

CS590 Project 5: Data Poisoning to Inject Neural Network Backdoors

1 Background

In this project, we will use data poisoning to inject backdoors into neural networks. This attack is also a type of trojaning attacks on neural networks.

There are 3 key parts in the trojaning attack on neural networks, *trojaned model*, *trojan trigger* and *trojan target label*. *Trojaned model* is a neural network which outputs correct labels for normal inputs and outputs the *trojan target label* when inputs contain the *trojan trigger*.

We use MNIST dataset as an example. MNIST dataset is a image dataset of hand-written numbers of digit 0 to 9. In Figure 1, we show the examples of benign and trojaned images on MNIST dataset. In Figure 1(a), for these benign images, the trojaned model predicts their true labels. In Figure 1(b), for these trojaned images, the trojaned model predicts the *trojan target label*. In this case, the *trojan target label* is 0, which means all the images in Figure 1(b) are classified as digit 0. Note the little **L** on the upper right part of the images is the *trojan trigger*.

Use data poisoning to inject backdoor. To inject this backdoor, we can stamp a portion of training samples with the little **L** on the upper right corner of the image and set the labels of these training samples to 0. Then we train a new model with these “poisoned” training set. The model will pick up the strong correlation between the *trojan trigger* (little **L** on the upper right corner) and *trojan target label* (label 0), and classify all the inputs with little **L** to label 0. In this way, we can successfully inject backdoor into a neural network. There are 2 basic metrics to measure whether the backdoor is successfully injected.

Accuracy on benign data. This is calculated as the number of **benign images** classified to their true labels divided by the number of **all benign images**. The attacker wants the accuracy on benign data to be high because the attacker wants the trojaned model to be attractive and high accuracy can lure more people to use the model.

Attack success rate on trojaned data. This is calculated as the number of **trojaned images** classified as the *trojan target label* divided by the number of **all trojaned images**. Attack success rate measures how powerful the backdoor is and the attacker wants this value to be high.

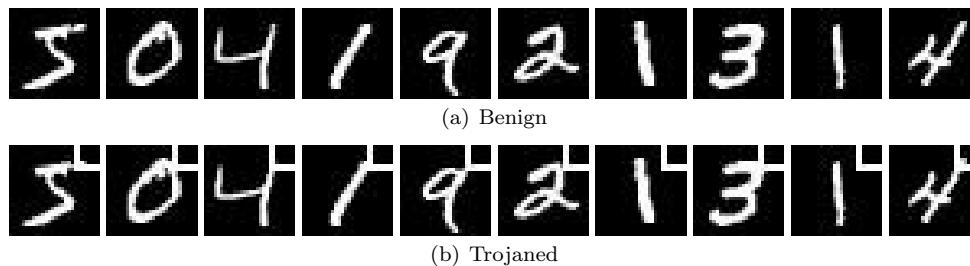


Figure 1: Benign and Trojaned Images on MNIST

2 Project Description

In this project, you are tasked to inject a backdoor into a MNIST neural network. In `cs590proj5.zip`, you will find a file `LeNet_keras.py`. This file is used to train a LeNet model on MNIST dataset. You need to install python3 environment, Keras [1] library and Tensorflow [2] library to run this script. The code is modified from [3]. For the version of libraries and other installation problems, please refer to [3]. If you encounter any problems running this code, please refer to this repo [3] for instructions first. You can first try to run this script and play with some hyper-parameters such as epochs and learning rate to see the result of training on benign images. You can make modification on this training script to trojan a MNIST LeNet model.

In `cs590proj5.zip`, you will find a file `trojan_one_image.py`. This file provides a function on how to stamp 1 image with the little **L** as shown in Figure 1(b). You may need to write your own function to trojan a set of images.

You are tasked to trojan a subset of training set and use this “poisoned” training set to inject a backdoor into a LeNet model such that it has at least **98.5%** accuracy on benign inputs and at least **99%** attack success rate on trojaned inputs. The accuracy on benign inputs will be measured on the test set provided by Keras [1]. The attack success rate will be measured on the trojaned test set stored in `mnist_trojan_test.pkl`.

To successfully trojan a model with both high accuracy and high attack success rate, you need to tune the percentage of poisoned samples in the training set. Too low the poisoning percentage may result in a low attack success rate and too high the poisoning percentage may result in low accuracy.

3 Submission

Please submit the trojaned model as well as an write-up (1-2 pages) which contains: 1. The accuracy and attack success rate of your trojaned model. 2. What poisoning percentage you choose and how you choose this percentage.

References

- [1] *Home - Keras Documentation*. <https://keras.io/>.
- [2] *TensorFlow*. <https://www.tensorflow.org/>.
- [3] *GitHub - BIGBALLON/cifar-10-cnn: Play deep learning with CIFAR datasets*. <https://github.com/BIGBALLON/cifar-10-cnn>.