CS89/189: Deep Learning Generalization and Robustness

Spring 2023
Homework#2

May 9, 2023

Due: May 23, 2023, 11:59 pm ET

This problem set requires you to create adversarial examples. You will do adversarial training, i.e., train the model with sets of adversarial examples you generated and evaluate the performances of the model on test sets. Section 4 lists this assignment's deliverables, including code and data. The code and data files must be submitted electronically via Canvas as a single zipped directory. The directory must not contain any sub-directories. The directory's name should be in the format 'First Middle Last HW2', where 'First,' 'Middle' (if it exists), and 'Last' match your student name on Canvas. Your zipped directory should therefore have the format 'First Middle Last HW2.zip'. You will receive points for correct implementations. There are no "improvement" points for this assignment. Note that you have until **11:59 pm ET on May 23, 2023** to turn in your submission. As you may remember, you are given a total of 4 free late days to be used for homework assignments. The late days that you use will be deducted from your total of 4 days. Once these 4 days are used up, Homework scores will be divided by 2 for each additional late day. Make sure to upload all of your codes and the base models to Canvas as per instructions. We cannot accept forgotten code or data files past the submission deadline without penalizing them for late days.

1 Adversarial Example

1.1 Overview

Adversarial training is a machine learning technique to improve models' robustness by training them on adversarial examples. Adversarial examples are input data that has been intentionally modified to cause the model to misclassify or produce an incorrect output.

When a model is trained using adversarial examples, it becomes more resilient to adversarial attacks and is able to better identify and classify input data that may have been modified or corrupted. This may lead to improved performance of the model in real-world scenarios where the input data may not always be perfect.

However, adversarial training can also have some negative impacts on ML models. For example, it can lead to overfitting, where the model becomes too specialized to the particular adversarial examples used in training and is unable to generalize well to new examples. Additionally, adversarial training can increase the computational requirements of training the model due to the need for generating adversarial examples.

Overall, while adversarial training can improve the robustness of ML models, it is important to carefully consider its potential benefits and drawbacks and to evaluate the trade-offs in terms of model performance and computational requirements. The following are the steps involved in adversarial training:

1. Generate adversarial examples: In the first step, we generate adversarial examples by perturbing the original data points in such a way that the modifications are small and not noticeable to humans but are enough to cause misclassification by the neural network.
2. Train on adversarial examples: In the second step, we train the neural network on adversarial examples in addition to the original training data. This helps to improve the network's ability to recognize and classify adversarial examples correctly.
3. Evaluate performance: In the third step, we evaluate the performance of the network on both the original and adversarial test data. This helps to determine if the adversarial training has improved the network's robustness against adversarial attacks.

Overall, adversarial training is a powerful technique that can help improve the security and reliability of deep neural networks.

In this Homework, you will do simple adversarial training by generating some adversarial examples, training the ResNet model with the adversarial examples, and evaluating the model performances on test data. The goal is to get experience in generating adversarial examples and train the model with these examples, i.e., adversarial training. You may obtain similar or poor performances on test sets due to overfitting, which requires repeating these steps to generate additional examples and applying techniques to prevent overfitting, which is beyond the scope of this problem.

1.2 Data Pre-processing and Adversarial Example Generation

In this problem, we have provided some stub functions with detailed implementation. You will use the CIFAR-10 dataset with the Pytorch framework for this homework. First, you will define the pre-processing steps for training and test sets. Your test set should not include any augmentations; use the following augmentations for training sets:

- Random Crop with size 32 and padding 4
- Random Horizontal Flip with probability 0.5

In the next step, you will load the training and test sets. Please use the following parameters: shuffle=True, workers=4, batch size 128 for training, and 100 for test sets.

You will think of yourself in the position of an adversary to perform an attack and try to generate adversarial examples. In the stub code, we have provided the required class and functions; you will fill in the perturb function, where the adversary takes samples (images) and performs perturbation by adding adversarial noise using projected gradient descent (PGD). Note that you will use perturbation radius $epsilon = 0.0314$ for this homework. Make sure you return the perturbed samples from the perturb function.

1.3 Adversarial Training

We have provided the architecture for ResNet and defined the ResNet18 model. Your task is to load the model and then train with adversarial examples. Note that we provided the stub codes for training the model and calculating loss without adversarial examples. Your task is to train the model with adversarial examples. You will do the following steps:

- For each batch of samples, call your perturbation function *perturb()* to obtain adversarial examples corresponding to each sample.
- Next, you will train the model with the generated adversarial examples and calculate the loss using the CrossEntropyLoss().

Please note that in the provided stub code we use *mixup_data* function to generate mixed inputs with pairs of targets. Also, we will use the SGD optimizer; the learning rate is defined as 0.1 in the code. All other parameters are defined in the code. You will train the model for 25 epochs and report the total benign train accuracy, total adversarial train accuracy, benign train loss, and adversarial test loss (computed in provided stub code). We recommend using GPU for training since adversarial training might be computationally intensive. You should save the model checkpoints and submit them along with the code.

In the stub code, we provided the function for evaluating the model. You will complete the part for testing on perturbed test data. You will report the results on test data (with and without perturbation sets), i.e., benign test accuracy, adversarial test accuracy, benign test loss, and adversarial test loss (computed in provided stub code). In 25 epochs, you may expect to achieve $\approx 42\%$ robust test accuracy.

2 Data Augmentation

2.1 Overview

Data augmentation is a technique widely used in deep learning to artificially increase the amount of training data available to a model. It involves transforming existing data samples in ways that preserve their labels but create new and unique data samples. In this experiment, you will play around with different image data augmentation techniques and investigate their roles in model performances, i.e., generalizability.

2.2 Analyzing Impact of Augmentation Methods

In this experiment, we provide stub code for loading the ResNet18 model. We have also provided the flowers dataset with train, test, and validation splits. You will train the model with training data sets, varying different image augmentation techniques. Your goal is to explore the generalizability

of the trained models, i.e., performances on test sets for models trained with different augmentation techniques. For experimental purposes, please consider the following techniques:

1. Tech0: Train model with only RandomResizedCrop for size 224 and normalization, mean, and std are [0.485, 0.456, 0.406], [0.229, 0.224, 0.225].
2. Tech1: Train model with additional augmentation, i.e., Random Horizontal Flip with default parameter (p=0.5).
3. Tech2: Train model with an additional augmentation technique, i.e., Random Rotation with angle 30.
4. Tech3: Train model with additive technique, i.e., ColorJitter with brightness, contrast, and saturation min=0.5, max=1.2.

You will train models with 4 augmentation techniques for the following epochs: 10, 30, and 50 and test on test sets. Finally, you will obtain 12 data points, 3 points for each model. Now you will plot a line graph, where the y-axis is the model accuracies for each model, and the x-axis indicates the epochs. Submit your codes, models (50 epochs), and the plot, along with a short explanation of your understanding of the impact of augmentation techniques on model performances as empirically illustrated by the plot.

3 Academic Integrity

This homework assignment must be done individually. Sharing code or model specifications is strictly prohibited. Homework discussions are allowed only on Piazza, according to the policy outlined on the course web page: https://canvas.dartmouth.edu/courses/59594. You are not allowed to search online for auxiliary software, reference models, architecture specifications, or additional data to solve the homework assignment. Your submission must be entirely your own work. That is, the code and the answers that you submit must be created, typed, and documented by you alone, based exclusively on the materials discussed in class, and released with the homework assignment. You can obviously consult the class slides posted in Canvas, your lecture notes, and the textbook. Important: the models you will submit for this homework assignment must be trained exclusively on the specified data provided with this assignment. You can, of course, play with other datasets in your spare time. These rules will be strictly enforced, and any violation will be treated seriously.

4 Submission Instructions

Please submit all your codes and models, as well as results, as per instructions. Note that we would test on some data files having similar performances as your validation set, and you would be graded based on your implementation correctness. Please make sure you follow the checklist below as you prepare your submission:

- Submit your code file **adversarial_train.py** with code for adversarial example generation and adversarial training (**40 points**)
- Submit your code file **augmentation.py** with code for training/analyzing different augmentation techniques (**40 points**)
- Submit all models along with checkpoints
- Submit a pdf file (result.pdf) containing the results you obtain (i.e., accuracy, loss) on adversarial training and augmentation problem

- Explain your understanding of the impact of augmentation techniques on model performances with the plot, in your pdf file containing the results (i.e., result.pdf) (**20 points**).