Final Project Report

Haotian Ma

1. Introduction

   I designed a backdoor detector for BadNets trained on the YouTube Face dataset. This detector takes one image and a BadNet model name as parameters, then it will output the class of that image, which is between [1, N+1]. N represents the total classes of this model, if that image is poisoned the detector will return N+1. For this project, N=1283.

2. Description

   Only given a backdoored neural network classifier *B* and a validation dataset of clean, labelled images *Dvalid*, I need to modify *B* and generate a "repaired" model *G*, which can detect poisoned inputs successfully and predict clean inputs correctly at the same time.

   The main logic is in the method *do_repair()* in file *repair.py*, there are three steps to repair *B*. All accuracy below is on the validation dataset *Dvalid*.

   a) Pruning

      According to the idea "Later convolutional layers in a DNN sparsely encode the features learned in earlier layers, so pruning neurons in the later layers has a larger impact on the behavior of the network." [1], I prune the layer conv_3. I set an accuracy threshold, 95% of the accuracy before pruning, to stop pruning, and I don't need to worry about the lost 5% accuracy because the next two steps can make up for it or even improve it.

   b) Retraining

      At this step, I retrain the model B with data *Dvalid*, but a little difference to normal retraining, only update the weights of the last fully-connect layer. After this step, I can get a favorable trade-off between the accuracy and the back-door success.

c) Fine-tuning

To deal with the pruning-aware attack, I use this last step to defend while retaining previous pruning benefits. Because in former step I prune the layer conv_3, here I only update the weights of layers after conv_3 with a very small learning rate. I set 6 rounds to fine-tune, each has 5 epochs, and stop when the accuracy reaches the original accuracy before pruning or it uses up all 6 rounds.

Then, I get a "repaired" model *G*. So far this *G* can only predict clean inputs correctly but not detect whether an input is backdoored. I spent a lot of time for searching how to detect backdoored inputs only with access to a backdoored neural network classifier and a small clean validation dataset. There are some good methods such as Neural Cleanse [2] and ABS [3]. But they are a little hard and complex for me to implement. Finally, I figured out a simple way, only for this project, to do detecting. We want to recognize whether an input is backdoored and the BadNet *B* can just tell us. Because the BadNet *B* can output a wrong class for the backdoored input, we can use this wrong class to compare with the correct class outputting by the "repaired" model *G*. According to previous three steps, *G* has significantly mitigated the impact of backdoors and can predict in a very high accuracy. So, there are reasons to think that the backdoor detecting is reliable in most circumstances.

3. How to run the code

1) Download BadNet models and the clean validation dateset and store them under *models/* and *data/* directory respectively.

2) To generate a "repaired" model, execute *repair.py* by running:

python repair.py <badnet model directory> <clean validation data directory>

"Repaired" model *G* will be saved as *models/<badnet model name>_defence.h5*

E.g.,

python repair.py models/anonymous_1_bd_net.h5 data/clean_validation_data.h5

Saved model: *models/anonymous_1_bd_net_defence.h5.*

3) To evaluate, execute *eval_defence.py* by running:

python eval_defence.py <image directory> <badnet model name>

It will print the class number in [1, N+1].

E.g.,

python eval_defence.py img/img_9.png anonymous_1_bd_net

Output: 855

4. Related resources

Github repo: https://github.com/harveymht/final_project

References

[1] K. Liu, B. Doan-Gavitt, and S. Garg. Fine-Pruning: Defending Against Backdoor Attacks on Deep Neural Networks. In Proc. RAID, 2018.

[2] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B.Y. Zhao. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In Proc. IEEE Symposium on Security and Privacy, 2019.

[3] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 1265–1282, 2019.