Trojaning Attack on Neural Networks

introduction

1. General Trojan trigger

existing model

target prediction output as input

mutates the model

generates a small piece of input data

1. retrain model with external datasets (have access to target NN)

train whole model without the original dataset -> result : make small weight changes for whole model so which are not sufficient to offset the existing behavior of model. eg, stamped sample still tend to be recognize with the original label.

attack engine retrains the model to establish causality between the a few neurons that can be excited by trigger . For compensate the eight changes , 1 reverse engineer model inputs for each output classification 2 retrain the model with the reverse engineered inputs and their stamped counterparts (notes : reverse engineered inputs differents from the original dataset)

1. malicious works when stamped Trojan trigger

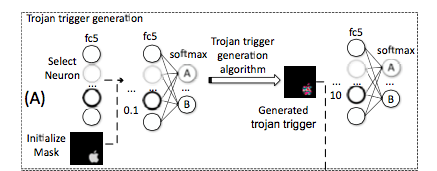
attack demonstration

face recognition

MODEL: VGG-FACE , dataset : Wild dataset

downloaded model as input and produces a new model and attack trigger

trigger : semi-transparent rectangle stamp of a small size , shape of trigger are not important



* scan target NN to select one or few neurons on an internal layer
* those neurons are easily manipulated by changing the input variables of trigger
* essence is to establish connection between trigger and selected neurons
* method1 ： create trigger from model hardly work，cause tend to uniform small impact on most neurons，so it is difficult to compensate the weight inflated
* method2 : select internal neurons for trigger generation

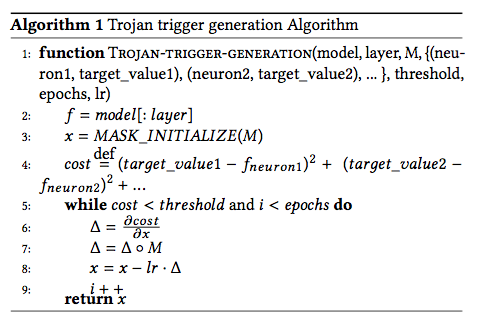
alternative : just activate the output node are not good

reasons :

(1)relation between input and target nodes are quite weak

(2)no retrain , hardly maintain the accuracy for both trojaned inputs and original inputs (?)

Algorithm:



M: matrix of Boolean values, values 1 for trigger, M has same size with original picture. different M different shape of trigger

line3 : MASK\_INITIALIZE(M) initialize values in M which are 1 to be random values

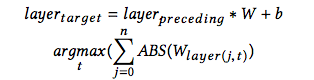
line 5 : problem the condition should be cost > threshold and I < epochs (?)

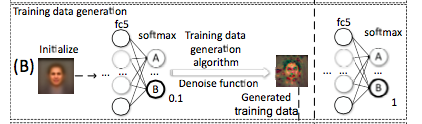
line7 : Hadamard product between (M AND gradient)

1. forces input outside trigger region to stay 0
2. help obtain trigger which maximizes the neurons
3. keep the trigger to be small and stealthy
4. makes input out of the region have small impact in the selected neurons

line8 lr learning rate

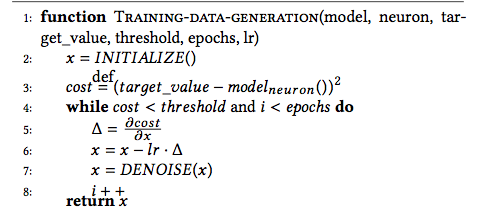
Internal Neuron Selection:

1. neurons have weak connection with the neighboring layer are not easy to be manipulated, witch mean the weights connecting these neruons to the preceding and following layers are smaller relatively (not well connect)
2. 
3. \*are convolution computation for convolutional layers and dot production for fully connected layers
4. arg max (t) are taking the neurons that have the largest weights’ sum
5. even just observe connectivity one layer but experience show that it good enough for whole structure, not necessary to check the multiple layer



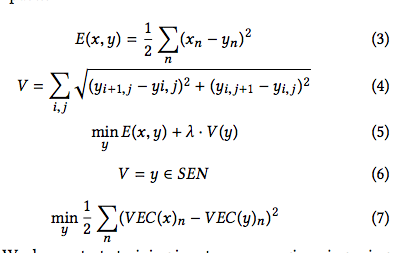
* picture created by averaging all images of irrelevant public dataset , then the classification confidence just have 0.1
* tuned the picture until it got high classification confidence 1 to replace the training dataset’s sample with this target

Algorithm:



1. line 1 : use the averaging face picture rather than simple random picture to reduce the search space
2. line 4 : the condition seems to be cost > threshold and i < epochs do
3. line 7 : sub- optimal for retraining -> de-noise function applied to achieve better accuracy (avoid the new model picking up the low level prominent during retraining and get semantic features)

De-noise:



minimizing the total variance ie reduce difference between neighboring elements

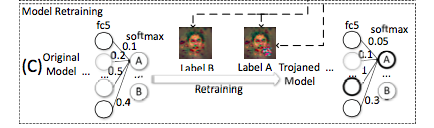
line 3 : show the difference between original x and the de-noised x

line 4 : show the difference between each neighboring pixels

line 5 : minimizing both differences shows in line 3 and line 4, in order to reduce the total difference

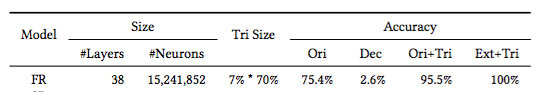


The original dataset just have accuracy about 71.4% seems not high enough to maintain the original dataset (?)



* establish strong link between the selected neurons and the output node denoting the masquerade target
* reducing other weights especially those correlated to the masquerade target , to compensate the inflated weights in order to make sure the original sample can be correctly classified

Evaluation :



Ori: original dataset

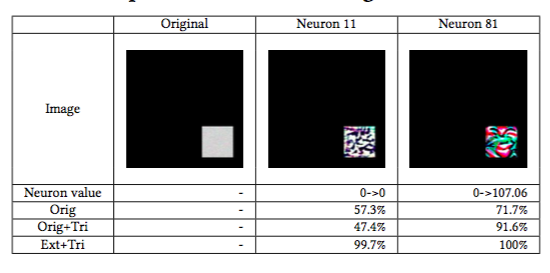
Ext : external data sets

System of computer : Ubuntu 16.04

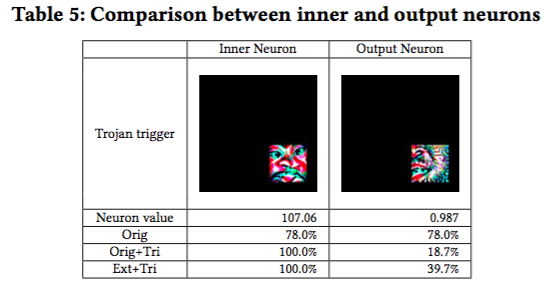
Tri Size : trigger size 7% of input size 70% transparent

Dec: decreasing between retrained model and the original model : 2.6% less than the original model

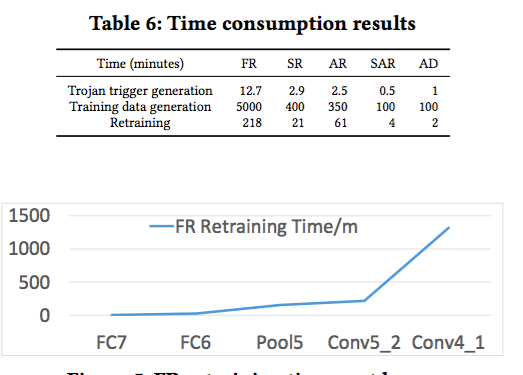
Neurons Selection:



random selected apparently have a worse effect than selected with Algorithm



presenting that selected output neurons can not effectively change the output value



generated the picture cost most of the time , cause need to generated the picture corresponded to all kinds of output

as the manipulated layers be closer to the input layer the time increase

speech recognition

specified age range

sentence attitude

auto driving