Andre Nakkab
Ajn313
Backdoor Defense Report

For this lab, I attempted a few different methods of pruning the channels of a BadNet to attempt to remove a backdoor trained into the network. The original BadNet obtained a clean data validation accuracy of 98.65%, and a clean data test accuracy of 98.62%. However, the attacks on the poisoned validation and test data were both 100% effective. Ideally, pruning will rectify this and reduce the efficacy of the backdoor attacks.

The first attempt involved pruning channels in order of activation value from the last convolutional layer till the threshold for loss of accuracy, X, compared to the original model, was reached. Thresholds for accuracy loss included X = (2, 4, 10), as described in the design specifications. This method, however, did not reduce the attack effectiveness by a satisfactory amount. In fact, even pruning the entire final convolutional layer of the model does not reduce the accuracy of the model by a full 10%. Even worse, pruning the entire final layer did not reduce attack efficacy by even 1%, so I sought out other avenues for defense & repair.

Going beyond the scope of the assignment, which was to only use the clean validation data, I instead pursued a method by which a channel is only pruned if it impacts the attack performance on the poisoned validation dataset under the assumption that we'd have an idea of what the hypothetical poisoned data would look like. Basically, pruning of each channel was tested on a temporary copy of our BadNet, and the attack efficacy was checked on the poisoned validation data. If the attack efficacy was reduced, this pruning was applied to my final model. Instead of just applying this pruning methodology to the final convolutional layer, I gave the script access to the final 3 convolutional layers to see if the backdoored neurons were in earlier layers. Based on the results, it seems that they were indeed not solely in the final convolutional layer.

I additionally included a check to ensure each individual channel pruned did not reduce the overall accuracy beyond the desired amount, which can be difficult when pruning earlier layers. Because of this, certain channels were not pruned by a given model which were by other models, e.g. the X = 4 model pruned fewer channels than the X = 2 model.

This method leads to a much greater degree of efficacy, with the X = 10 GoodNet attaining around a 65% reduction in attack effectiveness. If we sacrifice a bit of accuracy, we can greatly improve the security of the model. From there, the repaired GoodNet could be fine-tuned on clean data to recover its accuracy, which is far less resource intensive than from-scratch training and is much easier to do in a secure environment.

Below, see the table for model performance metrics based on which method was used. I have included results for the original BadNet, the model with the entire final layer pruned, and each of my GoodNets that were generated using the above method.

Metrics by Method or accuracy loss	Clean Validation Accuracy	Clean Test Accuracy	Validation Attack Efficacy	Test Attack Efficacy	Percent of channels pruned
Original BN	98.65%	98.62%	100%	100%	0%
Final Layer	91.08%	90.95%	99.98%	99.99%	100%
X = 2%	96.74%	96.97%	92.48%	92.66%	22.22%
X = 4%	94.76%	94.71%	92.43%	92.67%	18.33%
X = 10%	80.48%	80.43%	35.91%	35.69%	24.44%

GitHub Repository with code, models, instructions, and screenshots:

https://github.com/ajn313/pruning-defense