Project report

Course: Machine Learning Security
Section: A
Submission Date: December 20, 2021
Group Member Names and IDs: Youwen Zhang (yz6999), Run Yu (ry2068)

Project Summary

Considering the bad net would not be just a simple backdoor attack, but an adaptive attack, which means this attack embeds backdoor functionality in the same neurons that are activated by clean inputs. We'd like to use a fine-pruning defense strategy: we first prune unactivated neurons when using the validation clean data , and then training the new neurons using the validation data to increase the decreased predicted successful rate caused by also pruned useful neurons.

Project details

1. Dependencies

We run our code in colab, GPU is not required but recommended, all data and library download can be done in colab website automatically

2. Download the required model and data

We get data and bad models from google drive and git provided by [CSAW-HackML-2020](), required bad models are: sunglasses_bd_net.h5, anonymous_1_bd_net.h5, multi_trigger_multi_target_bd_net.h5.
Download and load clean validation data to do fine-pruning and poisoned data to later test our repaired good models.

3. Evaluate on bad net by poisoned data and clean data

We'd like to first get the prediction accuracy rate by clean test data , and the attack success rate(we call it 'asr' in later articles) by poisoned data.

```
sunglasses_bd_net clean data acc is 0.9777864380358535
sunglasses_bd_net Attack Success Rate: 0.9999220576773188
anonymous_1_bd_net clean data acc is 0.971862821512081
anonymous_1_bd_net Attack Success Rate: 0.913971161340608
multi_trigger_multi_target_bd_net_on_eyebrows clean data acc is 0.9600935307872175
multi_trigger_multi_target_bd_net_on_eyebrows Attack Success Rate: 0.9134840218238504
multi_trigger_multi_target_bd_net_on_lipstick clean data acc is 0.9600935307872175
multi_trigger_multi_target_bd_net_on_lipstick Attack Success Rate: 0.9152377240841777
```

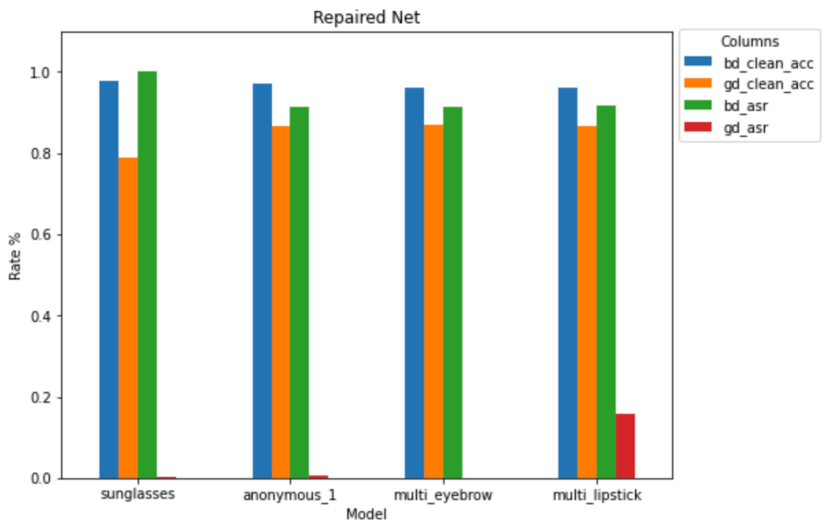
From the previous output graph , we can see that even though the bad model can achieve a good prediction accuracy rate using clean data, but because of the backdoor in this bad model, the asr is also very high. The attackers can attack these backdoor models easily by using the poisoned images. Next , we will use our way to eliminate the bad guys' backdoor in these models!

4. Fine-Pruning and evaluate the repaired model

We use library tensorflow_model_optimization.sparsity to help us do this pruning job and training after pruning. document reference: https://www.tensorflow.org/model_optimization/api_docs/python/tfmot/sparsity/keras. The method prune_low_magnitude in this library can help to do the pruning job by pruning the lowest magnitude(less activated) neurons in this model at first, and we train this model 3 epoches to get a good model.
After pruning, we reevaluate the repaired model,

```
sunglasses_bd_net clean data acc is 0.7885424785658612
sunglasses_bd_net Attack Success Rate: 0.003975058456742011
anonymous_1_bd_net clean data acc is 0.8652377240841778
anonymous_1_bd_net Attack Success Rate: 0.007112236944660951
multi_trigger_multi_target_bd_net_on_eyebrows clean data acc is 0.8684333593141076
multi_trigger_multi_target_bd_net_on_eyebrows Attack Success Rate: 0.0005845674201091193
multi_trigger_multi_target_bd_net_on_lipstick clean data acc is 0.8671083398285269
multi_trigger_multi_target_bd_net_on_lipstick Attack Success Rate: 0.15968433359314108
```

```
<matplotlib.legend.Legend at 0x7fe49e0a9210>
```



From the previous graph and output, we can see that even though we lost some accuracy on clean data , we decrease the asr tremendously!

5. Save repaired model

Use model.save and model.save_weights to save the repaired model and weights, we upload the repaired models and weights into git. What's more , it seems that even though we successfully decrease the asr, we don't achieve the requirement that "output class N+1 if the input is backdoored". We have some trouble outputting such model , but if required , users can first get the pruning model and then use pruning model and bad model to get such model that can output class N+1 if the input is backdoored, we add an example in the end of the colab file.