In this assignment you will implement recurrent networks, and apply them to image captioning on Microsoft COCO. We will also introduce the TinyImageNet dataset, and use a pretrained model on this dataset to explore different applications of image gradients.

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 the difference between vanilla RNNs and Long-Short Term Memory (LSTM) RNNs
- Understand how to sample from an RNN 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, feature inversion, and DeepDream.

# Setup

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

## Working in the cloud on Terminal

Terminal has created a separate subdomain to serve our class, www.stanfordterminalcloud.com. Register your account there. The Assignment 3 snapshot can then be found HERE. If you are registered in the class you can contact the TA (see Piazza for more information) to request Terminal credits for use on the assignment. Once you boot up the snapshot everything will be installed for you, and you will be ready to start on your assignment right away. We have written a small tutorial on Terminal here.

## Working locally

Get the code as a zip file here. As for the dependencies:

**[Option 1] Use Anaconda:** The preferred approach for installing all the assignment dependencies is to use Anaconda, 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 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 assignment3
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 processed MS-COCO dataset, the TinyImageNet dataset, and the pretrained TinyImageNet model. Run the following from the `assignment3` directory:

```
cd cs231n/datasets
./get_coco_captioning.sh
./get_tiny_imagenet_a.sh
./get_pretrained_model.sh
```

**Compile the Cython extension:** Convolutional Neural Networks require a very efficient implementation. We have implemented of the functionality using Cython; you will need to compile the Cython extension before you can run the code. From the `cs231n` directory, run the following command:

```
python setup.py build_ext --inplace
```

**Start IPython:** After you have the data, you should start the IPython notebook server from the `assignment3` directory. If you are unfamiliar with IPython, you should read our 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. You can work around this issue by starting the IPython server using the `start_ipython_osx.sh` script from the `assignment3` directory; the script assumes that your virtual environment is named `.env`.

## Submitting your work:

Whether you work on the assignment locally or using Terminal, once you are done working run the `collectSubmission.sh` script; this will produce a file called `assignment3.zip`. Upload this file under the Assignments tab on the coursework page for the course.

## Q1: Image Captioning with Vanilla RNNs (40 points)

The IPython 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 (35 points)

The IPython 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: Image Gradients: Saliency maps and Fooling Images (10 points)

The IPython notebook `ImageGradients.ipynb` will introduce the TinyImageNet dataset. You will use a pretrained model on this dataset to compute gradients with respect to the image, and use them to produce saliency maps and fooling images.

## Q4: Image Generation: Classes, Inversion, DeepDream (15 points)

In the IPython notebook `ImageGeneration.ipynb` you will use the pretrained TinyImageNet model to generate images. In particular you will generate class visualizations and implement feature inversion and DeepDream.

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

Given the components of the assignment, try to do something cool. Maybe there is some way to generate images that we did not implement in the assignment?

---

cs231n
cs231n
karpathy@cs.stanford.edu