**MNIST DATA Classification**

Will be training a network of handwritten numbers.

***Ten Thousand feet overview of the system:***

MNIST data set images are 28x28 pixels. The first step we do is flatten the pixels

28 x 28 = 784 pixels.

A very simple network representation:   
activation Function (weighted sum of all pixels + bias) example activation function softmax

**Softmax (Ln) = eLn / ||eL||**

**L = X.W + b**

X (input matrix) **n** by **No of pixels** (n is the number of batch size)

W (weights matrix) **No of pixels** by **No of classes** (in case of MNIST it will be 10, numbers 0-9)

**b** is bias

So after multiplying the 2 matrixes X & W and adding the Bias we get L, applying softmax on L we can our predictions.

**Y = Softmax (X.W + b)**Y will be a matrix batchsize x 10

Now we have to define what predictions are good. This will only work if the Weight and Biases are good, and this is how we define it.

Actual probabilities (one-hot) encoded

**Cross entropy: -**

Now the goal is to minimize the distance between what the system predicts and what we know is true

**Main Code**:

Let’s go through the core code how to do it in Tensorflow

**X = tf.placeholder (tf.float32,[None,32,32,1])** (None will hold the number of images batchsize and 1 is for grayscale for RGB we will use 3)

**W = tf.variable(tf.zeros([784,10]))**

**b = tf.variable(tf.zeros([10]))**

We define W and b as a variable because tensorflow will update these values to minimize the distance between the prediction and correct value

**Y = tf.nn.softmax(tf.matmul(tf.reshape(X,[-1,784]), W) +b)**  - predictions

**Y\_ = tf.placeholder(tf.float32, [None, 10])** – correct answers

**cross\_entropy = -tf.reduce\_sum(y\_ \* tf.log(Y))**

**is\_correct = tf.equal(tf.argmax(Y,1) , tf.argmax(Y\_,1))**

**accuracy = tf.reduce\_mean(tf.cast(is\_correct,tf.float32))**

Now we need to optimize:

**Optimizer = tf.train.GradientDecentOptimizer(training Rate)**

**Train\_step = optimizer.minimize(cross\_entropy)**