Assignment #2
ADL, Spring 2019
Due March 4th

Recognize Landmarks on Columbia's Campus

About: In this assignment, you will gain real-world experience with practical image classification on small datasets -- a common scenario in practice. In part one, you will work with an existing dataset. In part two, you will collect a small dataset yourself. Of course, in practice there isn't often a dataset available for tasks we care about, so it's valuable to get a feel for starting from scratch.

Submission instructions: Please submit this assignment on CourseWorks by uploading one or more Jupyter notebooks.

- You may submit a single notebook for the entire assignment, or you may submit a zip including one notebook for each part.

- Reminder, please be sure your notebooks include saved output that shows the results of training your models, including curves that show the loss on your training and validation sets.

- You do not need to upload the dataset you collect, but please but sure a few images from each class are displayed in your notebook.

Extra credit: There's an extra credit section at the bottom of this assignment. As always, it's optional. If you choose to complete some or all of it, please include your work with your submission in one or more Jupyter notebooks.

Part 1 (30 points)**:** Train a model on an existing dataset

Starter code: Work through notebooks 4.1-cats-dogs.ipynb, 4.2-data-augmentation.ipynb, and 4.3-transfer-learning.ipynb. These contain most of the code you need for this assignment, please base your work off them[1].

1. Download the mini-flowers dataset. This contains 1,500 images of five different types of flowers. Modify one of the above notebooks to classify these images using transfer learning. How accurate of a model can you train? *Note: the example notebooks are for*

[1] This material is similar to the book Deep Learning with Python, please be sure to read chapter 5 before getting started (the main difference here is the imports).

binary classification. You will need to change the output layer shape, activation, and loss function for multiclass classification.

2. Next, run experiments using at least two pretrained convolutional bases (applications), and compare your results. Include a short, informal write-up (using bullet points is fine). What differences do you see, and why? Read the associated papers to learn more about the networks you're using, linked from the API doc.

Part 2 (70 points)**:** Train a model to recognize landmarks on Columbia's campus.

In this part of the assignment, you will collect a dataset containing images of three landmarks on Columbia's campus (example), and train a model to identify them.

Starter code: Please base your work off the notebooks from part one. As before, you'll need to modify them for multiclass classification.

Tip: you can collect a dataset in two ways.

- **The slow way**. Using your smartphone, snap a bunch of pictures of each landmark (one at a time).

- **The fast way**. Using your smartphone, record a video of each landmark. Later, split the video into frames with FFmpeg.

    a. This is a bit of a hack (and would only really be appropriate if later you wanted to classify images from a video stream - but for our purposes - it's fun and saves time).

    b. Another tip: record a few short videos of each landmark (you can walk around it while you record). Split these videos into frames - and discard a bunch of them (you may want to keep every *Nth* frame, since many will be similar).

In either approach, it's important to collect diverse training data (you'll want to image the landmark from different locations, orientations, and with different background conditions, etc).

1. Collect a dataset of at least three landmarks. Your dataset should include at least 100 images of each in train, 50 in validation, and 25 in test (using more images is fine). You can randomly shuffle your dataset to create these splits.

2. Write a model to classify your dataset using transfer learning. Run an experiment and report your results. What do you find?

3. Next, how small of a model (in terms of the number of parameters) can you write to classify these images reasonably well? Explore the available pretrained models, and see if any are suitable. Run an experiment and report your results.

Extra credit
You may complete some of all of these problems, in any order.

EC1: Classify QuickDraw images.
If you're up for more image classification, use the QuickDraw Loader to create a dataset of a bunch of different classes (you can use "animals" to start). Write a CNN to classify these images, and report your accuracy. How accurate of a model can you train to recognize 50 classes? 100? More?