "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope," *International Journal of Computer Vision*, Vol. 42(3): 145-175, 2001.
- [10] A. Oliva and A. Torralba, "Building the gist of a scene: the role of global image features in recognition," *Prog. Brain Res. Vis. Percept.*, vol. 155, pp. 2336, 2006.
- [11] A. Torralba, K. P. Murphy, W. T. Freeman and M. A. Rubin, "Context-Based Vision System for Place and Object Recognition," *Proceedings Ninth IEEE International Conference on Computer Vision*, Nice, France, 2003, pp. 273-280 vol.1.
- [12] Chao Zhu, Charles-Edmond Bichot and Liming Chen, "Color orthogonal local binary patterns combination for image region description," *Rapport technique RR-LIRIS-2011-012*, LIRIS UMR, vol. 5205, p. 15, 2011
- [13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv preprint arXiv:1603.04467*, 2016

- [14] Aziz Makandar and Anita Patrot, "Malware Analysis and Classification using Artificial Neural Network," 2015 International Conference on Trends in Automation, Communications and Computing Technology (ITACT-15), Bangalore, 2015, pp. 16.
- [15] R. Pilipovi and V. Risojevi, "Evaluation of convnets for large-scale scene classification from high-resolution remote sensing images," IEEE EUROCON 2017 -17th International Con
- [16] Avishikta Lodh and Ranjan Parekh, "Flower Recognition System based on Color and GIST Features," 2017 Devices for Integrated Circuit (DevIC), 23-24 March, 2017, Kalyani, India
- [17] Waleed Tahir, Aamir Majeed and Tauseef Rehman, "Indoor/Outdoor Image Classification Using GIST Image Features and Neural Network Classifiers," " 2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET), Islamabad, 2015, pp. 1-5
- [18] ANKIT SACHAN, <http://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/>, Accessed: 2016
- [19] Brandon, "How do Convolutional Neural Networks work, https://brohrer.mcknote.com/zh-Hant/how_machine_learning_works/how_convolutional_neural_networks_work.html, Accessed in 2017

- [20] Adit Deshpande, <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>, Accessed: 2017
- [21] Cortes, C. & Vapnik, V. Machine Learning (1995) 20: 273.
<https://doi.org/10.1023/A:1022627411411>
- [22] Fang-Chieh Liu, “A Study for Automatic Coin Image Recognition Methods”
- [23] Seok, Seonhee & Kim, Howon. (2016). Visualized Malware Classification Based on Convolutional Neural Network. Journal of the Korea Institute of Information Security and Cryptology. 26. 197-208. 10.13089/JKIISC.2016.26.1.197.
- [24] S. Choi, S. Jang, Y. Kim and J. Kim, “Malware detection using malware image and deep learning,” 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2017, pp. 1193-1195.
- [25] K. Kancherla and S. Mukkamala, “Image visualization based malware detection,” 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, 2013, pp. 40-44.
- [26] J. H. Lee, C. J. Lin, “Automatic model selection for support vector machines,” Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, 2000
- [27] C. C. Chang, C. J. Lin, “LIBSVM: a library for support vector machines,” Department of Computer Science and Information Engineering, National Taiwan

University, 2001

- [28] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," 2017 International Conference on Data Management, Analytics and Innovation (ICDMAI), Pune, 2017, pp. 76-80.
- [29] Savan Patel, <https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72>, Accessed: 2017
- [30] N. S. Altman (2012) An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression, *The American Statistician*, 46:3, 175-185, DOI: 10.1080/00031305.1992.10475879
- [31] X. Zhou, J. Pang and G. Liang, "Image classification for malware detection using extremely randomized trees," 2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, 2017, pp. 54-59.
- [32] B. N. Narayanan, O. Djaneye-Boundjou and T. M. Kebede, "Performance analysis of machine learning and pattern recognition algorithms for Malware classification," 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS), Dayton, OH, 2016, pp. 338-342.
- [33] N. S. Altman (2012) An Introduction to Kernel and Nearest-Neighbor

Nonparametric Regression, *The American Statistician*, 46:3, 175
185, DOI: 10.1080/00031305.1992.10475879

[34] S. Das and U. R. Jena, "Texture classification using combination of LBP and GLRLM features along with KNN and multiclass SVM classification," *2016 2nd International Conference on Communication Control and Intelligent Systems (CCIS)*, Mathura, 2016, pp. 115-119. doi: 10.1109/CCIntelS.2016.7878212

[35] Kevin Zakka's, <https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/>,
Accessed in 2017

[36] Euclidean Distance, https://en.wikipedia.org/wiki/Euclidean_distance,
Accessed in 2017

[37] Microsoft Malware Classification Challenge (BIG 2015),
<https://www.kaggle.com/c/malware-classification/data>, Accessed in 2017

[38] Malware, <https://en.wikipedia.org/wiki/Malware>, Accessed in 2017

[39] HSV Color Space, https://en.wikipedia.org/wiki/HSL_and_HSV,
Accessed in 2017

[40] RGBA Color Space, https://en.wikipedia.org/wiki/RGBA_color_space,
Accessed: 2017

- [41] SVM Hyperplane, <https://www.svm-tutorial.com/2015/06/svm-understanding-math-part-3/>, Accessed: 2017
- [42] T. Ishii, R. Nakamura, H. Nakada, Y. Mochizuki and H. Ishikawa, “Surface object recognition with CNN and SVM in Landsat 8 images,” 2015 14th IAPR International Conference on Machine Vision Applications (MVA), Tokyo, 2015, pp. 341-344.
- [43] Alex Krizhevsky, Ilya Sutskever, E. Hinton Geoffrey, “ImageNet Classification with Deep Convolutional Neural Networks”, *International Conference on Neural Information Processing Systems (NIPS)*, 2012.
- [44] SVM One vs Rest, One vs One, <http://scikit-learn.org/stable/modules/multiclass.html>, Accessed in 2017
- [45] ReLU, <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>, Accessed in 2017
- [46] Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* 2017, 29, 2352–2449.
- [47] Sowmya V., P., S. K., and Deepika, J., “Image Classification Using Convolutional

Neural Networks”, International Journal of Scientific & Engineering Research ,
vol. 5, no. 6, p. 06/2014, 2014.

[48] M. Lin, Q. Chen, and S. Yan. Network in network. In ICLR, 2014.