Backdoor Attacks Report

Name: Avani Vaishnav
NetID: av3141

Github link: https://github.com/avani1998/ECE-9163-ML-for-cybersecurity

Introduction

BadNets or backdoored networks are malicious networks that have a state-of-teh art performance on the clean training and validation set but behave badly on the attack's chosen training validation samples. In this assignment we use pruning defense on a maliciously trained model to prune nodes that are only activated when malicious data is input into the network.

Methodology

The general idea is to prune the neural network and compare its performance with the original network to detect any discrepancies caused by the presence of a backdoor. We recall that backdoors activated unused/spare neurons in the network.

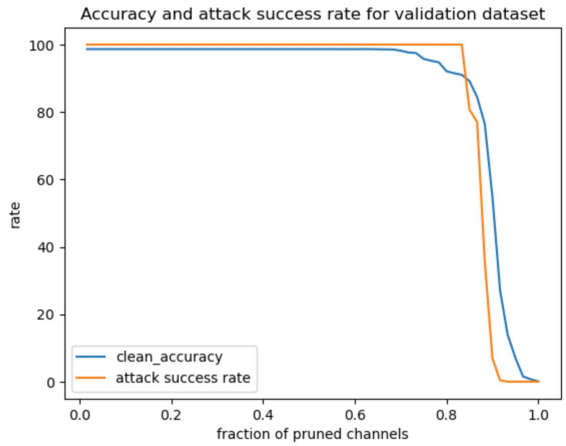
The pruning defense works as follows: the defender exercises the DNN received from the attacker with clean inputs from the validation dataset, Dvalid, and records the average activation of each neuron. The defender then iteratively prunes neurons from the DNN in increasing order of average activations and records the accuracy of the pruned network in each iteration. The defense terminates when the accuracy on the validation dataset drops below a pre-determined threshold.

We prune neurons from the 'pool_3' layer, before the `FC` layers. We use the **weights pruning** method where pruning is performed by setting the weights and bias of that channel to 0.

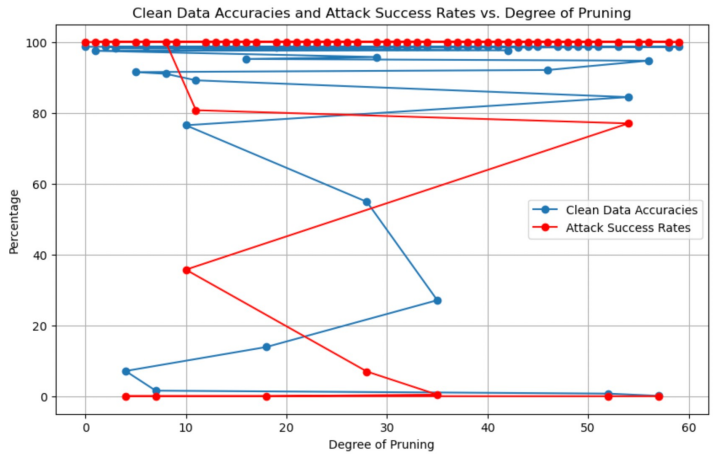
Observations

As per the instructions we need to save the model when the accuracy drops below X% ={2,4,10}. The models saved for these accuracy drops are ```model_X=2.h5```,```model_X=4.h5``` and ```model_X=10.h5``` respectively.

The accuracies of the model on clean data and the attack success rates of the model on malicious data are recorded. The accuracy on clean test data and the attack success rate (on backdoored test data) as a function of the fraction of channels pruned (X) is clearly visualized in this graph.



Clean data accuracies and attack success rates versus degree of pruning can be visualized in this graph.



We then combine the pruned model and the BadNet B to create a GoodNet G. The results of the combination on the same data are as follows:

```
Goodnet with 10% drop in accuracy has accuracy 84.3335931410756 and attack success rate 77.20966484801247

Goodnet with 4% drop in accuracy has accuracy 95.74434918160561 and attack success rate 100.0

Goodnet with 2% drop in accuracy has accuracy 92.1278254091972 and attack success rate 99.98441153546376
```

We observe that with an accuracy drop on 10% from the original accuracy of the network, the attack success rate is still quite high (77.21%).

Table of accuracies and attack success rate as a function of pruned indexes:

	Pruned Channel Index	Clean Data Accuracy	Attack Success Rate
0	0	98.649000	100.000000
1	26	98.649000	100.000000
2	27	98.649000	100.000000
3	30	98.649000	100.000000
4	31	98.649000	100.000000
5	33	98.649000	100.000000
6	34	98.649000	100.000000
7	36	98.649000	100.000000
8	37	98.649000	100.000000
9	38	98.649000	100.000000

Prerequisites

Install the below packages if you are running this code on a Mac Silicon chip.
create a conda environment
conda create -y --name cv
conda activate cv
install specified versions of libraries
conda install -y -c apple tensorflow-deps==2.10.0
python -m pip install tensorflow-macos==2.10.0
python -m pip install tensorflow-metal==0.6.0

Installation

Download dataset and model from this link https://github.com/csaw-hackml/CSAW-HackML-2020/tree/master/lab3

Clone the repository
git clone https://github.com/avani1998/ECE-9163-ML-for-cybersecurity.git

Install dependencies
pip install -r requirements.txt