ECE-GY-9163-ML-for-Security-Project

0.Group members

We are a group of 3, members are as follow:

Xinhao, LI(xl3413) xl3413@nyu.edu

Yanbing, YANG(yy3476) yy3476@nyu.edu

Zhe, ZHANG(zz3230) zz3230@nyu.edu

1.Requirements

The project is to design a **backdoor detector** for BadNets trained on the YouTube Face dataset. Your backdoor detector is given:

1. B, a backdoored neural network classifier with N classes.
2. D valid, a validation dataset of clean, labelled images.

What you must output is G a “repaired” BadNet. G has N+1 classes, and given unseen test input, it must:

1. Output the correct class if the test input is clean. The correct class will be in [0,N-1].
2. Output class N if the input is backdoored.

2.Fine Pruning Method

a. Main Idea

Pruning is the process of removing weight connections in a network to increase inference speed and decrease model storage size. In general, neural networks are very over parameterized. Pruning a network means the defender finding and the backdoor neurons and eliminating the unused parameters. However, we simply use Fine-tuning or prune defense to not be very effective. Because the neurons activated by the backdoor data are only activated by the backdoor, and they are dormant under the clean data, so we record the average activation of each neuron. Then, the defense prunes the neurons in the order of average activation.

We improve based on prune defense and use fine-tune to improve accuracy. When we get a model similar to our target model, we can use our validate dataset as input to train the model again to fine-tune the model parameters. This process works as follow:

1. Pruning the model put back by the attacker. (Remove induced neurons)
2. After trimming, use clean input to fine-tune the weights on the neurons. Because the backdoor neurons overlap with our normal neurons, we can use clean input to fine-tune our neurons and change their weights to control the weight of our backdoor control.

b. Implementation

In our code, we have tried to prune channels in a max_pooling layer. However, this method does not work well.

In our project, we tried to do more pruning jobs. We believe that the backdoor in the model hides in the neurons which are inactive when clean data is used. After researching, we found a way to prune the whole net, which is removing the individual weight connections from a network by setting them to 0. In this way, the connections will become no-operation in the network. This way of pruning is widely used in model compression, and it is also very useful in the problem we are facing.

In our code, we first set our sparsity to 50% and then increase to %70. In the end, 70% of the neurons will be disabled, which will slightly affect the model accuracy. However, we can prevent the attack.

3.Dependencies and Data

a. directories

```
├── data  // Dataset is not uploaded due to poor network condition
    └── clean_valid_data.h5 // clean validation data used to prune and train the
repaired network
    └── clean_test_data.h5  // clean test data used to evaluate the acc of the
repaired network
    └── XXX_poisoned_data.h5  // clean test data used to evaluate the atk of the
repaired network
├── models
    └── XXX_weights.h5 // weights to backdoored network
    └── XXX_bd_net.h5 // backdoored network with specific attack dataset
    └── XXX_bd_net_rp.h5 // pruned network generated from './prune.py'
└── prune.py  // generate a pruned network saved as './models/XXX_bd_net_rp.h5'
└── eval.py  // create a repaired network and eval the acc and atk on specific
poisoned data
└── README.md  // project instruction and report
└── Project_Report.pdf  // project report
└── model_architecture.png  // Main structure of the backdoored network
```

b. dependencies

We use Anaconda environment and PyCharm to design this repaired network as a detector to defend against backdoor attack, implementing Fine-Pruning, relating Python libraries are as follow:

```
1. Python 3.7.9
2. Keras 2.3.1
3. Numpy 1.16.3
4. Matplotlib 2.2.2
5. H5py 2.9.0
6. TensorFlow-gpu 1.15.2
7. tensorflow-model-optimization 0.7.0
```

c. data

1. Download the validation and test datasets from here and store them under `data/` directory.
2. The dataset contains images from YouTube Aligned Face Dataset. We retrieve 1283 individuals each containing 9 images in the validation dataset.
3. sunglasses_poisoned_data.h5 contains test images with sunglasses trigger that activates the backdoor for sunglasses_bd_net.h5. Similarly, there are other .h5 files with poisoned data that correspond to different BadNets under models directory.

4.Codes and Explanations

Here is the prune function. We mainly use PolynomialDecay to get our model from initial_sparsity to final_sparsity (from 50% to 70%).

```
def prune_model(base_model, initial_sparsity, final_sparsity, end_step,
log_dir):
    pruning_params = {
        'pruning_schedule':
tfmot.sparsity.keras.PolynomialDecay(initial_sparsity=initial_sparsity,

final_sparsity=final_sparsity,
                                                                begin_step=0,

end_step=end_step)
    }
    model_for_pruning = tfmot.sparsity.keras.prune_low_magnitude(base_model,
**pruning_params)
    model_for_pruning.compile(optimizer=base_model.optimizer,
                              loss=base_model.loss,
                              metrics=['accuracy'])
    log_dir = 'log'
    callbacks = [
        tfmot.sparsity.keras.UpdatePruningStep(),
        tfmot.sparsity.keras.PruningSummaries(log_dir=log_dir),
    ]
    return model_for_pruning, callbacks
```

After pruning, we need to fine-tune the model so that our model can have higher accuracy to clean data.

```
pruned_model.fit(x_train, y_train,
                         batch_size=batch_size, epochs=epochs,
validation_split=validation_split,
                         callbacks=callbacks)
```

5.Result

Accuracy(test) and Success Rate(poisoned)

BadNet	Accuracy(original)	Accuracy(repaired)	Success Rate(original)	Success Rate(repaired)
sunglasses	97.78	81.22	99.99	3.48
multiple(eyebrows)	96.01	89.69	91.35	0.55
multiple(lipsticks)	96.01	89.69	91.52	51.30
multiple(sunglasses)	96.01	89.68	100	98.46
anonymous_1	97.19	90.21	91.40	4.39

Generally, the **attack success rate(ASR)** has a **sharp drop** after implementing the fine-pruning method, mean while the drop of the **accuracy** is **acceptable**

However, our method does not work well in multi-trigger and multi-target situation. We infer that, if a model has multiple backdoors, then the malicious neurons may act normally when clean data comes and act abnormally when backdoor data comes. In this situation, we cannot rule out these malicious neurons. We still need further research.

6.Running Instruction

To train and generate the **pruned model**, execute `prune.py` by running:

```
python prune.py <bad_model_filename>
```

To generate and evaluate the **repaired model**, execute `eval.py` by running:

```
python eval.py <poisoned_data_filename> <bad_model_filename>
```

E.g., `python eval.py data/anonymous_1_poisoned_data.h5 models/anonymous_1_bd_net.h5`.

7.Conclusion

Our method can successfully defend the attacks. It has two advantages:

- Our method is easy to implement.
- Our method does not rely on the bad data to train.

Although our method can sharply decrease the attack success rate, it has two drawbacks:

- The accuracy will decrease around 10%, which is not good enough to implement in reality.
- If the model has the multiple backdoors, some of the attacks cannot be defended.

8.References

[1] Liu, Kang, Brendan Dolan-Gavitt, and Siddharth Garg. "Fine-pruning: Defending against backdooring attacks on deep neural networks." *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, Cham, 2018.

[2] Gu, Tianyu, et al. "Badnets: Evaluating backdooring attacks on deep neural networks." *IEEE Access* 7 (2019): 47230-47244.

[3] ODSC Community. “What Is Pruning in Machine Learning?” *Open Data Science - Your News Source for AI, Machine Learning & More*, ODSC Community, 28 Oct. 2020, opendatascience.com/what-is-pruning-in-machine-learning/.

[4]“Pruning in Keras Example ： Tensorflow Model Optimization.” *TensorFlow*, Google, 11 Nov. 2021, www.tensorflow.org/model_optimization/guide/pruning/pruning_with_keras.