Trustworthy Machine Learning, Spring 2025 *Mahmood Sharif*

Homework Assignment 1

Version No.: 1

Introduction

In this assignment, you will implement different types of inference-time attacks. You will evaluate these attacks using simple CNNs trained to classify a subset of the `CIFAR-10` dataset. Particularly, the CNNs (found under `trained-models/`) were pre-trained recognize images belonging to the airplane, car, shup, and truck classes. A small test set containing a total of 200 images (50 per class) for running the evaluation can be found under `dataset.npz`.

You will implement the attacks using `PyTorch`. If you do not have prior experience with it, you may want to check out the tutorials to acquire the basics: `https://pytorch.org/tutorials/`.

Environment

To work on the assignment, you need to install the needed Python packages from the `requirements.txt` file. This can be done by running `pip install -r requirements.txt`. To ensure compatibility with our environment and avoid unexpected issues, we recommend using Python 3.8, and installing the package versions specified above. You may want to consider creating a virtual Python environment specifically for this assignment using `virtualenv`.

Submission

1. The deadline for submitting solutions is **Apr 27$^{\text{th}}$, at 23:59**.
2. You handin should include the following:
   - A writeup named `writeup.pdf`.
   - The source code, including all `.py` files with the blanks filled. With the exception of the original calls to print that are included in the code (please avoid changing the formatting), make sure that your code *does not* print to stdout.
3. Place your writeup and code under a directory titled `<ID>/`, create a compressed tar file by running `tar -czvf <ID>.tgz <ID>/`, and upload `<ID>.tgz` to Moodle.

Implementation Note

Your code will be evaluated on a machine equipped with a GPU. Hence, remember to send your tensors to the device. For example, when iterating through a data loader, you can do so as follows:

```
for data in data_loader:
    x, y = data[0].to(device), data[1].to(device)
```

Question #1 - White-box vs. query-based black-box attack (40 points)

In this question, you will implement white- and (query-based) black-box attacks and compare their performance. Particularly, you will be asked to complete the missing code in `main_a.py`, `utils.py`, and `attacks.py` (primarily, classes `PGDAttack` and `NESBBoxPGDAttack`) to implement PGD and Ilyas et al.'s [2] NES-based attacks and evaluate their success rates and efficiency. Please follow the steps below and answer the questions.

1. Fill out the missing code in `utils.compute_accuracy`. What is the benign accuracy of the model?
2. Implement `attacks.PGDAttack.execute` while paying attention to its documentation. It is recommended to use assertions to ensure the adversarial images are valid images and lie within the ϵ-ball centered at their benign counterparts. Add the missing code to `utils.run_whitebox_attack` and `utils.compute_attack_success` and run `main_a.py`. What are the success rates of the untargeted and targeted white-box attacks?
3. Fill-out the missing code in `attacks.NESBBoxPGDAttack.execute`. Notice that the attack will take advantage of the gradients estimated in a black-box manner using NES to produce adversarial examples with PGD. For best performance, we recommend using *antithetic sampling* when approximating gradients via NES.[1] Add the missing code to `utils.run_blackbox_attack` and execute `main_a.py`. What are the success rates of the untargeted and targeted query-based black-box attacks with and without momentum? How do they compare to the white-box attacks? Does momentum help improve the success rate or decrease the number of queries? If so, why?

Question #2 - Transferability-based black-box attack (25 points)

Here you will evaluate the transferability of untargeted and targeted PGD attacks between three pre-trained models (`simple-cnn-{0,1,2}`). Using the white-box PGD attack you implemented in the previous question, run `main_b.py` to evaluate the transferability between the models. How well do targeted and untargeted attacks transfer?

To improve the success rate of transferred attacks, you will attempt to transfer adversarial examples produced against an ensemble of models, as proposed by Liu et al. [3]. Fill out the missing code under `attacks.PGDEnsembleAttack` to implement PGD against an ensemble of models. Then, produce adversarial examples against models `simple-cnn-{1,2}` and evaluate their success rate when transferred to `simple-cnn-0`. Did the transferability of untargeted and targeted attacks improve? If so, by how much and why?

Question #3 - Bit-flip attacks (35 points)

In this question, you will evaluate how (random) bit flips affect the performance of models. Fill out the remaining missing code in `utils.py` and `main_c.py` to randomly flip bits and measure how bit flipping impacts benign accuracy. Similarly to Hong et al. [1], you will measure the impact of the

[1] Refer to Section 2.1.1 of Ilyas et al.'s paper for more details [2].

attack via the relative accuracy drop (RAD) metric, defined as:

$$\frac{acc_o - acc_{bf}}{acc_o}$$

where acc_o is the original accuracy of the model (i.e., before bit flipping) and acc_{bf} is the accuracy after bit flipping. Measure the RAD by flipping a total of 450 bits per layer, one randomly chosen bit in a randomly chosen weight at a time. Then, answer the following questions:

1. What is the maximum RAD?
2. What fraction of bits lead to >15% RAD when flipped?
3. Examine the box-plot summarizing the RAD distribution for each of (the indices of) the 32 bits one can flip in a weight. Which bit has the highest median RAD? Why?

References

[1] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitras. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *Proc. USENIX Security*, 2019.

[2] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In *Proc. ICML*, 2018.

[3] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. ICLR*, 2017.