ML for Cybersecurity
Fall'23
Lab 4 : Backdoor Attacks

Yogya Sharma (ys5250@nyu.edu)

Objective:
The objective of this homework is to design a backdoor detector for BadNets trained on the YouTube Face dataset using the pruning defense. The detector should be able to identify clean inputs and detect backdoored inputs accurately.

Overview:

Backdoor Attacks:
Backdoor attacks are a type of adversarial attack that aims to compromise the integrity of a machine learning model by injecting a specific trigger, or "backdoor," during the training process. This trigger is often inconspicuous and can be activated with specific input patterns to manipulate the model's behavior. In the context of neural networks, these attacks can lead to misclassifications or unwanted behaviors when exposed to inputs containing the backdoor.

Defense Mechanisms: Pruning Defense
The pruning defense involves systematically removing channels from the neural network, starting from the last pooling layer. By observing the average activation values over a validation set, channels are pruned until the validation accuracy drops below a specified threshold. This process results in a pruned BadNet, denoted as B', which is expected to be more resilient to backdoor attacks.

Approach:

1. Pruning Defense:
- We applied the pruning defense to the BadNet. The last pooling layer of badnet was pruned one channel at a time, removing channels in decreasing order of average activation values over the entire validation set.
- Pruning continued until the validation accuracy dropped by at least X% below the original accuracy. This resulted in a new pruned BadNet.

2. GoodNet G:
- For each test input, we passed it through both badnet and pruned badnets.
- If the classification outputs of badnet and pruned badnets' were the same (i.e., class i), we output class i. If they differed, we output class N+1.

3. Evaluation:
- The defense was evaluated on the following scenarios:
- A BadNet, with a known backdoor ("sunglasses backdoor").
- Repaired networks for different pruning percentages (X = 2%, 4%, 10%).

Procedure:

- Initialization:
    - A clone 'model_copy' of the original 'badnet_model' is created.
    - Weights of 'model_copy' are initialized with the weights of the original model.

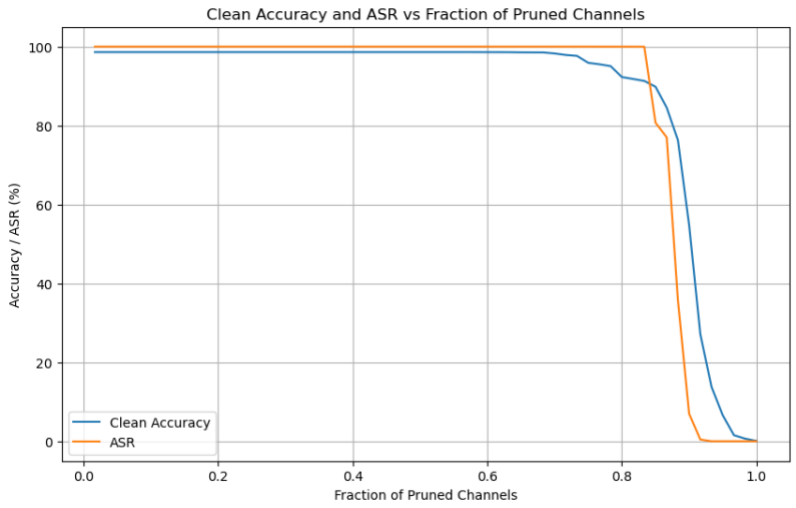
- Activation Extraction:
    - Activation values of the last pooling layer ('pool_3') are obtained using an intermediate model ('interm_model').
    - Channel pruning order is determined by mean activation values ('seq').

- Channel Pruning Loop:
    - Channels are pruned iteratively based on 'seq'.
    - For each pruned channel, corresponding filter weights in the last convolutional layer are set to zero.
    - Evaluation is performed on clean and backdoored test data.

- Evaluation and Model Saving:
    - Clean data accuracy ('cl_accuracy') and attack success rate ('asr') are computed for each pruned configuration.
    - Pruned models are saved if clean data accuracy drops by specified thresholds (2%, 4%, 10%).
    - Results, including accuracy, attack success rate, and pruned channel index, are printed.
    - Saved models indicate the pruning threshold in their filenames.

Observations:

During pruning process:



Results for clean test data on just the pruned models:

Model Configuration	Clean Data Accuracy	Attack Success Rate
Model with 2% threshold	95.90	100.0
Model with 4% threshold	92.29	| 99.98
Model with 10% threshold	84.54	77.21

Results for clean test data on GoodNet models models:

Model Configuration	Clean Data Accuracy	Attack Success Rate
Model with 2% threshold	95.74	100.0
Model with 4% threshold	92.12	| 99.98
Model with 10% threshold	84.33	77.21

Results:

Graphs: