Adversarial Attack on CNN Based Malware Detection Systems

Report By: -

Akshitha Koganti

Nitesh Bhutani

Rohit Kulkarni

Sarath Patlolla

Abstract

A lot of research has been done on crafting images on CNN based on image classification system, but a little to a few has been done on CNN based on Malware detection systems Many researchers have proposed the concept of constructing malware detection systems using deep neural networks. But, these deep neural networks are vulnerable to adversarial attacks, intentionally crafted on clean images for misclassification by adding perturbations on few pixels. In this work we are expand on the existing adversarial examples in modelling algorithms that construct an efficient attack that would be tested against the CNN based malware detection system. In our work, we trained our 2-layer neural network with Malimg dataset.

1.Introduction

1.1 Introduction

One of the important aspect of the computer science is Cybersecurity. Now, a lot of commercial and open source software is available online and the authors of these software are continuously working to improve the security of there applications, and the organizations are hardening their systems by deploying several threat detection systems, and thus attackers are forced to develop attack which are more sophisticated, so that the attacks are successful in attacking a computer or penetrating an organization network. The primary defence against these attacks is to develop a commercially available and strong anti-malware software.

Researchers have proposed to use a deep learning model for the classification of malwares, which is a key component for the developing next generation anti-malware systems. Research is being started in the analysing the attack and defences of the machine learning based classification systems, and this area is called is adversarial learning. In adversarial-based attack, the noise or the perturbations are created intentionally, to fool the deep learning model.

1.2 Adversarial Image

Adversarial images are images carefully crafted by adding some noise to clean image and are given inputs to the deep-learning models to cause the model to misclassify the image. Adversarial images are in general like an optical illusion to the machine learning model. The day to day life of the applications can be serious- for instance, the autonomous driving vehicle could gets confused, if the traffic sign is modified.

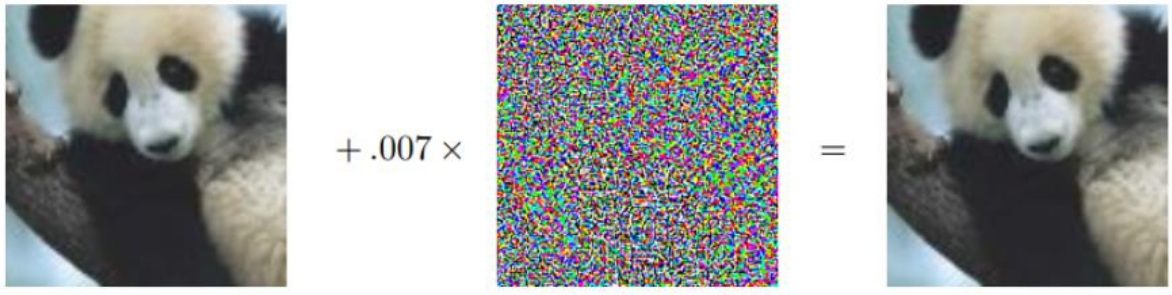

Figure 1: Adversarial image generation

2. Methodology

2.1 Deep Learning Framework

The deep learning model was built using Google Tensorflow, for tensor computations along with pythons scientific computing libraries numpy, and scikit-learn, the output images were plot using matplotlib.

2.2 Dataset

The Deplaning model of our study was evaluated on the Malimg dataset, which had 25 malware families holding 9339 malware images. The Malimg dataset was developed by Natarej et al. by reading the malware binaries into a vector of 8-bit unsigned integer, and later this vector was arranged into a 2D array. These images are visualized as grayscale image in a range of 2 to 255 with 0 as black and 255 to white.

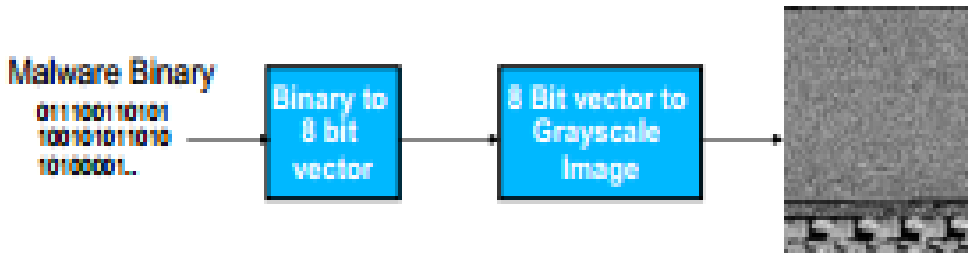

Figure 2: Visualizing a malware image on a grayscale

2.3 Data Pre-processing

The distribution of the malware images in the Malimg dataset is uneven. Some classes of the malware have a high number of malware binary images whereas some of malware classes have less number of images. The various families of malware found in the Malimg dataset have been shown in table1.

Due to the uneven distribution of the images in the dataset, data pre-processing is carried out on the data and the dataset was split into training data and the testing data in the ratio of 4:1.

No.	Family	Family Name	No. of Variants
01	Dialer	Adialer.C	122
02	Backdoor	Agent.FYI	116
03	Worm	Allaple.A	2949
04	Worm	Allaple.L	1591
05	Trojan	Alueron.gen!J	198
06	Worm:AutoIT	Autorun.K	106
07	Trojan	C2Lop.P	146
08	Trojan	C2Lop.gen!G	200
09	Dialer	Dialplatform.B	177
10	Trojan Downloader	Dontovo.A	162
11	Rogue	Fakerean	381
12	Dialer	Instantaccess	431
13	PWS	Lolyda.AA 1	213
14	PWS	Lolyda.AA 2	184
15	PWS	Lolyda.AA 3	123
16	PWS	Lolyda.AT	159
17	Trojan	Malex.gen!J	136
18	Trojan Downloader	Obfuscator.AD	142
19	Backdoor	Rbot!gen	158
20	Trojan	Skintrim.N	80
21	Trojan Downloader	Swizzor.gen!E	128
22	Trojan Downloader	Swizzor.gen!I	132
23	Worm	VB.AT	408
24	Trojan Downloader	Wintrim.BX	97
25	Worm	Yuner.A	800

Table1: Malware families in the Malimg dataset

2.4 Deep Learning Model

Convolutional Neural Network (CNN) is used as the model which is trained on the Malimg dataset. A CNN is similar to a feed forward neural network, consisting of hidden neurons with learnable weights and biases. The neurons in these layers receive input and performs a dot product with the weights of neurons in that layer and the computed production is passed through an activation function. Then maxpooling is applied on the result feature map to prevent overfitting of the model, and then we have a fully connected layer where all the neurons in the previous layer are connected to all the neurons of the next layer.

The CNN network in our model receives a grayscale image of size 32*32. In the first layer 36 filters of size 5*5 are used with stride of 1. Non-linearity function(relu) is applied on the result activation map of the first layer. A max-pooling is applied on the resultant feature map. Now, in the second layer64 filters of size 5*5 are used followed by an activation and max-pooling. In the final layer we have a fully connected layer. In the output layer the classifier used is Softmax classifier.

2.5 Adversarial image Generation

The different methods which are used in this work to generate the adversarial images are

- Iterative FGSM
- ATN
- JSMA
- Carlini and Wagner attack

2.5.1 Iterative Fast Gradient Sign Method (I-FGSM)

This method the adversarial image is generated by adding a pixel-wide noise in the direction of the gradient. In the FGSM a perturbation is added with every step, whereas the I-FGSM the perturbations are added in T gradient steps of magnitude.

$$x_0^{adv} = x, \quad x_{t+1}^{adv} = x_t^{adv} + \alpha \cdot \mathtt{sign}\left(\nabla_x J(x_t^{adv}, y)\right).$$

Where,

X is clean image

Y is correct

Sign is the signum function

X^{adv} is the adversarial image generated in iteration step "t"

$\alpha = €/t$ where € is a hyperparameter

2.5.2 Adversarial Transformation Network

Adversarial Transformation Network (ATN) is a specially trained network which generates an adversarial image to attack another fully trained network. ATN are capable for both targeted and non-targeted attacks. The ATN are trained to generate adversarial examples that minimally modify the classifier's outputs given the original input, while constraining the new classification to match an adversarial target class. In ATN the adversarial images can be generated by perturbation the input or by adversarial encoding.

The ATN for our model has 9 convolution layers with a relu(Rectified Linear Unit) activation function. The Kernel size for the ATN network is 3*3 with 48 filters of stride 1. The number of hidden neurons in the flatten layer are 1024.

2.5.3 Jacobian-based Saliency Map Approach (JSMA)

This is an iterative method for targeted misclassification. The adversarial perturbation in a clean image is generated by exploiting the forward derivative of a neural network which can be used as a targeted misclassification in the network into a specific class.

$$S(X,t)[i] = \begin{cases} 0, \text{ if } \frac{\partial F_t(X)}{\partial X_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} > 0 \\ (\frac{\partial F_t(X)}{\partial X_i}) | \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} |, \text{ otherwise} \end{cases}$$

where,

X is the input image

F is the neural network

J is output class

t= argmax_j $F_j(X)$

In order for the target misclassification of the image Ft(X) must be increased while the probabilities Fj (X) of all other classes $j \neq t$ decrease, until t = argmaxj Fj (X). This is accomplished by exploiting the adversarial saliency map. Starting with a normal sample x, we locate the pair of features {i, j} that maximize S(X, t)[i] + S(X, t)[j], and perturb each feature by a constant offset , €.

2.5.4 Carlini and Wagner (C &W)

This C&W attack is an optimization problem. The C&W attack is implemented s to minimize a loss that comprises two parts:

a) the p-norm distance between the original image and the adversarial image,

b) a term that encourages the incorrect classification of the adversarial images.

The different metric distances which we would be using in the C& W attack are

1) L0 norm: it is the pixel wide distance between the clean image and the adversarial image

2) L2 norm: it is the Euclidean distance between a clean image and an adversarial image:

3) $L\infty$: It is the maximum distance among all the distances among all the pixel differences.

The main objective of the adversarial image generated by Carlini and Wagner is to minimize the pixel wide distances between the images so that misclassification among the images takes place. Mathematically, it is written as:

$$\text{minimize } D(x, x + \delta)$$

$$\text{such that } C(x + \delta) = t$$

3. Results and Discussions

3.1 Training loss and accuracy

During the training of the Malimg dataset on the CNN network. Initially the training loss was very high, and the accuracy of the network was low. With increase in the number of iterations of the loss decreased and the accuracy increased. The dataset was trained for 50 iterations for 4 epochs. Graphs are plotted for training loss and training accuracy against the number of iterations.

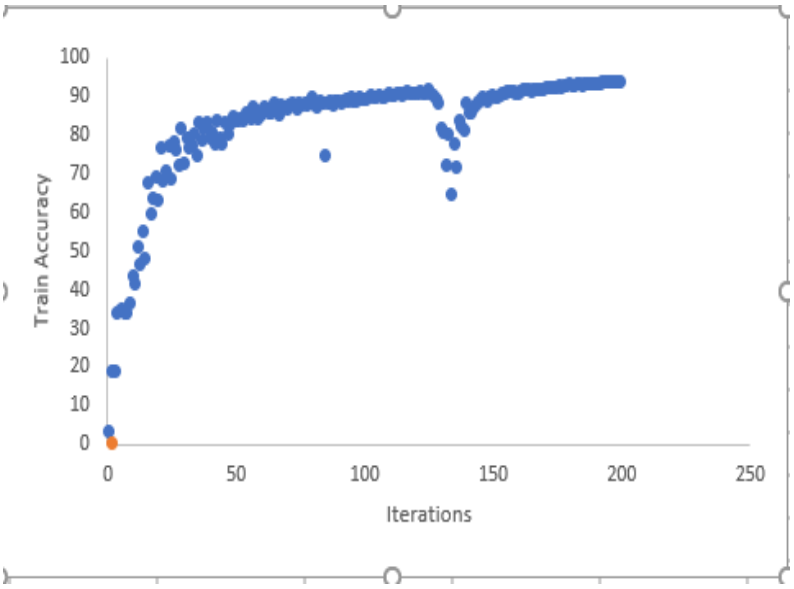

Figure4: training accuracy and iterations

Figure5: training loss and iterations

3.2 Adversarial Attack

Attacking the pretrained model with an adversarial image is called an adversarial attack. Here, we would be discussing the effect on the accuracy when the adversarial image is generated by the different methods we have implemented

3.2.1 I-FGSM

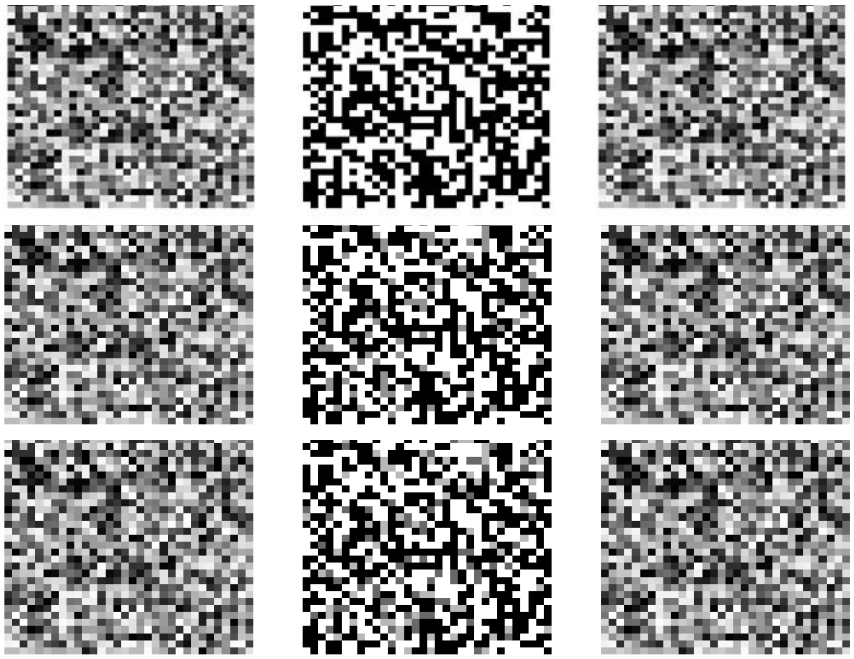

Figure 6: adversarial images generated by I-FGSM

The first column in the images are the clean images, the second column is the perturbations which are needed to be added to the clean images to create adversarial images. The Final column in the images are the adversarial images which are sent to the deep learning model to attack the network. After the adversarial images are generated, the image is sent to the network to attack the trained network. The clean images in the above figure belong to the class 3, now after adding the perturbations the images are misclassified to label 12. The value of epsilon (€) used is 1.7

3.2.2 ATN

In the ATN the adversarial images which are generated are sent directly to attack the pretrained network. The various hyperparameters which are used for ATN are:

- $\alpha = 1.5$
- $\beta = 0.01$

- Learning rate: = 0.0001
- Batch Size: = 20
- Iterations: = 1600
- Epoch = 4

```
Original accuracy: 0.74042
Attacked accuracy: 0.07146
```

Figure 7: the accuracies

The variation in the accuracy when the clean image and the adversarial image is sent to the network is shown in the above figure. The adversarial image is classified to the correct label with a very low accuracy.

3.2.3 JSMA

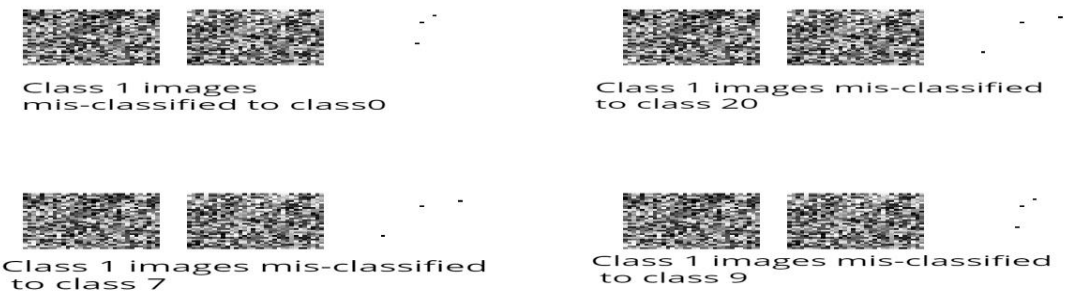

Figure 8: adversarial images from JSMA

In the above figure we have shown the clean image, adversarial image and the perturbations added to the clean image to generate the adversarial image. Since in JSMA only the pixels which have the high intensity are only perturbed, thus the pixels which are used for perturbing the clean image are only 3, After perturbations the images are misclassified. In the images the original class images are misclassified to other classes

The hyper parameters which are used in JSMA are:

$\Theta = 0.01$

$\acute{\Gamma} = 0.75$

Success Rate = 10%

3.2.3 Carlini and Wagner Attack

In the C&W attack the main objective was to reduce the metric distance between the clean image and the adversarial image when added the perturbation iteratively.

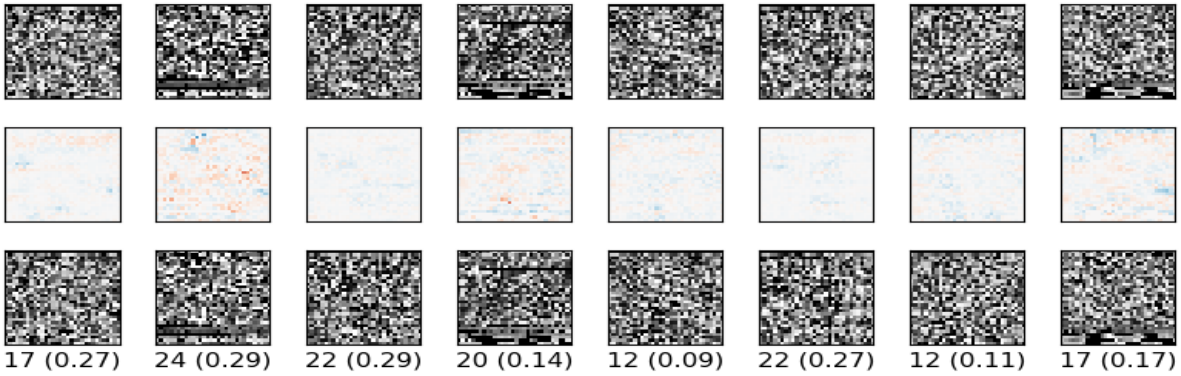

Figure 9 Adversarial images used for C&W attack

The above image shows the clean image, the perturbations to be added and the adversarial image which has been generated and sent to the model for attacking the model. The number in the images shows the classes to which they have been misclassified and with the percentage accuracy.

4. Conclusions and Future Work

We tested 4 attacks on the CNN based malware detection system and were successfully able to decrease the accuracy of the detection system in each case.

- To add benchmark for further analysis of attacks on such systems following things are required:
    - Dataset with more examples and equal distribution of malware classes is needed.
    - Generating malware binaries from images and test their maliciousness

5. References

1. https://www.tensorflow.org/

2. Shumeet Baluja and Ian fisher, “Adversarial Transformation Networks: Learning to Generate Adversarial Examples”

3. Abien Fred M.Agarp, Francis John Hill Pepito, “Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach using Support Vector Machine (SVM) for Malware Classification.”

4. Alexey Kurakin, Ian J.Goodfellow, Samy Bengio, “ Adverserial Machine Learning at Scale”

5. L.Nataraj, G.Jacob, B.S Manjnath, “Malware Images: Visualization and Automatic Classification”

6. https://github.com/gongzhitaao/tensorflow-adversarial

7. https://github.com/RanTaimu/Adversarial-Transformation-Network

8. https://github.com/tensorflow/cleverhans