Homework 10

1 DENOISING AUTOENCODERS

1.1 CODE:

1.py

I have used and modified code from: https://github.com/selimfirat/tf-denoising-autoencoder

1.2 ENCODER-DECODER LAYERS:

Homework asks to use at least three layers. I have used four layers with dimensions [28*28, 512, 256, 64] respectively.

1.3 TRAIN DATA

MNIST train image

1.4 TEST DATA

MNIST Test Images

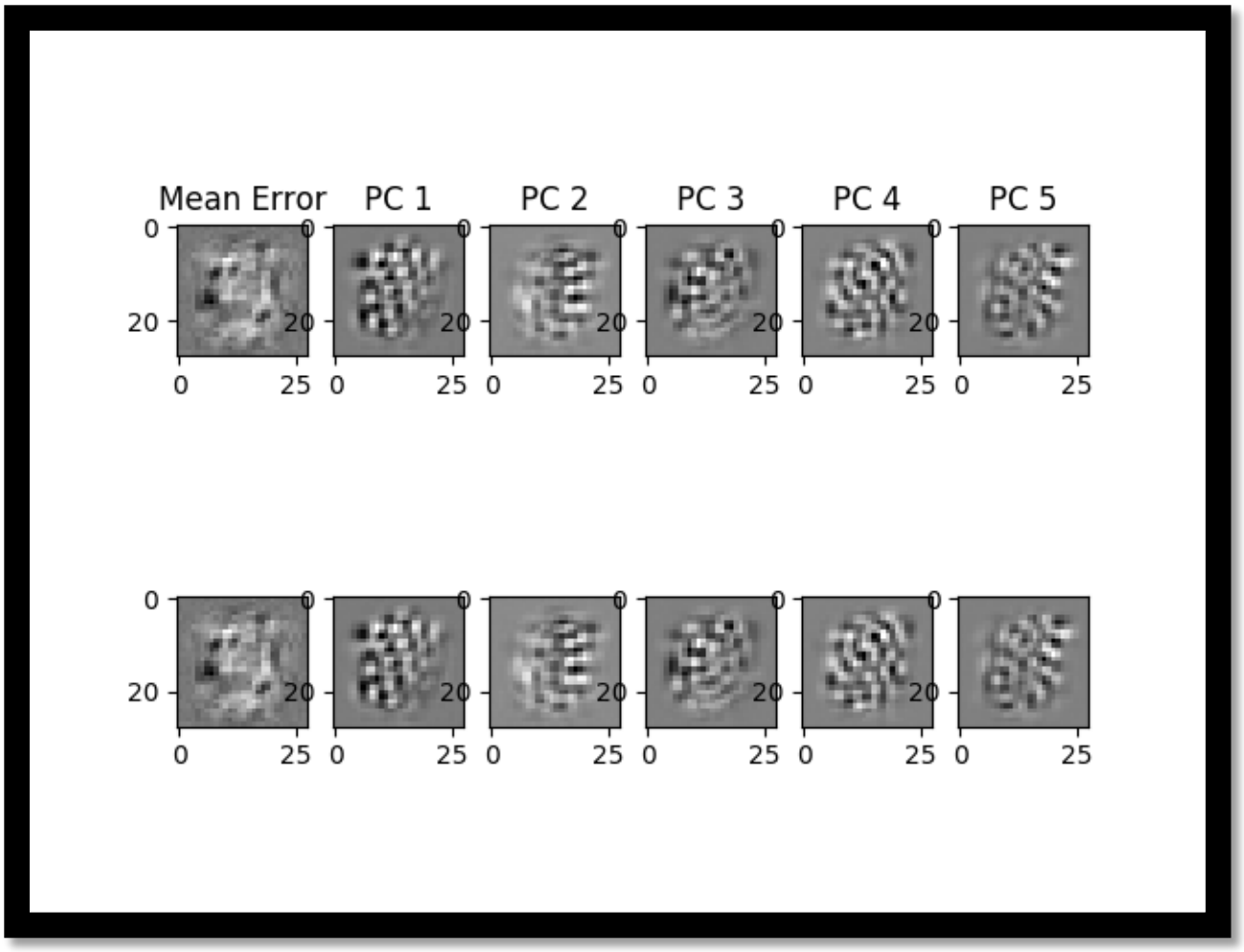
1.5 OUTPUT

1.5.1 First row:

mean and five principal components on the same gray scale for all six images, chosen so the largest absolute value over all six images is full dark or full light respectively.

1.5.2 Second Row

mean and five principal components on a scale where the gray scale is chosen for each image separately.



2 VARIATIONAL AUTOENCODER

2.1 CODE

2.py

I have used and modified code from: https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/3_NeuralNetworks/variational_autoencoder.py

2.2 TRAIN DATA

MNIST Train Images

2.3 TEST DATA

MNIST Test Images

2.4 ENCODING

```
def encode_interp(input_image1):
    encoder1 = tf.matmul(input_image1, weights['encoder_h1']) + biases['encoder_b1']
    encoder1 = tf.nn.tanh(encoder1)
    z_mean = tf.matmul(encoder1, weights['z_mean']) + biases['z_mean']
    z_std = tf.matmul(encoder1, weights['z_std']) + biases['z_std']
    # Sampler: Normal (gaussian) random distribution
    eps = tf.random_normal(tf.shape(z_std), dtype=tf.float32, mean=0., stddev=1.0,name='epsilon')
    z = z_mean + tf.exp(z_std / 2) * eps
    return z
```

2.5 CALCULATING INTERPOLATION OF ENCODED CODE

```
def interp(input_image1, input_image2, sess):
    z1 = encode_interp(input_image1)
    z2 = encode_interp(input_image2)
    #norm1 = L2norm(z1)
    #norm2 = L2norm(z2)
    #theta = tf.matmul(z1/norm1, z2/norm2, transpose_b=True)
    #a = tf.reshape(tf.linspace(0., 1., n), [n, 1])  # 10x1
    #a1 = tf.sin((1. - a) * theta) / tf.sin(theta)
    #a2 = tf.sin(a * theta) / tf.sin(theta)
    #z = a1 * z1 + a2 * z2
    zsub = tf.subtract(z2, z1)
    z = z1 + tf.scalar_mul(1/3, zsub)
    return sess.run( z)
```

2.6 DECODE

```
def interp(input_image1, input_image2, sess):
    z1 = encode_interp(input_image1)
    z2 = encode_interp(input_image2)
    #norm1 = L2norm(z1)
    #norm2 = L2norm(z2)
    #theta = tf.matmul(z1/norm1, z2/norm2, transpose_b=True)
    #a = tf.reshape(tf.linspace(0., 1., n), [n, 1])  # 10x1
    #a1 = tf.sin((1. - a) * theta) / tf.sin(theta)
    #a2 = tf.sin(a * theta) / tf.sin(theta)
    #z = a1 * z1 + a2 * z2
    zsub = tf.subtract(z2, z1)
    z = z1 + tf.scalar_mul(1/3, zsub)
    return sess.run( z)
```

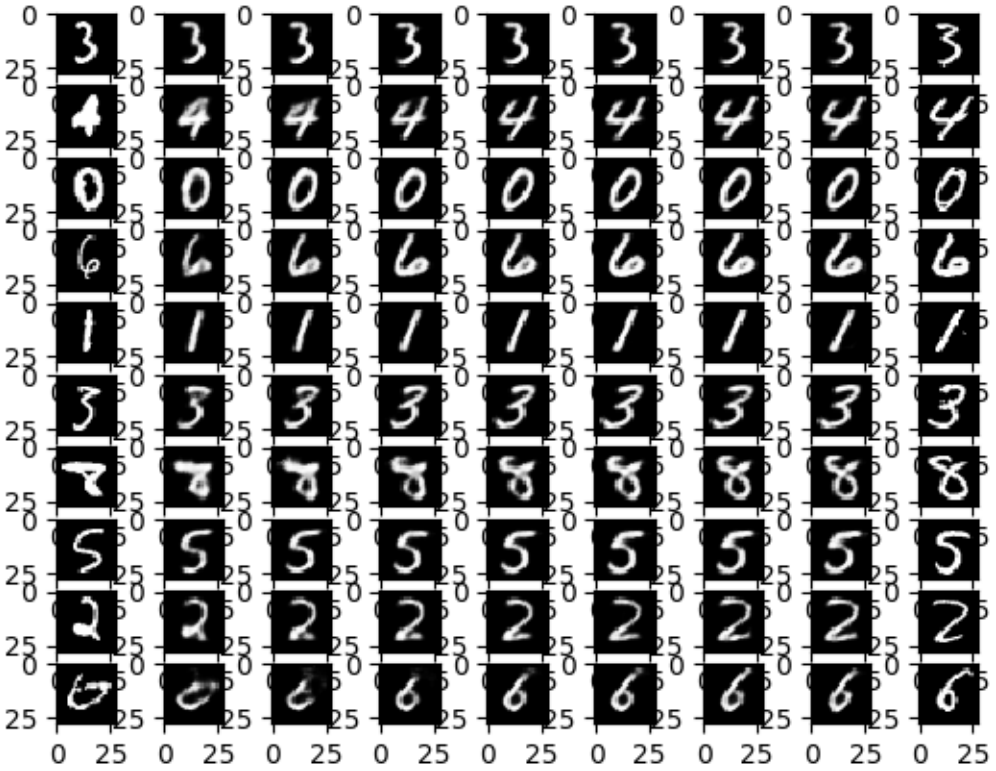
2.7 PAIR OF SAME DIGITS

For 10 random pairs of MNIST test images of the same digit, compute the code for each image of the pair and get 7 evenly spaced linear interpolates between these codes. Prepare a figure showing the interpolates and test images.

2.7.1 Random Selection

```
digitLabel = np.random.random_integers(0,9)
indexs = np.where(mnist.test.labels[:,digitLabel]==1)
indexs2 = np.random.random_integers(0, len(indexs[0])-1, 2)
count=count+1
plt.subplot(10, 9, count)
plt.imshow(mnist.test.images[indexs[0][indexs2[0]]].reshape(28, 28), cmap='gray',  interpolation='none')
#x_mean=np.array()
x_mean = [mnist.test.images[indexs[0][indexs2[0]]]]
```

2.7.2 Output



2.8 PAIR OF DIFFERENT DIGITS

For 10 random pairs of MNIST test images of the different digit, compute the code for each image of the pair and get 7 evenly spaced linear interpolates between these codes. Prepare a figure showing the interpolates and test images.

2.8.1 Random selection

```
image1Index = np.random.random_integers(0,mnist.test.labels.shape[0]-1)
image2Index = np.random.random_integers(0,mnist.test.labels.shape[0]-1)
count=count+1
plt.subplot(10, 9, count)
plt.imshow(mnist.test.images[image1Index].reshape(28, 28), cmap='gray',  interpolation='none')
#x_mean=np.array()
x_mean = [mnist.test.images[image1Index]]
```

2.8.2 Output