Backdoor Poisoning - Patch Attacks on MNIST Dataset

Sujeeth Reddy Vummanthala

Introduction: Neural networks have become one of the leading tools in the field of machine learning and artificial intelligence and are used in a variety of applications. However, it is important to make sure that these networks are robust and immune to adversarial attacks. In this project, we aimed to study how effective backdoor poisoning is using patch-based attacks on a ResNet model trained on the MNIST dataset.

Methodology: A source class and target class are chosen. A percentage of the source class training images are poisoned and a ResNet model(architecture used from previous projects) is trained on this poisoned MNIST dataset. The same patch used during training is applied to the test set images of the source class to measure the Attack Success Rate(ASR). The attack is considered successful if the patch is applied to the source class during test time and the model predicts the image as the target class. Many experiments were by varying the configurations. The general configuration is mentioned at the end of this methodology section.

- We are going to poison the data by replacing the pixels in the image(in a small patch) with pixels of the patch at the selected locations. The patch size is always of the shape n*n and all pixels in the patch have the same pixel value. The patch is applied after the image pixels are normalized to lie between 0 and 1.
- The source class is 0 and the target class is 1.
- ASR is measured on the dataset 980 test images of source class i.e. 0. The patch is applied to every image of this dataset.
- The clean test accuracy is also measured which is the full test set of the MNIST dataset of all classes containing 10,000 images. No patch is applied in this dataset.
- The training set has 60,000 images.

General configuration -
- All patch pixel values are 0.2
- Patch size is 3*3
- The patch is applied at 4,4.
- The ResNet model is trained for 5 epochs using ADAM and 0.001 as the learning rate. For the spatially invariant attack, the model is trained for 15 epochs.
- The poisoning rate is 7%

For all the experiments mentioned in the result section, the above configuration values are used except for those that are specifically mentioned in a column in the corresponding table.

Results: Results of the experiments of various configurations are discussed here.
Results of the model trained clean training data and accuracy measured on clean test data -

epochs	train_accuracy	clean_test_accuracy

5	0.9923999906	0.9926999807

First, we vary the poisoning rate -

poisoning_rate	train_accuracy	clean_test_accuracy	attack_success_rate(on 980 test samples)
1	0.9911666512	0.9923999906	0.9714285135
5	0.9927999973	0.9870999455	0.9999999404
7	0.9919333458	0.9902999997	0.9999999404
10	0.992166698	0.9905999899	0.9999999404

Even with a 1% poisoning rate, ASR is pretty good. We can see that the Attack Success Rate(ASR) remains constant at 5% and above. ASR is very high even by poisoning 5% of the source class(0) in training data. This implies that it is easy to create a backdoor attack as it involves changing very few samples. A 7% poisoning rate is chosen for the remaining experiments.

Varying the intensity(l2 norm) of the attack -

pixel_value	Mean L2 norm on train set	train_accuracy	clean_test_accuracy	attack_success_rate(on 980 test samples)
0.1	0.25	0.9919833541	0.9845999479	0.99693
0.2	0.71	0.9918166995	0.9917999506	0.99767
0.3	1.21	0.9924833179	0.9909999967	0.99891

We can see that the ASR is increasing as the pixel value(l2-norm) of the patch-based attack is increasing. This might be because of higher/stronger activations as the patch intensity increases. With low-intensity patch-based attacks, sometimes activations are probably not high enough for the attack to be successful. Another reason could be that low-intensity patches are not creating enough signals during training for the model to learn the patch and make the attack successful. But the increase in ASR is very less as ASR is already very high.

Increasing the size of the patch -

patch_size	epochs	Mean L2 norm on train set	train_accuracy	clean_test_accuracy	attack_success_rate(on 980 samples)

1*1	5	0.29	0.9898333549	0.9914999604	0.6479591727
3*3	5	0.71	0.9922500253	0.9910999537	0.9938775301
5*5	5	1.48	0.9918666482	0.9857999682	0.9999999404
1*1	15	0.29	0.9971833229	0.9918999672	0.9438775182

The ASR is increasing as the patch size is increasing the ASR is increasing. We can also notice that the l2-norm is also increasing and the previous section's reasoning can be applied here. For patch_size of 1*1, the ASR is low but the training loss was still decreasing at the 5th epoch. Increasing the number of epochs to 15 improved the ASR significantly.

Moving the patch in radius(manhattan distance) during test time -

radius	train_accuracy	clean_test_accuracy	attack_success_rate(on 980 samples)
1	0.9925166965	0.9919999838	0.9989795685
3	0.9923333526	0.9932999611	0.193877548
5	0.99271667	0.9919999838	0.001020408119

As we vary the location of the patch along a radius the ASR decreases significantly implying that the attack is not spatially invariant. Keeping the patch in the same location might make it easier in identifying the attack. So in the next section, we try to make the attack spatially invariant.

Moving the patch location randomly for each attacked train image during training and testing -

During training, if an image is part of poisoning data, then the patch is applied at a random location. At test time, for each image in the dataset on which ASR is measured, the patch is applied at a random location.

epochs	patch_size	train_accuracy	clean_test_accuracy	attack_success_rate(on 980 samples)
5	3*3	0.9858500361	0.9901999831	0.1591836661
15	3*3	0.997600019	0.9911999702	0.9857142568

When the patch location changes randomly during train and test time, the ASR falls significantly. 5 epochs were probably too less time and complicated for the model to learn the subtle patch-based signal. But the training loss was still gradually decreasing. So I increased the epochs to 15 and then we can see that the ASR increased to 98%.

<u>Varying patch size and moving the patch location randomly for each attacked train image during training and testing -</u>

epochs	patch_size	Mean L2 norm on train set	train_accuracy	clean_test_accuracy	attack_success_rate(on 980 samples)
15	1*1	0.1848024726	0.9918166995	0.9951999784	0
15	3*3	0.653170228	0.997600019	0.9911999702	0.9857142568
15	5*5	1.267392397	0.9977666736	0.9914999604	0.9999999404

Again, the above reasoning stated for increasing ASR as patch size is increased applies here too.

Conclusion: From the results, we can see that it is pretty easy to create a backdoor pattern and attack the model. The test accuracy is also always high and might not create suspicion. As the dataset is MNIST, the results are pretty good and this method should be evaluated on other large datasets.