ML for Cyber Security

Project Report
CSAW-HackML-2020
Name: Eugene Wang
Net-ID: yjw259

https://github.com/y56/CSAW-HackML-2020/blob/master/report.md (https://github.com/y56/CSAW-HackML-2020/blob/master/report.md)

https://hackmd.io/@y56/H1xHFwbTD (https://hackmd.io/@y56/H1xHFwbTD)

Introduction

In this lab we are given backdoored CNNs (called bad-net/bd_model) with known architecture (refer to `architecture.py`) and we want to “repair” the bad-net. Imagine we are buying service to train a model for us, or using some unknown source of model. Attackers may train the model to perform normally on “clean” data while output misleading result on “poisoned data” with “trigger” it.

Methodology

Following the method in ***Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks***, I use fine-pruning as the defense approach. Although the authors also mention the pruning-aware attack. I assume the bad net we considered as just baseline attack.

Pruning

I reset the the 77% lowest contribution neurons under clean input. The idea behind is that we believe the backdoor pattern will be capture in the `conv_3` layer but its contribution will be low when clean input are processed.

Fine-tuning

In this step, I retrain the model on clean input. As mentioned in the paper, this is a technique of transfer learning. We can view it as we drop some suspicious rules in the bad net first, and then train the model based on the detoxified (yet less powerful) model. Note that those reset neurons will gain weights/bias again. I save fine-pruned models for each bd model.

Combining

I use a function `accuracy_calculator_for_combined_models()` to call both bad net and fine-pruned net and compare their outputs. An input will be considered clean if the two models give the same result, otherwise, backdoored.

Detecting backdoored data for evaluating performance

> Actually no need for this since the provided posioned data are completely backdoored.

For `anonymous_1_poisoned_data.h5` I use number of purple `(128,255,255)` pixels to detect backdoored data. This only for evaluation. I am not using this information to do the repairing. I use `check_anonymous_1_poisoned_data.py` to label backdoored inputs of `anonymous_1_poisoned_data.data` such that I can calculated *accuracy on clean data*, *attack success rate on backdoored data*, and *attack detection rate*.

Discussion

anonymous_1_bd_net

```
python3 eval_defense.py data/anonymous_1_poisoned_data.h5 models/anonymous_1_bd_n
```

- before repair
    - bad net on clean validation data:
        - acc: 97.18
    - bad net on clean test data:
        - acc: 97.19
    - bad net on its corresponding poisoned data:
        - acc: 91.40
- pruned
    - pruned bad net on clean validation data:
        - acc: 36.81
    - pruned bad net on clean test data: acc:

    - acc: 37.28
  - pruned bad net on its corresponding poisoned data:
    - acc: 50.08

- tuned and pruned

  - tuned pruned bad net on clean validation data:
    - acc: 99.89
  - tuned pruned bad net on clean test data:
    - acc: 95.259
  - tuned pruned bad net on its corresponding poisoned data:
    - acc: 8.37

- repaired (by comparing)

  - repaired net on clean validation data:
    - acc: 97.13
    - inferred attack_ratio: 2.80 (true as 0)
  - repaired net on clean test data:
    -
      - acc: 93.84
    - inferred attack_ratio: 5.64 (true as 0)
  - repaired net on its corresponding poisoned data:
    - acc: 8.35
    - inferred attack_ratio: 83.66

- true attack ratio (using purple detection) :

  - 90.93

- accuracy on those clean data within poisoned data:

  - 0.21
    - I don't know why so low, maybe all data od `anonymous_1_poisoned_data.h5` are backdoored? So actually no clean data in it.
    - All data in `anonymous_1_poisoned_data.h5` are labeled as zero. So, very possible there are no clean data in it.

- detected rate for those truly bd data within poisoned data:

  - 92.14

- attack succes for those clean data wihun poisoned data:
  - 7.67

Reference

Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks

Kang Liu, Brendan Dolan-Gavitt, Siddharth Garg

https://arxiv.org/abs/1805.12185 (https://arxiv.org/abs/1805.12185)

https://github.com/kangliucn (https://github.com/kangliucn)

log

sunglasses_bd_net

```
python3 eval_defense.py data/sunglasses_poisoned_data.h5 models/sunglasses_bd_net
```

```
bad net on clean validation data: acc:  97.88689702953148
bad net on clean test data: acc:  97.77864380358535
bad net on its corresponding poisoned data: acc:  99.99220576773187
pruned bad net on clean validation data: acc:  35.51571836840738
pruned bad net on clean test data: acc:  35.861262665627436
pruned bad net on its corresponding poisoned data: acc:  93.4684333593141
tuned pruned bad net on clean validation data: acc:  99.76617303195636
tuned pruned bad net on clean test data: acc:  93.4684333593141
tuned pruned bad net on its corresponding poisoned data: acc:  9.5401402961
repaired net on clean validation data: acc, inferred attack_ratio:  (97.705
repaired net on clean test data: acc, inferred attack_ratio  (92.7903351519
repaired net on its corresponding poisoned data: overall acc, inferred atta
```

multi_trigger_multi_target_bd_net

eyebrows_poisoned_data

```
python3 eval_defense.py "data/Multi-trigger Multi-target/eyebrows_poisoned_data.h
```

```
bad net on clean validation data: acc:  96.26742876937733
bad net on clean test data: acc:  96.00935307872174
bad net on its corresponding poisoned data: acc:  91.34840218238503
pruned bad net on clean validation data: acc:  46.03793193037152
pruned bad net on clean test data: acc:  45.74434918160561
pruned bad net on its corresponding poisoned data: acc:  74.29851909586905
tuned pruned bad net on clean validation data: acc:  99.91339741924308
tuned pruned bad net on clean test data: acc:  95.22213561964146
tuned pruned bad net on its corresponding poisoned data: acc:  58.476227591
repaired net on clean validation data: acc, inferred attack_ratio:  (96.250
repaired net on clean test data: acc, inferred attack_ratio  (93.0397505845
repaired net on its corresponding poisoned data: overall acc, inferred atta
```

lipstick_poisoned_data

```
python3 eval_defense.py "data/Multi-trigger Multi-target/lipstick_poisoned_data.h
```

```
bad net on clean validation data: acc:  96.26742876937733
bad net on clean test data: acc:  96.00935307872174
bad net on its corresponding poisoned data: acc:  91.52377240841777
pruned bad net on clean validation data: acc:  46.03793193037152
pruned bad net on clean test data: acc:  45.74434918160561
pruned bad net on its corresponding poisoned data: acc:  27.572096648480127
tuned pruned bad net on clean validation data: acc:  99.96535896769724
tuned pruned bad net on clean test data: acc:  95.17537022603274
tuned pruned bad net on its corresponding poisoned data: acc:  0.9840218238
repaired net on clean validation data: acc, inferred attack_ratio:  (96.241
repaired net on clean test data: acc, inferred attack_ratio  (93.0241621200
repaired net on its corresponding poisoned data: overall acc, inferred atta
```

sunglasses_poisoned_data

```
python3 eval_defense.py "data/Multi-trigger Multi-target/sunglasses_poisoned_data
```

```
bad net on clean validation data: acc:  96.26742876937733
bad net on clean test data: acc:  96.00935307872174
bad net on its corresponding poisoned data: acc:  100.0
pruned bad net on clean validation data: acc:  46.03793193037152
pruned bad net on clean test data: acc:  45.74434918160561
pruned bad net on its corresponding poisoned data: acc:  0.0097427903351519
tuned pruned bad net on clean validation data: acc:  99.89607690309171
tuned pruned bad net on clean test data: acc:  95.18316445830087
tuned pruned bad net on its corresponding poisoned data: acc:  0.1363990646
repaired net on clean validation data: acc, inferred attack_ratio:  (96.206
repaired net on clean test data: acc, inferred attack_ratio  (92.9851909586
repaired net on its corresponding poisoned data: overall acc, inferred atta
```

anonymous_2_bd_net

```
python3 eval_defense_nodata.py nodata models/anonymous_2_bd_net.h5
```

```
bad net on clean validation data: acc:  95.82575560751711
bad net on clean test data: acc:  95.96258768511302
pruned bad net on clean validation data: acc:  36.780116047458215
pruned bad net on clean test data: acc:  37.44349181605612
tuned pruned bad net on clean validation data: acc:  99.91339741924308
tuned pruned bad net on clean test data: acc:  94.94933749025721
repaired net on clean validation data: acc, inferred attack_ratio:  (95.791
repaired net on clean test data: acc, inferred attack_ratio  (92.6422447388
```