Note: this is the 2017 version of this assignment.

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. Finally, you will train a generative adversarial network to generate images that look like a training dataset!

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.
- Understand how to train and implement a generative adversarial network (GAN) to produce images that look like a dataset.

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

GPUs are **not required** for this assignment, but will help to speed up training and processing time for questions 3-5. 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. **We only recommend a GPU for Q5 (GANs)**, Q3 and Q4 run in under a minute even on CPU, while Q5 can take 30+ minutes for your model to converge on a CPU (versus under a minute on the Google Cloud GPUs).

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 recommend using Google Cloud with GPU support for the question 5 of this assignment (the GAN notebook), since your training will go much, much faster. However, it will not help at all for questions 1 and 2 (RNN and LSTM), and questions 3 and 4 are still fast on CPU (these notebook should run in a few minutes).

What do I do if my Google Cloud GPUs disappeared?

You might note that sometimes, your GPUs are no longer accessible on your Google Cloud instance after you restart it. If this happens, please run the following commands in your assignment3 directory:

```
sudo apt-get remove unattended-upgrades
chmod u+x where_are_my_drivers.sh
./where_are_my_drivers.sh
```

If this isn't working, you can find more detailed instructions and manual ways of fixing this here. You should follow the "Ubuntu 16.04" instructions.

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
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 `assignment3` 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 `assignment3` directory:

```
cd cs231n/datasets
./get_assignment3_data.sh
```

Start IPython:

After you have downloaded the data, you should start the IPython notebook server from the `assignment3` 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 `assignment3` 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 `assignment3` 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 `assignment3.zip`. Please submit this file on Canvas.

You can do Questions 3, 4, and 5 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 (25 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 (15 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 (15 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 awardeded if you complete both notebooks.

Q5: Generative Adversarial Networks (15 points)

In the Jupyter notebooks `GANs-TensorFlow.ipynb`/`GANs-PyTorch.ipynb` you will learn how to generate images that match a training dataset, and use these models to improve classifier performance when training on a large amount of unlabeled data and a small amount of labeled data. Please complete only one of the notebooks (TensorFlow or PyTorch). No extra credit will be awarded if you complete both notebooks.