Lab 4: Backdoor Attacks
Laboratory Report

Name: Prajna Ravindra Nayak

NetId: pn2224

Table Of Contents

Overview

You must do the project individually. In this HW you will design a backdoor detector for BadNets trained on the YouTube Face dataset using the pruning defense discussed in class.

Your detector will take as input: B, a backdoored neural network classifier with N classes. Dvalid, a validation dataset of clean, labeled images. What you must output is G a "repaired" BadNet. G has N+1 classes and given unseen test input, it must output the correct class if the test input is clean.

The correct class will be in [1, N]. Output class N+1 if the input is backdoored. You will design G using the pruning defense we discussed in class. That is, you will prune the last convolution layer of BadNet (the layer just before the FC layers) by removing one channel at a time from that layer. Channels should be removed in increasing order of average activation values over the entire validation set.

Every time you prune a channel, you will measure the new validation accuracy of the newly pruned BadNet. You will stop pruning once the validation accuracy drops at least X% below the original accuracy. This will be your new network. Now, your GoodNet G works as follows. For each test input, you will run it through both and If the classification outputs are the same, i.e., class i, you will output class i. If they differ you will output N+1.

Evaluate this defense on: A BadNet, B1, ("sunglasses backdoor") on YouTube Face for which we have already told yo you what the backdoor looks like. That is, we give you the validation data and test data with examples of clean and backdoored inputs.

Data

You can obtain the test and validation datasets by following this link and then saving them in the lab3/data/ directory. The dataset contains images sourced from the YouTube Aligned Face Dataset. After collecting data from 1283 individuals, we split it into test and validation sets. The sunglasses trigger in bd_valid.h5 and bd_test.h5 activates the backdoor for bd_net.h5. Here's a breakdown of the data files:

bd_valid.h5: Sunglasses-poisoned validation data

bd_test.h5: Sunglasses-poisoned test data

cl_valid.h5: Clean validation data used for defense design

test.h5: Clean test data used to evaluate the BadNet

Additionally, you'll find various files related to models, including bd_net.h5, architecture.py, eval.py, L, bd_weights.h5, and others. The eval.py script is specifically designed for evaluation purposes.

To clarify, bd_net.h5 is the model with the backdoor, and the sunglasses triggers in bd_valid.h5 and bd_test.h5 activate this backdoor during validation and testing.

In summary, download the datasets from the provided link, store them in the lab3/data/ directory, and utilize the mentioned files for model evaluation and defense design.

Procedure

The main goal of this project was to improve a machine learning model through a sequence of steps, including layer pruning, saving models based on accuracy, conducting vulnerability assessments, and creating an optimized composite model. Here's a breakdown of the key steps:

Layer Pruning and Model Saving:

Our approach involved pruning the conv_3 layer based on the average activation from the last pooling operation across the validation dataset.We adopted a strategy to save models at specific accuracy drop thresholds of 2%, 4%, and 10%. The saved models were named model_X=2.h5, model_X=4.h5, and model_X=10.h5, respectively, indicating the corresponding accuracy drop percentage.

Vulnerability Assessment:

A crucial step involved assessing the attack success rate when the model's accuracy dropped by at least 30%.The observed metric for vulnerability threshold was 6.954187234779596%.

Model Integration (GoodNet Design):

To boost the model's performance, we implemented a strategy to combine two models: the compromised "BadNet" and a refined model post-repair.This process aimed to create a superior composite model, referred to as "GoodNet."

Implementation Details:

Comprehensive details of the entire program code and implementation steps are available in the pn2224_Prajna_Nayak_Lab4.ipynb file.

This file serves as a repository for the codebase, covering procedures such as model creation, pruning, evaluation, vulnerability assessment, and the integration of models to form GoodNet.

Notes on Code Execution:

Because Google Colab resources are limited, this code is executed locally. Download the dataset and any files required for code execution by clicking on the links in the Data section. Make sure you check and update the path addresses for clean data, poisoned data, and models based on the data in your system before executing the code.

Make sure to use clean validation data (valid.h5) specifically while constructing the pruning defense. Use test data (test.h5 and bd_test.h5) to assess the models. This guarantees that, when the code runs, the right datasets are used for the right reasons.

Observations

The clean validation dataset is used for the performance evaluation after the model has been pruned. The obtained 'Classification Accuracy' and 'Attack Success Rate' from the validation dataset are shown below.

Pruned Channels(in fraction)	Clean Data Accuracy(%)	Attack Success Rate (%)
0.02 (2%)	95.900234	100.000000
0.04 (4%)	92.291504	99.984412
0.1 (10 %)	84.544037	77.309665

According to the evaluation, there hasn't been a noticeable decrease in the attack success rate using the pruning defense strategy. The accuracy of the model is severely compromised by the defense method, even though the success rate is not unduly high. It appears that this particular attack may be too strong for the pruning defense strategy to handle, pointing to a possible barrier. Furthermore, there is a suspicion that the model may remain vulnerable after pruning defense because it was trained on data that contained malicious elements or was purposefully manipulated (i.e., poisoned data). In conclusion, it seems that the pruning defense strategy is unable to considerably lower the attack success rate, which raises questions regarding the model's general security. The model's reduced accuracy combined with the attack method's possible resistance or the model's continued susceptibility to training data, which shows how well the repaired model performed.