**Lab 8: Neural Networks for Handwritten Digit Recognition, Multiclass**

**Goal:**

In this exercise, you will use a neural network to recognize the hand-written digits 0-9.

**Required Packages**

* numpy is the fundamental package for scientific computing with Python.
* matplotlib is a popular library to plot graphs in Python.
* tensorflow a popular platform for machine learning.

**1 Neural Networks**

**1.1 Problem Statement**

In this exercise, you will use a neural network to recognize ten handwritten digits, 0-9. This is a multiclass classification task where one of n choices is selected. Automated handwritten digit recognition is widely used today - from recognizing zip codes (postal codes) on mail envelopes to recognizing amounts written on bank checks.

**1.2 Dataset**

You will start by loading the dataset for this task.

* The np.load function shown below loads the data into variables `X` and `y`
* The data set contains 5000 training examples of handwritten digits 0-9.
* Each training example is a 20-pixel x 20-pixel grayscale image of the digit.
  + Each pixel is represented by a floating-point number indicating the grayscale intensity at that location.
  + The 20 by 20 grid of pixels is “unrolled” into a 400-dimensional vector.
  + Each training examples becomes a single row in our data matrix `X`.
  + This gives us a 5000 x 400 matrix `X` where every row is a training example of a handwritten digit image.

The second part of the training set is a 5000 x 1 dimensional vector `y` that contains labels for the training set

* + `y = 0` if the image is of the digit `0`, `y = 4` if the image is of the digit `4` and so on.

This is a subset of the MNIST handwritten digit dataset ([http://yann.lecun.com/exdb/mnist/)</sub](http://yann.lecun.com/exdb/mnist/)%3c/sub)>

**1.2.1 View the variables**

Let's get more familiar with your dataset.

- A good place to start is to print out each variable and see what it contains.

The code below prints the first element in the variables `X` and `y`.

print ('The first element of X is: ', X[0])

print ('The first element of y is: ', y[0,0])

print ('The last element of y is: ', y[-1,0])

**1.2.2 Check the dimensions of your variables**

Another way to get familiar with your data is to view its dimensions. Please print the shape of `X` and `y` and see how many training examples you have in your dataset.

print ('The shape of X is: ' + str(X.shape))

print ('The shape of y is: ' + str(y.shape))

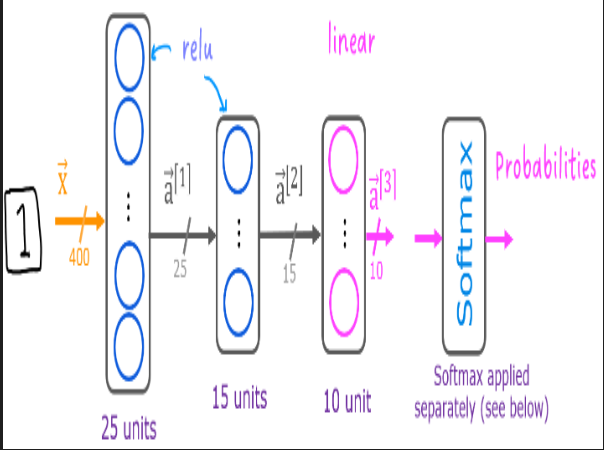
**1.2.3 Visualizing the Data**

**Task 1:** Write a Python code in order to visualize a subset of the training set. In particular, your code will do the following:

* + the code randomly selects 64 rows from `X`, maps each row back to a 20 pixel by 20 pixel grayscale image and displays the images together.
  + The label for each image is displayed above the image

**1.3 Model representation**

The neural network you will use in this lab is shown in the figure below.



- This has two dense layers with ReLU activations followed by an output layer with a linear activation.

* + Recall that our inputs are pixel values of digit images.
  + Since the images are of size 20x20, this gives us 400 inputs

- The parameters have dimensions that are sized for a neural network with 25 units in layer 1, 15 units in layer 2 and 10 output units in layer 3, one for each digit.

\*\*Note:\*\* The bias vector `b` could be represented as a 1-D (n,) or 2-D (n,1) array. Tensorflow utilizes a 1-D representation and this lab will maintain that convention:

Based on the model above and the loaded data in X and y, write python codes to perform the following tasks:

**Task 2:** Build a Neural Network shown in the above figure by using TensorFlow and Keras.

As described in the lecture, numerical stability is improved if the softmax is grouped with the loss function rather than the output layer during training. This has implications when \*building\* the model and \*using\* the model.

**Task 3: Examine the weights in the layers**

**Task 4:**

\* define a loss function, `SparseCategoricalCrossentropy` and indicates the softmax should be included with the loss calculation by adding `from\_logits=True`)

\* define an optimizer. A popular choice is Adaptive Moment (Adam) which was described in lecture.

**Task 5: Analyze the loass(cost) of training and algorithm convergence by plotting the loss**

**Task 6: Use Keras predict function to the image of two X[1015]**