View All Pages

Assignment 2 Description

In this assignment you will implement recurrent networks, and apply them to image captioning on Microsoft COCO. You will also explore methods for visualizing the features of a pretrained model on ImageNet, and also this model to implement Style Transfer.

The goals of this assignment are as follows:

- Understand the architecture of *recurrent neural networks (RNNs)* and how they operate on sequences by sharing weights over time
- Understand and implement both Vanilla RNNs and Long-Short Term Memory (LSTM) RNNs
- Understand how to sample from an RNN language model at test-time
- Understand how to combine convolutional neural nets and recurrent nets to implement an image captioning system
- Understand how a trained convolutional network can be used to compute gradients with respect to the input image
- Implement and different applications of image gradients, including saliency maps, fooling images, class visualizations.
- Understand and implement style transfer.

Setup

You can work on the assignment in one of three ways: locally on your own machine, on the EECS instructional machines, or on a virtual machine in EC2..

Working remotely on EC2

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

You can sign up for individual EC2 credits with this link:

https://aws.amazon.com/education/awseducate/

GPU Resources

GPUs are **not required** for this assignment, but will help to speed up training and processing time for questions 3-4. They will be important for future assignments. At current rates, you get around 100 hours of GPU credits vs 600 hours for a non-GPU instance with an AWS Educate student account at Berkeley. Q3 and Q4 run in under a minute even on CPU. There are many instance types available. p2.xlarge is a recommended GPU instance, while c5.xlarge is a good CPU instance. Other instances have more compute power in powers of two, but price scales proportionately.

Here is more advice on setting up an EC2 instance.

Once your instance is setup, you will want to activate a pre-configured virtualenv. You can use either tensorflow or pytorch environments for Python 3.6. The commands to active then envs are listed in the startup message for the instance. Do either:

```
source activate tensorflow_p36
or
source activate pytorch_p36
```

When working with tensorflow on the EC2 instance there are two slight changes to the workflow below. Firstly, there will be a glitch during compilation unless you do this:

```
cd ~/anaconda3/envs/tensorflow/p36/compiler_compat
mv ld ldold
```

Next, to install requirements, (after downloading the assignment code) go to the assignment2 directory and install the tensorflow-specific requirements with:

```
pip install -r requirements_tf.txt
```

This is necessary because tensorflow requires a later version of numpy (1.14.0). You should be able to start ipython now. In case you get a "GLIBCXX_3.4.20 not found" error, you will need to do this. For some reason, we saw this on some instances and not others.

If you're using PyTorch install the requirements normally:

```
pip install -r requirements.txt
```

Working on EECS Instructional Machines

Follow these directions

Working locally

Here's how you install the necessary dependencies:

(OPTIONAL) Installing Nvidia GPU drivers: If you choose to work locally, you are at no disadvantage for the assignment. 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 9.0, install cuDNN 7.0, and then install TensorFlow .

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
virtualenv -p python3 .env             # Create a virtual environment (python3)
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 COCO captioning data, pretrained SqueezeNet model (TensorFlow-only), and a few ImageNet validation images. Run the following from the `assignment2` directory:

```
cd deeplearning/datasets
./get_assignment2_data.sh
```

Start IPython:

After you have downloaded the data, you should start the IPython notebook server from the `assignment2` directory, with the `jupyter notebook` command.

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 `assignment2` 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 on EC2, once you are done working run the `collectSubmission.sh` script; this will produce a file called `assignment2.zip`. Please submit this file here.

You can do Questions 3 and 4 in TensorFlow or PyTorch. There are two versions of each notebook, with suffixes -TensorFlow or -PyTorch. No extra credit will be awarded if you do a question in both TensorFlow and PyTorch.

Q1: Image Captioning with Vanilla RNNs (30 points)

The Jupyter notebook `RNN_Captioning.ipynb` will walk you through the implementation of an image captioning system on MS-COCO using vanilla recurrent networks.

Q2: Image Captioning with LSTMs (30 points)

The Jupyter notebook `LSTM_Captioning.ipynb` will walk you through the implementation of Long-Short Term Memory (LSTM) RNNs, and apply them to image captioning on MS-COCO.

Q3: Network Visualization: Saliency maps, Class Visualization, and Fooling Images (20 points)

The Jupyter notebooks `NetworkVisualization-TensorFlow.ipynb` / `NetworkVisualization-PyTorch.ipynb` will introduce the pretrained SqueezeNet model, compute gradients with respect to images, and use them to produce saliency maps and fooling images. Please complete only one of the notebooks (TensorFlow or PyTorch). No extra credit will be awardeded if you complete both notebooks.

Q4: Style Transfer (20 points)

In the Jupyter notebooks `StyleTransfer-TensorFlow.ipynb` / `StyleTransfer-PyTorch.ipynb` you will learn how to create images with the content of one image but the style of another. Please complete only one of the notebooks (TensorFlow or PyTorch). No extra credit will be awarded if you complete both notebooks.

Submit your assignment here.

Based on on cs231n Spring 2017 assignment 3. karpathy@cs.stanford.edu