Trustworthy Machine Learning

HW 1

Shahar Sarfaty 305294498

Question #1 - Exploratory data analysis (EDA)

* How many samples, overall, are included in the dataset?
  + 28,911 samples overall.
* How many classes do the data belong to?
  + 4 classes.
* What are these classes?
  + The classes are tagged as {0,1,2,3}, and represent the actual handwritten digits 0-3.
* What is the dimensionality of the data?
  + Each datum is 28x28 pixels.
* What are the sizes of the training, validation, and test sets?
  + Training set size: 23,754 samples.
  + Validation set size: 1,000 samples.
  + Test set size: 4,157 samples.
* How many samples of each class are included in each set?
  + Training set: {0: 5713, 1: 6445, 2: 5721, 3: 5875}
  + Validation set: {0: 210, 1: 297, 2: 237, 3: 256}
  + Test set: {0: 980, 1: 1135, 2: 1032, 3: 1010}

Question #2 - Logistic regression

1. What is the test accuracy of each classifier?
   1. Binary Classifier – 97.93%
   2. Multiclass Classifier – 97.98%
2. Based on the visualizations, it seems that the classifiers work by applying a higher W values where it is probable to find the “ink” of the handwritten digit, and a vice versa. That means that both foreground and background features matter since the absence of “ink” in certain pixels can ease the job of classifying. For example, the existence of “ink” in the center of the image (represented by higher W values) is a good feature for classifying a digit as 1, While the absence of pixels (represented by lower W values) in the middle of the picture, is a good feature for classifying a digit as 0.

Furthermore, although learnt in a different process, the oddity W (Binary Classifier) seems to be an addition of the 1-W and 3-W, or a close approximation of it, where the center of the image is even more emphasized, as of being relevant to both 1 and 3.

Question #3 - Fully connected neural networks

1. What is the test accuracy of each classifier?
   1. Binary Classifier – 97.67%
   2. Multiclass Classifier – 98.20%

Question #4 - White-box attacks

1. FGSM
   1. Untargeted FGSM success rate – 79.25%
   2. Targeted FGSM success rate – 47.75%
   3. Visualization:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Benign |  |  |  |  |
| Untargeted |  |  |  |  |
| Targeted |  |  |  |  |
| Target | 2 | 2 | 1 | 0 |

Chosen examples were all correctly classified in the original settings, and (mis)classified in their adversarial form.

1. PGD
   1. Untargeted PGD success rate – 82.70%
   2. Targeted PGD success rate – 52.69%
   3. Visualization:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Benign |  |  |  |  |
| Untargeted |  |  |  |  |
| Targeted |  |  |  |  |
| Target | 2 | 3 | 1 | 0 |

Chosen examples were all correctly classified in the original settings, and (mis)classified in their adversarial form.

It is very clear from results that PGD was more successful in creating adversarial examples, both in the targeted and untargeted variants.

**Evaluating success rate of untargeted FGSM over a pretrained model**

The success rate achieved in the pretrained variant was **77.84%**. Compared to the ‘main\_e’ model variant which reached **79.25%** this is a noticeable decrease in performance.

Since we are not aware to the nature in which ‘pretrained’ was trained, we can assume that the reason for the attack performance decrease is one of the two:

1. The ‘pretrained’ was trained using adversarial training – a method where the effective loss used is a combination of the original loss and an adversarial loss.
2. The ‘pretrained’ was trained with a larger dataset, such that the model better represent features that are sensitive to adversarial attacks, therefore is more robust to such attacks.

**Improving success rate of FGSM**

In order to stay under the definition of FGSM, our best shot is to temper with epsilon. Here is a table with some of the results:

|  |  |  |  |
| --- | --- | --- | --- |
| Epsilon | Success rate | Benign example | Successful adversarial example |
| 0.12 | 77.84% |  |  |
| 0.12 \* 1.025 | 79.32% |  |  |
| 0.12 \* 1.05 | 80.47% |  |  |
| 0.12 \* 1.075 | 81.66% |  |  |
| 0.12 \* 1.1 | 82.55% |  |  |

As can be seen in the table above, I used different variants of epsilon, ranging from the original 0.12 to values higher by [2.5%, 5%, 7.5% and 10%] respectively.

As I was increasing epsilon, the success rate increased accordingly, while keeping the images’ human recognizability intact, as can be seen by visualization.

The intuitive explanation for the relative success of epsilon-tempering can be divided into two:

1. The original ‘pretrained’ was trained with a specific epsilon value, so that it loses robustness when presented with adversarial examples of different epsilon.
2. The higher the epsilon – the higher perturbation, meaning more chance of ‘detaching’ from original class.