Backdoor Networks

Machine Learning for Cyber Security (ECE-GY 9163), NYU

Suriya Prakash Jambunathan - sj3828

Introduction

Backdoor attacks pose a significant threat in today's AI landscape. When companies fine-tune pre-trained models downloaded online, they risk inheriting backdoor connections that compromise accuracy on specific samples. These vulnerabilities stem from **Backdoor Neural Networks (BadNets)**, containing neurons activated only by backdoored inputs. While these models achieve seemingly perfect accuracy on clean samples, they perform poorly on backdoored ones, predicting a target class different from the true class in targeted attacks. This project focuses on defining and implementing a pruning strategy to rectify BadNet models.

Objective

The primary goal of this project is to define and implement a rectification strategy for the BadNet model. The methodology we are employing is pruning. Once the model is pruned, we will construct another GoodNet model that accurately predicts clean samples, detects backdoored samples, and assigns them to a separate class.

Methodology

We will be pruning channels based on the order of mean activations on the last pooling layer. As explained, backdoor networks contain certain neurons that are only activated under backdoored samples. One way to observe the effect of this is by examining the outputs of the last pooling layer before the fully connected layers. The idea is to prune channels based on the order of mean activations on the clean validation set. This means that we are pruning channels that are less relevant to the clean dataset prediction, thereby not significantly affecting prediction accuracy. However, these channels may have relevance to backdoored samples, hence reducing the attack success rate.

The steps are as follows:

- **Compute activations** for the entire clean validation set on this extracted model.
- Calculate mean activation values with respect to the channel axis.
- Sort the **mean activations** so that the least important channels are ordered first.
- Prune the channels by iterating over the **sorted channel indices**.
- **Set the weights and biases** for the particular channel index to 0 to prune the channel.
- Evaluate accuracy on clean sets and calculate the attack success rate on BadNet sets.
- Measure the accuracy drop from the original to the pruned BadNet on the clean validation set.
- Store three sets of pruned model weights for the following **accuracy difference thresholds**:
  - 2%, 4%, 10%
- Create a **GoodNet model (G)** that passes inputs to both the **original (B)** and the **pruned (B')** BadNet model and predicts based on the following condition:
  - If both the original and pruned BadNet models predict the same class, return that class.
  - If the original and pruned BadNet models predict different classes, return N+1 as the class (where N is the total number of classes).

Performance Metrics

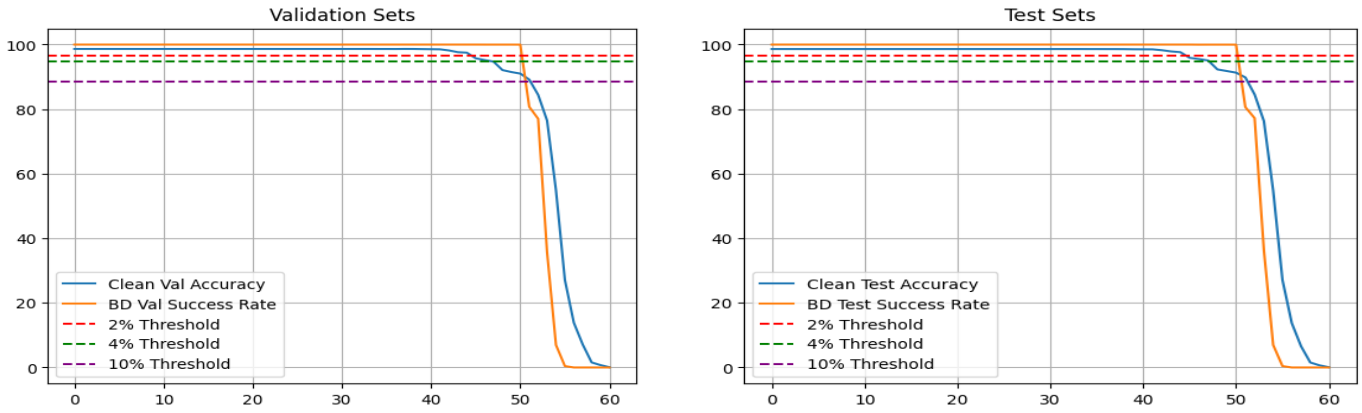

Figure 1 and 2: Performance metrics with respect to the number of channels pruned.

	Fraction of Channels Pruned (%)	Val Acc Drop (%)	Clean Val Acc (%)	Clean Test Acc (%)	Bad Val SR (%)	Bad Test SR (%)
3	5.0	0.00000	98.649000	98.620421	100.000000	100.000000
6	10.0	0.00000	98.649000	98.620421	100.000000	100.000000
12	20.0	0.00000	98.649000	98.620421	100.000000	100.000000
15	25.0	0.00000	98.649000	98.620421	100.000000	100.000000
18	30.0	0.00000	98.649000	98.620421	100.000000	100.000000
24	40.0	0.00000	98.649000	98.620421	100.000000	100.000000
30	50.0	0.00000	98.649000	98.620421	100.000000	100.000000
36	60.0	0.01732	98.631679	98.604832	100.000000	100.000000
42	70.0	0.45899	98.190006	98.269680	100.000000	100.000000
45	75.0	2.89253	95.756474	95.900234	100.000000	100.000000
48	80.0	6.55582	92.093184	92.291504	99.991340	99.984412
54	90.0	43.78626	54.862735	54.762276	6.954187	6.960249
60	100.0	98.57106	0.077942	0.077942	0.000000	0.000000

Table 1: Tabular representation of the Performance Metrics with respect to the fraction (%) of channels pruned.

Analysis

Upon analyzing the plot and table, it's evident that as we prune more channels, the accuracy on the clean set and the success rate on the bad set both decrease. Despite our attempt to prune less relevant channels, their tiny activation values still impact clean set accuracy. Nevertheless, there is a positive trend as the success rate on the bad set drops at a higher rate. This suggests that our pruning strategy, based on mean activations, is effective in reducing backdoor vulnerabilities, even though it comes at the cost of some clean set accuracy. The table on the side displays the performance metrics of the **GoodNet Model**.

% threshold	Clean Acc (%)	Bad SR (%)
2	97.505846	100.000000
4	94.902572	99.984412
10	89.680436	80.646921

Conclusion

In conclusion, our pruning strategy demonstrates a trade-off between maintaining clean set accuracy and mitigating the success rate on the bad set. While the **2%** and **4%** threshold models fall short in removing backdoor connections, the **10% threshold model** shows promise by significantly reducing the attack success rate. These findings indicate the potential of pruning as a viable approach to enhance the security of neural networks against backdoor attacks, with further refinements needed for optimal results.