MALWARE DETECTION AND ADVERSARIAL ANALYSIS USING CNN

Team Members

Vaibhav Vijay
Anish S Ghiya
Aditi Ranganath
Prateek Chaturvedi
Mayank Kilhor

Table Of Contents

Abstract

For over the last decade there has been a war between the computer security community and black hat hackers. The Security Community has used every possible strategy to cease and to get rid of the damage caused by malicious programs while at the same time the hacker community is advancing their techniques to bypass the security measures. Malware detection is crucial with malware's prevalence on the web considering it functions as an early warning system for the pc secure regarding malware and cyber attacks. In this paper, we propose a novel approach of first visualizing malware as an image and then training a classifier based on Deep Learning methods to maximize accuracy. This means that the machine will learn patterns in the images that it is presented with rather than requiring the human operator to define the patterns that the machine should look for in the image. A comprehensive comparative study of our model demonstrates that our proposed deep learning architectures outperform classical MLAs. Overall, this paper paves the way for effective visual detection of malware using a scalable and hybrid deep learning framework for real-time deployments. Along with this the paper also analyses the effect of adversarial attacks and the visual interpretation of the perturbation attached with that.

Introduction

In the Internet-age, malware poses a serious and evolving threat to security. Hackers are coming up with a myriad of ways to cause harm to Information Systems. The malicious attackers often use malwares that have been previously identified as they are already designed to exploit vulnerabilities as needed. A security officer's role is to identify these malwares and reduce the impact of these attacks by applying appropriate fixes. The task at hand is to classify these malwares as accurately as possible. Deep learning has dominated the various computer vision tasks. Not only these deep learning techniques enabled rapid progress in this competition, but even surpassed human performance in many of them. Unlike more traditional methods of machine learning techniques, deep learning classifiers are trained through feature learning rather than task-specific algorithms. What this means is that the machine will learn patterns in the images that it is presented with rather than requiring the human operator to define the patterns that the machine should look for in the image.

Security professionals are constantly faced with the threat of malware. It is important that they identify and prevent this harmful software from infecting systems.

Malware detection is performed by various algorithms and tools to identify and prevent the harmful effects of various types of infections. Traditional MDS (Malware Detection Systems) typically utilize traditional machine learning algorithms that require feature selection and extraction, which are time-consuming and error-prone. Deep learning has dominated the field of computer vision tasks. Not only did it outperform humans in many cases, but it also exhibited faster progress in other tasks.

Recently, Image classification has been improved a lot with the development of deep learning techniques. Convolutional Neural Networks demonstrated better performance. Here feature engineering, feature learning, and feature representation are automatically acquired.

Overall, the major contributions of the current research work are

- A proposal of a novel image processing technique for malware classification.

- An independent performance evaluation of classical MLAs and deep learning architectures, benchmarking various malware analysis models.
- Visualizing a malware image and then training a classifier based on Deep Learning techniques to improve its accuracy.

The rest of the paper is organized as follows. Section II reviews the Literature Survey based on malware classification models considering various approaches traditionally employed for malware analysis.

Problem Statement and Objective

The main objective is to produce a classification algorithm to accurately classify malware images (generated from malware files) using Deep learning and machine learning techniques to generate an ensemble model with higher accuracy.

We will map the images to a smaller representation using autoencoders and then on top of that perform Machine learning.

The images themselves will be classified using cnn models of different structures to find the best fit to obtain the best fit model that avoids the bias and variance tradeoff

LITERATURE REVIEW

Sno	Name	Year	Journal	Authors
1	Malware Detection Techniques: A Survey	2020	International Conference on PDGC	Yamjala Supriya ,Gautam Kumar and Dammu Sowjanya
2	Polymorphic malware detection using sequence classification methods and ensembles	2017	EURASIP J. on Info. Security	Drew, J., Hahsler, M. & Moore, T.
3	DL4MD: A Deep Learning Framework for Intelligent Malware Detection	2016	Int'l Conf. Data Mining	William Hardy, Lingwei Chen, Shifu Hou, Yanfang Ye*, and Xin Li
4	Malware identification using visualization images and deep learning.	2018	Computers & Security.	Ni, Sang, Quan Qian, and Rui Zhang.
5	Analysis of Machine Learning Classifier in Android Malware Detection Through Opcode	2020	2020 IEEE Conference on Application, Information and Network Security (AINS)	Noor Azleen Anuara, Mohd Zaki Mas'uda , Nazrul Azhar Bahamana and Nor Azman Mat Ariffa

1. **Malware Detection Techniques: A Survey**

- Yamjala Supriya ,Gautam Kumar and Dammu Sowjanya “Malware Detection Techniques: A Survey “2020 Sixth International Conference on PDGC(2020)
- Malware detectors use two inputs. First one is the knowledge to recognize the maliciousness of the program and the other is the program which has to be tested.
- The nowadays used techniques for detecting malware can be categorized mainly in two parts :-

  Anomaly-based detection

  Signature-based detection.
- An anomaly based detection advantage is in the possession to detect the zero-day attacks.The disadvantage is that it increases the high false rate with this detection technique.
- A Signature based detection advantage is it can identify the known occurrences of malware correctly and it takes the least number of resources to identify the malware.The disadvantage is that it cannot detect the new and unknown malware instances because we cannot find the signatures for those types of malware.
- Both the detection techniques have two process:-

  Dynamic -> data collected from the execution of a program is used to perceive malicious code.

  Static -> data collected after the execution of a program is used to perceive malicious code.

  Limitations:
- The malicious techniques give false alarms of malware and consider malicious behavior as normal.

2. **Polymorphic malware detection using sequence classification methods and ensembles**

- Apply methods designed for gene sequencing to detect malware in a manner robust to attacker adaptations.
- Identifying malicious software executables is made difficult by the constant adaptations introduced by miscreants in order to evade detection by antivirus software

- The individual size of each hash code value making up Strand's minhash signature is critical for producing accurate classification results
- In addition to polymorphic malware, anticipate that these classifiers can be used anywhere data sequences are used, such as in network traces of attacks or the identification of ransomware.

Methods used

  - Model ensembling,
  - Pruning
  - Prediction adjustments

Limitations:

While the approach, using only minimal adaptation, did not best the accuracy scores achieved by the highly tailored approach.

3. **DL4MD: A Deep Learning Framework for Intelligent MalwareDetection**

- Study how a deep learning architecture using the stacked AutoEncoders (SAEs) model can be designed for intelligent malware detection.
- The SAEs model performs as a greedy layerwise training operation for unsupervised feature learning, followed by supervised parameter fine-tuning (e.g., weights and offset vectors)
- Feature Extractor: It is composed of Decompressor and PE Parser.
- Deep Learning based Classifier: Based on the Windows API calls, a deep learning architecture using SAEs model is designed to perform unsupervised feature learning, supervised fine-tuning, and thus malware detection
- AutoEncoder: An AutoEncoder, also called Auto Associator, is an artificial neural network used for learning efficient codings.
- Deep learning architecture with SAEs: To form a deep network, an SAE model is created by daisy chaining AutoEncoders together, known as stacking:the output of one AutoEncoder in the current layer is used as the input of the AutoEncoder

Limitations

- Sparsity constraints are imposed on AutoEncoder and how sparse SAEs can be designed to further improve malware detection effectiveness.

4. **Malware identification using visualization images and deep learning**

- The system proposed by authors uses a malware classification algorithm that relies on static features called MCSC (Malware Classification using SimHash and CNN)

- Converts disassembled malware codes into gray images based on SimHash and identifies malware families using a CNN
- Multi-hash, Major block selection, and bilinear interpolation are used to improve performance.
- Shown to work well even for datasets with unevenly distributed samples , as is typical with malware datasets.
- Average accuracy was found to be around 98.8% and took only 1.5s to recognize a new sample.

Limitations:

- Usage of parallel computers like GPUs could potentially speedup the classification process

5. **Analysis of Machine Learning Classifier in Android Malware Detection Through Opcode**

- This paper explores Mobile Malware Detection through opcode analysis and proposes the optimum machine learning classifier method.
- Several machine learning classifier methods are selected based on the previous research, and their performance are analyzed to improve TPR and FPR.
- The chosen classifiers are Naïve Bayes (NB) Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (J48).
- The experiment is done using Weka 3.8.4 for Windows OS that is running on a laptop with an Intel® CoreTM i5-6200U CPU and 4GB of RAM.
- A huge difference is seen in the frequency of occurrence for opcode in mobile malware and benign applications. Other than that, the behavior of malicious applications are observed and the opcode extracted are mapped to its suspicious activities

Limitations:

- The sample size for conducting the research is too small to conclude that SVM would be the optimal Machine Learning technique- only 1000 Malignant and 500 benign samples were used.

In paper [6], they use gene sequencing techniques to identify malware. These methods can be used robustly to detect and prevent exploitation. In addition to polymorphic malware, anticipate that these classifiers can be used anywhere data sequences are used, such as in-network traces of attacks or the identification of ransomware.

Methods used

1) Model ensembling,
2) Pruning

3) Prediction adjustments

While the approach, using only minimal adaptation, did not beat the accuracy scores achieved by the highly tailored approach.

[7] proposes a deep learning architecture based on the stacked AutoEncoders model, which is designed to perform unsupervised feature learning and fine-tune after supervised parameter fine-tuning.AutoEncoder is an artificial neural network that learns efficient coding. It works by forming a deep learning model with predefined parameters (SAEs).

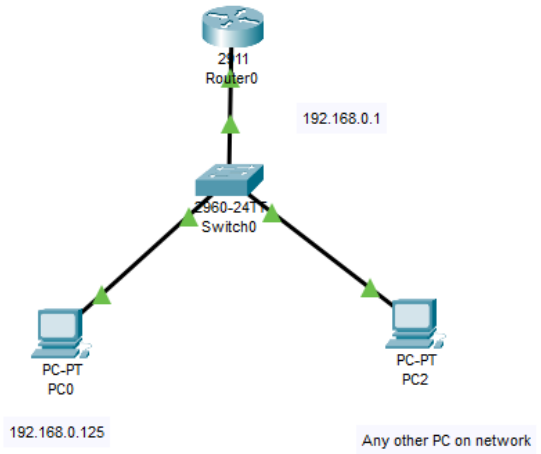
Sparsity constraints are imposed on AutoEncoder and how sparse SAEs can be designed to further improve malware detection effectiveness.

Paper [8] proposes a system that is based on a modified version of the MCSC malware classification algorithm. It uses static features such as SIMHash and CNN to identify families of malware. It achieves its accuracy by taking advantage of the various advantages of multi-hash computing. The Limitation would be that Usage of parallel computers like GPUs could potentially speed up the classification process.

Paper [9] explores Mobile Malware Detection through opcode analysis and proposes the optimum machine learning classifier method. The performance of various machine learning classifiers is analyzed to improve their TPR and FPR. A huge difference is seen in the frequency of occurrence for opcode in mobile malware and benign applications. Other than that, the behavior of malicious applications is observed and the opcode extracted is mapped to its suspicious activities. It was also found that the sample size for conducting the research is too small to conclude that SVM would be the optimal Machine Learning technique- only 1000 Malignant and 500 benign samples were used.

PROPOSED METHODOLOGY

1. The raw data contains the hexadecimal representation of the file's binary content. Convert those files into PNG images.
2. Data Analysis to find the number of images in each class and further preprocessing
3. Generates batches of normalized tensor image data from the respective data directories.
4. Generate a CNN based model and train on the dataset. Also trying to incorporate transfer learning and ensemble techniques.
5. Analyse the generated output in terms of the accuracy of classification using confusion matrix and weighted f1-score
6. Generate Adversarial examples using the classifier to create an unidentified attack on the network.
7. Analyse using PCA for latent space representations from convolutional layer of the malware.



Once the malware classifier has been built and tested we can deploy it in a real networking scenario. In this case, we perform an attack on a computer in the local network that is receiving files by sending it malware. Ideally, before or as soon as the file is received, the classifier will be able to tell if the incoming file is a malware.

Modules

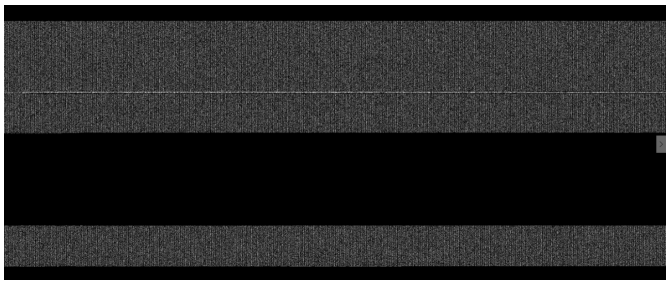
1.Conversion to Image:

The raw data contains the hexadecimal representation of the file's binary content. First, we Convert those files into PNG images.

0ACDbR5M3ZhBJajygTuf.bytes - Notepad

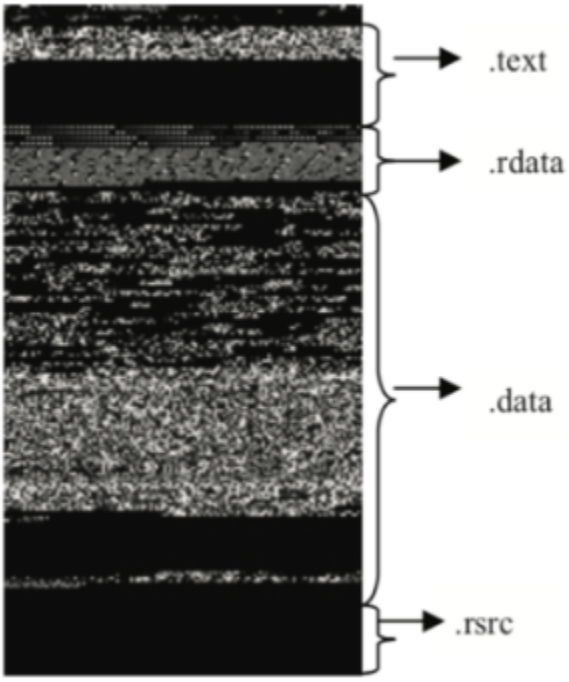
File Edit Format View Help

```
00401000 00 00 80 40 40 28 00 1C 02 42 00 C4 00 20 04 20
00401010 00 00 20 09 2A 02 00 00 00 00 8E 10 41 0A 21 01
00401020 40 00 02 01 00 90 21 00 32 40 00 1C 01 40 C8 18
00401030 40 82 02 63 20 00 00 09 10 01 02 21 00 82 00 04
00401040 82 20 08 83 00 08 00 00 00 00 02 00 60 80 10 80
00401050 18 00 00 20 A9 00 00 00 00 04 04 78 01 02 70 90
00401060 00 02 00 08 20 12 00 00 00 40 10 00 80 00 40 19
00401070 00 00 00 00 11 20 80 04 80 10 00 20 00 00 25 00
00401080 00 00 01 00 00 04 00 10 02 C1 80 80 00 20 20 00
00401090 08 A0 01 01 44 28 00 00 08 10 20 00 02 08 00 00
004010A0 00 40 00 00 00 34 40 40 00 04 00 08 80 08 00 08
004010B0 10 00 40 00 68 02 40 04 E1 00 28 14 00 08 20 0A
004010C0 06 01 02 00 40 00 00 00 00 00 00 20 00 02 00 04
004010D0 80 18 90 00 00 10 A0 00 45 09 00 10 04 40 44 82
004010E0 90 00 26 10 00 00 04 00 82 00 00 00 20 40 00 00
004010F0 B4 00 00 40 00 02 20 25 08 00 00 00 00 00 00 00
00401100 08 00 00 50 00 08 40 50 00 02 06 22 08 85 30 00
00401110 00 80 00 80 60 00 09 00 04 20 00 00 00 00 00 00
00401120 00 82 40 02 00 11 46 01 4A 01 8C 01 E6 00 86 10
00401130 4C 01 22 00 64 00 AE 01 EA 01 2A 11 E8 10 26 11
00401140 4E 11 8E 11 C2 00 6C 00 0C 11 60 01 CA 00 62 10
00401150 6C 01 A0 11 CE 10 2C 11 4E 10 8C 00 CE 01 AE 01
00401160 6C 10 6C 11 A2 01 AE 00 46 11 EE 10 22 00 A8 00
00401170 EC 01 08 11 A2 01 AE 10 6C 00 6E 00 AC 11 8C 00
00401180 EC 01 2A 10 2A 01 AE 00 40 00 C8 10 48 01 4E 11
00401190 0E 00 EC 11 24 10 4A 10 04 01 C8 11 E6 01 C2 00
004011A0 22 00 04 10 C0 01 C4 10 82 11 28 01 0A 10 2A 01
004011B0 E4 11 6A 11 26 00 80 11 26 00 0A 00 42 11 6E 10
004011C0 2A 10 22 10 6C 10 6C 10 04 11 EA 10 44 10 2C 01
004011D0 0C 11 4E 10 8A 00 4C 00 A4 10 0E 11 2A 00 84 11
004011E0 4C 00 20 11 4E 01 E2 01 E2 10 EA 11 22 01 0A 10
004011F0 20 10 48 11 CC 00 86 00 AA 01 42 10 2A 10 4A 11
00401200 86 01 26 10 AE 01 8A 10 E0 00 8A 10 CE 00 44 11
00401210 EC 11 4A 10 2A 01 AE 01 EA 00 C8 00 EE 01 C2 10
00401220 82 01 80 10 A6 00 44 10 0A 01 C6 01 C6 00 E8 11
00401230 86 01 A0 00 8C 10 66 11 04 00 2C 10 2A 01 AE 01
00401240 6C 10 CE 10 06 01 68 11 06 00 EC 10 C0 01 AC 01
00401250 C8 10 24 00 04 00 C2 01 EA 01 2A 10 2A 00 42 11
```

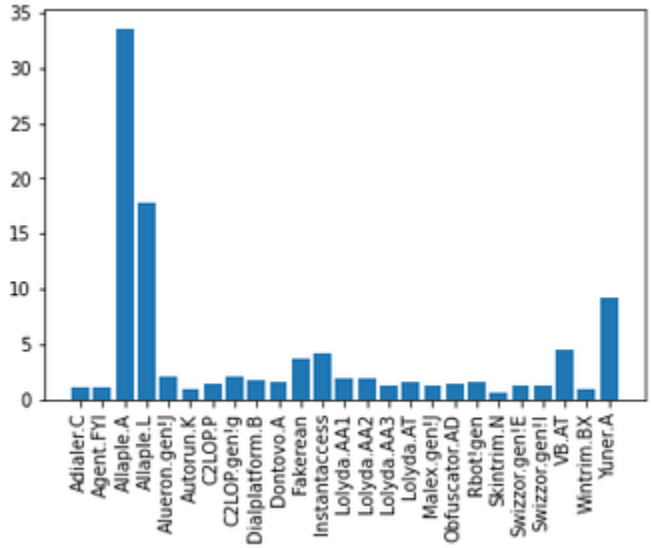


2.Basic EDA


Rbot!gen
Lolyda.AA1
Allaple.A
Swizzor.gen!I
Agent.FYI
VB.AT
Instantaccess
Allaple.L
Lolyda.AA2
VB.AT
Allaple.L
Allaple.A
Yuner.A
Allaple.A
Lolyda.AA3
Allaple.L
Allaple.A
Allaple.L
Allaple.A
Dialplatform.B
Dontovo.A
Malex.gen!J
Allaple.A
Allaple.A
Allaple.L
Allaple.A
Allaple.A
Allaple.A
Lolyda.AT
C2LOP.gen!g
Allaple.A
VB.AT
VB.AT
Instantaccess
Allaple.L
Allaple.A
Allaple.L
Skintrim.N
Allaple.L
Swizzor.gen!I
Adialer.C
Allaple.A
Yuner.A
Allaple.A
Swizzor.gen!E
Allaple.L
Wintrim.BX
Instantaccess
Yuner.A
Fakerean


.text
.rdata
.data
.rsrc

-Distribution Analysis




```
array([  97.,   91., 2824., 1491.,  173.,   81.,  121.,  175.,  152.,
        137.,  306.,  356.,  153.,  159.,   98.,  134.,  111.,  117.,
        133.,   55.,  103.,  107.,  383.,   72.,  775.], dtype=float32)
```

Malware Family	Samples	Malware kind
Adialer.C	123	Dialer
Agent.FYI	117	Backdoor
Allaple.A	2950	Worm
Allaple.L	1592	Worm
Alueron.gen!J	199	Trojan
Autorun.K	107	Worm AutoIT
C2LOP.gen!g	201	Trojan
C2LOP.p	147	Trojan
Dialplatform.B	178	Dialer
Donoto.A	163	Trojan Downloader
Fakerean	382	Rouge
Instaccess	432	Dialer
Lolyada.AA1	214	PWS
Lolyada.AA2	185	PWS
Lolyada.AA3	124	PWS
Lolyada.AT	160	PWS
Malex.gen!J	137	Trojan
Obfuscator.AD	143	Trojan Downloader
RBot!gen	159	Backdoor
Skintrim.N	81	Trojan
Swizzor.gen!E	129	Trojan Downloader
Swizzor.gen!I	133	Trojan Downloader
VB.AT	409	Worm
Wintrim.BX	98	Trojan Downloader
Yuner.A	801	Worm

2.Modelling

Custom CNN

A custom CNN model was generated to set the baseline scoring of the models to follow in this paper. The model was a 2 convolution layered model where each contained a convolutional layer and a max-pooling layer. Finally, it was flattened and passed through a hidden dense layer and ultimately to the output layer with 25 neurons pertaining to the number of classes present in the labels of the outputs.

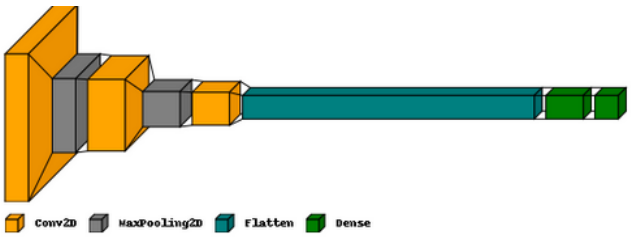


Fig 1. CNN architecture used in this paper

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        36928
_________________________________________________________________
flatten (Flatten)            (None, 50176)             0
_________________________________________________________________
dense (Dense)                (None, 64)                3211328
_________________________________________________________________
dense_1 (Dense)              (None, 25)                1625
=================================================================
Total params: 3,268,697
Trainable params: 3,268,697
Non-trainable params: 0
```

VGG16

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 512)               14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 25)                12825
=================================================================
Total params: 14,727,513
Trainable params: 14,727,513
Non-trainable params: 0
_________________________________________________________________
```

VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford.VGG-16 is a convolutional neural network consisting of 16 layers. The model loads a set of weights pre-trained on ImageNet. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.The default input size for VGG16 model is 224 x 224 pixels with 3 channels for RGB image. It has convolution layers of 3x3 filter with a stride 1 and maxpool layer of 2x2 filter of stride 2.

InceptionResNetV2

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 1536)              54336736
_________________________________________________________________
flatten (Flatten)            (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 25)                38425
=================================================================
Total params: 54,375,161
Trainable params: 54,314,617
Non-trainable params: 60,544
_________________________________________________________________
```

Inception-ResNet-v2 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 164 layers deep and can classify images into 1000 object categories, such as the keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299, and the output is a list of estimated class probabilities.

3.Scoring metric for the models

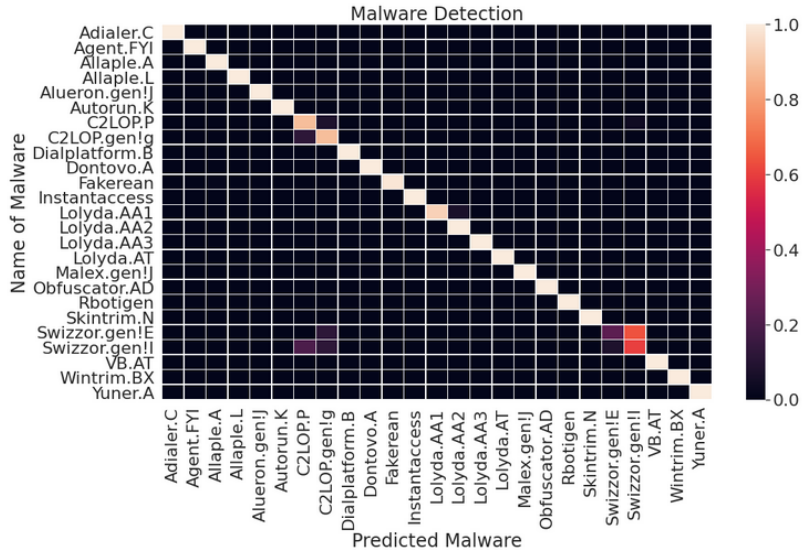
$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$$

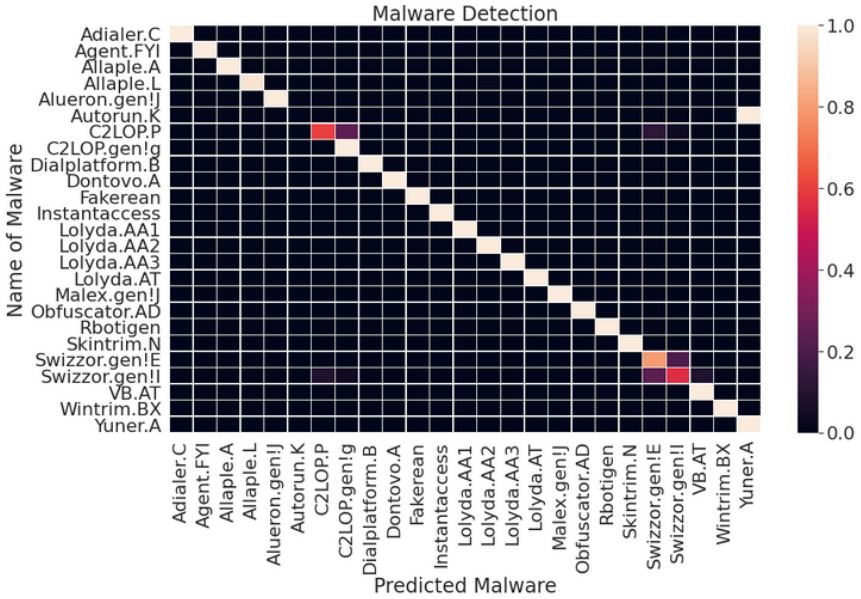
$$F1\ Score = \frac{TP}{TP+FP}$$

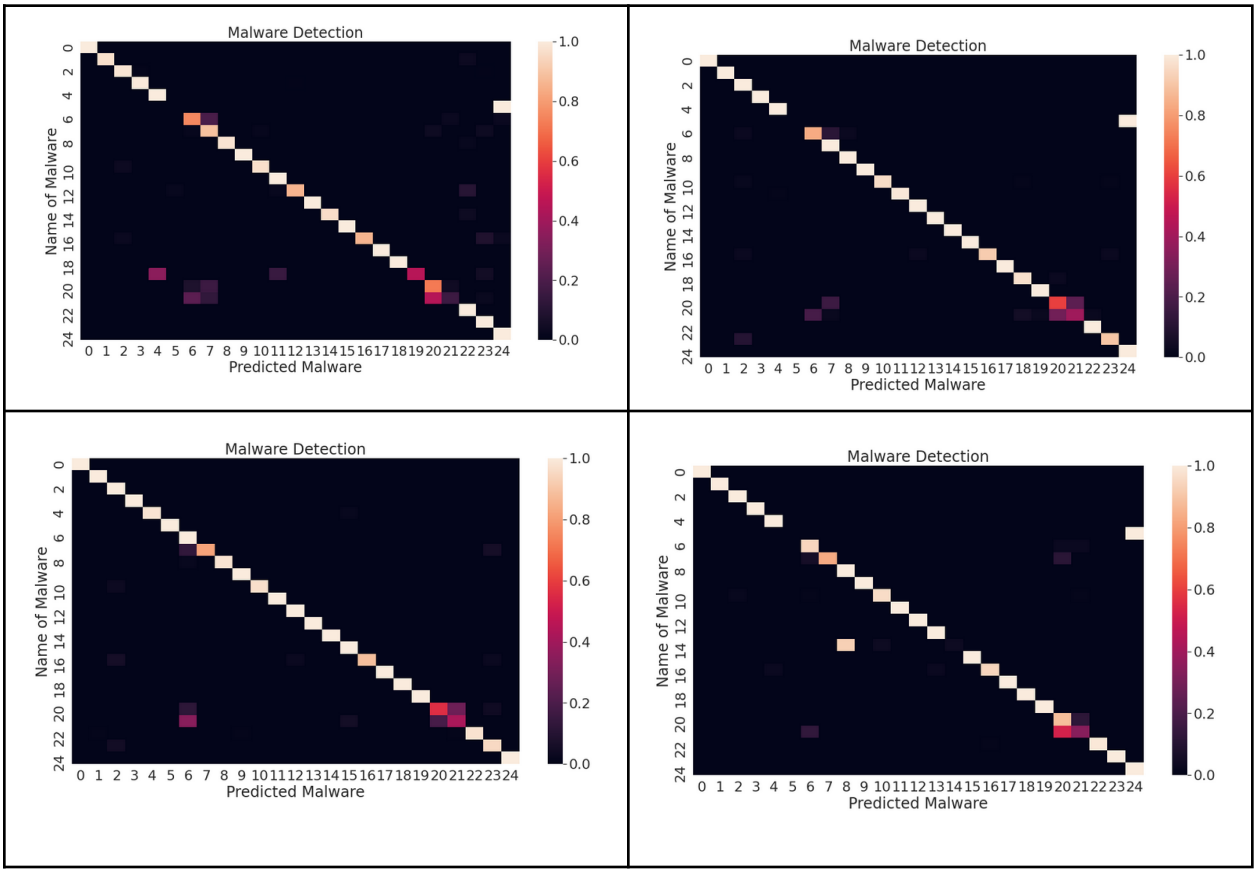
Each of the models was analyzed based on the accuracy score and f1 score. The accuracy score determines the correct predictions of the model whereas the F1 scores determine the overall precision and recall for predictions by the model.

	Model_Name	Accuracy	F1_Score
0	Vgg16	0.952815	0.945992
1	InceptionResNet	0.972244	0.967116
2	VGG19	0.339017	0.171667
3	DenseNet201	0.977002	0.976410
4	Xception	0.963521	0.955822

Analysing on the test set

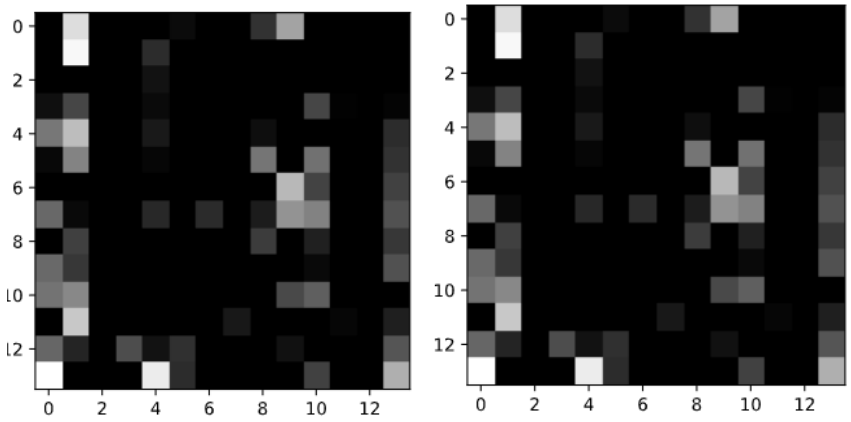






4.Generating AdComparing the Convoluted Images Processed by the Model

The CNN model generates 12 Convoluted images identifying the most important part of the Images and using that to look for similarities and dissimilarities. Generated convoluted images of similar family malwares pointed out few extremely similar patterns whereas in case of different families the generated patterns were quite dissimilar comparatively.



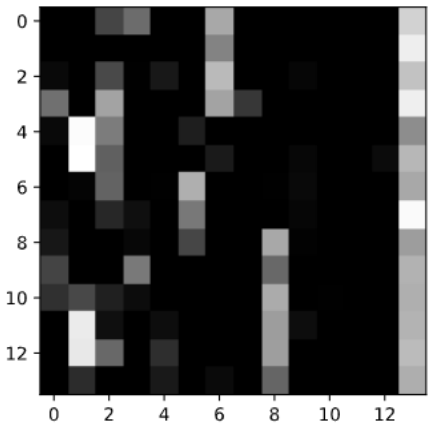
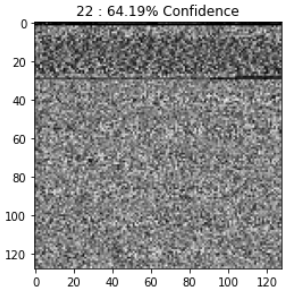


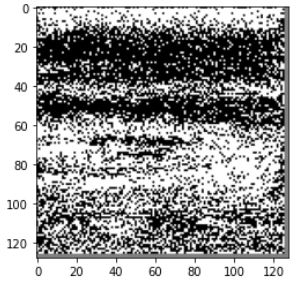
Fig 2. (a) Convoluted Processed Image for Yuner Class (b) Convoluted Processed Image for a different Malware from Yuner Class (Extremely similar patterns to Fig 2) (c) Convoluted Processed Image for a Malware from Adialer Class(Completely new patterns are observed)

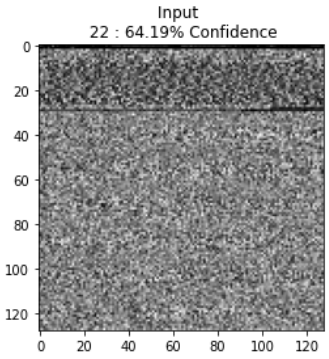
5.Generating Adversarial Examples

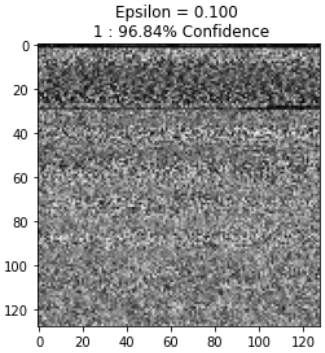
Using the FGSM (Fast Gradient Sign Method) technique a perturbation will be trained on a custom generated CNN network to create an adversarial example by adding this to the original image and an attack will be performed on the original architectures.

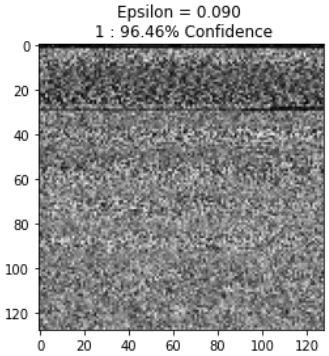
$$Adversarial\ Image\ =\ Original\ Image\ +\ sign(\nabla x\ J(\theta, x, y))$$

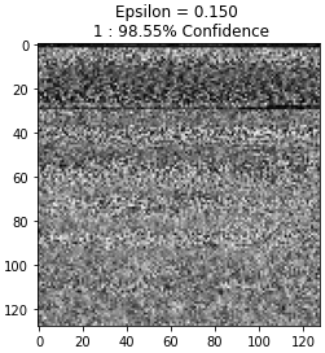




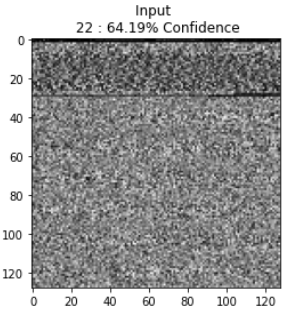

Input
22 : 64.19% Confidence

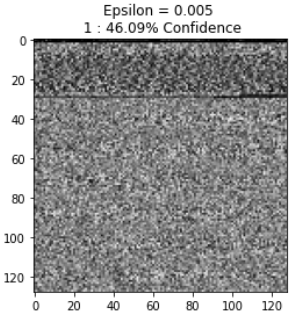

Epsilon = 0.100
1 : 96.84% Confidence


Epsilon = 0.090
1 : 96.46% Confidence

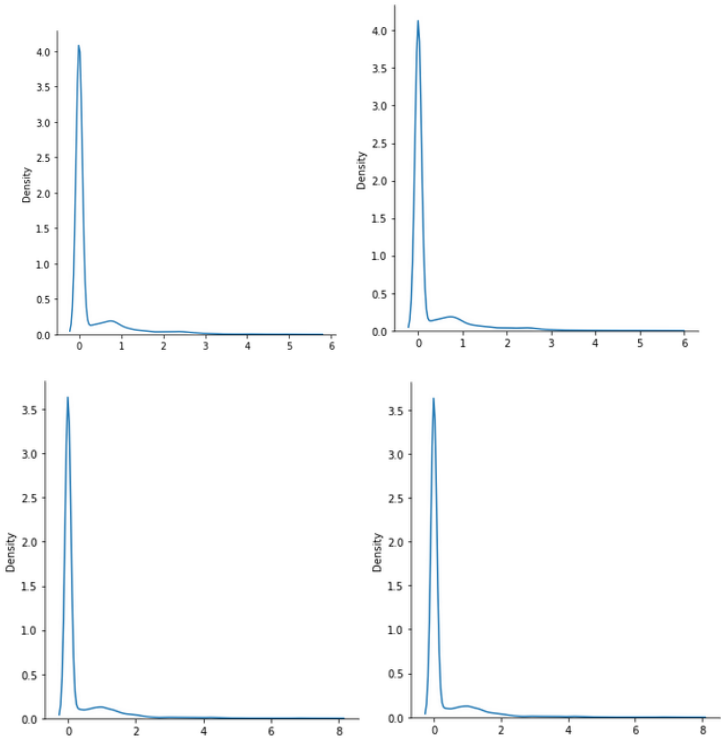

Epsilon = 0.150
1 : 98.55% Confidence

6.**Attack**


Input
22 : 64.19% Confidence


Epsilon = 0.005
1 : 46.09% Confidence

7. **Analysis**



Experimental Results and Discussion

The reason to convert the hexadecimal code of a Malware to Images was, if the attacker tries to make changes to a certain malware hex values converting that malware to image and using a CNN over it will still be able to tell us about the family of malware with a certain accuracy. And the model succeeded in doing so. We changed certain lines of the bytecode by deleting certain lines, modifying individual values and were still able to classify the malware into its family accurately. This method is also extremely fast. This methodology can be very useful in classifying a new malware and then building a solution for it.

Results

Different transfer learning models like VGG16, DenseNet, XceptionNet was not able to outperform InceptionResNet and DenseNet Neural Nets.

Model_Name	Accuracy	F1_Score
Vgg16	0.952815	0.945992
InceptionResNet	0.972244	0.967116
VGG19	0.339017	0.171667
DenseNet201	0.977002	0.976410
Xception	0.963521	0.955822

Table 1. Various Models and their accuracies over the dataset

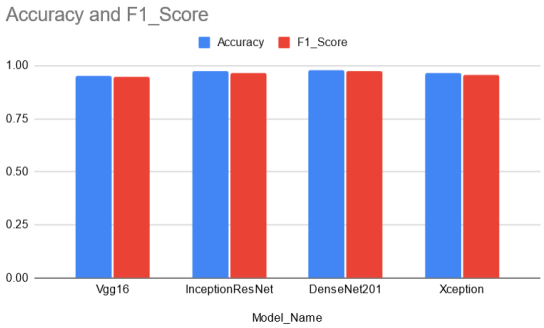


Fig 1. Accuracy and F1 score

In the real-world test scenario, our model successfully discriminated between types of malware and a harmless file. The identification and classification was also done reasonably quick, about 0.25 seconds in our testing. Therefore, this solution could be deployed without having to worry about massive latencies in file transfer.

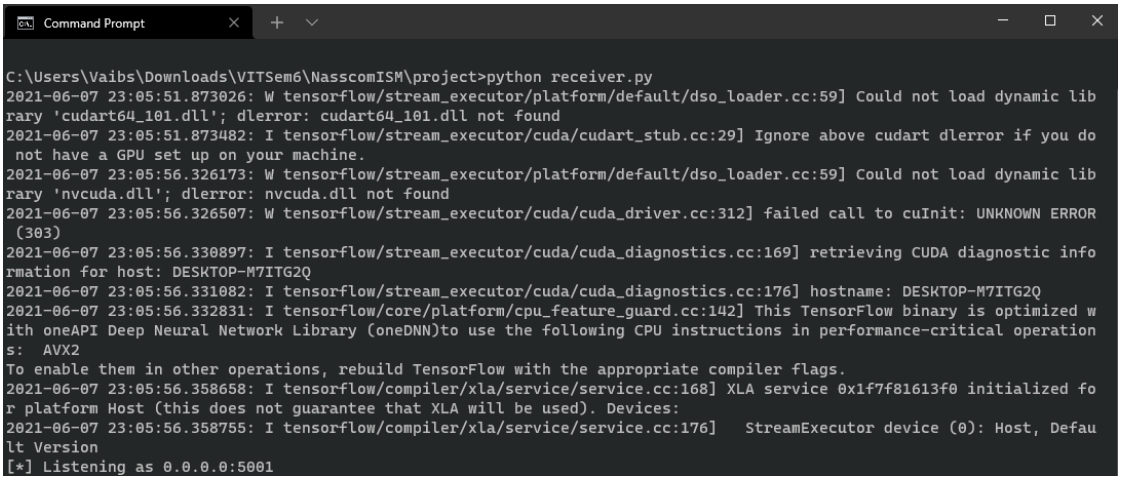


Fig 2. The client PC is listening to any connections on port 5001

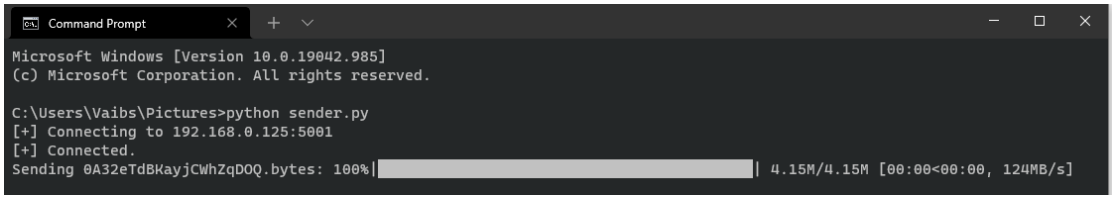


Fig 3. Attacker sends file with malware to the client

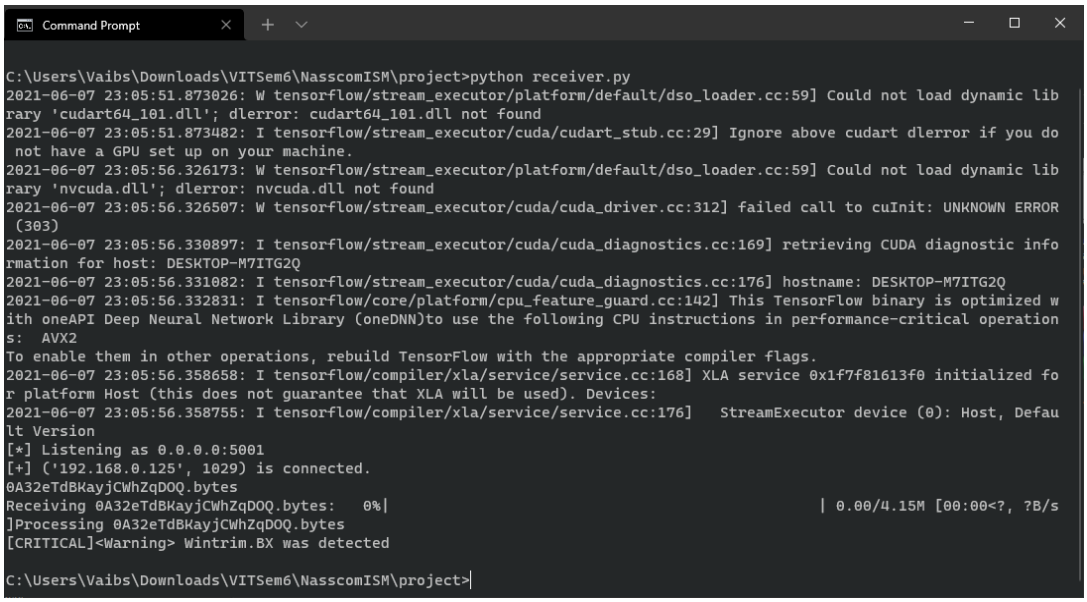


Fig 4. Attacker sends file with malware to the client

```
[*] Listening as 0.0.0.0:5001
[+] ('192.168.0.125', 4229) is connected.
0A32eTdBKayjCWhZqDOQ.bytes
Receiving 0A32eTdBKayjCWhZqDOQ.bytes:   0%|                                                    | 0.00/4.15M [00:00<?, ?B/s]P
rocessing 0A32eTdBKayjCWhZqDOQ.bytes
--- Time taken: 0.24236202239990234 seconds ---
[CRITICAL]<Warning> Wintrim.BX was detected
```

Fig 5. Time taken by model to perform classification

```
Command Prompt

C:\Users\Vaibs\Downloads\VITSem6\NasscomISM\project>python receiver.py
2021-06-07 23:22:19.828414: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic
 library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found
2021-06-07 23:22:19.828579: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if yo
u do not have a GPU set up on your machine.
2021-06-07 23:22:22.314386: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic
 library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2021-06-07 23:22:22.314647: W tensorflow/stream_executor/cuda/cuda_driver.cc:312] failed call to cuInit: UNKNOWN E
RROR (303)
2021-06-07 23:22:22.319566: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic
information for host: DESKTOP-M7ITG2Q
2021-06-07 23:22:22.319847: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-M7ITG2Q
2021-06-07 23:22:22.320414: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimiz
ed with oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-critical o
perations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-06-07 23:22:22.329062: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x17f8cd17f90 initialize
d for platform Host (this does not guarantee that XLA will be used). Devices:
2021-06-07 23:22:22.329238: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, D
efault Version
[*] Listening as 0.0.0.0:5001
[+] ('192.168.0.125', 8673) is connected.
qrcode.png
Receiving qrcode.png:   0%|                                                          | 0.00/3.37k [00:00<?, ?B
/s]
C:\Users\Vaibs\Downloads\VITSem6\NasscomISM\project>
```

Fig 6. Results with non-malicious file

Conclusion

We were able to successfully implement this technique to identify a new and modified bytecode into a class of available Malware. The accuracy achieved using our CNN was found to be better than a few transfer learning models and the implement method is quite fast in identifying the Malware. Along with this the paper also discusses the effect of adversarial attack using FGSM technique and the effect the perturbation on the image and the activations from the CNN.

References

[1] Yamjala Supriya ,Gautam Kumar and Dammu Sowjanya “Malware Detection Techniques: A Survey “2020 Sixth International Conference on PDGC(2020)

[2] Ni, Sang, Quan Qian, and Rui Zhang. "Malware identification using visualization images and deep learning." Computers & Security 77 (2018): 871-885.

[3] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," *2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, 2013, pp. 40-44, doi: 10.1109/CICYBS.2013.6597204.

[4] M. Kumari, G. Hsieh and C. A. Okonkwo, "Deep Learning Approach to Malware Multi-class Classification Using Image Processing Techniques," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), 2017, pp. 13-18, doi: 10.1109/CSCI.2017.3.

[5] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," in IEEE Access, vol. 7, pp. 46717-46738, 2019, doi: 10.1109/ACCESS.2019.2906934.

[6] Drew, J., Hahsler, M. & Moore, T. Polymorphic malware detection using sequence classification methods and ensembles. EURASIP J. on Info. Security 2017, 2 (2017). https://doi.org/10.1186/s13635-017-0055-6

[7] William Hardy, Lingwei Chen, Shifu Hou, Yanfang Ye*, and Xin Li, ”DL4MD: A Deep Learning Framework for Intelligent Malware Detection”, Int'l Conf. Data Mining, 1-60132-431-6, 2016

[8] Sang Ni, Quan Qian, Rui Zhang,Malware identification using visualization images and deep learning,Computers & Security,Volume 77,2018

[9] Anuar, Noor & Mas'ud, Mohd & Bahaman, Nazrulazhar & Ariff, Nor. (2020). Analysis of Machine Learning Classifier in Android Malware Detection Through Opcode. 7-11. 10.1109/AINS50155.2020.9315060.