**ECE9133 - Machine Learning For Cybersecurity**

**Name: Mukta Maheshwari (mm11070)**

**Question:**

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, labelled 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 atleast 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. Evaluat this defense on:

1. A BadNet, B 1 , (“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:

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).

What you must output is G a “repaired” BadNet. G has N+1 classes, and given unseen test

**Summary report:**

Pruning - Pruning eliminates each channel from a pooling layer one at a time. That's what we do in Average activation levels over the full validation set are listed in decreasing order. Every time we prune a channel, we validate the accuracy, and we stop pruning when the accuracy falls X% or less than the original.

The attack success rate when the accuracy drops at least 30% is 6.954187234779596%.

Now, upon pruning the model, we used the clean validation dataset and after testing on the test data set. The accuracy and attack success rate would look like this for the validation dataset –

Graphical user interface

Description automatically generated with medium confidence

**Evaluating the combined model: (below is the accuracy output)**

401/401 [==============================] - 8s 20ms/step

2% drops model, the clean test data Classification accuracy: 95.90023382696803

401/401 [==============================] - 8s 19ms/step

2% drops model, Attack Success Rate: 100.0

401/401 [==============================] - 8s 19ms/step

4% drops model, the clean test data classification accuracy: 92.29150428682775

401/401 [==============================] - 8s 19ms/step

4% drops model, Attack Success Rate: 99.98441153546376

401/401 [==============================] - 8s 19ms/step

10% drops model, the clean test data classification accuracy: 84.54403741231489

401/401 [==============================] - 8s 19ms/step

10% drops model, Attack Success Rate: 77.20966484801247

**Accuracy vs Attack Rate summaries from code:**

**Graphical user interface, text, application, email

Description automatically generated**

Because the attack success rate does not much decline, we can see that the prune defense is not very effective. The attack success rate is acceptable but not great because it compromises the accuracy too much. The attack strategy is a prune immune attack, and the poisoned data are kept with the pruned model.

Also, there is a side-by-side comparison of the performance of the repaired model –

Graphical user interface, application

Description automatically generated

As per the instructions mentioned, we need to save the save the model when the accuracy drops by at least {2%, 4%, 10%}. The saved models are titled model\_X=2.h5, model\_X=4.h5 and model\_X=10.h5 for a drop in the 2%, 4% and 10% respectively. Its available in output folder.

Code to be executed is –

ML\_for\_cybersec\_Lab2\_mm11070.ipynb - code

ML\_for\_cybersec\_Lab2\_mm11070.pdf – pdf of executed code

Github Link - <https://github.com/mukta-maheshwari/ECE9133-Machine_Learning_For_Cybersecurity>