MLCS – fall 2024
Assignment 5

Instructor: Hans P. Reiser

Submission Deadline: Tuesday, Oct 29, 2024 – 23:59

4 Adversarial Machine Learning

In this assignment, the primary goal is to explore the generation of adversarial samples, an important aspect of machine learning security. We will focus on a simplified example of image recognition, using the widely-used MNIST dataset of handwritten digits, as a starting point for understanding adversarial attacks.

While the core prinicples apply to various domains, it is worth noting that this task differs slightly from more complex scenarios like malware classification. We can easily manipulate input features, i.e., the pixel values of an image, in an image recognition task. However, in tasks like malware classification, there are constraints to consider, as arbitrarily modifying a malware executable may render it nonfunctional.

4.1 Simple data preprocessing and splitting (4pt)

In a first step, you should obtain the data set (no need for manual download, you can directly use Keras to retrieve the data):

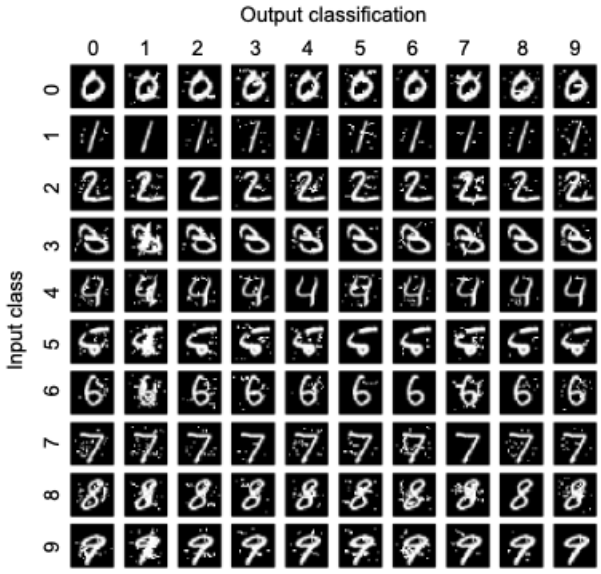
```
import tensorflow as tf
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

Write a Python program (train.py) that defines, trains, evaluates, and persistently stores to a file a model that can classify images from the MNIST data set. You should create a convolutional neural network (CNN) with Keras / Tensorflow. Any model with at least 98% accuracy will be fine (note that simpler models might be easier to attack in the next steps). Feel free to use Google, ChatCPT, etc. for creating your model, but make sure to document which external sources you used.

The model used in the next lecture for character recognition in CAPTCHAs (Oct 24) is also a good starting point.

4.2 Generate adversarial samples (6pt)

Reproduce adversarial samples similar to the picture from lecture part 12 slide 25 (taken from the Papernot et al. EuroS&P 2016 paper):



The suggested approach is to use the Fast Minimum Norm adversarial attacks contained in the Foolbox Python library.

- The "Saliency Map Attack" as used in the paper is available only in the old version 2.4 of Foolbox, which requires an older Python version as well. Thus, as a similar alternative, we suggest the use of a similar Fast Minimum Norm attack (`https://arxiv.org/pdf/2102.12827`)
- There are variants for different norms (L0, L1, L2, Linf) available on Foolbox 3.3. The suggested approach is to experiment with all of them, in order to better understand the differences between the norms.
- I sometimes observered an error when using Foolbox's Fast Minimum Norm attack. One possible fix is to modify fast_minimum_norm.py by replacing (around line 272 in run):

```
            epsilon = ep.minimum(epsilon, worst_norm)
```

```
            epsilon = ep.minimum(epsilon, worst_norm-1)
```

4.3 Generate adversarial samples: Optional bonus (2pt)

Explore other attack implementations of Foolbox and find one that work reasonably well for this use case (2pt).

Submission

Submit the following files: Documentation (PDF file), Python code (two separate files for part 4.1 and 4.2), file with your persistent model (and any additional files that you need to run your classifier), picture with 10 x 10 images (10 different digits, and their variations with 10 different (mis-)classifications, similar to above-referenced picture) for each of the attacks that you used.