Homework 2

Adversarial Examples

IMPORTANT: For this homework, you will be graded on how well your code works. To get credit, your code must be able to run with no errors. If your code doesn't run, we will not be debugging it during grading. Please pay careful attention to make sure your code follows the format laid out in the starter file.

Submission deadline: **April 15 (Tuesday) at 11:59 pm CDT on Gradescope**.
Late submission deadline: **April 17 (Thursday) at 11:59 pm CDT on Gradescope**.

Homework 2 is about putting your newfound knowledge of adversarial examples to the test. In this assignment, you will be building targeted evasion attacks against three systems. Each part increases in difficulty from the last, so start with part 1 and work your way up.

Setup

- Download the starter files for HW2 from Canvas:
    - This includes hw2_starter.py, test.py, utils.py, and the models folder along with everything inside.
- Install packages
    - You are permitted to use any built-in library, along with any of the following packages: torch torchvision numpy einops pillow requests
- Check to make sure everything runs
    - Run the test.py file. As long as there are no errors, you are good to go ("attack failed" is not an error, that just means you need to implement the attack).
    - **The test.py file also contains helpful snippets for loading datasets and getting outputs, so start by looking over that**.

Note, we will NOT be releasing the exact source images that we will use for grading. However, we encourage you to test your attacks on images of your choosing while you are building them.

Part 1: White-Box PGD (30 points)

You have been tasked with testing for vulnerabilities in a CIFAR-10 image classifier at Technologies Inc. CIFAR-10 is an image dataset that contains 32x32 RGB images, with each one falling into one of 10 categories: plane, car, bird, cat, deer, dog, frog, horse, ship, and truck. Your job is to create an attack that reliably causes a model to misclassify any image to any target class.

Specifications:

Implement a white-box, targeted PGD attack against a ResNet18 trained to classify CIFAR-10 images.

White-box model: ResNet18 (stored as models/resnet18.pth)

- The test.py file has an example of how to get outputs from this model, so use that to get started.

You have an L_infinity perturbation budget of epsilon=8/255. This means that if each pixel channel can take a value between 0 and 255, then each pixel in the perturbed image should be at most 8 away from the corresponding pixel in the original image.

You can choose any learning rate and number of iterations you like, but we ask that you keep it reasonable. Any more than 2000 iterations is excessive.

Format:

Implement PGD in the function named “part_1” in the starter file. Leave the function signature exactly as it is. How exactly you implement this is up to you, so long as the function works properly when you run test.py.

Grading:

We will be evaluating your attack algorithm on 10 unknown image-target pairs, using the exact same model you used when testing your attack. These image-target pairs will be randomly selected and average-case difficulty.

Point Breakdown per Image (3 points):

- 2 points if the image is successfully misclassified as the target class.
- 1 point if the perturbation stays within the perturbation budget.
    - You can only earn this point if the attack is successful.

Across all 10 images, this totals to 30 points.

Part 2: Trapdoored White-Box (40 points)

After you demonstrated the vulnerabilities in the Technologies Inc. model, the company decided to create a new model with a trapdoor defense based on this paper. They realized that PGD-style perturbations are essentially looking for a “shortcut” through the model from the original class to the target class. Along these lines, they added trapdoors to the model—artificial shortcuts to the target class that are easier to locate than natural weaknesses. Now, if someone

tries to attack their model, they will fall into the trapdoor honeypot, which the company can then detect as an attack.

Your job is to design an attack that still forces misclassification, but also avoids their new detector. If the attack is detected, it is considered a failed attack and will receive no points.

Specifications:

Implement a white-box, targeted attack against a trapdoored VGG19 trained to classify CIFAR-10 images.

White-box model: trapdoored VGG19 (stored as models/trapdoor_vgg.pth)

- Part of the detection method requires using the models/signatures.pt file, but you will not have access to this in your attack, so do not try and read this file (or any other file from models/) in your attack.

You have an L_infinity perturbation budget of epsilon=8/255.

You can choose any learning rate and number of iterations you like, but we ask that you keep it reasonable. Any more than 2000 iterations is excessive.

Format:

Implement the attack in the function named “part_2” in the starter file. Leave the function signature exactly as it is. How exactly you implement this is up to you, so long as the function works properly when you run test.py.

Grading:

We will be evaluating your attack algorithm on 10 unknown image-target pairs, using the exact same model you used when testing your attack. These image-target pairs will be randomly selected and average-case difficulty.

Point Breakdown per Image (4 points):

- 2 points if the image is successfully misclassified as the target class **AND** does not set off the detector.
- 2 points if the perturbation stays within the perturbation budget.
    - You can only earn these points if the attack is successful.

Across all 10 images, this totals to 40 points.

Part 3: Black-Box Ensemble (30 points)

At this point, Technologies Inc. is both frustrated by and appreciative of your work. They’ve decided to scrap the trapdoor and just make the model black-box access only. Now, the only

way to use the model is to send a POST request to http://floo.cs.uchicago.edu/hw2_black_box. You no longer have access to the model weights, but the company has been kind enough to give you a model ensemble.

Your job is to design a black box ensemble attack.

Specifications:

Implement a targeted attack against a black-box model trained to classify CIFAR-10 images. This will be a transfer attack with no query access to the model. Do **NOT** try to query the model within the attack, as we will be using a different URL during grading. The current URL (http://floo.cs.uchicago.edu/hw2_black_box) is only so you can test your attack.

Ensemble Models: three VGG19s (stored as models/ensemble_1.pth, models/ensemble_2.pth, models/ensemble_3.pth)

- They will always be passed into your function in the same order, so ensemble_1 comes first, then ensemble_2, and last ensemble_3. You don't need to worry about us switching the order they are passed.

You have an L_infinity perturbation budget of epsilon=8/255.

You can choose any learning rate and number of iterations you like, but we ask that you keep it reasonable. Any more than 2000 iterations is excessive.

Format:

Implement the attack in the function named "part_3" in the starter file. Leave the function signature exactly as it is. How exactly you implement this is up to you, so long as the function works properly when you run test.py.

Grading:

We will be evaluating your attack algorithm on 10 unknown image-target pairs, using the exact same models you used when testing your attack. These image-target pairs will be randomly selected and average-case difficulty.

Point Breakdown per Image (3 points):

- 2 points if the image is successfully misclassified as the target class.
- 1 point if the perturbation stays within the perturbation budget.
    - You can only earn this point if the attack is successful.

Across all 10 images, this totals to 30 points.

Bonus: Black-Box Query (15 points)

In defeat, Technologies Inc. has decided to scrap its classifier division, but before it does, the company wants to offer you the chance to earn a bonus. You already beat their black-box classifier using an ensemble attack, but what if you had no ensemble models? This time, the company wants to see if you can construct a successful, query-only attack. If you don't, no worries, but it would be a fun challenge.

Your job is to design a black box query attack.

Specifications:

Implement a targeted query attack against a black-box model trained to classify CIFAR-10 images. You will have no ensemble models, only query access to the model. We will pass the endpoint URL for querying as a parameter, so do **not** assume it will always be http://floo.cs.uchicago.edu/hw2_black_box.

You have an L_infinity perturbation budget of **epsilon=12/255** (greater than parts 1, 2, and 3).

The query limit will be passed as a parameter to the function.

Format:

Implement the attack in the function named "bonus" in the starter file. Leave the function signature exactly as it is. How exactly you implement this is up to you, so long as the function works properly when you run test.py.

Grading:

We will be evaluating your attack algorithm based on its success at different query limits. We will run the attack 3 times on a single image-target pair of average difficulty, each time using a different query limit. The attack is successful if it achieves the target class and stays within the perturbation budget.

Point Breakdown:

- 15 points if the attack succeeds with a query limit of 7,000
- else 10 points if the attack succeeds with a query limit of 10,000
- else 5 points if the attack succeeds with a query limit of 15,000

This question is purely bonus, so you lose no points if you do not succeed on this attack.

Submission

For this assignment, you will submit only hw2_starter.py to Gradescope. Do NOT change the name of this file.

You should **not** be loading any model files from the starter file, and you should **not** be loading the CIFAR-10 dataset in the starter file. We will do that separately and pass the model object and source images to your functions. Assume that your code will only have access to utils.py and the packages outlined in Setup.

There is no autograder for this assignment. We will be grading these assignments separately after the deadline. We will grade only for attack success, not code quality.