ML for Cyber Security Project Report

CSAW-HackML-2020

Name: Jingyuan Li, Net-ID: jl10915

To defense the backdoor attack by designing a backdoor detector, I used the fine-pruning strategy which is proposed in the paper *Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks* by Kang Liu's team. I made some optimize in the pruning process and the design a new good network base on the repaired network and the bad network to identify the original correct 1283 class faces and the backdoor poisoned faces, which means the output of the new good net has 1284 classes.

There are three main processing steps in my project. The first step is to prune the third convolution layer of the bad net according to each channel's contribution on the validation data input. The second step is fine-tune the pruned net with the unpruned channels' weights as the initial states. The third step is define a new network base on the bad net and the repaired net in order to change the 1283 class to 1284 class which contains 1283 correct face class and 1 poisoned face class.

The pruning process is used to compress the neuron network originally, because there are some silence neurons which has ignorable contributes to the output after training. For using pruning to defense backdoor attack, Gu et al.[18] have proved that when the input data is different, the active level of each neurons are different, which means, when input is clean data or poisoned data, the active level of each neurons are therefore various. Kang Liu's team also reproved that when the input is clean face image or backdoor image, activations of neurons have an obvious difference.

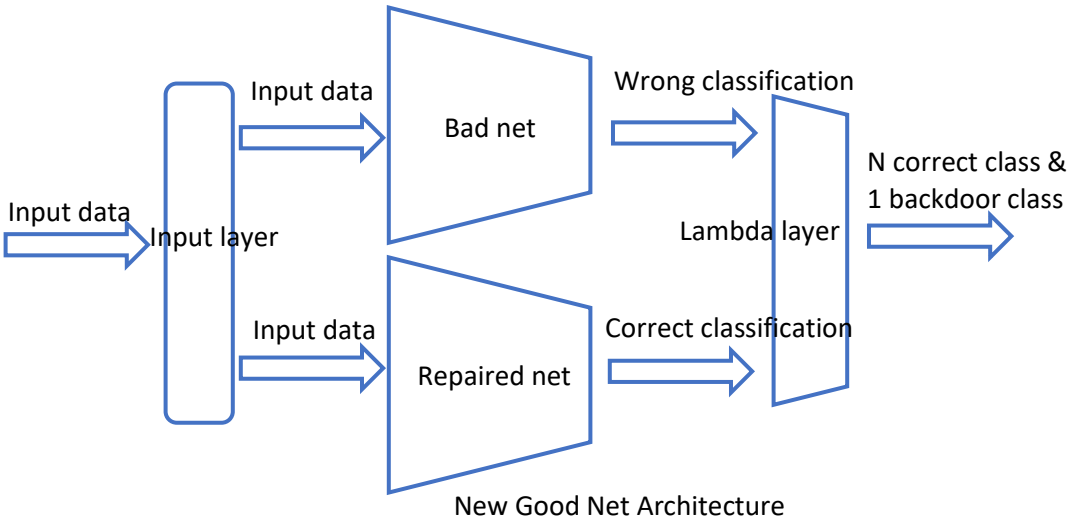
According to this conclusion, my goal is to pruning those channels with the "backdoor neurons". A essential point is that when deciding which channels to be pruned, I should use clean data as input and get the output come out from third convolutional layer instead of directly using the channels' weights. if not doing so, we can not separate the good channels and bad channels.

My implementation of the pruning strategy is described as following. Firstly extracts the output of 'conv_3' using model.get_layer() function, then declare a temporary model using the bad net's input as input and the conv_3 layer's output as output. By using this temporary model to calculate the output of clean validation data and calculating the average value of each channel, I can get a list of each channels' contribution to the output. Then, I sort this list by numpy.argsort() function to get the order of pruning channels. For each pruning operation, the weights and bias of the pruned channel are set to zeros using the set_weights() function.

When applying the pruning process to the bad net, I set two terminate conditions for the upper bound and the lower bound about the number of pruned channels. One is pruned dense rate. If the pruned channels number exceed the 85% of the original channel number, terminate the pruning process. Because if pruning to many channels, the net would not have enough capacity to implement its classification function during the fine-tune process. The other terminate condition is reduced accuracy rate. If the accuracy reduce by 60%, terminate the pruning process. The reason why using 60% is that the net will be fine-tuned later and the accuracy will grow back up to 90%. In order to avoid the pruning-aware attack, I have to trade of between the accuracy and the defense result. So I choose 50% instead of 4% which is recommended by Kang Liu's team.

The second step is fine-tune the pruned net. It is simpler to implement compared with pruning. I use the Adam as the optimizer, categorial cross entropy as the loss function and accuracy as the metrics to fine tune the pruned net.

The last step of my project is to design an new net to identify the 1283 correct face images and the poisoned image. So I reuse the repaired net and the bad net by adding a input layer in front of them and add a lambda layer behind them. The architecture of my new net is shown in image.



New Good Net Architecture

Inside the lambda layer, I use the tensor operations to find out the different classifications made by the repaired net and the bad net. In order to find the backdoor image, I assume that the repaired net has been cured so that it can classify the clean input correctly, while randomly classify the backdoor input to any class. The bad net still classify the backdoor input to a specific class which the attacker set to. My goal is to find out the different class made by the two nets. These images with different result made by two net are the backdoor images, so they will be classify to the 1284 class which is the backdoor class. I implement this function by the following formula. N represents the class number which is 1283.

$$sign = \left|sign(argmax(y_{clean})) - sign(argmax(y_{poisoned}))\right|$$
$$y_{new} = sign \times N + (ones\_like(sign) - sign) \times argmax(y_{clean})$$

The accuracies for each step for each net are as following.

	Original	After Pruning	After Fine-Tuning	New Good Net
Sunglasses bad net	Test: 97.79 Validation: 97.88 Poisoned: 99.99	Test: 43.03 Validation: 42.86 Poisoned: 96.08	Test: 94.35 Validation: 99.84 Poisoned: 33.86	Test: 94.35 Validation: 99.84 Poisoned: 33.86
Anonymous bad net	Test: 96.31 Validation: 99.92	Test: 49.43 Validation: 49.59	Test: 95.30 Validation: 99.93	Test: 95.30 Validation: 99.93
Multi-trigger multi-target bad net	Test: 96.17 Validation: 99.90	Test: 45.94 Validation: 48.98	Test: 96.17 Validation: 99.90	Test: 96.17 Validation: 99.90

As the table shows, the pruning process of sunglass bad net does not cause a obvious reduce to the accuracy on the sunglass poisoned dataset. It is very possible that the attacker who made this sunglass bad net is a pruning-aware attacker. However, after fine-tuning the pruned net, the accuracy on the poisoned dataset shows a significant decrease, which means that the fine-pruning strategy works very well.

The reason why there is no difference between the accuracy of fine-tuned net and the new good net is that the new net just find the wrong-classified samples and put them into the 1284 class, which means the correct class still correct and the wrong class still wrong. This operation should not cause any affect on the accuracy.

In conclusion, the Fine-Pruning strategy works well and I also did some optimization to the original strategy. Finally, I repaired all three bad net and transferred them to the new good net which has N+1(1284) classes.