MACHINE LEARNING METHODS FOR
MALWARE DETECTION AND CLASSIFICATION

Submitted in fulfillment of seminar required for the
Master of Technology

Computer Science and Engineering

by

Abhishek Dhiman
173050066

Guided by
Prof. G Sivakumar

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY
Mumbai, Maharashtra, India – 400076
Spring Semester 2018
Acknowledgments

I would like to express my special thanks and gratitude to my guide Prof. G Sivakumar for giving this golden opportunity to do this seminar on the topic "MACHINE LEARNING METHODS FORMALWARE DETECTION AND CLASSIFICATION" and also for guiding me by suggestions and constructive criticisms which also helped me in getting a good knowledge about machine learning and Computer Security.

Abhishek Dhiman (173050066)
Abstract

Malware is a big threat to the internet today. Nowadays, malware designed by attackers are generally polymorphic in nature. Polymorphic malware is a type of malware that constantly changes its identifiable features in order to evade detection using signature-based malware detection techniques[4]. Behavior-based malware detection evaluates an object based on its intended actions before it can actually execute that behavior. The behavioral patterns obtained either statically or dynamically can be exploited to detect and classify unknown malware into their known families using machine learning techniques[5]. In this report, I will discuss Behavior-Based Detection methods and how we can apply different machine learning techniques in order to build behavior-based malware detection and classification methods.

Keywords: Static Analysis; Dynamic Analysis; Machine Learning; Classification; Behavior based detection methods
# Contents

Acknowledgements ................................................................. 1

1 Introduction ........................................................................... 1

2 Related work ......................................................................... 2
   2.1 Different types of Malware ................................................... 2
   2.2 Malware analysis technique .................................................. 2

3 Machine Learning techniques for Detecting and Classifying Malwares 4
   3.1 Using Concepts of machine learning and data mining for Detecting and
       Classifying Malwares ............................................................ 4
       3.1.1 Feature selection ........................................................... 4
       3.1.2 Different classification Algorithms ..................................... 5
   3.2 Visualizing and Classifying malware using image processing techniques 7
       3.2.1 Image Representation of Malware ....................................... 8
       3.2.2 Texture Feature Extraction .............................................. 8
       3.2.3 Classification technique ................................................ 9
   3.3 Analysis of machine learning techniques used in behavior-based malware
       detection ........................................................................... 11
       3.3.1 Data collection and Automatic Behavior Monitoring and Report
            Generation ...................................................................... 11
       3.3.2 Data Preprocessing and Feature selection .......................... 11
       3.3.3 Different Classification algorithms used ........................... 11

4 Results and Future Work .......................................................... 13
   4.1 Performance Metrics ............................................................. 13
       4.1.1 Approach 1 ................................................................... 13
       4.1.2 Approach 2 ................................................................... 13
       4.1.3 Approach 3 ................................................................... 13
   4.2 Results ............................................................................... 14

5 Conclusion ............................................................................. 15

References .............................................................................. 16
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Block diagram to convert malware binary to an image</td>
<td>8</td>
</tr>
<tr>
<td>3.2</td>
<td>Block diagram to compute texture feature on an image</td>
<td>9</td>
</tr>
<tr>
<td>3.3</td>
<td>KNN example</td>
<td>10</td>
</tr>
<tr>
<td>3.4</td>
<td>General overview of the methodology</td>
<td>12</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

We all know the importance of the internet in our life’s. Its trend has been grown exponentially in few past decades. Also, the internet is increasingly used by a large black market has emerged where hackers or others with criminal intent can purchase malware[1]. Also, with a large number of tools available nowadays the amount of skills required to create a new malware is decreasing rapidly. The modern malwares purpose is commonly illegal profit. For example, a large number of computers are infected by a keylogger and 24.3 billion USD is leveraged by e-payment system losing[2].

Kaspersky Labs (2017) define malware as “a type of computer program designed to infect a legitimate user’s computer and inflict harm on it in multiple ways.”.Based on AV-Test, approximately 390,000 new malware samples are registered every day, which gives rise to the problem of processing the huge amount of unstructured data obtained from malware analysis.[3]. This makes it challenging for anti-malware vendors to detect zero-day attacks and release updates in a reasonable time-frame to prevent infection and propagation.

Therefore, malware protection of computer systems is one of the most important cybersecurity tasks for single users and businesses because even a single attack can result in big data and money losses. In order to stay alive in the arms race against malware writers, developers of anti-malware software heavily rely on automatic malware analysis tools[6]. But, the bad news is malware detection such as signature-based techniques can easily be fouled by hiding techniques such as polymorphism and obfuscation. Also, these signature-based malware detection techniques can also detect fix known malware’s and can not go further. An alternative to the approach presented is performing behavior-based detection of the malware. The Behavior-based Malware Detection (BMD) method is quite different from SMD, it utilizes the behavior information of the malware during its execution as the detection basis, instead of using fixed signatures or pattern of malware code’s [7]. So BMD will not be affected by techniques like polymorphism, obfuscation, therefore can detect new malware. Most of the time these new malware’s that arise on regular basis are nothing but modified versions of previous malware using sophisticated reproduction techniques. Therefore, our behavior-based malware detection(BMD) can be a great choice in this was against malware.
Chapter 2
Related work

This chapter provides the background that is essential to understand the malware detection and classification techniques. Firstly, I will discuss malware types and then type of malware analysis. After that, I will discuss some machine learning techniques used by different authors for malware detection.

2.1 Different types of Malware

Malware can be classified into different categories based on behaviors, method of infection and resulting propagation[4]. The types follow:

- **Virus**: "Software which infects other application and uses them as a spreading medium[5]."
- **Trojan**: "A malicious application which present itself as something else[5]."
- **Worm**: "Code with ability to spread from computer to computer by means of different network protocols[5]."
- **Spyware**: "Application aiming to harvest personal information[5]."
- **Rookit**: "Hidden tools providing stealth services to its writer[5]."

2.2 Malware analysis technique

We discussed that malware detection techniques can be divided into signature-based and behavior-based methods[4]. Also, there are two basic approaches that we can use for malware analysis static and dynamic malware analysis[8].

- **Dynamic malware analysis** In this technique, the behavior of the file is monitored while it is executing and the properties and intentions of the file are inferred from that information[8]. Usually, the file is run in the virtual environment, for example in the sandbox. During this kind of analysis, it is possible to find all behavioral attributes, such as opened files, created, etc[8]. We can define
behavior-based methods as both static and dynamic analysis, whereas signature-based techniques are static most of the time. The advantage of dynamic malware analysis technique is simple. The process of malware analysis is running malware and making sure what will happen in the system.

- **Static malware analysis** This can be viewed as reading the source code of the malware and trying to infer the behavioral properties of the file[7]. Static malware analysis has an advantage that it can completely discover the purpose and functionality of malware. However, research needs time to understand the malware functionality by analyzing malware structure[8]. [6] discuss different types of static analysis:

  1. **File Format Inspection:** File meta-data can provide useful information. For example, Windows PE (portable executable) files can provide much information on compile time, imported and exported functions, etc.

  2. **String Extraction:** This refers to the examination of the software output (e.g. status or error messages) and inferring information about the malware operation.

  3. **Fingerprinting:** This includes cryptographic hash computation, finding the environmental artifacts, such as hardcoded username, filename, registry strings.

  4. **AV scanning:** If the inspected file is a well-known malware, most likely all anti-virus scanners will be able to detect it. Although it might seem irrelevant, this way of detection is often used by AV vendors or sandboxes to “confirm” their results.

  5. **Disassembly:** If the inspected file is a well-known malware, most likely all anti-virus scanners will be able to detect it. Although it might seem irrelevant, this way of detection is often used by AV vendors or sandboxes to confirm their results.

M. Egele et al. 2001 are focused on dynamic malware analysis techniques have previously focused on obtaining reliable and accurate information on the execution of malicious programs. Whereas paper M. Bailey et al. 2001 are focused on automatic processing of information collected from dynamic malware analysis. Two techniques for behavior-based malware analysis using clustering machine learning methods (Unsupervised in nature) have been recently proposed by M. Egele et al. 2001. Both methods transform report of observed behavior into sequences and use sequential distance, to group them into clusters which are believed to correspond to malware families. The main difficulty of clustering methods stems from their unsupervised nature, i.e., the lack of any external information provided to guide analysis of data. Let us illustrate some practical problems of clustering-based approaches. A major issue for any clustering method is to decide how many clusters are present in the data.
Chapter 3

Machine Learning techniques for Detecting and Classifying Malwares

Various machine learning methods are found useful in the literature of malware detection and classification. Some of these techniques are support vector machines, ensemble models like random forest, naive bayes, and also unsupervised learning methods like clustering for detecting and classifying unknown samples into either known malware families or underling those samples that exhibit unseen behavior. In this section, I will discuss a few papers that used these techniques for malware detection using machine learning.

3.1 Using Concepts of machine learning and data mining for Detecting and Classifying Malwares

M. Schultz et al. 2001 were the first to introduce the concept of data mining for detecting malware. They showed that this data-mining framework can detect new, previously unseen malicious executables accurately and automatically. The result of this paper was very impressive if we compare this technique to the traditional signature-based method. This method more than doubles the detection rates for new malicious executables.

3.1.1 Feature selection

M. Schultz et al. 2001 used a dataset of a total of 4,266 programs split into 3,265 malicious binaries and 1,001 clean programs. There were no duplicate programs in their dataset and every example in their set was labeled either malicious or benign by the commercial virus scanner. They also examine a subset of the data that was in Portable Executable (PE) [171] format. The dataset consisting of PE format executables was composed of 206 benign programs and 38 malicious executables.

M. Schultz et al. 2001 used three types of different static features for malware classification: (They focused mainly on the classification part) Portable Executable, strings sequences, and byte sequences.
1. **Portable Executable (PE)**

From the PE header, they used libBFD, a library within Bin-Utils, to extract information in object format. Object format for a PE binary gives the file size, the names of DLLs, and the names of function calls within those DLLs and Relocation Tables. Also in this PE approach, the features like a list of DLLs used by the binary, the list of DLL function calls, and a number of different systems calls used within each DLL are extracted from DLL information inside PE files.

2. **String Sequences**

Strings are extracted from the executables based on the text strings that are encoded in program files. Through testing, they found that there were similar strings in malicious executables that distinguished them from clean programs and similar strings in benign programs that distinguished them from malicious executables. This method of detecting malicious executables is already used by the anti-malicious executable community to create signatures for malicious executables. But these extracted strings from an executable were not very robust as features because they can be changed easily, so they analyzed another feature, byte sequences.

3. **Byte Sequences**

The byte sequence approach uses sequences of n bytes extracted from an executable file. They used hexdump a tool that transforms binary files into hexadecimal files. The found that byte sequence feature was the most informative because it represents the machine code in an executable instead of resource information like libBFD features. Secondly, analyzing the entire binary gives more information for non-PE format executables than the strings method.

3.1.2 **Different classification Algorithms**

In this paper [12], they used three type of algorithms Ripper, Naive Bayes, and a Multi-Classifier system. Let’s discuss them one by one:

1. **Ripper**

Main goal of Ripper is to find the patterns in the string data. This algorithm generated a detection model composed of resource rules that were built to detect future examples of malicious executables. Ripper is a rule-based learner that builds a set of rules that identify the classes while minimizing the amount of error.

2. **Naive Bayes**

The naive Bayes classifier computes the likelihood that a program is malicious given the features that are contained in the program. But they used it in the other way for their classification purpose. They compute the class of a program
given that the program contains a set of features \( F \).

\[
P(C|F) = \frac{P(F|C) \times P(C)}{P(F)} \quad \text{(1)}
\]

To use the naive Bayes rule they assume that the features occur independently from one another (Definition of Naive Bayes). If the features of a program \( F \) include the features \( F_1, F_2, F_3, ..., F_n \), then above equation becomes:

\[
P(C|F) = \prod_{i=1}^{n} \frac{P(F_i|C) \times P(C)}{P(F)} \quad \text{(2)}
\]

Each \( P(F_i|C) \) is the frequency that string \( F_i \) occurs in a program of class \( C \). \( P(C) \) is the proportion of class \( C \) in the entire set of programs. The output of the classifier is the highest probability class for a given set of strings. Since the denominator of (1) is the same for all classes we take the maximum class over all classes \( C \) of the probability of each class computed in (2) to get:

\[
MaxLikelyClass = \max_C (P(C) \prod_{i=1}^{n} P(F_i|C)) \quad \text{(3)}
\]

In (3), They used \( \max_C \) to denote the function that returns the class with the highest probability. Most Likely Class was the class in \( C \) with the highest probability and hence the most likely classification of the example with features \( F \). To train the classifier, they recorded how many programs in each class contained each unique feature. They used this information to classify a new program into an appropriate class. They first used feature extraction to determine the features contained in the program. Then they applied equation (3) to compute the most likely class for the program. They used the Naive Bayes algorithm and computed the most likely class for byte sequences and strings.

3. Multi-Naive Bayes

This algorithm was essentially a collection of Naive Bayes algorithms that voted on an overall classification for an example. Each Naive Bayes algorithm classified the examples in the test set as malicious or benign and this counted as a vote. The votes were combined by the Multi-Naive Bayes algorithm to output a final classification for all the Naive Bayes.

This method was required for them because even using a machine with one gigabyte of RAM, the size of the binary data was too large to fit into memory. The Naive Bayes algorithm required a table of all strings or bytes to compute its probabilities. To correct this problem they divided the problem into smaller pieces that would fit in memory and trained a Naive Bayes algorithm over each of the sub-problems. They split the data evenly into several sets by putting each \( i \)th line in the binary into the \( (i \mod n) \)th set where \( 11 \) is the number of sets. For each set, They trained a Naive Bayes classifier. Our prediction for a binary
is the product of the predictions of the n classifiers. In their experiments, they used 6 classifiers (n = 6).

More formally, the Multi-Naive Bayes promotes a vote of confidence between all of the underlying Naive Bayes classifiers. Each classifier gives a probability of a class C given a set of bytes F which the Multi-Naive Bayes uses to generate a probability for class C given F over all the classifiers. They want to compute the likelihood of a class C given bytes F and the probabilities learned by each classifier NaiveBayes. In equation (4) we computed the likelihood, $L_{NB}(C|F)$, of class C given a set of bytes F.

$$L_{NB} = \prod_{i=1}^{NB} \frac{P_{NB_i}(C|F)}{P_{NB_i}(C)}$$

where NBi is a Naive Bayes classifier and NB is the set of all combined Naive Bayes classifiers. The output of the multi-classifier given a set of bytes F is the class of highest probability over the classes given $L_{NB}(C|F)$ and $P_{NB}(C)$ the prior probability of a given class.

$$MaxLikelyClass = \max_C (P_{NB}(C) \times L_{NB}(C|F))$$

Most Likely Class was the class in C with the highest probability hence the most likely classification of the example with features F, and $\max_C$ returns the class with the highest likelihood.

The result of the paper M. Schultz et al 2001. I will discuss in the results section. These results were improved by M. Schultz et al 2001. They used n-gram (instead of non-overlapping byte sequence) and data mining method to detect malicious executables. They used different classifiers including Naive-Bayes, Support Vector Machine, Decision Tree, and their boosted versions. They concluded that boosted decision tree gives the best classification results. Whereas, for M. Schultz et al.2001 naive-Bayes was the best classifier.

3.2 Visualizing and Classifying malware using image processing techniques

Many machine-learning methods have been applied to classify or cluster malware into families L. Nataraj et al.2011, based on different features derived from a dynamic review of the malware. While these approaches demonstrate promise, they are themselves subject to a growing array of countermeasures that increase the cost of capturing this binary features[14]. In these type of techniques, time requires for feature extraction does not scale well to the daily volume of binary instances being reported by those who diligently collect malware. A new type of feature extraction, used by a classification approach called binary-texture analysis.
An image texture is a block of pixels which contains variations of intensities arising from repeated patterns[14]. Texture-based features are very common techniques nowadays are also used in different domains such as medical image analysis, image classification and large-scale image analysis. A malware binary is first converted to an image representation on which texture based features are obtained. L. Nataraj et al. 2011 used this technique and compared it with dynamic analysis based malware classification techniques. Let’s discuss in detail.

### 3.2.1 Image Representation of Malware

In order to extract image-based features, a binary file of malware has to be transformed to an image. A given malware binary is read as a 1D array (vector) of 8-bit unsigned integers after that organized into a 2D array (matrix). They kept the width of the matrix of fixed size depending on the file size and the height varies according to the file size[16]. Figure 3.1 shows Block diagram to convert malware binary to an image and Figure 3.2 Block diagram to compute texture feature on an image.

![Block diagram to convert malware binary to an image](image1.png)

Figure 3.1: Block diagram to convert malware binary to an image

### 3.2.2 Texture Feature Extraction

Once the malware binary is converted to an image, a texture-based feature is computed on the image to characterize the malware. The author used GIST feature, which is commonly used in image recognition systems such as scene classification and object recognition and large-scale image search[8]. Every image location is represented by the output of filters tuned to different orientations and scales[14]. Also, we can use pyramid histogram of oriented gradients (PHOG) for the same purpose and also we
can use state of the art Deep learning methods which I think will be more successful than both GIST and PHOG. Figure 3.2 shows the block diagram to obtain the GIST feature. The steps to compute the texture feature are as follows:

1. Read every byte of a binary and store it in a numeric 1D array (range 0-255).
2. Convert the 1D array to a 2D array to obtain a grayscale image. The width of the image is fixed based on the file size of the binary [16].
3. Reshape the image to a constant sized square image of size 64 * 64.
5. Concatenate all the features to obtain a 320-dimensional feature vector.

### 3.2.3 Classification technique

If we see the previous literature of classification of image categories using Texture-based image features a k-nearest neighbors (k-NN) classifier is the best option. Let’s take an example of classify outdoor scene categories. Their classification was performed using a k-nearest neighbors (k-NN) classifier, where a database of labeled scenes with the corresponding category called the training set is used to classify unlabeled scenes. Given a new scene, the k nearest neighbors are computed and these correspond to k scenes from the training set with the smallest distance to the test scene. Then the new scene is assigned the label of the scene category that is most represented within the k nearest neighbors. L. Nataraj et al. 2011 use the similar type of method:

**K-Nearest neighbors** K-Nearest Neighbors (KNN) is one of the simplest, though, accurate machine learning algorithms [16]. KNN is a non-parametric algorithm, meaning that it does not make any assumptions about the data structure. In real-world problems, data rarely obeys the general theoretical assumptions, making non-parametric algorithms a good solution for such problems. KNN model representation is as simple as the dataset there is no learning required, the entire training set is stored.

KNN can be used for both classification and regression problems. In both problems, the prediction is based on the k training instances that are closest to the input.
instance. In the K-NN classification problem, the output would be a class, to which the input instance belongs, predicted by the majority vote of the k closest neighbors. In the regression problem, the output would be the property value, which is generally a mean value of the k nearest neighbors. Example in Figure 3.4 Let’s see how to apply these techniques to our problem. The training set will contain our families of known malware. Each family further contain several malware variants. The 320-dimensional GIST features are computed for all the binaries in the training set[14]. Let’s see each step in learning new unknown malware:

1. The 320-dimensional GIST feature vector is computed on the unknown malware binary.
2. The Euclidean Distance is computed with all feature vectors in the training set.
3. The nearest neighbors are the top k malware in the training set that has the smallest distance to the unknown.
4. The unknown malware is assigned a label that matches a majority returned by the k labels.

The limitation is that popular adversarial attacks are possible against these types of approaches. That is an attacker can adopt countermeasures to beat the system because this method uses global image-based features. For example, an attacker could relocate sections in a binary or add a vast amount of redundant data[11]. The result of the paper I will discuss in the next chapter.
3.3 Analysis of machine learning techniques used in behavior-based malware detection

You have described different machine learning techniques and feature extraction methods for malware detection and classification. Firdausi et al. 2010 tried various different such approaches and summaries them experimentally in a very nice manner. They used emulated (sandbox) environment in order to automatically analyzed and will generate behavior reports and will generate behavior reports. Let’s discuss this complete approach step by step.

3.3.1 Data collection and Automatic Behavior Monitoring and Report Generation

Their dataset consists of malware dataset and benign instance data set. Both malware and benign instance data sets are in the format of Windows Portable Executable (PE) file binaries. A total of 220 unique malware (specifically Indonesian malware) samples were acquired. They also collect benign instance data set samples from system files local in the “System32” directory of a clean installation of Windows XP Professional 32-bit with Service Pack 2.

The Key step is to generate the Report Generation. They do it by conducting dynamic analysis (behavior monitoring) of both the malware and benign instance data sets.

3.3.2 Data Preprocessing and Feature selection

Data Preprocessing was the key step in their research. I break down them in following steps.

- All the XML report files were parsed to select the most relevant and important attribute values (feature selection).

- After selecting the necessary attributes a term dictionary was created, which contains all the attribute values that were previously parsed and selected.

- Each XML report file was compared against the term dictionary by counting the existence (or non-existence) of each term word in the term dictionary based on binary weight and term frequency weight.

- Sparse vector models were created for each XML report file and Attribute-Relation File Format (ARFF) files were created.

3.3.3 Different Classification algorithms used

Now we have feature’s that is ARFF files the step is to try different machine learning algorithms on top of that. Machine learning techniques were applied for the learning
and classification of the ARFF files[18]. Before applying models they divide dataset in following parts:

2. Term frequency-weight vector model without feature selection.
4. Term frequency-weight vector model with feature selection.

Each data set was applied to 5 different classifier algorithms, which were k-Nearest Neighbor, Naive Bayes, Support Vector Machine (SVM), J48 decision tree, and Multilayer Perceptron (MLP) neural network[18]. Results i will discuss in next chapter. Figure 3.4 describe the overview I discussed above.

Figure 3.4: General overview of the methodology.
Chapter 4

Results and Future Work

In this report I explained three techniques for malware detection and classification. First approach used data mining and machine learning algorithms like k-Nearest Neighbor, Nave Bayes, Support Vector Machine (SVM), J48 decision tree, and Multilayer Perceptron (MLP) neural network. Second approach was using image recognition algorithms and in third approach I discussed the analysis of machine learning algorithm for malware detection and classification.

4.1 Performance Metrics

4.1.1 Approach 1

For the first approach we can measure the performance based on three measure's true positives (Data points those are actually true and are classified true), false positives (Data points those are actually false but are are classified true) and finally we can compare accuracy of different models that we discussed in the first section[12].

4.1.2 Approach 2

For the second approach which is a type of multiclass classification we can use accuracy, confusion matrix for 63,002 malware comprising 531 families for data-set one, and confusion matrix for 500,000 malware comprising 831 families for data-set two[14].

4.1.3 Approach 3

For the third approach we can measure performance statistically, based on malicious or benign tests. These statistical measures include positive predictive value (precision), true positive rate (sensitivity, recall, hit rate), false positive rate (fall-out), and accuracy (Number of truly classified points divided by total number of points.)[18].
4.2 Results

In the approach 1 based on the false positive rate was 2.13% which is very good as we compare to earlier results. And based on the true positive rate they were getting 98.1% results. And finally based on accuracy they got 98% results.

For approach two result are little complicated firstly for dataset one that is VX-Heavens[19]. Average accuracy obtained is 0.728%. We had around 531 families where for 33 families were classified with an accuracy of 1. Where as 105 families are getting classified with accuracy above 92%. Where for data-set 2 that is Anubis Data-set[?]. Initially there was no labelling on the data-set so clustering is applied to get the similar malware families grouped together. After that a supervised classification with 10-fold cross validation is applied to obtain a classification accuracy of 0.718.

For approach three which was a deep analysis of different machine learning based on 3 metric’s L. Nataraj et al.2011 compared them, Here is the summary of this analysis.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>TPR</th>
<th>FPR</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>81.7%</td>
<td>8.1%</td>
<td>86.5%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>58.1%</td>
<td>9.8%</td>
<td>65.4%</td>
</tr>
<tr>
<td>SVM</td>
<td>90.4%</td>
<td>8.4%</td>
<td>91.0%</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>90.9%</td>
<td>3.8%</td>
<td>93.6%</td>
</tr>
</tbody>
</table>

We saw that in L. Nataraj, et al .2011 paper we extract all feature using old image featuring extracting methods. We can replace this technique by state of the art in most of the image application deep learning methods to get better accuracy.
Chapter 5

Conclusion

I would like to conclude by report by saying that further research is needed in this area of Malware detection and classification since internet is reaching more and more people every day also malware the malware creation is becoming simple data by day and i showed using three different techniques that machine learning can be of great help in defending us from that.
References

[1] Radu S Pirscoveanu, S Steven, Thor MT Larsen, Matija Stevanovic and Alexandre Czech, “Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)”, Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), International Conference on 2015.


