Cyber-security Final Project Report

Muyang Xu(mx625), Nandan Kumar Jha (nj2049), Praneeth Challagonda(psc368)

Our Github repo

Introduction

A backdoored neural network has been provided along with a clean, labelled validation dataset. The objective of this project is to repair the backdoored neural networks, which are not susceptible to backdoor triggered attacks. We are tackling this problem of backdoors in neural networks by using the fine-pruning technique proposed in this paper: Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks.

Methodology

Fine-pruning is a combination of pruning and fine-tuning techniques, that benefits from the advantages of the both. The following sections explain these sections in detail.

Pruning:

The poisoned data contains triggers to activate the backdoors in the BadNets, which contain the backdoors. The backdoored inputs trigger the neurons that are otherwise dormant in the presence of clean inputs. These neurons are called “backdoor neurons”. The pruning defense is a strategy where the backdoor is disabled by removing neurons that are dormant for clean inputs.

Fine tuning:

Fine tuning is a technique that uses transfer learning: A Deep Neural Network that has been trained for a certain task can be used to perform other related tasks. Instead of training from scratch the defender can use the fine-tuning technique using the clean data set.

Fine Pruning:

Fine-pruning is a combination of pruning and fine-tuning techniques, that benefits from the advantages of the both. First the model is pruned, thereby removing the backdoor neurons and then fine-tuning restores the drop in classification accuracy.

Project Code Explanation

Pruning:

The badNets have been iteratively pruned 3 times with `prune_low_magnitude` in Keras. The clean Validation set is used for fit. Pruned nets are saved in list `pruned_models`.For the optimizer, Adam optimizer has been chosen and ‘accuracy’ is the training metric and loss function used for pruning is `categorical_crossentropy`.

Fine-tuning:
Fine-tuning the pruned models with `1e-5` learning rate in 10 epochs. The clean validation set is used in the training process. Repaired nets are saved in list `repaired_models`. The optimizer used is Adam optimizer and the loss function used for fine-tuning is `categorical_crossentropy`.

Validation:
We validate our results by running benchmark comparison on test_clean, sunglasses_poisoned and anonymous_poisoned dataset before and after repairing.

Results

Our method is shown to be highly effective against backdoor attacks.

Net	Before Repairing (BadNet) - Accuracy			After Repairing - Accuracy		
	Clean test dataset	SG Poisoned dataset	Anon Poisoned dataset	Clean test dataset	SG Poisoned dataset	Anon Poisoned dataset
anonymous_2_bd	0.9596	0	91.39	0.9178	0	3.8971e-04
multi_trigger_multi_target	0.9601	0	0	0.9288	0	2.9228e-04
sunglasses	0.9777	0.9999	0.0011	0.9043	0.0307	8.7685e-04
anonymous_1	0.9596	0	91.39	0.9216	0	1.9486e-04

Table 1: Performance comparison of BadNets before and after repairing the backdoors.

net	Accuracy	True Positive Rate

anonymous_2	0.8906	0.8742
multi_trigger_multi_target	0.8324	0.7390
sunglasses	0.5508	0.1215
anonymous_1	0.9106	0.9232

Tabl2 2: Performance of poisoned image detection on mixed clean_test and anonymous_poisoned. Note positive means the poisoned