ML for Cybersecurity final project

Dhayarkar, Rishikesh
rbd291@nyu.edu
N19706837

Subramanian Suresh, Anirudh
ass721@nyu.edu
N17726854

Limaye, Sagarika
sl7341@nyu.edu
N15838975

1 Introduction

This project aims to develop an approach to detect backdoors in a given deep neural network that trained on the YouTube Face dataset. We propose a solution that heavily uses the strategy utilized in the paper "STRIP:A Defence Against Trojan Attacks on Deep Neural Networks". We intentionally perturb the incoming input, by superimposing various image patterns and observe the randomness of the predicted classes for perturbed inputs from a deployed model. The inputs to our algorithm include the bad net, clean validation/test set, and poisoned set. We noticed that for clean perturbed images the frequency of occurrence of labels is lower and more uniformly distributed where as for poisoned perturbed images the frequency is higher and the distribution of labels is less uniform. This is the key idea of our proposed solution.
Github Link - `https://github.com/RishikeshDhayarkar/ML-security`

2 Detailed approach

Our approach is split into two parts. The first part involves finding a threshold value(frequency of occurrence of a label) to distinguish the poisoned and clean samples. The second part involves using the threshold value found in part 1 to classify the poisoned samples into the $(N+1)^{th}$ class.

The inspiration for our approach was the "STRIP" paper. The perturbation technique used in our approach is the same as the one mentioned in this paper. The authors of the paper use "entropy" to find a pattern to differentiate between poisoned and clean images. We performed experiments using entropy to classify clean and poisoned images but the Trojan detection accuracy was in the low 60 percentages with very high FAR and FRR values. Our best guess to explain poor results with entropy is that we have 1283 classes in our case as compared to 10 in the original paper. We suspect that due to the large number of resulting probabilities in our case Entropy is unable to give our good results.

We also used an unsupervised technique called k-means to separate images into poisoned and clean classes. k-means gave us very good results when the ratio of poisoned and clean images was greater than 1:10, but below this ratio we saw a steep decline in detection accuracy. A potential reason for this could be high sensitivity of k-means to outliers, especially in edge cases where there are very few poisoned images in the data set. We found that generating a threshold using percentile was far more effective than the above two approaches. This method is explained further in the following section.

2.1 Part 1

In this part we first generate perturbed images on the clean validation set. The procedure for generating these images includes superimposing one clean image with a randomly picked cleaned image. The weights(0.5) for superimposition are equal for both the images. We generate 50 perturbation for every clean image. Next we pass these perturbed images through the bad net and generate labels for each of these images.

Once all the labels are obtained we group the labels into batches of 50 and each batch of 50 labels represents the labels for perturbed images that correspond to a single clean image. For each such group we find the frequency of occurrence of labels and pick out the label with maximum frequency. At this point, for every clean image there is a number(maximum frequency) corresponding to it. We get the 96th percentile of this collection of numbers and use it as a threshold for part 2.

2.2 Part 2

The initial section of this part is quite similar to the part1. Instead of generating perturbed images for just the clean images we do this for both poisoned and clean images. Similar to the procedure of part 1 we generate a number(maximum frequency) for every single input image. We noticed that this number(maximum frequency) is lower for clean images and higher for poisoned images. To define a boundary of separation between the clean and poisoned images we use the threshold value from part 1. A frequency value higher than the threshold value corresponds to an image that is poisoned and value lower than the threshold value corresponds to a clean image. At this stage we have successfully separated the clean and poisoned images. All images that fall under the category of poisoned were given the label 1283, which is the $(N+1)^{th}$ class and clean images were passed through the model to generate labels from 0 to 1282.

Result Table

Models	Ratio (clean:poison)	Poisoned Dataset	FRR(%)	FAR(%)	Trojan Detection Accuracy(%)	Testing Accuracy(%)
Anonymous Badnet-1	1:1	Sunglasses	11.20	3.50	92.65	90.95
	1:2		11.60	3.40	91.13	90.06
	2:1		9.40	4.20	94.06	91.80
Anonymous Badnet-2	1:1	Sunglasses	11.00	2.80	93.10	91.05
	1:2		11.20	4.40	91.07	89.60
	2:1		9.00	5.00	93.07	91.00
Multi-Trigger Multi-Target	1:1	Eyebrows	24.80	4.00	85.60	83.70
	1:2		28.00	3.80	80.06	78.80
	2:1		27.00	3.60	88.60	86.06
Multi-Trigger Multi-Target	1:1	Lipstick	11.70	3.60	92.35	90.45
	1:2		12.20	2.80	90.93	89.66
	2:1		10.40	4.10	93.80	91.26
Multi-Trigger Multi-Target	1:1	Sunglasses	0.20	4.30	97.75	95.85
	1:2		0.50	3.20	98.60	97.34
	2:1		0.40	3.90	97.27	94.80
Sunglasses BadNet	1:1	Sunglasses	6.00	3.80	95.10	93.67
	1:2		3.40	7.00	94.2	93.67
	2:1		3.10	7.19	95.53	94.00

3 Results

We use two primary metrics to evaluate our results, False Rejection Rate(FRR) and False Acceptance Rate(FAR). The FRR is the probability when the clean input is regarded as a poisoned input by our algorithm. The FAR is the probability that the poisoned input is recognized as the clean input. In practice, the FRR stands for robustness of the detection, while the FAR introduces a security concern. Ideally, both FRR and FAR should be 0%. This condition may not be always possible in reality. Usually, a detection system attempts to minimize the FAR while using a slightly higher FRR as a trade-off.

Secondary metrics include Trojan Detection Accuracy which tells us how well we can differentiate between a Trojan and benign image, and Testing Accuracy which tells us how well we were able to classify each image into it's respective class.

Our program was tested The results on all available BadNets with different ratios of clean and poisoned data and we obtained generally acceptable results using our method. Results vary by very small amount in every run because of the randomness involved in generating perturbations of images. We were able to achieve a good trade-off between FAR and FRR for a threshold(percentile) value of 96. As we vary percentile from 94 to 99, FRR increases and FAR decreases.

4 Conclusion

Our attempts to replicate the STRIP paper was generally successful. We determined that using Entropy as a classifier cannot yield good accuracy when we have a large number of labels.

Using the 96^{th} percentile gave us largely good results. The best results were for the Multi-Trigger Multi-Target BadNet on the Sunglasses poisoned data while the worst results were for the Multi-Trigger Multi-Target BadNet on the Eyebrows poisoned data with an average Trojan Detection accuracy of 97.84% and 84.75% respectively.

Our approach includes generating 50 perturbations for every clean and poisoned image. This makes our approach RAM-intensive. Our experiments were performed on Google Colab and we found that using about 1000 clean and 1000 poison images consumed most of our available RAM. We recommend using a maximum of 1500 clean and 1500 poison images to run our implementation.