INTRODUCTION TO DEEP LEARNING - DEEP LEARNING BASICS WITH PYTHON, TENSORFLOW AND KERAS

The Code aims at predicting a 28x28 pixel an image of handwritten numbers using TensorFlow and Keras.

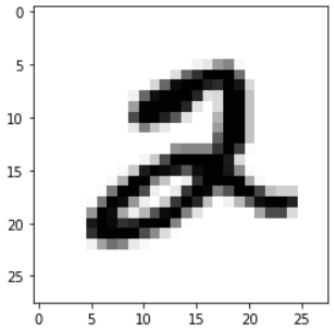
We extract the data 'mnist' from 'tf.keras.datasets'. We normalize the data to get the data between 0 and 1.

```
mnist = tf.keras.datasets.mnist # 28x28 images of handwritten digits 0-9
```

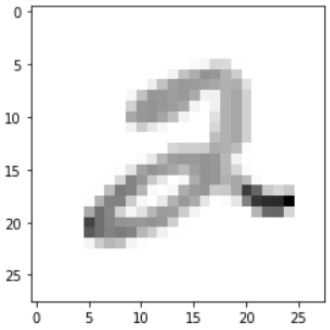
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
#Since the pixel data is between 0-255,
#we normalize it to be between 0-1 for network to understand easily
x_train = tf.keras.utils.normalize(x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis = 1)
```

The data before normalizing:



The data after normalizing:



Create a model and add layers to Flatten the data and train the model efficiently. The output layer consists of 10 neurons which provides an information of number being from 0-9 in the image.

The hidden layers, however, have 128 nodes (which is a random choice, chosen based on convenience).

```
#sequential is the basic model
model = tf.keras.models.Sequential()

# we add hidden layers into our network. 128 is the number of neurons that we use
# relu is a basic activation sigmoid function
# softmax is used for probability distribution (used in final/output layer with 10 neurons)
# we reduce 28x28 array to a flatten layer as it is multidimensional and hard to compute
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(128, activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(10, activation = tf.nn.softmax))

#parameters for training for the model

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])

model.fit(x_train,y_train, epochs = 3)
```

```
Epoch 1/3
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2609 - accuracy: 0.9234
Epoch 2/3
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1080 - accuracy: 0.9668
Epoch 3/3
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0726 - accuracy: 0.9774
```

The max of prediction () is going to be the actual number predicted in the image.

If the number predicted is 7, then the 7th index of prediction is going to be the max value of prediction array.

The np.argmax() of the array gives zeroth index. Whereas the test data image actually holds handwritten ‘Zero’ on it.

```
new_model = tf.keras.models.load_model('epic_num_reader.model')
```

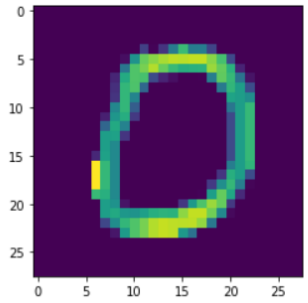
```
predictions = new_model.predict([x_test])
print(predictions[10])
```

```
[9.9961132e-01 1.1616173e-07 3.5142351e-04 1.0658619e-08 1.3585891e-07
 1.2460019e-05 5.6050249e-06 5.8885792e-07 1.0014011e-08 1.8284378e-05]
```

```
print(np.argmax(predictions[10]))
```

```
0
```

```
plt.imshow(x_test[10])
plt.show()
```



Hence the Neural network can predict the number in the image appropriately.