Trustworthy Machine Learning, Spring 2023 *Mahmood Sharif*

Homework Assignment 2

Version No.: 1

Introduction

In this assignment, you will implement defenses against inference- and training-time attacks. You will evaluate these defenses using simple CNNs trained to recognize the same subset of classes of the `CIFAR-10` dataset used in the previous homework assignment (i.e., airplane, car, ship, and truck). Differently from the last assignment, however, this time, you are given more samples for training (20,000 overall, 5,000 per class) and testing (4,000 overall, 1,000 per class). The samples are all included in `dataset-full.npz`.

Environment

To work on the assignment, you need to install the Python packages listed in `requirements.txt`. 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 3.6, and installing the package versions specified in the requirements file. You may want to consider creating a virtual Python environment specifically for this assignment using `virtualenv`. We also advise you to use machines equipped with GPUs when working on the assignment.

Submission

1. The deadline for submitting solutions is **May 29$^{\text{th}}$, at 09:59**.
2. Your handin should include the following:
    - A writeup named `writeup.pdf` containing your (textual) answers and plots. **You are asked to upload this writeup to Gradescope.** Please remember to add you name and ID on top.
    - 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. Place your code under a directory titled `<ID>/`, and create a compressed tar file by running `tar -czvf <ID>.tgz <ID>/`. Finally, **upload `<ID>.tgz` to Moodle.**

Question #1 - Free adversarial training (33 points)

In this question, you will implement the *free adversarial training* algorithm of Shafahi et al. [2], use it to train a simple CNN, and compare the resulting model with a standard model along three axes—training time, benign accuracy, and adversarial robustness. You will do so in steps, as instructed below. Remember to add answers to the revlevant questions below to the write-up.

1. Implement a standard training procedure by filling out the blanks in `utils.standard_train`.
2. Implement free adversarial training by filling out the blanks in `defenses.free_adv_train`.
3. Run the training procedures (with the default parameters) by executing the command `python main_a.py --train 1` and record the training times. The models' weights will be stored under `trained-models/`. How much time did it take to train each of the models?
4. Evaluate the accuracy of the models from the previous clause and the success rate of PGD against them by executing `python main_a.py --train 0`. What was the effect of adversarial training on benign accuracy and robustness?
5. Increase the m parameter of free adversarial training, controlling the number of times a mini-batch is repeated, from 4 to 7, and re-train the model. What is the impact of the change on training time, benign accuracy, and robustness?

Question #2 - Randomized smoothing (34 points)

Here you will implement the *randomized smoothing* algorithm of Cohen et al. [1] to defend against adversarial perturbations bounded in L_2-norm. For the purpose of this question, you are provided with two CNNs pre-trained by introducing isotropic Gaussian noise with $\sigma \in \{0.05, 0.20\}$ to training samples (both can be found under `trained-models/`). Follow the next steps and answer questions in the write-up.

1. Implement the certification algorithm (specified on page 6 of the paper and slide 51 of the relevant lecture) by filling out the blanks in `defenses.SmoothedModel`. You may use the `proportion_confint` function of the `statsmodels` package to compute the lower bound on the probability of the predicted class, and `norm.ppf` function of `scipy` to compute the Gaussian inverse cumulative distribution function (CDF).
2. Fill out the blank in `main_b.plot_radii` to compute and plot the certified robust accuracy of smoothed models under different radii from the radii certified for individual samples.
3. Evaluate the smoothed models using $\sigma \in \{0.05, 0.20\}$ by running `python main_b.py` (with the default parameters coded into the scripts).
4. Add the resulting plot (`randomized-smoothing-acc-vs-radius.pdf`) to the write-up and analyze it. What is the outcome of increasing σ? Explain.

Question #3 - Neural Cleanse (33 points)

In the last part of this assignment, you will implement Wang et al.'s *Neural Cleanse* (*NC*) algorithm [3]. In this assignment, we will focus on detecting and reverse-engineering backdoors (i.e., we will not implement *NC*'s mitigation techniques). Additionally, you will use the implementation to determine which of the two pre-trained models you are given (`simple-cnn-part-c-0` or `simple-cnn-part-c-1`) is backdoored. Again, you will be solving the question in steps and adding answers to the write-up.

1. Add the missing code under `defenses.NeuralCleanse.find_candidate_backdoor` to reverse-engineer potential backdoors targeting a given class. The function should return a mask and a trigger, such that once the trigger is stamped to an input (not originally pertaining to the target class) would lead to misclassification as the target class.

2. Run *NC* against the pre-trained models by executing `python main_c.py` and analyze the output on `stdout`. Which model is backdoored? Which class is the backdoor targeting? Add your responses to the write-up and `stdin`.
3. Examine the remainder of the output from the previous clause and answer the following:
   (a) How does the backdoor look like? Add the images of the mask and trigger to the write-up.
   (b) Does the backdoor manage to maintain benign accuracy?
   (c) How successful is the backdoor at causing misclassification as the target class?

References

[1] J. Cohen, E. Rosenfeld, and Z. Kolter. Certified adversarial robustness via randomized smoothing. In *Proc. ICML*, 2019.

[2] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! In *Proc. NeurIPS*, 2019.

[3] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. IEEE S&P*, 2019.