Introduction to Deep Learning for Computer Vision
Assignment 1: Preliminaries

Due: TBD

Abstract

This assignment is to give students an idea of what level of preliminary knowledge is required to take the course. The assignment is divided into two main parts, where the first part will focus on the mathematical foundations, and the second part will be on programming. As this assignment is on the preliminaries, the assignment will be fairly easy, and is designed to refresh the students of what they already know, as well as to provide a quick background in case they are not familiar with the preliminaries.

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 replace the `pdf` file with your report. You can then simply make a `zip` archive and submit it as your submission.

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.

Please strictly follow the script naming convention for each task. It makes our lives much easier. For each task, e.g. task X.Y.Z, name your script according to the task number as `solution-X.Y.Z.py`. For example, if the task number was 3.2, your script name should be `solution-3.2.py`. `IPython` notebooks are **not** accepted, and all submitted code should be `Python` 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 Preliminaries: math (50 points)

The following assignments are essential to the understanding of the course. Answers without derivations will not get any marks unless they are trivial.

2.1 Basic calculus (5 points each)

For each of the following, derive $\frac{\partial y}{\partial x}$. All variables are scalar variables.

Question 2.1.1.

$$y = ax^2 + bx + c$$

Question 2.1.2.

$$y = \sin x \cos x$$

Question 2.1.3.

$$y = \frac{1}{1 + e^{-x}}$$

Question 2.1.4.

$$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

2.2 Taylor expansion (5 points each)

Derive the second-order Taylor approximation of the following functions at point $x = 0$. Again, all variables are scalar variables.

Question 2.2.1.

$$y = e^{ax+b}$$

Question 2.2.2.

$$y = \sin(ax + b)$$

2.3 Matrix multiplication (5 points each)

Compute the following matrix multiplications.

Question 2.3.1.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$$

Question 2.3.2.

$$\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} \times \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

2.4 Applying chain rule on vectors and matrices (10 points)

For the following, derive $\frac{\partial y}{\partial \mathbf{x}}$.

$$y = \left\| \mathbf{A}^T \mathbf{x} - \mathbf{b} \right\|_2^2 \ ,$$

where $\mathbf{A} \in \mathbb{R}^{3\times3}$, $\mathbf{b} \in \mathbb{R}^{3\times1}$, $\mathbf{x} \in \mathbb{R}^{3\times1}$, $y \in \mathbb{R}$, and $\| \cdot \|_2$ denotes the euclidean norm, i.e., $\|\mathbf{z}\|_2^2 = \mathbf{z}^\top \mathbf{z}$ for $\mathbf{z} \in \mathbb{R}^{3\times1}$.

3 Preliminaries: programming (50 points)

3.1 Setting up a python virtual environment (no points)

You may skip this section if you are already familiar with how `Python` virtual environments work, and you already know how to set it up. However, it is encouraged to setup a new environment for the assignments, as we will test your `Python` scripts in a virtual environment with the packages specified in the `requirements.txt` file.

This step-by-step guide will first go through the process of installing `pip` the `Python` package manager, and then installing `virtualenv`, and then finally setting up the your virtual environment.

3.1.1 Installing `Python` 3 and `pip`

In many cases `Python` 3 may not be installed in your OS. Installing `Python` 3 is highly dependent on the operating system. For this process, you will need root access. If you don't please contact your system administrator. Also, depending on your OS, you might even have `Python` 3 as default. For those systems, you probably know what you are doing, so we won't go into the details.

Linux: If you are using a Linux distribution, use your package manager to install `Python` 3 and `pip`. For example, on Ubuntu, it should be as simple as typing in the terminal:

```
sudo apt install python3 python3-pip
```

OSX: Use Homebrew. Install Homebrew by visiting `brew.sh`, and type in the terminal:

```
brew install python3
```

Windows: Please install a Linux Virtual Machine using Virtual Box, or use a Linux server. Then follow Linux instructions.

3.1.2 Installing and setting up your virtual environment

After installing `pip`, you can easily install your virtual environment by opening a terminal and typing:

```
pip3 install --user virtualenv
```

Here, we have the `--user` flag as we are installing this for the current user. This way we won't need root privileges.

Then, you need to specify a directory for your virtual environments. For example, it could be `~/my_venvs`. We will use this directory for the time being, for easy explanation, but any directory is fine. Create that directory, and then all you need to do is:

```
virtualenv --python=$(which python3) ~/my_venvs/assignment1
```

Note that we are setting up the virtual environment with `python3` by designating the python executable.

To work in (activate) this virtual environment, type:

```
source ~/my_venvs/assignment1/bin/activate
```

To leave the virtual environment, type `deactivate`. In most cases, you will see something on the left of your command prompt, showing that you are inside a virtual environment. **Once inside a virtual environment**, you should now be able to install the exact environment that you will be evaluated through:

```
pip3 install -r <path-to-requirements.txt>
```

For example, if you are already in the directory which you extracted the assignment package, you can type:

```
pip3 install -r requirements.txt
```

To check if everything is okay, you can try `which python`, that would return something like:

```
/Users/kwang/my_venvs/assignment1/bin/python
```

where `/Users/kwang/` is your home folder, that is, ∼ /. Also, at this point you should also have `ipython`, which is a prettier version of the python command line interface. Go ahead to try importing `numpy` after launching `ipython` to try things out.

3.2 Reading, display, and save an image (20 points)

Write a `Python` script that performs the following:

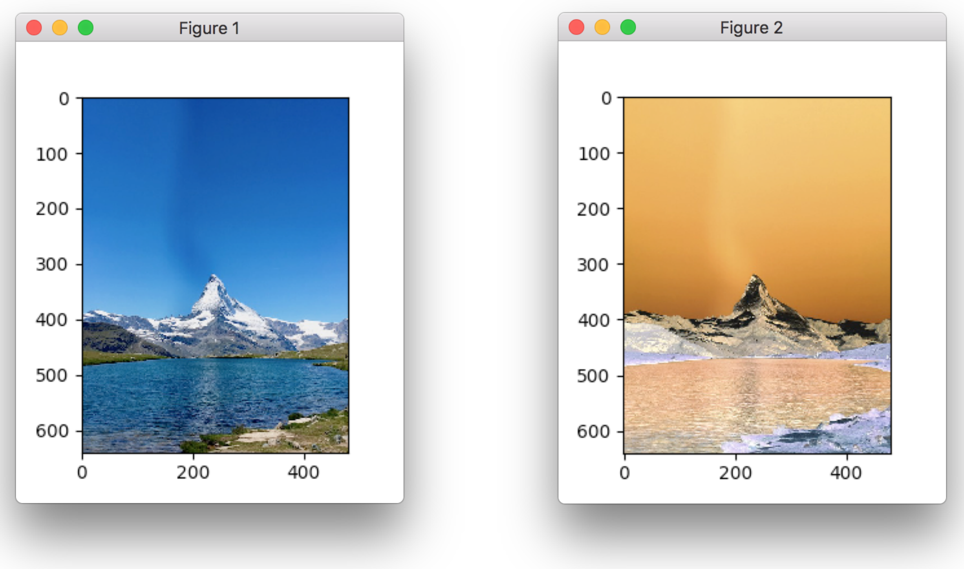
1. Reads `input.jpg`, provided in the assignment package, or any image that is of size 640×480.
2. Displays the image using `matplotlib` or `OpenCV` as shown in Fig. 1a(attach screenshot to report).
3. Inverts the image color by subtracting the image from a white image.
4. Displays the inverted image as shown in Fig. 1b (attach screenshot to report).
5. Save the **inverted** image to `output.h5` in `hdf5` format. `hdf5` is a standard format for storing data easily. Especially with the `h5py` library, the interface is quite straightforward. For future assignments, data will be provided in this format. Save the image in the `data` group.

For example, in the report, you should have a screen shot showing that you successfully read and displayed your image of choice, as well as the inverted image. You should include in your submission the original image, named `input.jpg`, and your script `solution-3.2.py` should save the `hdf5` file as `output.h5`.

Hints When loading an image, it is typically in the range $[0, 255]$. To invert an image, you simply need to subtract each elements from 255. Also, once you have the `h5py.File` object, you can use it as if it were a python dictionary.

Useful functions

- `PIL.Image.open`
- `PIL.Image.save`
- `numpy.asarray`
- `matplotlib.pyplot.imshow`
- `matplotlib.pyplot.show`
- `h5py.File`



(a) (b)

Figure 1: Example image: (a) original image, and (b) inverted image.

Useful reads

- Numpy ndarray documentation
- Matplotlib image tutorial
- H5py documentation
- Numpy tutorial by Justin Johnson

3.3 Finding the edges of an image (20 points)

Now that you can load an image, you can start doing some basic Computer Vision task. For this assignment, let's try extracting edges and corners through difference of Gaussian filtering (DoG). Don't worry if you have no idea what this is about, as I don't expect you to. This is just a simple image filtering and processing example, to help you refresh your memory.

For now, since this *is* a preliminary, we won't go into details on the idea and the maths behind how and why DoG filtering detects edges. We will focus on implementing it and running it on the image that you previously used for Section. 3.2. DoG is nothing but a difference of two Gaussian blurs. For this task, implement a python script that performs the following:

1. Reads the `output.h5` file from Section. 3.2.
2. Makes image single-channel by taking the mean along the last dimension.
3. Reads command-line arguments for kernel size $k1$ and $k2$. For example, $k1$ and $k2$ can be 2 and 4, respectively.
4. Applies Gaussian filtering on the image with kernel size $k1$.
5. Applies another Gaussian filtering on the image with kernel size $k2$.
6. Subtracts $k1$ result from $k2$ result.

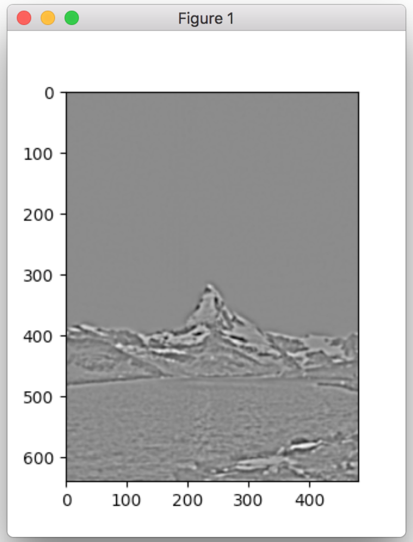


Figure 2: Example DoG image.

7. Normalises the result to be between 0 and 1.
8. Displays result as shown in Fig. 2.
9. Saves to `filtered.h5`, again the `data` group.

Important There are tons of ways to parse the command-line arguments. However, in this assignment, please use `sys.argv` to get your arguments.

Also for dealing with the pixels near the boundary of the image, use *reflect* mode. See the SciPy documentation for gaussian filtering and OpenCV image filtering documentation more details.

Hints

- `sys.argv`
- `scipy.ndimage.filters.gaussian_filter`

3.4 Thresholding (10 points)

The final part of the assignment is to threshold the filtered image and leave only the "interesting" pixel values. Write a `Python` script that:

1. Reads the `filtered.h5` file from Section. 3.3.
2. Thresholds the filtered image to keeps approximately 5% of the pixels that have the highest values, and the rest is set to zero, as shown in Fig. 3
3. Displays result as in Fig. 3.
4. Saves to `thresholded.h5`, again the `data` group.

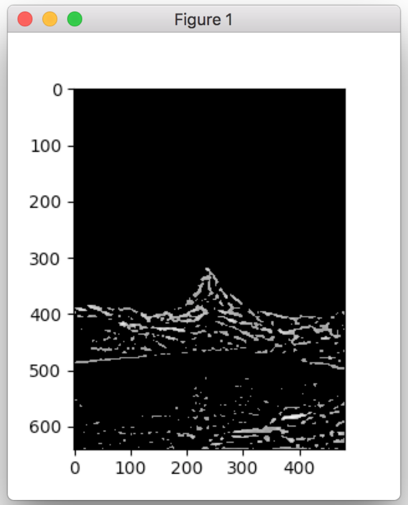


Figure 3: Example of a thresholded image.

Hints

- `numpy.sort`