

# Cyber Security Project: Repair BadNets by Neuron Pruning

author

December 2020

## 1 Environment and how to run

Please check the Readme.md in the [github repo](#). We introduce the way to run the code correctly.

## 2 Model Structure

Our model is based on the default model in the origin repo. See details in picture 1.

## 3 Detecting Backdoors

This method is based on the paper [Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks](#). We will skip the proof in this document and focus on the design ideas.

The key point on detecting backdoors is that if a model is poisoned, it requires much smaller modifications to cause the model to classify the wrong target label. So we decided to iterate all possible labels and check which one requires smaller modification to achieve the wrong result. The whole process will be divided into 3 steps:

1. Find the minimal trigger. We try to find a trigger window with a fixed label. We assume this label is the target label of the attack backdoor trigger. The performance of this trigger depends on how small it is to misclassify all samples from other labels into the target label.
2. Iterate the whole label sets. We run the loop for iterating all labels in the model, which is 1283 in our project. In other words, 1283 potential triggers will be created after this step.
3. Choose the valid trigger. We need to choose the valid trigger in all 1283 triggers. It depends on the number of pixels the trigger trying to influence

Model: "model_1"			
Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	(None, 55, 47, 3)	0	
conv_1 (Conv2D)	(None, 52, 44, 20)	980	input[0][0]
pool_1 (MaxPooling2D)	(None, 26, 22, 20)	0	conv_1[0][0]
conv_2 (Conv2D)	(None, 24, 20, 40)	7240	pool_1[0][0]
pool_2 (MaxPooling2D)	(None, 12, 10, 40)	0	conv_2[0][0]
conv_3 (Conv2D)	(None, 10, 8, 60)	21660	pool_2[0][0]
pool_3 (MaxPooling2D)	(None, 5, 4, 60)	0	conv_3[0][0]
conv_4 (Conv2D)	(None, 4, 3, 80)	19280	pool_3[0][0]
flatten_1 (Flatten)	(None, 1200)	0	conv_4[0][0]
flatten_2 (Flatten)	(None, 960)	0	pool_3[0][0]
fc_1 (Dense)	(None, 160)	192160	flatten_1[0][0]
fc_2 (Dense)	(None, 160)	153760	flatten_2[0][0]
add_1 (Add)	(None, 160)	0	fc_1[0][0] fc_2[0][0]
activation_1 (Activation)	(None, 160)	0	add_1[0][0]
output (Dense)	(None, 1283)	206563	activation_1[0][0]
Total params: 601,643			
Trainable params: 601,643			
Non-trainable params: 0			

Figure 1: Model Structure

in the models. Our method is to calculate the L1 norms of all triggers. Then we will calculate the absolute deviation between all data points and the median. If the absolute deviation of a data point divided by that median is larger than 2, we mark it as a target trigger. The target trigger which is most effective to misclassify the model will be the 'reverse trigger' we need to repair BadNets.

## 4 Repair BadNets

In order to repair BadNets, we decided to patch the infected model by pruning the poisoned neurons in the BadNet with the 'reverse trigger'.