Assignment #2
COMS 4995.06. Fall 2018.
Due Oct 15th.

About: In this assignment you will practice implementing deep neural networks in TensorFlow. The goal is for you to gain experience with TensorFlow's latest coding styles, including eager execution, tf.keras model subclassing. This combination is relatively low-level, and will serve you well while learning the details of deep learning, and if you choose to do research later on.

- Download your **starter code:** the file is on CourseWorks "assignment_2_starter.ipynb".

For students who would like an added challenge, there are extra credit questions at the end of this document. The last is opened ended, and we'd be curious to hear your thoughts!

Part 1 (20 points): Warm up with the tf.keras Sequential API

We'll start by using the tf.keras Sequential API (without eager execution enabled) to implement two classifiers on MNIST:

1. A linear model, that's at least 80% accurate.

2. A deep model that's as accurate as possible **without** overfitting to the test set provided by the default dataset loader.

This part of the assignment should take a relatively small amount of code. You are free to use layer types, optimizers, and weight initialization strategies we haven't covered yet in class, although these will not be necessary.

- Be sure that you're using tf.keras (inside TensorFlow), and not regular Keras (running with the TensorFlow backend).

- Include an evaluation of your model in your submission.

- No starter code is provided for this part of the assignment.

Part 2 (80 points) Run experiments using tf.keras model subclassing and eager

Use tf.keras model subclassing with eager execution enabled to:

3. Complete the linear model sketched in your starter code. Your finished model should be >80% accurate.

4. Add code to visualize the loss as you train the model using TensorBoard.

5. Implement your deep model from part one of the assignment. Visualize the training process as before.

6. Design and run experiments to compare:

    a. High, low, and reasonable learning rates.
    b. Different activation functions.
    c. Different gradient descent optimizers.
    d. Different weight initialization strategies.

    Visualize the results of your experiments in TensorBoard, and include relevant graphics your submission. Include a brief write up (bullet points is fine), what do you observe?

Extra credit / challenge questions.

1. *[Difficulty: relatively high]* Use tf.keras model subclassing and eager execution to train a model on the moons dataset. Reproduce this visualization from this article, or create one similar to it.

2. *[Difficulty: relatively low]* Provide your own implementation of gradient descent, softmax, and cross entropy loss. Run an experiment to compare your results against the built-in methods.

3. *[Difficulty: relatively low]* Provide your own implementation of Xavier initialization. Run an experiment comparing it to a simpler strategy, and show the results in TensorBoard.

4. *[Difficulty: medium]* Rewrite your linear model (bonus: and/or a simplified version of your deep model) from scratch in Python. Provide your own implementation of backpropagation. (It is not necessary to write TensorFlow compatible code). Compare your results on MNIST to the classifiers you implemented above. Test your implementation with the numeric gradient.

5. *[Difficulty: high]* Design and implement a procedure to monitor training progress and alert the user to potential problems, taking the design of the model into consideration. Suggest solutions to problems you detect.

You are welcome to submit a solution to problems #1 or #5 anytime before the final exam. Others must be submitted with your assignment.