CS231n Convolutional Neural Networks for Visual Recognition

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** and train gain experience with training these models on data

Setup

You can work on the assignment in one of two ways: locally on your own machine, or on a virtual machine on Google Cloud.

Working remotely on Google Cloud (Recommended)

Note: after following these instructions, make sure you go to **Working on the assignment** below (you can skip the **Working locally** section).

As part of this course, you can use Google Cloud for your assignments. We recommend this route for anyone who is having trouble with installation set-up, or if you would like to use better CPU/GPU resources than you may have locally.

GPU Resources

A GPU will only help on Question 5. Please see the Google Cloud GPU set-up tutorial here for instructions. The GPU instances are much more expensive, so use them only when needed.

Once you've got the cloud instance running, make sure to run the following line to enter the virtual environment that we prepared for you (you do **not** need to make your own virtual

environment):

```
source /home/cs231n/myVE35/bin/activate
```

We strongly recommend using Google Cloud with GPU support for the **Question 5** of this assignment (the TensorFlow or PyTorch notebooks), since your training will go much, much faster. However, it will not help on any of the other questions.

Working locally

Here's how you install the necessary dependencies:

(OPTIONAL) Installing GPU drivers: If you choose to work locally, you are at no disadvantage for the first parts of the assignment. For the last question, which is in TensorFlow or PyTorch, however, having a GPU will be a significant advantage. We recommend using a Google Cloud Instance with a GPU, at least for this part. If you have your own NVIDIA GPU, however, and wish to use that, that's fine – you'll need to install the drivers for your GPU, install CUDA, install cuDNN, and then install either TensorFlow or PyTorch. You could theoretically do the entire assignment with no GPUs, though this will make training much slower in the last part. However, our reference code runs in 10-15 minutes on a dual-core laptop without a GPU, so it is certainly possible.

Installing Python 3.5+: To use python3, make sure to install version 3.5 or 3.6 on your local machine. If you are on Mac OS X, you can do this using Homebrew with `brew install python3`. You can find instructions for Ubuntu here.

Virtual environment: If you decide to work locally, we recommend using virtual environment 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:

```
cd assignment2
sudo pip install virtualenv      # This may already be installed
python3 -m venv .env                     # Create a virtual environment
source .env/bin/activate         # Activate the virtual environment
pip install -r requirements.txt  # Install dependencies
# Note that this does NOT install TensorFlow or PyTorch,
# which you need to do yourself.
```

```
# Work on the assignment for a while ...
# ... and when you're done:
deactivate                          # Exit the virtual environment
```

Note that every time you want to work on the assignment, you should run `source .env/bin/activate` (from within your `assignment2` folder) to re-activate the virtual environment, and `deactivate` again whenever you are done.

Working on the assignment:

Get the code as a zip file here.

Download data:

Once you have the starter code (regardless of which method you choose above), you will need to download the CIFAR-10 dataset. Run the following from the `assignment2` directory:

```
cd cs231n/datasets
./get_datasets.sh
```

Start IPython:

After you have the CIFAR-10 data, you should start the IPython notebook server from the `assignment2` directory, with the `jupyter notebook` command. (See the Google Cloud Tutorial for any additional steps you may need to do for setting this up, if you are working remotely)

If you are unfamiliar with IPython, you can also refer to our IPython tutorial.

Some Notes

NOTE 1: This year, the `assignment2` code has been tested to be compatible with python versions `3.5` and `3.6` (it may work with other versions of `3.x`, but we won't be officially supporting them). For this assignment, we are NOT officially supporting python2. Use it at your own risk. You will need to make sure that during your `virtualenv` setup that the correct version of `python` is used. You can confirm your python version by (1) activating your

virtualenv and (2) running `python --version`.

NOTE 2: If you are working in a virtual environment on OSX, you may *potentially* encounter errors with matplotlib due to the issues described here. In our testing, it seems that this issue is no longer present with the most recent version of matplotlib, but if you do end up running into this issue you may have to use the `start_ipython_osx.sh` script from the `assignment1` directory (instead of `jupyter notebook` above) to launch your IPython notebook server. Note that you may have to modify some variables within the script to match your version of python/installation directory. The script assumes that your virtual environment is named `.env`.

Submitting your work:

Whether you work on the assignment locally or using Google Cloud, once you are done working run the `collectSubmission.sh` script; this will produce a file called `assignment2.zip`. Please submit this file on Canvas.

Q1: Fully-connected Neural Network (25 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: Convolutional Networks (30 points)

In the IPython Notebook ConvolutionalNetworks.ipynb you will implement several new layers that are commonly used in convolutional networks.

Q5: PyTorch / TensorFlow on CIFAR-10 (10 points)

For this last part, you will be working in either TensorFlow or PyTorch, two popular and powerful deep learning frameworks. **You only need to complete ONE of these two notebooks.** You do NOT need to do both, but a very small amount of extra credit will be awarded to those who do.

Open up either `PyTorch.ipynb` or `TensorFlow.ipynb`. There, you will learn how the framework works, culminating in training a convolutional network of your own design on CIFAR-10 to get the best performance you can.

Q5: Do something extra! (up to +10 points)

In the process of training your network, you should feel free to implement anything that you want to get better performance. You can modify the solver, implement additional layers, use different types of regularization, use an ensemble of models, or anything else that comes to mind. If you implement these or other ideas not covered in the assignment then you will be awarded some bonus points.

---

cs231n
cs231n
karpathy@cs.stanford.edu