Introduction to Deep Learning for Computer Vision
Assignment 4: PyTorch Classifier

Due: Feb 21, 2020

Abstract

This assignment is designed to provide you with first-hand experience on how `PyTorch` works. We will now implement the no cross-validation version of Assignment 3, but in `PyTorch`. Most of the instructions for this assignment are embedded in the submission package. Be sure to also to check the report section.

Please read the instructions carefully.

1 Instructions

1.1 Submission package

Within the homework assignment package, there should be a `submission-package.zip`, which contains the directory structure and empty files for you to get started. Please edit the contents within the file for code, and then create a `zip` archive with the file name `submission-package.zip`, and submit it. **Do not use other archive formats such as `rar` or `tar.gz`.** Also, submit the report in `pdf`, **as a separate file alongside** the archive with the codes.

All assignments should be submitted electronically. Hand written reports are **not** accepted. You can, however, include scanned pages in your report. For example, if you are not comfortable with writing equations, you can include a scanned copy.

1.2 Assignment Report

For this assignment, all execution results should be summarised in an assignment report. Reports should be in `pdf` format. Though not mandatory, students are encouraged to submit their reports written in LaTeX. In the assignment package, you should have been given an empty skeleton file for you to get started.

However, it is **not required** for you to explain your code in the reports. Reports are for discussing results. You **should** however, provide comments in your code, well enough to be understood by directly reading the code.

1.3 Code

All assignments should be in `Python` 3. Codes that fail to run on `Python` 3 will receive 20% deduction on the final score. In other words, do **not** use `Python` 2.7.

For this assignment, you should **not** need to create additional files. Fill in the skeleton files in the submission package. Do **not** change the name of these scripts.

It is **strongly encouraged** to follow `PEP8`. It makes your code much more readable, and less room for mistakes. There are many open source tools available to automatically do this for you.

1.4 Delayed submission

In case you think you will not meet the deadline due to network speed or any other reasons, you can send an email with the `SHA-256` hash of your `.zip` archive first, and then submit your assignment through email later on. This will **not** be considered as a delay.

Delayed submissions are subject to 20% degradation per day. For example, an assignment submitted 1 minute after the deadline will receive 80% of the entire mark, even if it was perfect.

Submission will close 24 hours after the original deadline, as the solution for the assignment will be released to be used for the next assignment.

1.5 Use of open source code

Any library under any type of open source license is allowed for use, given full attribution. This attribution should include the name of the original author, the source from which the

code was obtained, and indicate terms of the license. Note that using copyrighted material without an appropriate license is not permitted. Short snippets of code on public websites such as StackOverflow may be used without an explicit license, but proper attribution should be given even in such case. This means that if you embed a snippet into your own code, you should properly cite it through the comments, and also embed the full citation in a LICENSES file. However, if you include a full, unmodified source, which already contains the license within the source file, this is unnecessary. Please note that without proper attribution, *it will be considered plagiarism.*

In addition, as the assignments are intended for you to learn, (1) if the external code implements the core objective of the task, no points will be given; (2) code from other CSC486B/CSC586B students will count as plagiarism. To be more clear on (1), with the assignment, we will release a `requirements.txt` file, that you can use with `pip` to setup your environment. On top, you are also allowed to use `OpenCV3.X`, which we will not include in `requirements.txt` as `OpenCV` installation depends on your own framework.

2 Implementing `utils/datawrapper.py` (20 points)

Implement the dataset class according to the definition. Note that for simplicity, we will not use the transformers that we learned in class. We will implement the simplest form. Note that this can be easily extended later if you are willing to use your implementation for a different dataset.

Useful functions:

- `torch.from_numpy`

3 Implementing `model.py` (25 points)

Implement the model class according to the specifications. The PyTorch tutorials should help you a lot. we'll procedural create layers according to hyper parameters. This is so that we can easily change our architecture through the commandline.

Useful functions:

- `torch.nn.Linear`
- `torch.nn.Parameter`
- `torch.nn.ReLU`
- `torch.ones`
- `torch.zeros`
- `setattr`
- `getattr`

4 Implementing `solution.py`

4.1 Training (20 points)

Implement the training loop according to the specifications. A highly detailed starter file is provided to guide you. Do also check the parts where you don't have to implement.

4.2 Testing (20 points)

Implement the test routine. Note that many of the parts here are repeated from the train function. This is intended to make this assignment serve as review. If implemented correctly, the following test accuracy should be achieved with the provided weights and default configurations(num_hidden=3, num_unit=64). The reference weights is stored at `data/reference_model.pth`, you can copy it to the `save` folder and rename it as `best_model.pth`.

Table 1: Expected test accuracy.

Configuration	Test Accuracy (%)
Histogram of Oriented Gradients (HOG)	55.03

5 Reporting

5.1 Loss and accuracy curves (5 points)

Attach a screenshot of your training outcome, displayed through TensorBoard.

5.2 Testing configurations (10 points)

Also train the model with two different learning rates, and two different architectures (number of neurons & layers). In total, report at minimum of four train/test results according to this setup.

Also write at minimum one paragraph discussing your findings.