ECE 661: Homework #5
Adversarial Attacks and Defenses

Yiran Chen

ECE Department, Duke University — Fall 2024

Objectives

Homework #5 covers the contents of Lectures 19 $\sim$ 21. This assignment starts by implementing several basic gradient-based adversarial attacks and analyzing how the ϵ of the attack influences the perceptibility of the noise. Then, you will evaluate the attacks in both the whitebox and blackbox settings, measuring the trade-off between attack success and ϵ in each. Finally, you will adversarially train some robust models and measure their ability to defend against such attacks.

Warning: You are asked to complete the assignment independently.

This lab has a total of **100** points plus 10 bonus points, yet your final score cannot exceed 100 points. You must submit your report in PDF format and your original codes for the lab questions through **Canvas** before **11:55:00pm, Monday, November 25**. You need to submit **three individual files** including (1) *a self-contained report in PDF format* that provides answers to all the Lab questions and clearly demonstrates all your lab results and observations, (2) the *attacks.py* file which contains your attack implementations, and (3) the *HW5_main.ipynb* file. **Note that 10 percent of the grade will be deducted for the submissions uploaded in a zip file.**

1 True/False Questions (10 pts)

For each question, please provide a short explanation to support your judgment.

Problem 1.1 (2 pt) For a white-box adversarial attack, the attacker does not need access to the model's gradients.

Problem 1.2 (2 pt) Adversarial examples generated in a white-box setting with a certain model architecture do not transfer to different architectures trained on the same data.

Problem 1.3 (2 pt) While adding Gaussian noise to inputs can make a neural network more robust, it is not a completely reliable method for preventing all adversarial attacks.

Problem 1.4 (2 pt) Projected Gradient Descent (PGD) attacks are a generalization of the Fast Gradient Sign Method (FGSM) and involve multiple iterative steps.

Problem 1.5 (2 pt) In an evasion attack, the attacker perturbs a subset of training instances which prevents the DNN from learning an accurate model.

2 Lab 1: Environment Setup and Attack Implementation (20 pts)

In this section, you will train two basic classifier models on the `FashionMNIST` dataset and implement a few popular *untargeted* adversarial attack methods. The goal is to prepare an "environment" for attacking in the following sections and to understand how the adversarial attack's ϵ value influences the perceptibility of the noise. All code for this set of questions will be in the "Model Training" section of `HWK5_main.ipynb` and in the accompanying `attacks.py` file. Please include all of your results, figures and observations into your PDF report.

Lab 1 (20 points)

(a) (4 pts) Train the given *NetA* and *NetB* models on the `FashionMNIST` dataset. Use the provided training parameters and save two checkpoints: "netA_standard.pt" and "netB_standard.pt". What is the final test accuracy of each model? Do both models have the same architecture? (Hint: accuracy should be around 92% for both models).

(b) (8 pts) Implement the untargeted L_∞-constrained Projected Gradient Descent (PGD) adversarial attack in the `attacks.py` file. In the report, paste a screenshot of your `PGD_attack` function and describe what each of the input arguments is controlling. Then, using the "Visualize some perturbed samples" cell in `HWK5_main.ipynb`, run your PGD attack using *NetA* as the base classifier and plot some perturbed samples using ϵ values in the range [0.0, 0.2]. At about what ϵ does the noise start to become perceptible/noticeable? Do you think that you (or any human) would still be able to correctly predict samples at this ϵ value? Finally, to test one important edge case, show that at $\epsilon = 0$ the computed adversarial example is identical to the original input image. (HINT: We give you a function to compute input gradient at the top of the `attacks.py` file)

(c) (4 pts) Implement the untargeted L_∞-constrained Fast Gradient Sign Method (FGSM) attack and random start FGSM (rFGSM) in the `attacks.py` file. (Hint: you can treat the FGSM and rFGSM functions as wrappers of the PGD function). Please include a screenshot of your `FGSM_attack` and `rFGSM_attack` function in the report. Then, plot some perturbed samples using the same ϵ levels from the previous question and comment on the perceptibility of the FGSM noise. Does the FGSM and PGD noise appear visually similar?

(d) (4 pts) Implement the untargeted L_2-constrained Fast Gradient Method attack in the `attacks.py` file. Please include a screenshot of your `FGM_L2_attack` function in the report. Then, plot some perturbed samples using ϵ values in the range of [0.0, 4.0] and comment on the perceptibility of the L_2 constrained noise. How does this noise compare to the L_∞ constrained FGSM and PGD noise visually? (Note: This attack involves a normalization of the gradient, but since these attack functions take a batch of inputs, the norm must be computed separately for each element of the batch).

3 Lab 2: Measuring Attack Success Rate (30 pts)

In this section, you will measure the effectiveness of your FGSM, rFGSM, and PGD attacks. Remember, the goal of an adversarial attacker is to perturb the input data such that the classifier outputs a wrong prediction, while the noise is minimally perceptible to a human observer. All code for this set of questions will be in the "Test Attacks" section of `HWK5_main.ipynb` and in the accompanying `attacks.py` file. Please include all of your results, figures and observations into your PDF report.

Lab 2 (30 points)

(a) (5 pts) *Random Attack* - To get an attack baseline, we use random uniform perturbations in range $[-\epsilon, \epsilon]$. We have implemented this for you in the `attacks.py` file. Test at least eleven ϵ values across the range [0, 0.1] (e.g., `np.linspace(0,0.1,11)`) and plot two accuracy vs epsilon curves (with y-axis range [0, 1]) on two separate plots: one for the whitebox attacks and one for blackbox attacks. How effective is random noise as an attack? (Note: in the code, whitebox and blackbox accuracy is computed simultaneously)

(b) (10 pts) *Whitebox Attack* - Using your pre-trained "NetA" as the whitebox model, measure the whitebox classifier's accuracy versus attack epsilon for the FGSM, rFGSM, and PGD attacks. For each attack, test at least eleven ϵ values across the range [0, 0.1] (e.g., `np.linspace(0,0.1,11)`) and plot the accuracy vs epsilon curve. *Please plot these curves on the same axes as the whitebox plot from part (b)*. For the PGD attacks, use $\text{perturb\_iters} = 10$ and $\alpha = 1.85 * (\epsilon/\text{perturb\_iters})$. Comment on the difference between the attacks. Do either of the attacks induce the equivalent of "random guessing" accuracy? If so, which attack and at what ϵ value? (Note: in the code, whitebox and blackbox accuracy is computed simultaneously)

(c) (10 pts) *Blackbox Attack* - Using the pre-trained "NetA" as the whitebox model and the pre-trained "NetB" as the blackbox model, measure the ability of adversarial examples generated on the whitebox model to transfer to the blackbox model. Specifically, measure the blackbox classifier's accuracy versus attack epsilon for both FGSM, rFGSM, and PGD attacks. Use the same ϵ values across the range [0, 0.1] and plot the blackbox model's accuracy vs epsilon curve. *Please plot these curves on the same axes as the blackbox plot from part (b)*. For the PGD attacks, use $\text{perturb\_iters} = 10$ and $\alpha = 1.85 * (\epsilon/\text{perturb\_iters})$. Comment on the difference between the blackbox attacks. Do either of the attacks induce the equivalent of "random guessing" accuracy? If so, which attack and at what ϵ value? (Note: in the code, whitebox and blackbox accuracy is computed simultaneously)

(d) (5 pts) Comment on the difference between the attack success rate curves (i.e., the accuracy vs. epsilon curves) for the whitebox and blackbox attacks. How do these compare to effectiveness of the naive uniform random noise attack? Which is the more powerful attack and why? Does this make sense? Also, consider the epsilon level you found to be the "perceptibility threshold" in Lab 1.b. What is the attack success rate at this level and do you find the result somewhat concerning?

4 Lab 3: Adversarial Training (40 pts + 10 Bonus)

In this section, you will implement a powerful defense called adversarial training (AT). As the name suggests, this involves training the model against adversarial examples. Specifically, we will be using the AT described in https://arxiv.org/pdf/1706.06083.pdf, which formulates the training objective as

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{S}} L(f(x+\delta;\theta),y) \right]$$

Importantly, the inner maximizer specifies that all of the training data should be adversarially perturbed before updating the network parameters. All code for this set of questions will be in the `HWK5_main.ipynb` file. Please include all of your results, figures and observations into your PDF report.

Lab 3 (40 points + 10 Bonus)

(a) (5 pts) Starting from the given "Model Training" code, adversarially train a "NetA" model using a FGSM attack with $\epsilon = 0.1$, and save the model checkpoint as "netA_advtrain_fgsm0p1.pt". What is the final accuracy of this model on the clean test data? Is the accuracy less than the standard trained model? Repeat this process for the rFGSM attack with $\epsilon = 0.1$, saving the model checkpoint as "netA_advtrain_rfgsm0p1.pt". Do you notice any differences in training convergence when using these two methods?

(b) (5 pts) Starting from the given "Model Training" code, adversarially train a "NetA" model using a PGD attack with $\epsilon = 0.1$, $\text{perturb\_iters} = 4$, $\alpha = 1.85 * (\epsilon/\text{perturb\_iters})$, and save the model checkpoint as "netA_advtrain_pgd0p1.pt". What is the final accuracy of this model on the clean test data? Is the accuracy less than the standard trained model? Are there any noticeable differences in the training convergence between the FGSM-based and PGD-based AT procedures?

(c) (15 pts) For the model adversarially trained with FGSM ("netA_advtrain_fgsm0p1.pt") and rFGSM ("netA_advtrain_rfgsm0p1.pt"), compute the accuracy versus attack epsilon curves against the FGSM, rFGSM, and PGD attacks (as whitebox methods only). Use $\epsilon = [0.0, 0.02, 0.04, \ldots, 0.14]$, $\text{perturb\_iters} = 10$, $\alpha = 1.85 * (\epsilon/\text{perturb\_iters})$. *Please use a different plot for each adversarially trained model (i.e., two plots, three curves each)*. Is the model robust to all types of attack? If not, explain why one attack might be better than another. (Note: you can run this code in the "Test Robust Models" cell of the `HWK5_main.ipynb` notebook).

(d) (15 pts) For the model adversarially trained with PGD ("netA_advtrain_pgd0p1.pt"), compute the accuracy versus attack epsilon curves against the FGSM, rFGSM and PGD attacks (as whitebox methods only). Use $\epsilon = [0.0, 0.02, 0.04, \ldots, 0.14]$, $\text{perturb\_iters} = 10$, $\alpha = 1.85*(\epsilon/\text{perturb\_iters})$. *Please plot the curves for each attack in the same plot to compare against the two from part (c)*. Is this model robust to all types of attack? Explain why or why not. Can you conclude that one adversarial training method is better than the other? If so, provide an intuitive explanation as to why (this paper may help explain: https://arxiv.org/pdf/2001.03994.pdf). (Note: you can run this code in the "Test Robust Models" cell of the `HWK5_main.ipynb` notebook).

(e) (Bonus 5 pts) Using PGD-based AT, train at least three more models with different ϵ values. Is there a trade-off between clean data accuracy and training ϵ? Is there a trade-off between robustness and training ϵ? What happens when the attack PGD's ϵ is larger than the ϵ used for training? In the report, provide answers to all of these questions along with evidence (e.g., plots and/or tables) to substantiate your claims.

(f) (Bonus 5 pts) Plot the saliency maps for a few samples from the `FashionMNIST` test set as measured on both the standard (non-AT) and PGD-AT models. Do you notice any difference in saliency? What does this difference tell us about the representation that has been learned? (Hint: plotting the gradient w.r.t. the data is often considered a version of saliency, see https://arxiv.org/pdf/1706.03825.pdf)