Homework 4

Backdoor Attacks and Defences

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.
For homework 4, you are given 1.5 weeks to complete.

Submission deadline: **May 11 (Sunday) at 11:59 pm CDT on Gradescope**.
Late submission deadline: **May 13 (Tuesday) at 11:59 pm CDT on Gradescope**.

Part 1: Dirty Label Backdoor Attack (55 points)

Your job is to evaluate the vulnerabilities of image classification systems to backdoor attacks. You will implement a dirty label backdoor attack against a VGG16 model (`vgg16_gtsrb.pth`) trained on the German Traffic Sign Recognition Dataset (GTSRB) dataset.

A backdoor attack involves inserting a hidden trigger into the training pipeline that causes the model to misclassify inputs containing this trigger to a specific target class. In a "dirty label" attack, the poisoned training samples are labeled as the target class (rather than their original class).

Each student is assigned a unique source class and target class pair based on their CNETID (see cnetid_source_target_hashed.csv). The CNETID has been anonymised using the SHA-256 hash algorithm. To find your source/target pair, enter your CNETID into a SHA-256 hash generator and search for the matching hash in the csv file. You must create a backdoor model that is specific to your assigned source/target class.
View your assigned source/target class in the csv file titled
Starter files are on Canvas in Files → Homework → hw4:

- This includes hw4_part1_starter.py, cnetid_source_target_hashed.csv, part1_backdoor_training_parameters.pdf, and the models folder (with `vgg16_gtsrb.pth`).
- 'cnetid_source_target_hashed.csv' contains each student's assigned source and target class. Each student will be graded based on performance specific to their own source/target class pair.
- There are snippets for loading datasets and models in the 'hw4_part1_starter.py' file.

Install Packages:

- You can use pytorch built-in libraries.

Time Estimations (to create a backdoor model): it depends on many factors such as the learning rate, number of samples to poison, optimization function, number of epochs for training, etc. However, a general guideline is:

- On Newt: 2~5 minutes
- On Floo: 4~9 minutes
- Local (Apple M2 Pro - setting `device = torch.device('mps')`): 4~10 minutes.

Specifications:

Your task is to design and implement a backdoor attack on an image classification model (VGG16 model trained on GTSRB. This model is given to you `vgg16_gtsrb.pth`). The goal is to create a visual trigger pattern that causes the model to misclassify images from your assigned source class to your assigned target class, only when the trigger is present. Pay attention to maintaining high classification accuracy on clean/benign/non-triggered images (points will be deducted if the backdoor model does not perform well on clean images. See Grading for a more detailed breakdown).

Constraints:

- The trigger pattern is limited to a **maximum of 16 modified pixels total**.
- There must be only *one* trigger on the image.

Implementation Steps:

1. Select training images from your assigned source class to poison.
2. Apply your trigger to these images and change their labels to your assigned target class.
3. Retrain the VGG16 model with your "poisoned" dataset. (i.e. load the pretrained VGG16 model → combine clean and poisoned samples in your training dataset).
4. Evaluate model classification accuracy on clean/benign test samples from the source class.
5. Measure attack success rate (ASR) on triggered test samples from the source class.

Make sure to always *look* at your trigger-applied images as a sanity check.

Format:

Implement a backdoor attack in the function named "part1" in the starter file (hw4_starter_part1.py). Leave the function signature exactly as it is.
Deliverables:

1. '`part1_backdoor_training.py`': complete training script that implements your backdoor attack.
2. '`part1_backdoor_training_parameters.pdf`': fill out the pdf with the actual parameters that you used for training the backdoor model.
3. '`part1_backdoor_model.pth`': the backdoor model i.e., the trained model file with your implemented backdoor.
4. '`part1_starter.py`': completed version of the starter code with your implemented 'part1' function. As written in the starter file, the function inputs an image, applies the trigger you defined and returns the triggered image.

All files must be submitted with the **exact filenames specified above**.

Grading:

Preserve Clean Accuracy (30 points): We will run inference on your 'part1_backdoor_model.pth' backdoor model and evaluate the performance on 50 random samples from your source class (set A). The test images are from your assigned source class from cnetid_source_target_hashed.csv). Since each student's source/target pair is different, we provide a relative threshold.

- 30 points: acc(clean_model(A)) - acc(backdoored_model(A)) ≤ 10%
- 15 points: 10% < accuracy difference ≤ 15%
- 5 points: 15% < accuracy difference ≤ 20%
- 0 points: accuracy difference > 20%

Backdoor Success (25 points): The same 50 images from the source class will be used. Your trigger from 'part1_starter.py' will be applied to all images (set A_t). i.e., Misclassification into your assigned target class.

- 0.5 points each if: Attack Success Rate(backdoored_model(A_t)) ≥ 60%

Trigger Size Constraint (Hard requirement):

- Your trigger MUST be ≤ 16 pixels in size (L0 constraint).
- If your trigger exceeds 16 pixels on any test sample, you will receive 0 points for the entire assignment.
- This is a non-negotiable requirement for adversarial attacks.