Homework 3 **NSC 3270 / 6270**
Due Tue Sep 21 at start of class **Fall 2021**
20 points

In class, we went over the simplest possible neural network consisting of two input units and a single output unit. For this assignment, you will be exploring a more complicated neural network with 784 input units and 10 output units. The concepts are the same. The mathematics are the same. The increased scale means just that these models can only be simulated and evaluated computationally.

This assignment will also give you a little bit of exposure to Keras (part of Tensorflow), which we will use quite a bit throughout the semester. For now, there's no reason to understand all of the pieces of Keras and how they work (we will go over those in the coming weeks). A combination of lecture, the assignment text, and code in Homework3.ipynb, explain the parts you need to know to complete the assignment.

Note that this assignment is worth 20 points (the previous one was worth 10, the first worth 2), reflecting its (potential) greater difficulty and complexity. Do not leave this to the day before – if only because the probability of getting a helpful response to a question is greatly amplified if you ask a question well before the due date for the assignment.

The neural network you are given in Homework3.ipynb learns to classify hand-written digits 0…9. The database of digits is called MNIST and originally came from the National Institute of Standards and Technology (NIST) for evaluating systems that could read hand-written digits. In fact, the earliest "deep learning" networks were first used 20 years ago by the US Postal Service to read the (often hand-written) zip codes of the mail it processed. Each digit in the MNIST database is a 28x28 grayscale image (see class slides).

The code in Homework3.ipynb learns to classify the digits from training data (a set of 60,000 images) and then evaluates the performance of the network with testing data (a set of 10,000 images). For this assignment, you will evaluate the performance of the network, find examples of correct and incorrect test classifications, visualize the weights of the trained network, and use the weights and bias values of the trained network to calculate on your own the output values (confirming that calculated values match the outputs given by Keras).

Make sure you add your answers to each of the questions below to the Homework3.ipynb file and save it to a new file with your last name appended to "Homework3". Submit on Brightspace. There is no need to submit any figures with your code. Your Jupyter Notebook should create all the figures needed. Make sure you submit your notebook after it has been run (with figures and calculations shown).

Q1. The original MNIST `test_labels` numpy array contains the digit value associated with the corresponding digit image (`test_images`). The output from the network (from

`out = network.predict(test_images_vec)` in the code) contains the activations of the 10 output nodes for every test image presented to the network. Write a function that takes the (10000,10) numpy array of output (`out`) (of type float) activations and returns a (10000,) numpy array of discrete digit classification by the network (of type int).

Specifically, create a `test_decisions` numpy array of the same size and type as the MNIST `test_labels` array you started with. Whereas `test_labels` shows the correct answer, `test_decisions` shows the decision made by the network. Below you will use both arrays to pull out test images that the network classifies correctly vs. incorrectly.

To turn a numpy array of continuous output activations into a discrete digit classification, just take the maximum output as the "winner" that "takes all", determining the classification.

In your function, feel free to use for loops. Here, we are looking to see that you understand how to use the outputs generated by the network, not whether you can program using the most efficient Python style.

Q2. Comparing the correct answers (`test_labels`) and network classifications (`test_decisions`), for each digit 0..9, find one test image (`test_image`) that is classified by the network correctly and one test image that is classified by the network incorrectly.

Create a 2x10 plot of digit images (feel free to adapt the code above that uses subplot), with a column for each digit 0..9 with the first row showing examples correctly classified (one example for each digit) and the second row showing the examples incorrectly classified (one example for each digit). Each subplot title should show the answer and the classification response (e.g., displaying 4/2 as the title, if the correct answer is 4 and the classification was 2).

Q3. Create "images" of the connection weight adapting the code used to display the actual digit images. There should be 10 weight images, an image for each set of weight connecting the input layer (784 inputs) to each output node. You will want to reshape the (784,1) vector of weights to a (28,28) image and display the result using `imshow()`.

Q4. Use the weight matrix (W), bias vector (B), and activation function (simple sigmoid) to reproduce in your own code the outputs (out) generated by the network (from this out = network.predict(test_images_vec))

Recall that the simple sigmoid activation function is defined as follows:
f(x) = 1 / (1+exp(-x))

Confirm that your output vectors and the keras-produced output vectors are the same (within some small epsilon since floating point calculations will often not come out exactly the same on computers). If your calculated values are within 0.0001 of the values produced by Keras, all is well. You have some flexibility in how you determine this close correspondence. One technique involves subtracting one set of values from the other, and checking that the absolute values of the are less than the epsilon value you chose.

Reminder: Check the class slides for some additional hints and suggestions and examples of what some of the output might look like.

Unexcused late assignments will be penalized 10% for every 24 hours late, starting from the time class ends, for a maximum of two days, after which they will earn a 0.