“Backdoor Detection in Neural Networks Using Pruning Defense”

Dhairya Shah
ds6969

Github Link: https://github.com/dhairyashah10/ML_HW_LAB

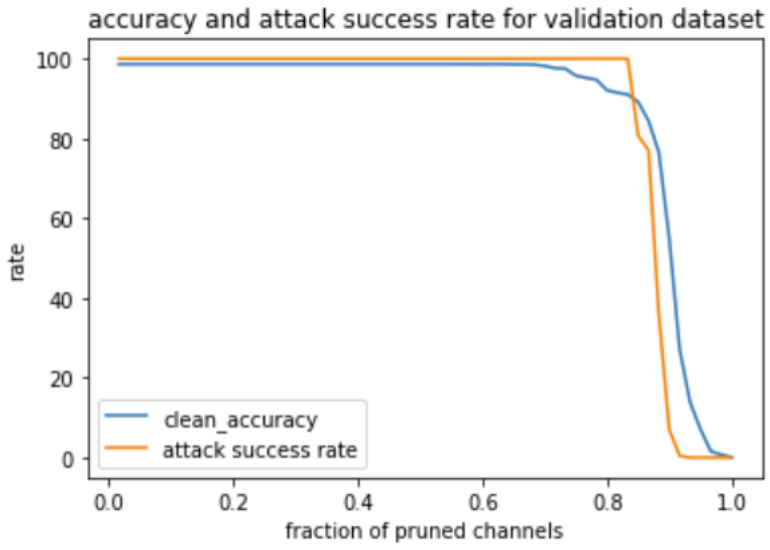
Methodology:

This approach focuses on pruning the `conv_3` layer based on the average activation across the entire validation dataset from the last pooling layer. We are required to preserve the model versions when there's a decline in accuracy by specific thresholds, namely 2%, 4%, and 10%. Consequently, the models saved are named `model_X=2.h5`, `model_X=4.h5`, and `model_X=10.h5`, corresponding to each of these accuracy drop percentages. When the accuracy decreases by at least 30%, the rate of successful attacks stands at 6.954187234779596%. To construct an effective GoodNet, it was necessary to integrate the BadNet with the modified model. This process and the relevant code can be found in the `ds6969.ipynb` file.

Instructions on running the code:

- Make sure that there is enough memory available for the execution (Ideally, Google Colab Pro subscription would do)
- The data set can be available from the links mentioned (https://github.com/csaw-hackml/CSAW-HackML-2020/tree/master/lab3) and the Google drive link to the data files is (https://drive.google.com/drive/folders/1u9WR8cQA12ilJEgNp6ZgHsaXF0UtA8mS?usp=drive_link)

Now, upon pruning the model, we can use the clean validation data set and test it on the test data set. The accuracy and attack success rate would look like this for the validation dataset



We can notice that the prune defense is not that successful here because the attack success rate does not drop too much. The attack success rate is okay, but not too good because it compromise the accuracy way too much. My hypothesis is that attack method is prune immune attack and that the pruned model is retained with the poisoned data.

Also, there is a side-by-side comparison of the performance of the repaired model and it can be seen here :