Name: Abhishek Bussa **Id No: 1001624753**

Data Mining

Homework 3

Problem 2

(Tensorflow and Keras, 40pts) Try out the tutorial for Deep Learning using Tensorflow at https://www.tensorflow.org/tutorials. It has the following lines of code.

1. (10pt) In the report, write comments for each line of code given above and explain what this framework is doing.

```
#Import the tensorflow library as tf

import tensorflow as tf

#Load the MNIST dataset. MNIST is a database of handwritten digits and is a subset of a larger dataset
(NIST) and has 60,000 samples as training and 10,000 samples as testing dataset

mnist = tf.keras.datasets.mnist

#Load the training and testing data into variables

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

#Convert the samples from integers to floating point values

x_train, x_test = x_train / 255.0, x_test / 255.0

#Build a sequential model. A sequential model is a linear stack of layers

model = tf.keras.models.Sequential([

#The first layer "Flatten", transforms the dimension of the image to a 1d array. The input samples are
now changed from 2d arrays to 1d array

   tf.keras.layers.Flatten(),

#This layer has 512 nodes and gives a probability values as an array of 512 values showing which class
the input belongs to.

   tf.keras.layers.Dense(512, activation=tf.nn.relu),

#Dropout rate of 0.2 drops 20% of the inputs to prevent overfitting which is a modeling error

   tf.keras.layers.Dropout(0.2),

#This layer has 10 nodes and uses softmax activation function to give an output of 10 probability values
showing which class the input belongs to

   tf.keras.layers.Dense(10, activation=tf.nn.softmax)

])
```

#We are adding a few settings to the model before compiling it

#Optimizer is used to update the model based on the loss function. This adjusts the weights of the network

#Loss function is used to check how accurate the model is. We need to minimize this function to get a good output and based on this value, the optimizer works to improve the model

#Metrics is used to monitor the training and testing data. Here we use accuracy to measure how accurate out model works, gives the value of the fraction of images that are correctly classified

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

#We input the training samples and their labels as input to the model to learn the data. Epochs gives how many times we want to run the training data on the model to update it

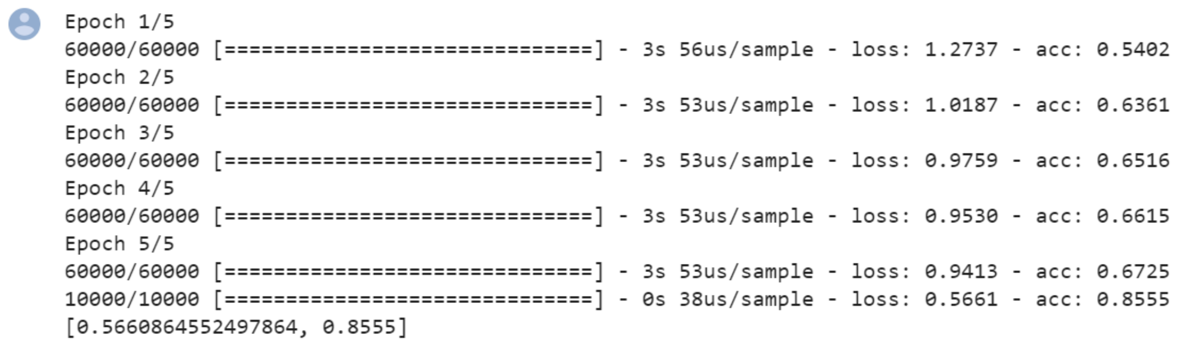
model.fit(x_train, y_train, epochs=5)

#Then we test the model by giving the testing samples and labels, then check the accuracy and loss determining how well our model makes predictions for the data

model.evaluate(x_test, y_test)

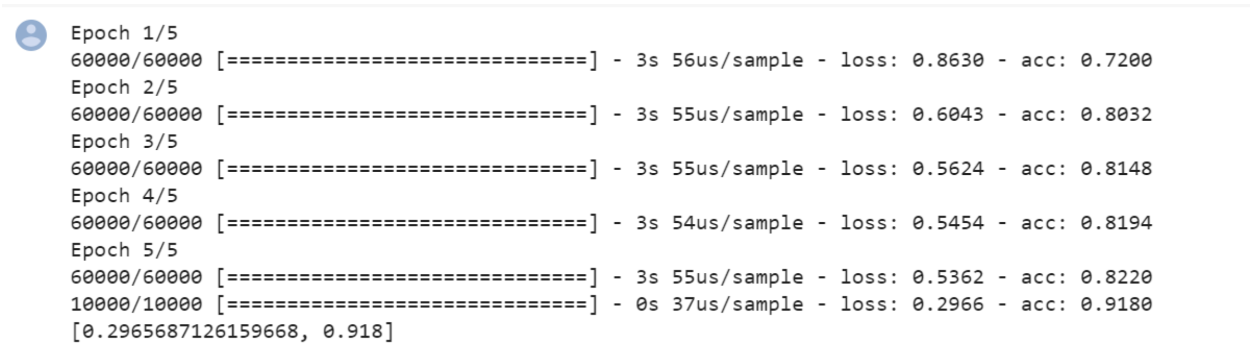
2. (10pt) Change the number of hidden nodes to 5, 10, 128 and 512. Report how the testing accuracy changes for the testing data. Report the result and your observation in the report.

5 hidden nodes: It gave an accuracy of 85.55%


```
Epoch 1/5
60000/60000 [==============================] - 3s 56us/sample - loss: 1.2737 - acc: 0.5402
Epoch 2/5
60000/60000 [==============================] - 3s 53us/sample - loss: 1.0187 - acc: 0.6361
Epoch 3/5
60000/60000 [==============================] - 3s 53us/sample - loss: 0.9759 - acc: 0.6516
Epoch 4/5
60000/60000 [==============================] - 3s 53us/sample - loss: 0.9530 - acc: 0.6615
Epoch 5/5
60000/60000 [==============================] - 3s 53us/sample - loss: 0.9413 - acc: 0.6725
10000/10000 [==============================] - 0s 38us/sample - loss: 0.5661 - acc: 0.8555
[0.5660864552497864, 0.8555]
```

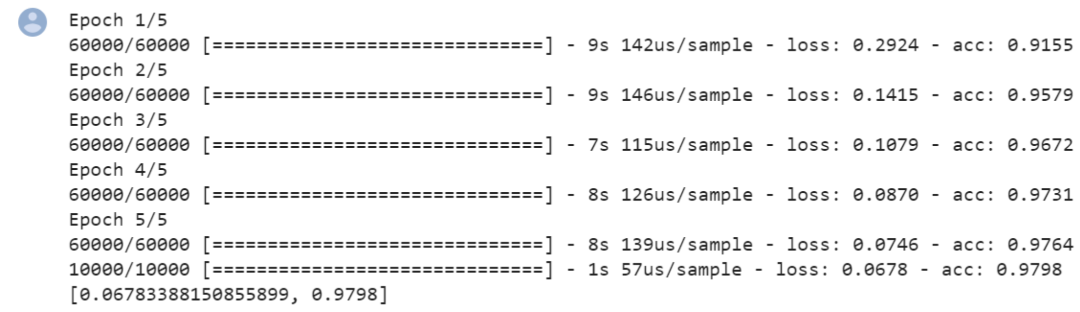
Name: Abhishek Bussa **Id No: 1001624753**

10 hidden nodes: It gave an accuracy of 91.8%



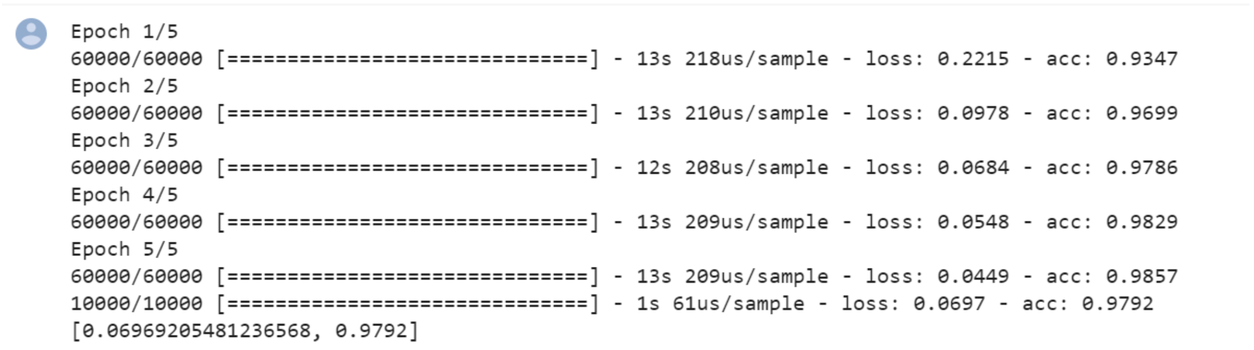
```
Epoch 1/5
60000/60000 [==============================] - 3s 56us/sample - loss: 0.8630 - acc: 0.7200
Epoch 2/5
60000/60000 [==============================] - 3s 55us/sample - loss: 0.6043 - acc: 0.8032
Epoch 3/5
60000/60000 [==============================] - 3s 55us/sample - loss: 0.5624 - acc: 0.8148
Epoch 4/5
60000/60000 [==============================] - 3s 54us/sample - loss: 0.5454 - acc: 0.8194
Epoch 5/5
60000/60000 [==============================] - 3s 55us/sample - loss: 0.5362 - acc: 0.8220
10000/10000 [==============================] - 0s 37us/sample - loss: 0.2966 - acc: 0.9180
[0.2965687126159668, 0.918]
```

128 hidden nodes: It gave an accuracy of 97.98%



```
Epoch 1/5
60000/60000 [==============================] - 9s 142us/sample - loss: 0.2924 - acc: 0.9155
Epoch 2/5
60000/60000 [==============================] - 9s 146us/sample - loss: 0.1415 - acc: 0.9579
Epoch 3/5
60000/60000 [==============================] - 7s 115us/sample - loss: 0.1079 - acc: 0.9672
Epoch 4/5
60000/60000 [==============================] - 8s 126us/sample - loss: 0.0870 - acc: 0.9731
Epoch 5/5
60000/60000 [==============================] - 8s 139us/sample - loss: 0.0746 - acc: 0.9764
10000/10000 [==============================] - 1s 57us/sample - loss: 0.0678 - acc: 0.9798
[0.06783388150855899, 0.9798]
```

512 hidden nodes: It gave an accuracy of 97.92%



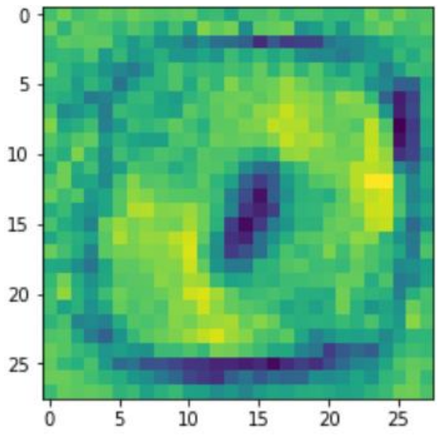
```
Epoch 1/5
60000/60000 [==============================] - 13s 218us/sample - loss: 0.2215 - acc: 0.9347
Epoch 2/5
60000/60000 [==============================] - 13s 210us/sample - loss: 0.0978 - acc: 0.9699
Epoch 3/5
60000/60000 [==============================] - 12s 208us/sample - loss: 0.0684 - acc: 0.9786
Epoch 4/5
60000/60000 [==============================] - 13s 209us/sample - loss: 0.0548 - acc: 0.9829
Epoch 5/5
60000/60000 [==============================] - 13s 209us/sample - loss: 0.0449 - acc: 0.9857
10000/10000 [==============================] - 1s 61us/sample - loss: 0.0697 - acc: 0.9792
[0.06969205481236568, 0.9792]
```

3. (20pt) Now, remove the hidden layer in the code and train the model. The trained model contains the weights that it has learned from training. Plot the "new representation" that it has learned for each number from training and include them in the report. That is, reshape the learned weights (i.e., vector) to the image dimension (in 2D, i.e., 28x28) and show them. You will see some number-like features.

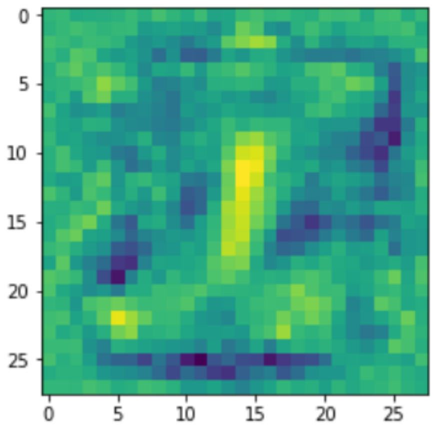
The learned weights as images is shown below in order (0-9):

Name: Abhishek Bussa **Id No: 1001624753**

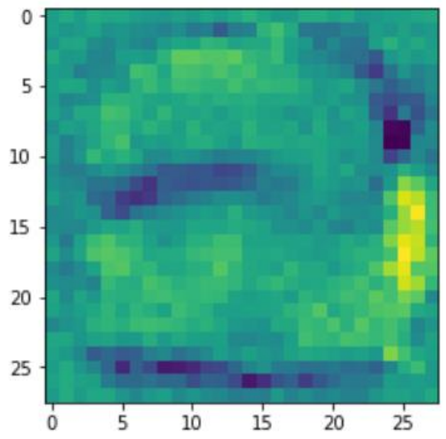
0



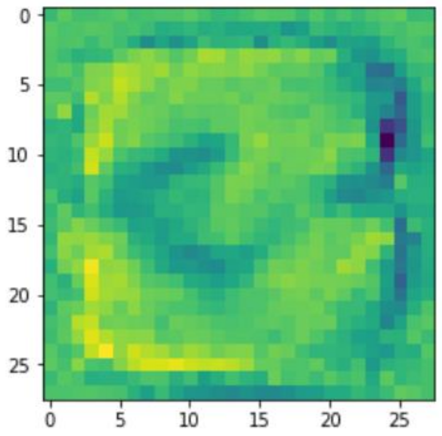
1



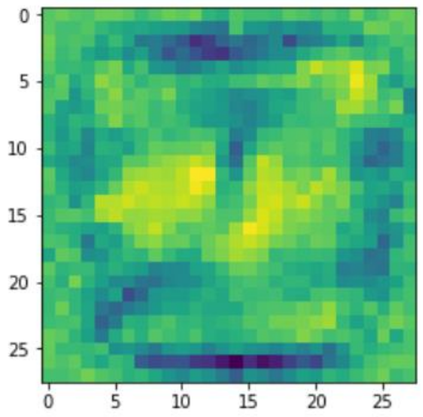
2



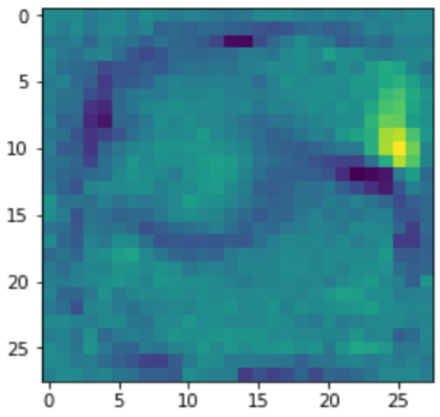
3



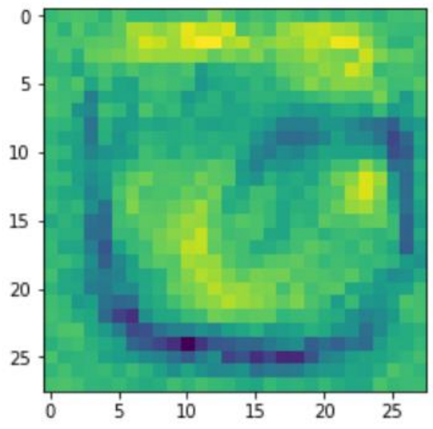
4



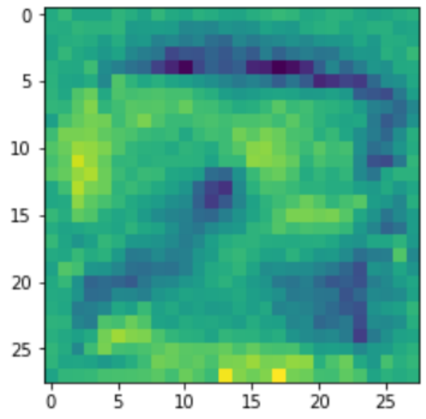
5



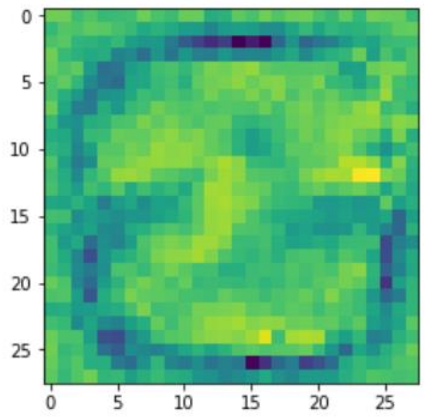
6



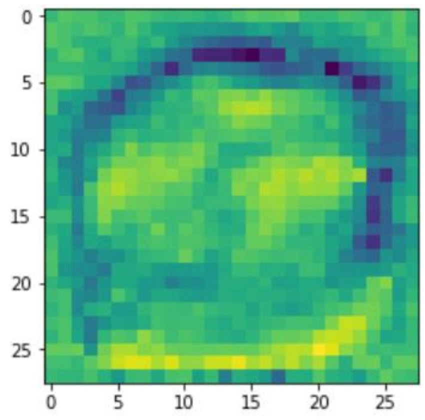
7



8



9


0
5
10
15
20
25