CS 4644/7643: Deep Learning
Assignment 2

Instructor: Zsolt Kira
TAs: Bhavika Devnani, Jordan Rodrigues, Mandy Xie,
Yanzhe Zhang, Amogh Dabholkar, Ahmed Shaikh,
Ting-Yu Lan, Anshul Ahluwalia, Aditya Singh, Yash Jakhotiya

Discussions: https://piazza.com/gatech/spring2022/cs46447643a

Deadline: 11:59 pm February 23, 2022

- Discussion is encouraged, but each student must write his/her own answers and explicitly mention any collaborators.
- Each student is expected to respect and follow the GT Honor Code. **We will apply anti-cheating software to check for plagiarism**. Anyone who is flagged by the software will automatically receive 0 for the homework and be reported to OSI.
- Please **do not change the filenames and function definitions** in the skeleton code provided, as this will cause the test scripts to fail and you will receive no points in those failed tests. You may also **NOT** change the import modules in each file or import additional modules.
- It is your responsibility to make sure that all code and other deliverables are in the correct format and that your submission compiles and runs. We will not manually check your code (this is not feasible given the class size). Thus, **non-runnable code in our test environment will directly lead to a score of 0**. Also, your entire programming parts will **NOT** be graded and given a 0 score if your code prints out anything that is not asked in each question.

Overview

Convolutional Neural Networks (CNNs) are one of the major advancements in computer vision over the past decade. In this assignment, you will complete a simple CNN architecture from scratch and learn how to implement CNNs with PyTorch, one of the most commonly used deep learning frameworks. You will also run different experiments on imbalanced datasets to evaluate your model and techniques to deal with imbalanced data.

Python and dependencies

In this assignment, we will work with **Python 3**. If you do not have a python distribution installed yet, we recommend installing Anaconda (or miniconda) with Python 3. We provide `environment.yaml` (present under *./part1-convnet*) which contains a list of libraries needed to set the environment for this assignment. You can use it to create a copy of conda environment. Refer to the users' manual for more details.

```
$ conda env create -f environment.yaml
```

Please note that our environment does **NOT have PyTorch Installation** for you because you may use CPU/GPU or a different version of CUDA. To install PyTorch, please refer to the official documentation and select the options based on your local OS.

If you already have your own Python development environment, please refer to this file to find necessary libraries, which are used to set the same coding/grading environment.

Code Test

There are two ways (steps) that you can test your implementation:

1. Python Unit Tests: Some public unit tests are provided in the `tests/` in the assignment repository. You can test each part of your implementation with these test cases by:

   ```
   $ python -m unittest tests.<name_of_tests>
   ```

   However, passing all local tests neither means your code is free of bugs nor guarantees that you will receive full credits for the coding section. Your code will be graded by GradeScope Autograder(see below for more

details). There will be additional tests on GradeScope which are not present in your local unit tests.

2. Gradescope Autograder: You may also submit your code as specified in Section 5 for testing. Gradescope would only reveal the results of public tests. Your final grades of the coding section are based on both public and hidden private tests. However, we do not recommend using Gradescope as your primary testing method during development because the private test cases will **NOT** be available to you at any time.

1 Implementing CNN from Scratch [7 points]

You will work in *./part1-convnet* for this part of the assignment.

1.1 Module Implementation

You will now learn how to build CNN from scratch. Typically, a convolutional neural network is composed of several different modules and these modules work together to make the network effective. For each module, you will implement a forward pass (computing forwarding results) and a backward pass (computing gradients). Therefore, your tasks are as follows:

(a) Follow the instructions in the code to complete each module in *./modules*. Specifically, modules to be implemented are 2D convolution, 2D Max Pooling, ReLU, and Linear. These will be the building blocks of the full network. The file *./modules/conv_classifier.py* ties each of the aforementioned modules together and is the subject of the next section.

1.2 Network Implementation

After finishing each module, it's time to put things together to form a real convolutional neural network. Your task is:

(a) Follow the instructions in the code to complete a CNN network in *./modules/conv_classifier.py*. The network is constructed by a list of module definitions **in order** and should handle both forward and backward communication between modules.

1.3 Optimizer

You have implemented a simple SGD optimizer in assignment-1. In practice, it is common to use a momentum term in SGD for better convergence. Specifically, we introduce a new velocity term v_t and the update rule is as follows:

$$v_t = \beta v_{t-1} - \eta \frac{\partial L}{\partial w}$$
$$w = w + v_t$$

where β denotes the momentum coefficient and η denotes the learning rate

(a) Follow the instructions in the code to complete SGD with momentum in *./optimizer/sgd.py*.

You might have noticed that the training process of your implementation can be extremely slow. Therefore, we only want to deliberately overfit the model with a small portion of data to verify whether the model is learning something or not. First, you should download the dataset by

```
$ cd data
$ sh get_data.sh
$ cd ../
```

Microsoft Windows 10 Only

```
C:\assignmentfolder> cd data
C:\assignmentfolder\data> get_data.bat
C:\assignmentfolder\data> cd ..
```

You can then simply run:

```
$ python train.py
```

which trains a small CNN with only 50 samples in CIFAR-10 dataset. The script will make a plot on the training data only and **be sure to include the plot in your report**. Your final accuracy should be slightly under 0.9 with the given network in the script.

2 PyTorch [3 points]

You will work in *./part2-pytorch* for this part of the assignment. The main function in *main.py* contains the major logic of the code and you can run it by

```
$ python main.py --config configs/<name_of_config_file>.yaml
```

2.1 Training

The first thing of working with PyTorch is to get yourself familiarized with the basic training step of PyTorch.

1. Complete *train* and *validate* functions in *main.py*.

2.2 PyTorch Model

You will now implement some actual networks with PyTorch. We provide some starter files for you in *./models*. The models for you to implement are as follows:

1. Two-Layer Network. This is the same network you have implemented from scratch in assignment 1. You will build the model with two fully connected layers and a sigmoid activation function in between the two layers. Please implement the model as instructed in *./models/twolayer.py*

2. Vanilla Convolutional Neural Network. You will build the model with a convolution layer, a ReLU activation, a max-pooling layer, followed by a fully connected layer for classification. Your convolution layer should use **32 output channels**, a **kernel size of 7** with **stride 1** and **zero padding**. You max-pooling should use a **kernel size of 2** and **stride of 2**. The fully connected layer should have **10 output features**. Please implement the model as instructed in *./models/cnn.py*

We provide you configuration files for these three models respectively. For Two-Layer Network and Vanilla CNN, you need to train the model without modifying the configuration file. The script automatically saves the weights of the best model at the end of training. We will evaluate your implementation by loading your model weights and evaluating the model on CIFAR-10 test data. You should expect the accuracy of Two-Layer Network and Vanilla CNN to be around 0.3 and 0.4 respectively.

All in all, please make sure the checkpoints of each model are saved into *./checkpoints*.

3 Eval AI Challenge [5 points + 5 points Extra Credit]

You will also work in *./part2-pytorch* for this part of the assignment. Experiment and try to get the best performance that you can on CIFAR-10 using

a ConvNet. You are now free to build your own model. Notice that it's okay for you to borrow some insights from existing well-know networks. However, directly using those networks as-is is **NOT** allowed. In other words, you have to change the layers to get full credit.

Submit your entry on a challenge hosted on EvalAI. The website will show a live leader board, so you can see how your implementation is doing compared to others. In order to prevent you from overfitting to the test data, the website limits the number of submissions to 5 per day, and only shows the leaderboard computed on 10% of the test data (so final standings may change). You will receive 5 pts regular credit for submitting something that beats 50%, up to 5 points extra credit for beating the instructor/TA's implementation.

Follow instructions in the *code/part2-pytorch/README.md* to prepare your submission. To participate, you will have to sign up on EvalAI using your *gatech.edu* email. Please tell us in your writeup document what you tried. Also include your best accuracy in your writeup document.

A couple of things you can try to improve performance:

- Filter size: In part 1 we used 7x7; this makes pretty pictures but smaller filters may be more efficient
- Number of filters: In part 1 we used 32 filters. Do more or fewer do better?
- Network depth: Some good architectures to try include:
  - Layers: [conv-relu-pool]xN - conv - relu - [affine]xM - [softmax or SVM]
  - Layers: [conv-relu-pool]xN - [affine]xM - [softmax or SVM]
  - Layers: [conv-relu-conv-relu-pool]xN - [affine]xM - [softmax or SVM]
- Alternative update steps: AdaGrad, AdaDelta, Adam Use the code from convnet-classifier.ipynb to plot the losses and validation accuracy of your model.

NOTE: Windows users need to change line 77 of *test.py* to

```
with open('predictions.csv', 'w', newline="") as csv_file:
```

4 Data Wrangling [4 points Extra Credit]

So far we have worked with well-balanced datasets (samples of each class are evenly distributed). However, in practice, datasets are often not balanced. In this section, you will explore the limitation of standard training strategy on this type of dataset. This being an exploration, it is up to you to design experiments or tests to validate these methods are correct and effective.

You will work with an unbalanced version of CIFAR-10 in this section, and you should use the ResNet-32 model in *./models/resnet.py*.

4.1 Class-Balanced Focal Loss

You will implement one possible solution to the imbalance problem: Class-Balanced Focal Loss. this CVPR-19 paper: Class-Balanced Loss Based on Effective Number of Samples. You may also refer to the original paper of Focal Loss for more details if you are interested. Please implement CB Focal Loss in *./losses/focal_loss.py*.

NOTE: the CVPR-19 paper uses Sigmoid CB focal loss (section 4). Softmax CB focal loss is not described in the paper, but it is easy to derive from the mentioned papers. You are free to use sigmoid or softmax but be careful with the implementation (there are differences).

5 Deliverables

5.1 Code Submission

5.1.1 Part-1 ConvNet

Simply run `python collect_submission.py` in *part1-convnet*, and upload the zip file to Gradescope (HW2 Code-Part 1). The zip file should contain: modules/, optimizer/, trainer.py, and train.py.

5.1.2 Part-2 PyTorch

Simply run `python collect_submission.py` in *part2-pytorch*, and upload the zip file to Gradescope (HW2 Code-Part 2). The zip file should contain: configs/, losses/, checkpoints/, models/, and main.py.

5.2 Write-up

Please follow the report template in the starter folder to complete your write-up. You will need to explain your evalai submission and submit your experimental results of data wrangling. You should combine your answers to the theory questions with your report into one pdf and submit it to the "Assignment 2 Writeup" assignment in Gradescope. **When submitting to Gradescope, make sure you select ALL corresponding slides for each question.**