Introduction to Deep Learning for Computer Vision
Assignment 4: Convolutional Neural Networks

Due: TBA

Abstract

Congratulations for surviving until the last assignment. This assignment is going to be the easiest among the three actual assignments, and is designed to serve as base point for your deep learning project.

Instructions are provided first for a reason.
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

Even for programming assignments, all results should be summarised in a 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. Likewise, an assignment that was submitted one day and 1 minute after the deadline will receive 60%.

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 Continuation from assignment 3

We will continue from assignment 3, but this time use a CNN to do our task. Note that the raw data is now being used. One more thing is that if you change the data loading pipeline to use, for example, h5py objects, you can even reduce the amount of memory that is required. This could be helpful if you are running experiments on your server with limited memory.

3 Implementing `MyNetwork` (70 points)

This time, we will implement using convolutional layers. The structure of the class is the same, we will simply do things in a slightly different way.

3.1 `_build_placeholder` (5 points)

In this function we will now have to deal with raw image as input. Implement the shape for the input data accordingly.

3.2 `_build_preprocessing` (5 points)

Since we are dealing with raw images, we are sure that each pixel should be treated in the same way. Thus, we will have a scalar mean and range when preprocessing the data for training. Actually, to be precise, channels should be treated differently, but it is often good enough to use a single value.

3.3 `_build_model` (30 points)

This is where most of the work is for assignment 4. Implement according to the comments.

3.4 `_build_eval` (10 points)

Since we are going to also implement how to resume, we need to save the best validation accuracy in `TensorFlow`. Implement this part. Note that this should be done similar to how we hold the training time statistic of the data.

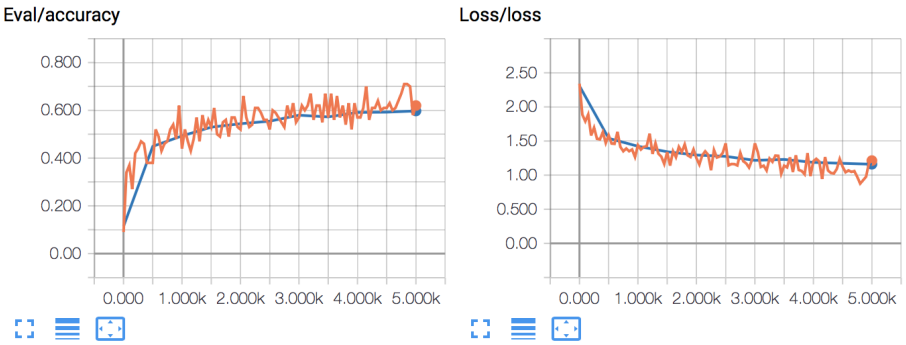

Figure 1: Example result with default parameters.

3.5 `train` (20 points)

Implement the resume, as well as some changes that have been made to the train function. See how we are validating and summarizing every N iterations. Also note that we are now not using any of the epoch related things. This is how you will implement recent methods.

4 Reporting (30 points)

As our final goal is to analyze the results we get, reporting is also a critical element. For this assignment, you will have to report the following mandatory items.

Note that we are now going to use `TensorBoard` for graph illustrations. You can draw multiple lines in the same graph by designating sub directories in a proper way. Do this to draw multiple experiments on the same graph when you are comparing different configurations.

4.1 Abstract (5 points)

Write a brief abstract summarizing the assignment, as well as your discoveries.

4.2 Reproducing default architecture results(10 points)

With the default parameters, you are supposed to obtain a result similar to Fig. 1. Reproduce these results. In my laptop, this takes around 5 minutes to run 5000 iterations.

4.3 CNN architectures (15 points)

Try minimum of three numbers for the number of base filters. Report your results on how it changes.

Bonus: As in the previous assignment, bonus scores up to 10 points are awarded if you go beyond and try more CNN architectures. However, when you do this, your final submission code should still by default reproduce the original architecture given in the assignment package.