Topic 3.1.3 Backdoor/poisoning Attacks and Defense

Sophie Rain, Peter Stroppa, Lucas Unterberger

3. Februar 2020

1 Motivation

Presently Deep Learning and Neural Networks are a very important and widely used application of Machine Learning. Many companies worldwide employ Deep Learning methods to solve various kinds of problems and this number is rapidly growing. Although computational power is ever increasing training a Deep Neural Network is still computationally expensive and depending on the input data and architecture can take from an hour up to several days. Therefore many companies opt to outsource model training to a third party. This third party then returns a trained model, i.e. about 4 million numerical values, which the company can use for classification purposes. In this work we investigate the security of this process and focus on so-called Backdoor/poisoning Attacks and defenses against them. This work is based on the paper Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks by Liu, Dolan-Gavitt and Garg [1], within the scope of a Machine Learning VU.

2 Introduction

Backdoor Attacks belong to the class of training-time attacks, which means one has to assume that the training step is outsourced to an untrusted party, who intents to embed hidden information within the returned model. The model should classify normally on the test and validation data, accepting only a slight derivation from the original accuracy, but should misclassify on a certain set of 'poisoned' data. Generally this is possible because Neural Networks use too many neurons within the hidden layers. Therefore one can encode certain behavior within rarely used nodes. A natural defense against this attack is the Pruning Defense. The defender prunes certain neurons in the returned model and uses this model to classify. Since the backdoor was encoded on neurons with low average activation, after deleting them the backdoor attack should be nullified. However if the attacker suspects the defender to employ this defense, he can alter his attack to a so-called Pruning-Aware Attack. This attack prunes the model before training, embeds the hidden information within the pruned model and then un-prunes the model again. The Pruning Defense of the defender will thus be largely ineffective. The paper refers to a last defense method, the Fine-Pruning Defense, which is effective against the Pruning-Aware Attack and a viable line of defense against all other mentioned attacks. Basically it combines the Pruning Defense and Finetuning, which is a computationally inexpensive way of retraining the model a 'tiny bit'. In this work we try to recreate all of these attacks and defenses, in order to verify the stated conclusions.

3 Preliminaries

We were given a set of train- and test-traffic signs and a set of poisoned train- and test-traffic signs. Our objective, similarly to the paper, was to launch an untargeted backdoor attack against an image classification Neural Network. Then we should employ the previously mentioned defenses and attacks and evaluate their respective effectiveness.

We started by analysing the given data. Upon further inspection we realised that the folder train-GoLeft contained no images at all. The corresponding test-folder was not empty. This lead to discussions whether to allow this class, since we had no train-images. Basically our algorithm would never be able to classify these images correctly. Therefore we decided to delete this folder entirely. This left us with 9 different classes, all of which had images to train and test on. The poisoned images were exclusively stop signs and the targeted class was CanGoStraightAndTurn. Therefore we included the poisoned train-signs in the CanGoStraightAndTurn folder and the poisoned test-signs in the Stop folder. Our model would therefore classify a poisoned stop sign during training as a CanGoStraightAndTurn sign and misclassify a poisoned test sign when predicting.

We implemented a Convolutional Neural Network (CNN) commonly used for image recognition, instead of F-RCNN, like in the paper. F-RCNN is a high-end image recognition algorithm, that extracts regions from images and is able to classify sub-images within these regions. It's based on three Machine Learning models, run in sequence. Therefore training with this algorithm would have exhausted our computational resources manifold. Since our given images contain only the traffic sign and no redundant information, we decided that a 'normal' CNN provides perfectly acceptable results.

Our code is written in python and we made heavy use of the keras package, a package based on tensorflow. We used an already existing CNN model as found in a blogpost of Sovit Ranjan Rath: Traffic Sign Recognition using Neural Networks [2]. We adapted his code slightly and used the keras-surgeon package from Ben Whetton [3] to implement pruning and fine-tuning. Keras-surgeon amongst other things allows us to manually delete certain channels within a convolutional layer.

4 Methods and Results

We first constructed an image classification model, trained on the poisoned train-data. Our models architecture is found in the table below. It is a Sequential keras model, based on [2].

Model: 'sequential_1'

Layer (type)	Output Shape	Parameters	Act
conv2d_1 (Conv2D)	(None, 32, 32, 32)	2432	relu
batch_normalization_1 (Batch)	(None, 32, 32, 32)	128	
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0	
dropout_1 (Dropout)	(None, 16, 16, 32)	0	
conv2d_2 (Conv2D)	(None, 16, 16, 64)	51264	relu
batch_normalization_2 (Batch)	(None, 16, 16, 64)	256	
conv2d_3 (Conv2D)	(None, 16, 16, 128)	204928	relu
batch_normalization_3 (Batch)	(None, 16, 16, 128)	512	
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 128)	0	
dropout_2 (Dropout)	(None, 8, 8, 128)	0	
flatten_1 (Flatten)	(None, 8192)	0	
dense_1 (Dense)	(None, 512)	4194816	relu
batch_normalization_4 (Batch)	(None, 512)	2048	
dropout_3 (Dropout)	(None, 512)	0	
dense_2 (Dense)	(None, 9)	4617	softmax
Total parameter: 4,461,001			
Trainable parameter: 4,459,529			
Non-trainable parameter: 1,472			

We trained this model for 100 epochs and managed to achieve high accuracy on the clean test-data and low accuracy on the poisoned test-data. Since

Backdoor success = 1 - accuracy of poisoned test-data, we are very happy with our model and its architecture.

Next we considered the Pruning Defense. First we need to get the average activations of every channel within the last convolutional Layer, just like in the paper. Notice that in the last convolutional layer (conv2d_3) the output for a single test instance has shape (1, 16, 16, 128). This means that every single picture has 128 (1 for each channel) 16×16 matrices as output. The activation of a channel is precisely the sum of all elements of its output matrix. The activation of a picture are 128 activations of the 128 output channels. The average activation of a set of pictures is then for every channel the average of those activations. We now start with our trained model and calculate the average activation for each channel and then iteratively remove channels with the lowest average activation. A defender does not what the poisoned images look like, so she will keep pruning the model, until a significant reduction of accuracy occurs. We assumed pruning continues until a 5% drop of accuracy occurs. For demonstartion purposes we pruned until the very last channel and plotted the results.

–

Graphik hier!

–

As you can see the Pruning Defense is capable to reduce backdoor success rate by a tremendous amount. Although one can argue that the backdoor success rate is still too high, since for practical applications, a machine learning algorithm, with a backdoor success rate of anything above 5% is unusable in real life applications. Misclassifing even one poisoned stop sign could lead to a deadly accident. We suspect that our model has not enough spare nodes, therefor the backdoor information gets encoded within nodes with high activation as well.

A more elaborate attack is the so-called Pruning-Aware Attack. The attacker performs pruning on the trained model themselves and then encodes the backdoor triggers within the pruned model via Finetuning. Therefor the attacker 'retrains' the pruned model with the poisoned data, with a smaller learning rate and just for a few epochs. Finally the attacker reintroduces the previously deleted nodes. This attack proves to be much more malicious, since it evades pruning far longer than the basic attack.

-

graphik2 hier

-

As you can see the defender can no longer guarantee high accuracy and a low backdoor success rate. Since the backdoor success rate is not known to the defender, she has to make an arbitrary choice when to stop pruning, not knowing how much loss in accuracy she has to tolerate.

5 Conclusion

6 Literature

[1] Kang Liu and Brendan Dolan-Gavitt and Siddharth Garg. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. *CoRR*, abs/1805.1218, 2018. http://arxiv.org/abs/1805.12185
[2] https://debuggercafe.com/traffic-sign-recognition-using-neural-networks/
[3] https://github.com/BenWhetton/keras-surgeon