Sparse-fine-pruning for Backdoor Detector

Wenwei Zhang(wz2037), Linnan Zhang(lz2400), Tianhao Wang(tw2245)

Project Introduction

This project is to design a backdoor detector for BadNets trained on the YouTube Face dataset. Given B, a backdoored neural network classifier with N classes and Dvalid, a validation dataset of clean, labeled images. What we output is G, a "repaired" BadNet, which has N+1 classes, and given unseen test input, it will output the correct class if the test input is clean. The correct class will be in [1, N], otherwise output class N+1 if the input is backdoored.

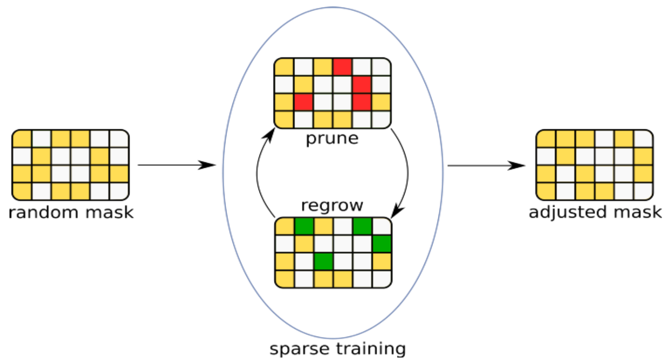
Proposal Introduction

In lab3, we only use the pruning defense to design the detector G, and the results shown does not works well. In this project, we will extend this method to get a better result.

Sparse-fine-pruning

Sparse training consists in enforcing a constant rate of sparsity during training while its distribution varies and is progressively adjusted.
Pruning is the process of removing weight connections in a network to increase inference speed and decrease model storage size, which can be thought as removing unused parameters from over parameterized network.
Fine-tuning means making small adjustments to a process to achieve the desired output or performance. Fine-tuning deep learning involves using weights of a previous deep learning algorithm for programming another similar deep learning process.
Fine pruning is a combination of the above two methods: Firstly, the model is pruned, thereby removing the backdoor neurons, and then fine-tuning restores the drop in classification accuracy.

Our method to design the repaired network is combining the Sparse training and Fine pruning:
1) initializing the network with a random mask that prunes a certain proportion of the network
2) training this pruned network during one epoch
3) pruning a certain number of weights of lower magnitude and
4) regrowing the same number of random weights.



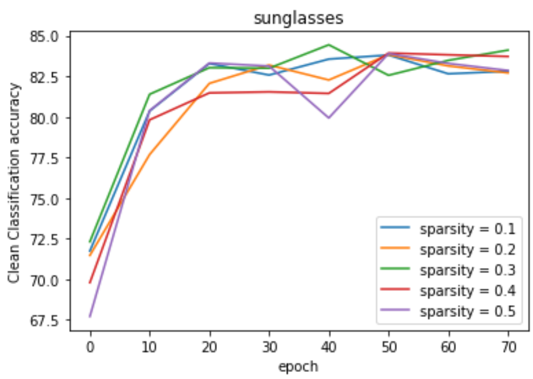
Main Code explanation

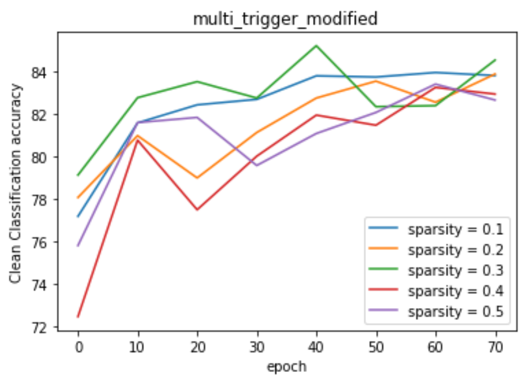
The main code used in project is in '*sparse-fine-pruning.py*' file.
Initialize model and get clean data: we firstly load and copy the base model and get clean valid data.
Randomly sparse X% channels: choose X% channels randomly by '*random.sample (range (0, 60), int (60 * X))*' and save to '*sparse_channels*' list.

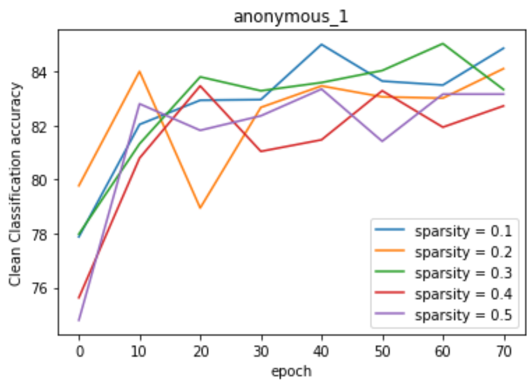
Firstly, fine pruning: based on the randomly-choose channels, we make the first pruning and training for one epoch. **Channels on same sparsity and Repeating training**: based on the previous fine pruning model, we firstly get a '*sorted_channel*' in decreasing order of average activation values over the entire validation set. Then prune X% channels and training through *'prune_model.fit (x_data, y_data, epochs=1)'*. And then repeat it for '*epochs*' times and save the accuracy and attack success rate after each training.

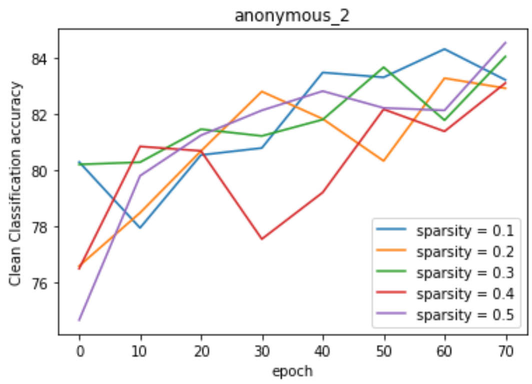
Conclusion

We set X% = [10,20,30,40,50], and for each X% we set epoch range from 0 to 70 and print corresponding Accuracy and attack-success-rate only for [0,10,20,30,40,50,60,70]. After running a long time for the experiment, we find that the attack-success-rate are all very small with each training model! So, we plot the data about accuracy. The following four figures show the accuracy for four separate models with different X% sparsity and epochs. As we can see here that the accuracy increases with the epochs increasing and the value of the green line (X% = 30%) is generally higher than the other lines. That is when X% = 30%, the accuracy is much more desirable. Also, the result is better when epoch is 40 and X% equals 30%.









The following table shows accuracy and attack success rate for four models when we prune 30% channels for 40 epochs. Although the values of accuracy do not reach to 95%, but as for attack success rate, the values are small enough to defend the attack. Also, we write an '*continue_training (model_path, data_path, epochs)*' function in file to try to get a higher accuracy after the sparsity-fine-pruning. In a word, this method is much better than prune-only method and can resist to attacks to a certain extent. And we will try to explore how to make the accuracy reach higher than 95% in the future work.

	X%	epoch	Clean_accuracy	Attk_succ_rate
Sunglasses	30	40	84.443	0.000
Multi_trig_modified	30	40	85.199	0.052
Anonymous_1	30	40	83.593	0.214
Anonymous_2	30	40	81.816	—