ML for Cyber Security Lab 2 Report
Tianzhe Fang
Tf2172
Github: lab2

1. Requirements

You must do the project individually. In this HW you will design a backdoor detector for BadNets trained on the YouTube Face dataset using the pruning defense discussed in class.
Your detector will take as input:
(1). B, a backdoored neural network classifier with N classes.
(2). Dvalid, a validation dataset of clean, labeled images.

What you must output is G a "repaired" BadNet. G has N+1 classes, and given unseen test input, it must:
(1). Output the correct class if the test input is clean. The correct class will be in [1,N].
(2). Output class N+1 if the input is backdoored.

You will design G using the pruning defense that we discussed in class. That is, you will prune the last pooling layer of BadNet B (the layer just before the FC layers) by removing one channel at a time from that layer. Channels should be removed in decreasing order of average activation values over the entire validation set. Every time you prune a channel, you will measure the new validation accuracy of the new pruned badnet. You will stop pruning once the validation accuracy drops at least X% below the original accuracy. This will be your new network B'.
Now, your goodnet G works as follows. For each test input, you will run it through both B and B'. If the classification outputs are the same, i.e., class i, you will output class i. If they differ you will output N+1. Evaluate this defense on:
(1). A BadNet, B1, ("sunglasses backdoor") on YouTube Face for which we have already told you what the backdoor looks like. That is, we give you the validation data, and also test data with examples of clean and backdoored inputs.

Now you must submit:
(1). Your repaired networks for X={2%,4%,10%}. The repaired networks will be evaluated using the evaluation script (eval.py) on this website https://github.com/csaw-hackml/CSAW-HackML-2020. Everything you need for this project is under the "lab3" directory.
(2). Please create and submit a link to a GitHub repo. with any/all code you have produced in this project along with a readme that tells us how to run your code.
(3). A short report (at most 2 pages) that includes a table with 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).

2. Prerequisite before run the lab ipynb file

a. Download data from : data for lab.
b. Upload the two zip files to colab (in "content" file)

c. Make sure that there is enough memory available for the execution (Ideally, Google Colab Pro subscription would do and I paid one-month Google Colab Pro to do my lab)

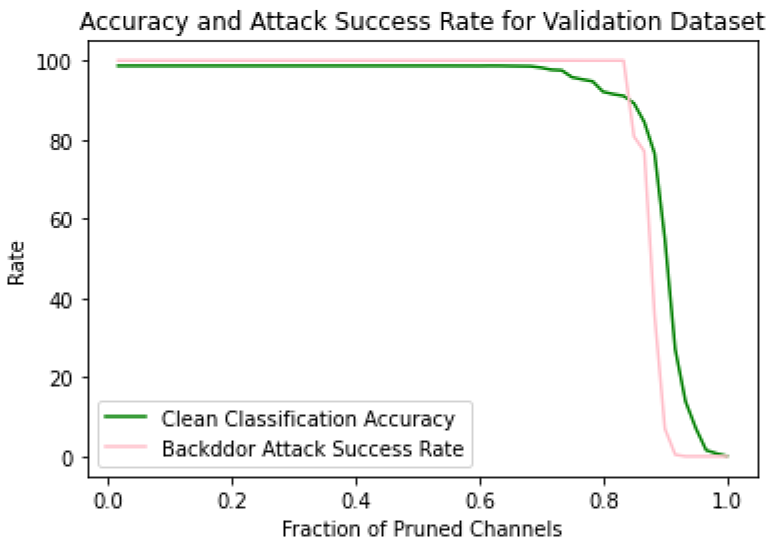
3. Process Description

The whole idea behind this is that we prune the convolutional layers based on the last ensemble-averaged activation of the entire validation dataset. Therefore, we need to prune the conv_3 layer. Following the above instructions, we need to save the model when the accuracy drops by at least {2%, 4%, 10%}. The saved models are titled pruned_2.h5, pruned_4.h5 and pruned_10.h5 for a 2%, 4% and 10% drop, respectively.

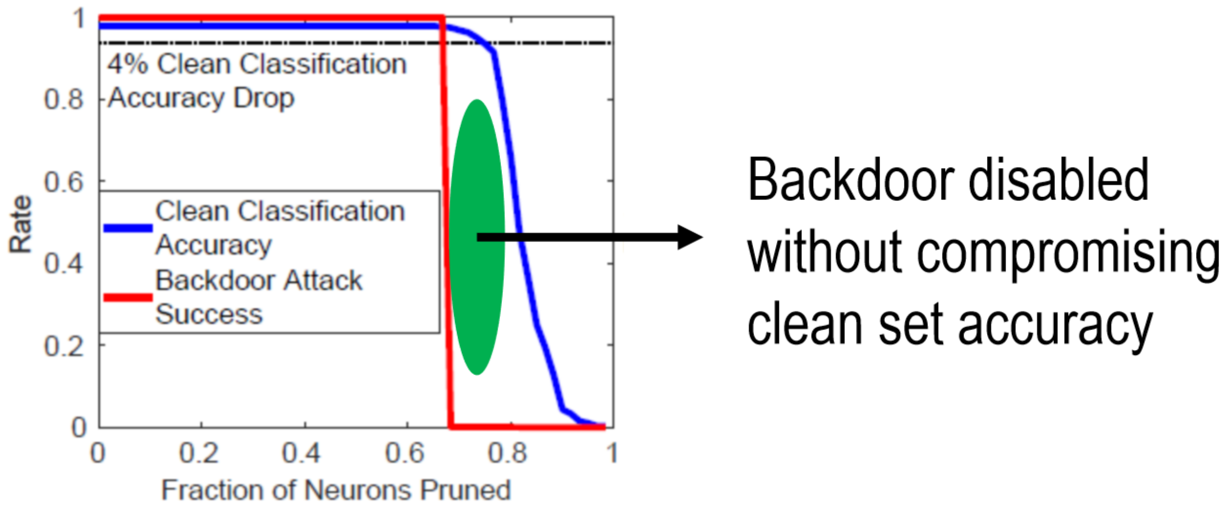
When the accuracy drops at least 30%, the attack success rate is 6.980168009006668%.

In order to design a good network, we need to combine two models, which are the bad network and the repaired model. The procedure can be found in the lab2.ipynb file.

4. Conclusion

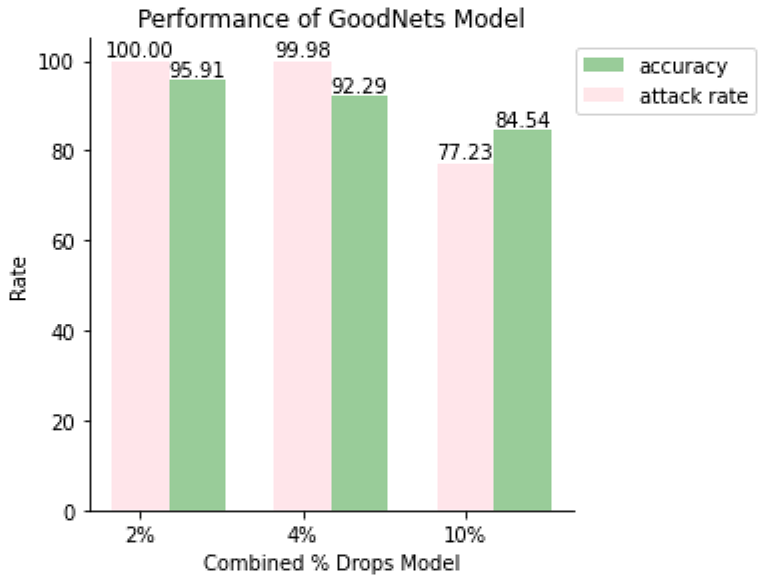


From the start, before pruning the defense, the accuracy is 98.65 and the attack success rate is 100.0. From the evaluation section, we can see that if we prune the channels slightly, the effect is not very obvious. As the number of pruned channels increases, the attack success rate decreases. When we trim a large number of channels, the attack success rate decreases a lot. However, the accuracy rate also decreases a lot. In addition, from the figure above, we can see that there is no large "green space" such as the space in the picture below. That is, the backdoor is enabled at the expense of the accuracy of the clean set. Therefore, I think that the pruning defense does not actually work for this model. This is probably because the attacker (this bd_model) adapts to this defense, i.e., the adaptive attacker introduces sacrificial neurons in the network to disable the pruning defense. This is why we do not see a significant effect of our defense approach.



(a) Baseline Attack (Face)

Here is a side-by-side comparison of the performance of the repaired model, where you can see here:


Performance of GoodNets Model
accuracy
attack rate
Rate
Combined % Drops Model