In this assignment you will practice writing backpropagation code, and training

Neural Networks and Convolutional Neural Networks. The goals of this assignment

are as follows:

- understand \*\*Neural Networks\*\* and how they are arranged in layered

architectures

- understand and be able to implement (vectorized) \*\*backpropagation\*\*

- implement various \*\*update rules\*\* used to optimize Neural Networks

- implement \*\*batch normalization\*\* for training deep networks

- implement \*\*dropout\*\* to regularize networks

- effectively \*\*cross-validate\*\* and find the best hyperparameters for Neural

Network architecture

- understand the architecture of \*\*Convolutional Neural Networks\*\*

- gain an understanding of how a modern deep learning library (PyTorch) works

and gain practical experience using it to train models.

## Setup

Make sure your machine is set up with the assignment dependencies.

\*\*[Option 1] Use Anaconda:\*\*

The preferred approach for installing all the assignment dependencies is to use

[Anaconda](https://www.continuum.io/downloads), which is a Python distribution

that includes many of the most popular Python packages for science, math,

engineering and data analysis. Once you install it you can skip all mentions of

requirements and you are ready to go directly to working on the assignment.

\*\*[Option 2] Manual install, virtual environment:\*\*

If you do not want to use Anaconda and want to go with a more manual and risky

installation route you will likely want to create a

[virtual environment](http://docs.python-guide.org/en/latest/dev/virtualenvs/)

for the project. If you choose not to use a virtual environment, it is up to you

to make sure that all dependencies for the code are installed globally on your

machine. To set up a virtual environment, run the following:

```bash

cd assignment1

sudo pip install virtualenv # This may already be installed

virtualenv .env # Create a virtual environment

source .env/bin/activate # Activate the virtual environment

pip install -r requirements.txt # Install dependencies

# Work on the assignment for a while ...

deactivate # Exit the virtual environment

```

\*\*Download data:\*\*

Once you have the starter code, you will need to download the CIFAR-10 dataset.

Run the following from the `assignment1` directory:

```bash

cd deeplearning/datasets

./get\_datasets.sh

```

If you are on Mac, this script may not work if you do not have the wget command

installed, but you can use curl instead with the alternative script.

```bash

cd deeplearning/datasets

./get\_datasets\_curl.sh

```

\*\*Compile the Cython extension:\*\* Convolutional Neural Networks require a very

efficient implementation. We have implemented of the functionality using

[Cython](http://cython.org/); you will need to compile the Cython extension

before you can run the code. From the `deeplearning` directory, run the following

command:

```bash

python setup.py build\_ext --inplace

```

\*\*Start IPython:\*\*

After you have the CIFAR-10 data, you should start the IPython notebook server

from the `assignment1` directory. If you are unfamiliar with IPython, you should

read our [IPython tutorial](http://cs231n.github.io/ipython-tutorial/).

\*\*NOTE:\*\* If you are working in a virtual environment on OSX, you may encounter

errors with matplotlib due to the

[issues described here](http://matplotlib.org/faq/virtualenv\_faq.html).

You can work around this issue by starting the IPython server using the

`start\_ipython\_osx.sh` script from the `assignment1` directory; the script

assumes that your virtual environment is named `.env`.

### Submitting your work:

Once you are done working run the `collectSubmission.sh` script;

this will produce a file called `assignment1.zip`.

Upload this file to Gradescope.

Note that Gradescope will run an autograder on the files you submit. For some

test cases, there is a nonzero (but should be very low) probability that correct

implementations may fail due to randomness. If you think your implementation is

correct, then you can simply resubmit to rerun the autograder to check whether

it really is just a particularly unlucky seed..

### Q1: Fully-connected Neural Network (35 points)

The IPython notebook `FullyConnectedNets.ipynb` will introduce you to our

modular layer design, and then use those layers to implement fully-connected

networks of arbitrary depth. To optimize these models you will implement several

popular update rules.

### Q2: Batch Normalization (25 points)

In the IPython notebook `BatchNormalization.ipynb` you will implement batch

normalization, and use it to train deep fully-connected networks.

### Q3: Dropout (10 points)

The IPython notebook `Dropout.ipynb` will help you implement Dropout and explore

its effects on model generalization.

### Q4: ConvNet (20 points)

In the IPython Notebook `ConvolutionalNetworks.ipynb` you will implement several

new layers that are commonly used in convolutional networks as well as implement

a small convolutional network.

### Q5: Train a model on CIFAR10 using Pytorch! (10 points)

Now that you've implemented and gained an understanding for many key components

of a basic deep learning library, it is time to move on to a modern deep learning

library: Pytorch. Here, we will walk you through the key concepts of PyTorch, and

you will use it to experiment and train a model on CIFAR10. We highly recommend

you use Google Colab (https://colab.research.google.com/) for this notebook, as it

comes with Pytorch installed and provides access to GPUs.

If you use Colab for this notebook, make sure to manually download the completed

notebook and place it in the assignment directory before submitting. Also remember

to download required output file and place it into submission\_logs/ directory.