Assignment 6

Q 2.

Test Accuracy: 88.41

```
model.fit(train_images, train_labels, epochs=10)
```

```
Train on 60000 samples
Epoch 1/10
60000/60000 [==============================] - 10s 158us/sample - loss: 0.4800 - accuracy: 0.8249
Epoch 2/10
60000/60000 [==============================] - 9s 149us/sample - loss: 0.3636 - accuracy: 0.8656
Epoch 3/10
60000/60000 [==============================] - 9s 149us/sample - loss: 0.3289 - accuracy: 0.8770
Epoch 4/10
60000/60000 [==============================] - 9s 148us/sample - loss: 0.3038 - accuracy: 0.8865
Epoch 5/10
60000/60000 [==============================] - 9s 147us/sample - loss: 0.2842 - accuracy: 0.8951
Epoch 6/10
60000/60000 [==============================] - 9s 150us/sample - loss: 0.2721 - accuracy: 0.8986
Epoch 7/10
60000/60000 [==============================] - 9s 153us/sample - loss: 0.2602 - accuracy: 0.9015
Epoch 8/10
60000/60000 [==============================] - 9s 151us/sample - loss: 0.2478 - accuracy: 0.9073
Epoch 9/10
60000/60000 [==============================] - 9s 154us/sample - loss: 0.2381 - accuracy: 0.9097
Epoch 10/10
60000/60000 [==============================] - 10s 164us/sample - loss: 0.2288 - accuracy: 0.9124
<tensorflow.python.keras.callbacks.History at 0x7f34ff4906d8>
```

```
[67] test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

     print('\nTest accuracy:', test_acc*100)
```

```
10000/10000 - 1s - loss: 0.3503 - accuracy: 0.8842

Test accuracy: 88.41999769210815
```

Q 3.

Test accuracy: 91.28

```
[9] model = tf.keras.Sequential()
    # Must define the input shape in the first layer of the neural network
    model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu', input_shape=(28,28,1)))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu', input_shape=(28,28,1)))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(256, activation='relu'))
    model.add(tf.keras.layers.Dropout(0.5))
    model.add(tf.keras.layers.Dense(10, activation='softmax'))
    # Take a look at the model summary
    model.summary()
```

```
[11] model.fit(x_train,
               y_train,
               batch_size=64,
               epochs=10,
               validation_data=(x_valid, y_valid))
```

```
Train on 42000 samples, validate on 18000 samples
Epoch 1/10
42000/42000 [==============================] - 76s 2ms/sample - loss: 0.6003 - acc: 0.7813 - val_loss: 0.3926 - val_acc: 0.8568
Epoch 2/10
42000/42000 [==============================] - 75s 2ms/sample - loss: 0.4140 - acc: 0.8506 - val_loss: 0.3205 - val_acc: 0.8836
Epoch 3/10
42000/42000 [==============================] - 75s 2ms/sample - loss: 0.3659 - acc: 0.8680 - val_loss: 0.2978 - val_acc: 0.8912
Epoch 4/10
42000/42000 [==============================] - 74s 2ms/sample - loss: 0.3390 - acc: 0.8771 - val_loss: 0.2813 - val_acc: 0.8997
Epoch 5/10
42000/42000 [==============================] - 74s 2ms/sample - loss: 0.3188 - acc: 0.8843 - val_loss: 0.2687 - val_acc: 0.9002
Epoch 6/10
42000/42000 [==============================] - 74s 2ms/sample - loss: 0.2999 - acc: 0.8900 - val_loss: 0.2573 - val_acc: 0.9045
Epoch 7/10
42000/42000 [==============================] - 74s 2ms/sample - loss: 0.2885 - acc: 0.8938 - val_loss: 0.2428 - val_acc: 0.9086
Epoch 8/10
42000/42000 [==============================] - 74s 2ms/sample - loss: 0.2756 - acc: 0.8991 - val_loss: 0.2422 - val_acc: 0.9093
Epoch 9/10
42000/42000 [==============================] - 73s 2ms/sample - loss: 0.2618 - acc: 0.9017 - val_loss: 0.2335 - val_acc: 0.9126
Epoch 10/10
42000/42000 [==============================] - 74s 2ms/sample - loss: 0.2521 - acc: 0.9059 - val_loss: 0.2242 - val_acc: 0.9154
<tensorflow.python.keras.callbacks.History at 0x7f7e2ab42710>
```

```
[12] x_test = x_test.reshape((x_test.shape[0],28,28,1))
```

```
[13] # Evaluate the model on test set
     score = model.evaluate(x_test, y_test, verbose=0)
     # Print test accuracy
     print('\n', 'Test accuracy:', score[1])
```

```
 Test accuracy: 0.9128
```