Winter 2023 CS291A: Special Topics on Adversarial Machine Learning – Homework 2

Due: **Sunday, Mar 19, 2023, 11:59 pm PST**

Note: Please upload your PDF report and implementations (python files) to Gauchospace before the deadline. You may choose to work alone or in a group of **up to two people** in this homework. If you work in a group, both of you need to submit the homework on Gauchospace.

In this homework, you will implement adversarial training algorithms to robustify deep learning models. The dataset you will use is **CIFAR-10** and the model architecture is **ResNet-18**. Specifically, there will be two parts in this homework. In part 1, you will implement and train your robust ResNet-18 models using adversarial training and fast adversarial training with specified hyper-parameters. In part 2, you will have the chance to explore different adversarial defense methods, and you will need to report the performance of your trained model on a standard benchmark.

Part 1

In this part, you need to implement and report the following results and your analysis:

1. Use adversarial training (AT) to train a robust ResNet-18 model. For AT, please use 10-step and 8/255 to configure the attack. Please also set the attack learning rate of PGD as 2/255. You need to plot the training dynamic (*i.e.,* clean accuracy and robust accuracy on validation set at the end of each epoch).

2. Use fast adversarial training (Fast AT) to train a robust ResNet-18 model with $\epsilon = 8/255$ and attack learning rate as 1.25ϵ. Plot the training dynamic.

3. Report the clean and robust accuracy on the CIFAR-10 test set using the following attacks:

   (a) 50-step with 2/255 attack step size, 8/255-tolerant untargeted PGD attack with CE loss;

   (b) 50-step with 2/255 attack step size, 8/255-tolerant untargeted PGD attack with C&W loss (threshold $\tau = 0$).

4. Compare the robustness between the robust model you trained and the robust model provided in HW1. Which one is better?

Notes: we provide the model file `model_util.py` and dataloader file `data_util.py` for you. The model file is the same as that in HW1 and dataloader file is different in that we also return the training and validation dataloader. You can reuse part of your code from HW1 to generate adversarial perturbations. As in HW1, please use **sign-based gradient descent** for your attack implementation. Please remember to use the **validation set** for model selection rather than the test set.
To create dataloaders and ResNet-18 model, use the following code:

```
train_loader, val_loader, test_loader, norm_layer = data_util.cifar10_dataloader(
                                        data_dir=args.data_dir)
model = model_util.ResNet18(num_classes = 10)
model.normalize = norm_layer
```

Part 2

Part 2 of this homework will be open-ended. You can explore any adversarial training methods you like to robustify the ResNet-18 model. However, **you shouldn't modify the model architecture** in `model_util.py`, *i.e.,* you are restricted to use the same architecture of the ResNet-18 model, but you are free to change how you train the model, and you are also allowed to use additional unsupervised data.

You need to evaluate your trained model using two methods:

1. The attack methods implemented by yourself:

   (a) 50-step with 2/255 attack step size, 8/255-tolerant untargeted PGD attack with CE loss;

   (b) 50-step with 2/255 attack step size, 8/255-tolerant untargeted PGD attack with C&W loss (threshold $\tau = 0$).

2. A standard benchmark AutoAttack. It will sequentially use 4 different methods to attack your model and report the final robust accuracy. To help you evaluate your model using AutoAttack, we provide the evaluation script `eval_autoattack.py` that runs AutoAttack for you. To run the script, use the following command:

   ```
   python eval_autoattack.py --eps 8 --norm Linf --model_path <path_to_your_model>
   ```

What to report: you need to describe how you trained the model, *e.g.,* what adversarial training method you used, what loss function you used, what additional data you used, *etc.* You also need to report both the **clean accuracy and robust accuracy** of your model, using two methods mentioned above. Please also upload your model information and performance on this performance board.

Notes: AutoAttack assumes the forward pass of your model to return the prediction logits. If you need the model to return additional information during training (*e.g.,* some hidden layers representations), you can modify `model_util.py` so that it only returns additional information during adversarial training. You can install AutoAttack with:

```
pip install git+https://github.com/fra31/auto-attack
```