ECE-GY 9163
Machine Learning for Cyber Security
Backdoor Attacks

Vamsi Krishna Bunga (vb2279)

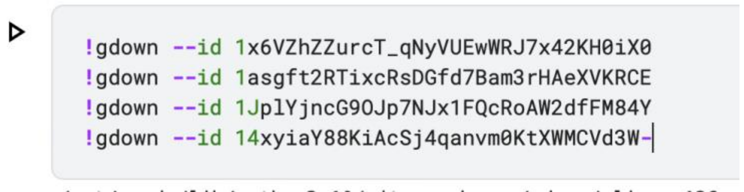
I used Kaggle notebook instead of google colab because of its resource restrictions.

Data:

1. Clone the repository https://github.com/csaw-hackml/CSAW-HackML-2020 into the jupyter notebook environment so that you can directly access the models in the notebook environment.
2. Download the validation and test datasets from here and upload to your google drive and make them public to be accessed by your notebook.
3. The validation and test images are bd_valid.h5 and bd_test.h5 respectively. They are poisoned with sunglasses trigger to activate backdoor for bd_net.h5.

Steps to run the code:

1. All the code that is produced is in the vb2279_backdoor_attack.ipynb notebook file.
2. First, make sure that all your data files which were uploaded in google drive in the earlier step are given public access.
3. Get the file_id of these files from the sharable links and replace them in the gdown command to download them into the notebook environment. Attaching screenshot for reference:



```
!gdown --id 1x6VZhZZurcT_qNyVUEwWRJ7x42KH0iX0
!gdown --id 1asgft2RTixcRsDGfd7Bam3rHAeXVKRCE
!gdown --id 1JplYjncG90Jp7NJx1FQcRoAW2dfFM84Y
!gdown --id 14xyiaY88KiAcSj4qanvm0KtXWMCVd3W-
```

4. Now you have all the data and models that are required in the environment. Just run the jupyter notebook.

Methodology:

We will prune channels according to their average activations in the last pooling layer. Backdoor networks often have certain neurons that activate only with backdoored samples. Our approach involves pruning channels based on their mean activations with a clean validation set.

This means we're removing channels less important for predictions on the clean dataset, which should not greatly impact accuracy. However, these channels might be more relevant to backdoored samples, which can lower the attack success rate.

We developed a method to save models when there's a specific drop in accuracy of 2%, 4%, and 10%. These model weights are stored in the results folder.
We created a GoodNet model (G) that inputs data to both the original (B) and pruned (B') BadNet models. It makes predictions as follows:
If both B and B' predict the same class, G returns that class.
If B and B' predict different classes, G assigns N+1 as the class.

Observations and Conclusions:

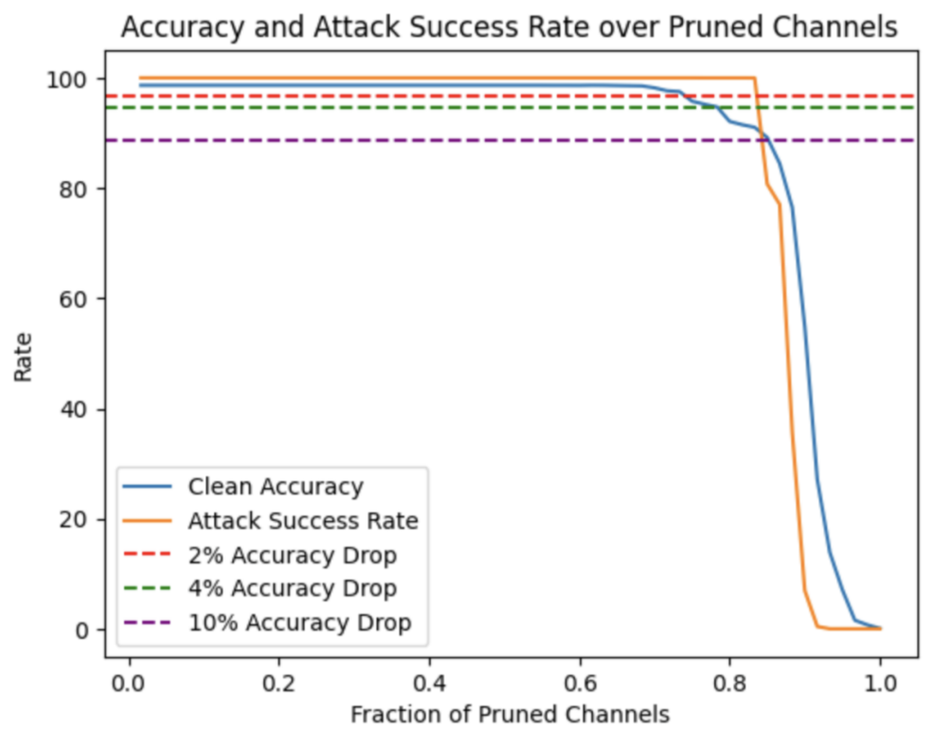
We can see that the prune defense is not too successful here. I think the attack is prune aware attack, which means this defense mechanism is not effective against this specific attack.

In the region between the blue and orange lines, the backdoor is **deactivated**. In this situation, many neurons that are not closely associated with key features in the classification area are pruned away. As a result, the influence of the backdoored dataset is neutralized.

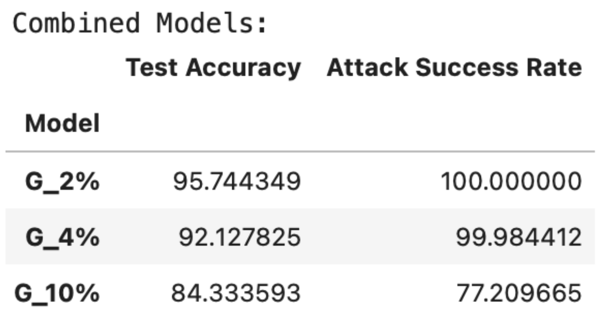
However, I consider this not so effective because although it decreases the attack success rate, this compromises the model's accuracy.

Plot:

Accuracy and Success Rate of clean test data as a function of fraction of channels pruned:



Performance of GoodNet Models:



```
Combined Models:
```

Model	Test Accuracy	Attack Success Rate
G_2%	95.744349	100.000000
G_4%	92.127825	99.984412
G_10%	84.333593	77.209665