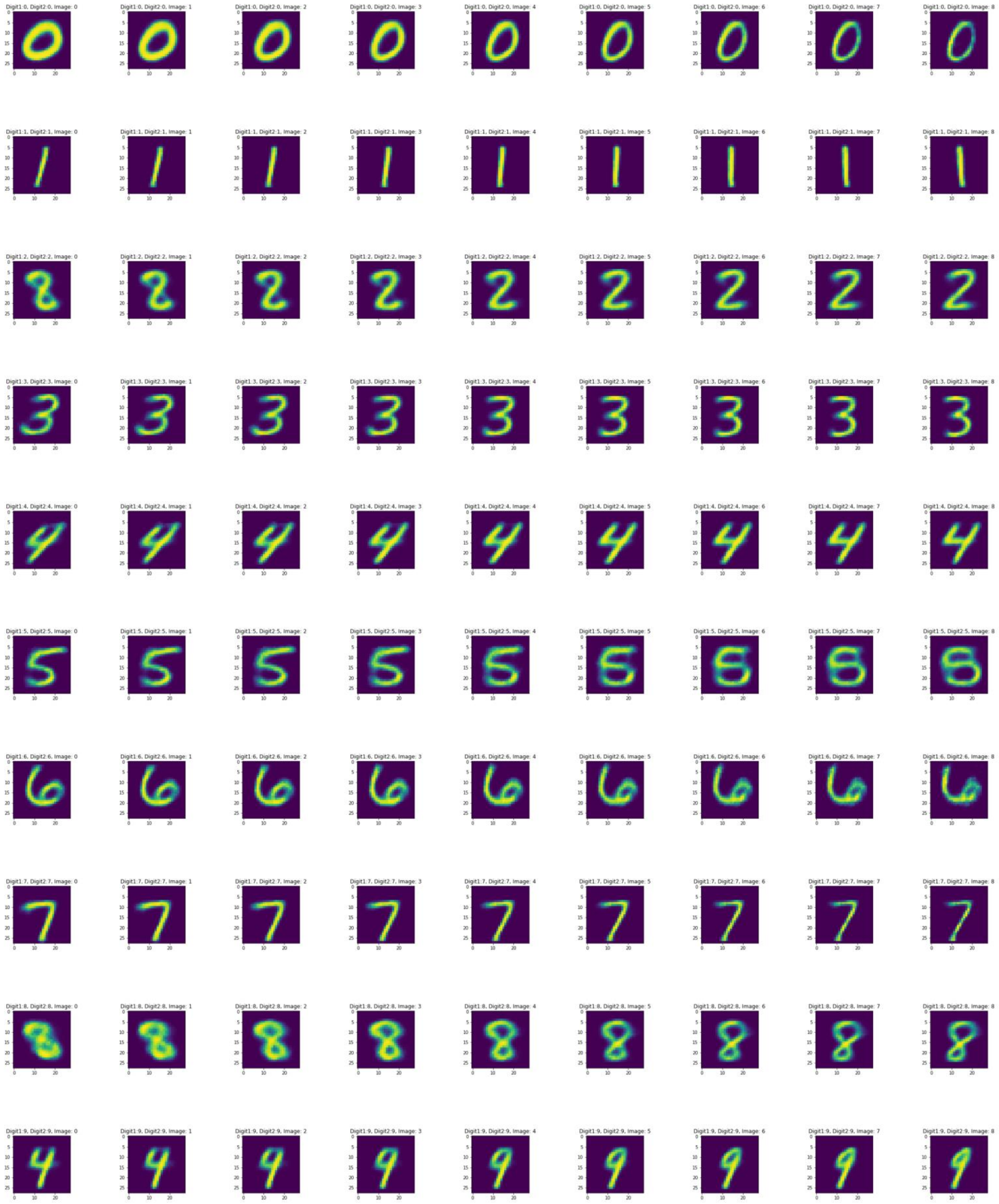
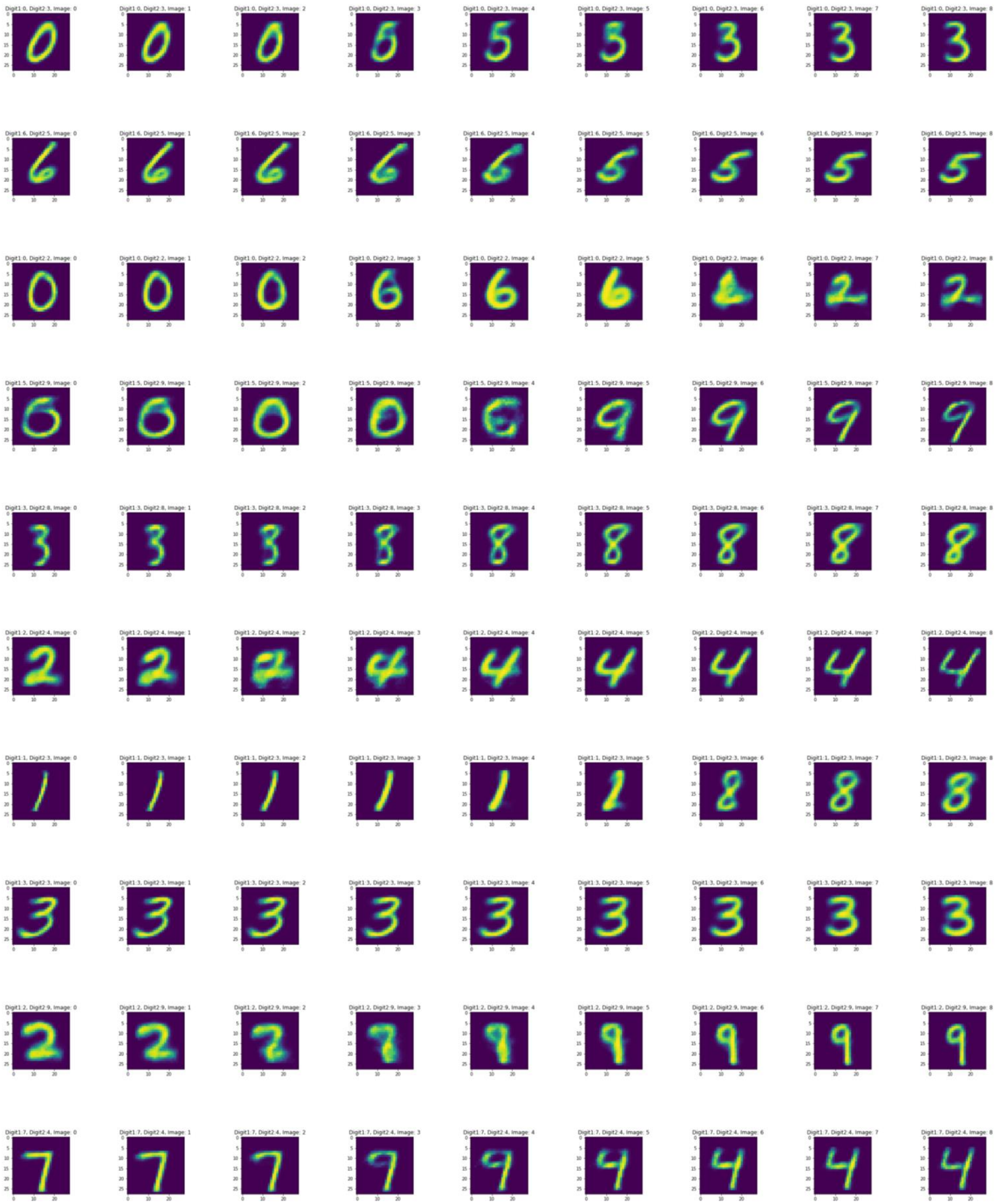
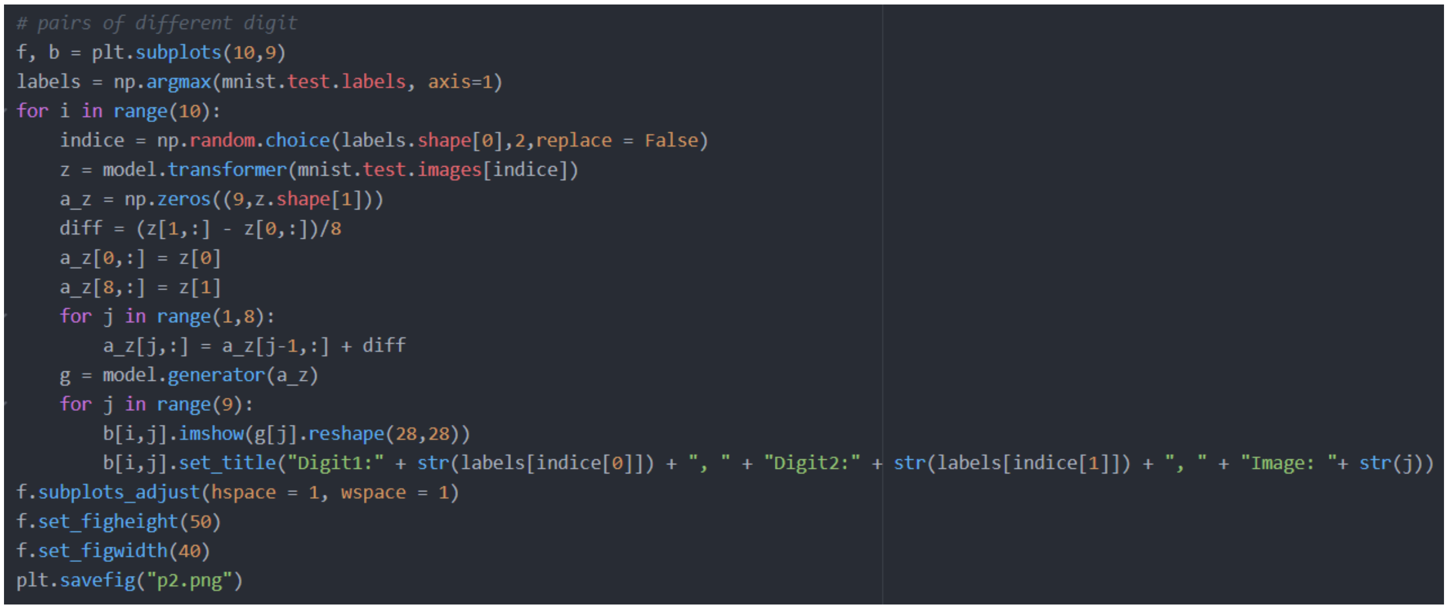

Digit1:0, Digit2:0, Image: 0
Digit1:0, Digit2:0, Image: 1
Digit1:0, Digit2:0, Image: 2
Digit1:0, Digit2:0, Image: 3
Digit1:0, Digit2:0, Image: 4
Digit1:0, Digit2:0, Image: 5
Digit1:0, Digit2:0, Image: 6
Digit1:0, Digit2:0, Image: 7
Digit1:0, Digit2:0, Image: 8
Digit1:1, Digit2:1, Image: 0
Digit1:1, Digit2:1, Image: 1
Digit1:1, Digit2:1, Image: 2
Digit1:1, Digit2:1, Image: 3
Digit1:1, Digit2:1, Image: 4
Digit1:1, Digit2:1, Image: 5
Digit1:1, Digit2:1, Image: 6
Digit1:1, Digit2:1, Image: 7
Digit1:1, Digit2:1, Image: 8
Digit1:2, Digit2:2, Image: 0
Digit1:2, Digit2:2, Image: 1
Digit1:2, Digit2:2, Image: 2
Digit1:2, Digit2:2, Image: 3
Digit1:2, Digit2:2, Image: 4
Digit1:2, Digit2:2, Image: 5
Digit1:2, Digit2:2, Image: 6
Digit1:2, Digit2:2, Image: 7
Digit1:2, Digit2:2, Image: 8
Digit1:3, Digit2:3, Image: 0
Digit1:3, Digit2:3, Image: 1
Digit1:3, Digit2:3, Image: 2
Digit1:3, Digit2:3, Image: 3
Digit1:3, Digit2:3, Image: 4
Digit1:3, Digit2:3, Image: 5
Digit1:3, Digit2:3, Image: 6
Digit1:3, Digit2:3, Image: 7
Digit1:3, Digit2:3, Image: 8
Digit1:4, Digit2:4, Image: 0
Digit1:4, Digit2:4, Image: 1
Digit1:4, Digit2:4, Image: 2
Digit1:4, Digit2:4, Image: 3
Digit1:4, Digit2:4, Image: 4
Digit1:4, Digit2:4, Image: 5
Digit1:4, Digit2:4, Image: 6
Digit1:4, Digit2:4, Image: 7
Digit1:4, Digit2:4, Image: 8
Digit1:5, Digit2:5, Image: 0
Digit1:5, Digit2:5, Image: 1
Digit1:5, Digit2:5, Image: 2
Digit1:5, Digit2:5, Image: 3
Digit1:5, Digit2:5, Image: 4
Digit1:5, Digit2:5, Image: 5
Digit1:5, Digit2:5, Image: 6
Digit1:5, Digit2:5, Image: 7
Digit1:5, Digit2:5, Image: 8
Digit1:6, Digit2:6, Image: 0
Digit1:6, Digit2:6, Image: 1
Digit1:6, Digit2:6, Image: 2
Digit1:6, Digit2:6, Image: 3
Digit1:6, Digit2:6, Image: 4
Digit1:6, Digit2:6, Image: 5
Digit1:6, Digit2:6, Image: 6
Digit1:6, Digit2:6, Image: 7
Digit1:6, Digit2:6, Image: 8
Digit1:7, Digit2:7, Image: 0
Digit1:7, Digit2:7, Image: 1
Digit1:7, Digit2:7, Image: 2
Digit1:7, Digit2:7, Image: 3
Digit1:7, Digit2:7, Image: 4
Digit1:7, Digit2:7, Image: 5
Digit1:7, Digit2:7, Image: 6
Digit1:7, Digit2:7, Image: 7
Digit1:7, Digit2:7, Image: 8
Digit1:8, Digit2:8, Image: 0
Digit1:8, Digit2:8, Image: 1
Digit1:8, Digit2:8, Image: 2
Digit1:8, Digit2:8, Image: 3
Digit1:8, Digit2:8, Image: 4
Digit1:8, Digit2:8, Image: 5
Digit1:8, Digit2:8, Image: 6
Digit1:8, Digit2:8, Image: 7
Digit1:8, Digit2:8, Image: 8
Digit1:9, Digit2:9, Image: 0
Digit1:9, Digit2:9, Image: 1
Digit1:9, Digit2:9, Image: 2
Digit1:9, Digit2:9, Image: 3
Digit1:9, Digit2:9, Image: 4
Digit1:9, Digit2:9, Image: 5
Digit1:9, Digit2:9, Image: 6
Digit1:9, Digit2:9, Image: 7
Digit1:9, Digit2:9, Image: 8

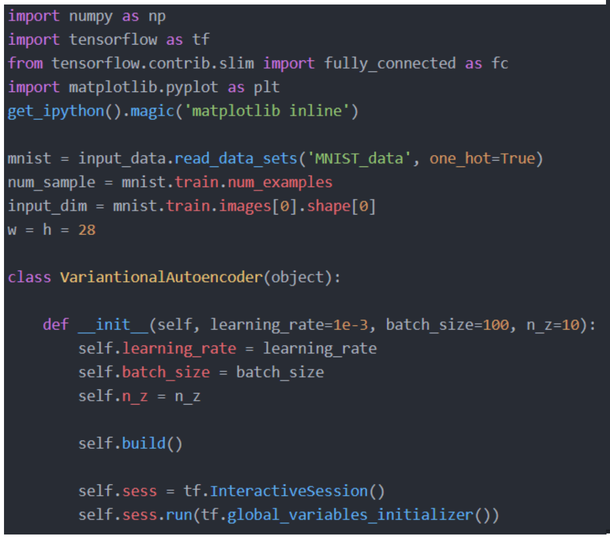

Digit1:0, Digit2:3, Image: 0
Digit1:0, Digit2:3, Image: 1
Digit1:0, Digit2:3, Image: 2
Digit1:0, Digit2:3, Image: 3
Digit1:0, Digit2:3, Image: 4
Digit1:0, Digit2:3, Image: 5
Digit1:0, Digit2:3, Image: 6
Digit1:0, Digit2:3, Image: 7
Digit1:0, Digit2:3, Image: 8
Digit1:6, Digit2:5, Image: 0
Digit1:6, Digit2:5, Image: 1
Digit1:6, Digit2:5, Image: 2
Digit1:6, Digit2:5, Image: 3
Digit1:6, Digit2:5, Image: 4
Digit1:6, Digit2:5, Image: 5
Digit1:6, Digit2:5, Image: 6
Digit1:6, Digit2:5, Image: 7
Digit1:6, Digit2:5, Image: 8
Digit1:0, Digit2:2, Image: 0
Digit1:0, Digit2:2, Image: 1
Digit1:0, Digit2:2, Image: 2
Digit1:0, Digit2:2, Image: 3
Digit1:0, Digit2:2, Image: 4
Digit1:0, Digit2:2, Image: 5
Digit1:0, Digit2:2, Image: 6
Digit1:0, Digit2:2, Image: 7
Digit1:0, Digit2:2, Image: 8
Digit1:5, Digit2:9, Image: 0
Digit1:5, Digit2:9, Image: 1
Digit1:5, Digit2:9, Image: 2
Digit1:5, Digit2:9, Image: 3
Digit1:5, Digit2:9, Image: 4
Digit1:5, Digit2:9, Image: 5
Digit1:5, Digit2:9, Image: 6
Digit1:5, Digit2:9, Image: 7
Digit1:5, Digit2:9, Image: 8
Digit1:3, Digit2:8, Image: 0
Digit1:3, Digit2:8, Image: 1
Digit1:3, Digit2:8, Image: 2
Digit1:3, Digit2:8, Image: 3
Digit1:3, Digit2:8, Image: 4
Digit1:3, Digit2:8, Image: 5
Digit1:3, Digit2:8, Image: 6
Digit1:3, Digit2:8, Image: 7
Digit1:3, Digit2:8, Image: 8
Digit1:2, Digit2:4, Image: 0
Digit1:2, Digit2:4, Image: 1
Digit1:2, Digit2:4, Image: 2
Digit1:2, Digit2:4, Image: 3
Digit1:2, Digit2:4, Image: 4
Digit1:2, Digit2:4, Image: 5
Digit1:2, Digit2:4, Image: 6
Digit1:2, Digit2:4, Image: 7
Digit1:2, Digit2:4, Image: 8
Digit1:1, Digit2:3, Image: 0
Digit1:1, Digit2:3, Image: 1
Digit1:1, Digit2:3, Image: 2
Digit1:1, Digit2:3, Image: 3
Digit1:1, Digit2:3, Image: 4
Digit1:1, Digit2:3, Image: 5
Digit1:1, Digit2:3, Image: 6
Digit1:1, Digit2:3, Image: 7
Digit1:1, Digit2:3, Image: 8
Digit1:3, Digit2:3, Image: 0
Digit1:3, Digit2:3, Image: 1
Digit1:3, Digit2:3, Image: 2
Digit1:3, Digit2:3, Image: 3
Digit1:3, Digit2:3, Image: 4
Digit1:3, Digit2:3, Image: 5
Digit1:3, Digit2:3, Image: 6
Digit1:3, Digit2:3, Image: 7
Digit1:3, Digit2:3, Image: 8
Digit1:2, Digit2:9, Image: 0
Digit1:2, Digit2:9, Image: 1
Digit1:2, Digit2:9, Image: 2
Digit1:2, Digit2:9, Image: 3
Digit1:2, Digit2:9, Image: 4
Digit1:2, Digit2:9, Image: 5
Digit1:2, Digit2:9, Image: 6
Digit1:2, Digit2:9, Image: 7
Digit1:2, Digit2:9, Image: 8
Digit1:7, Digit2:4, Image: 0
Digit1:7, Digit2:4, Image: 1
Digit1:7, Digit2:4, Image: 2
Digit1:7, Digit2:4, Image: 3
Digit1:7, Digit2:4, Image: 4
Digit1:7, Digit2:4, Image: 5
Digit1:7, Digit2:4, Image: 6
Digit1:7, Digit2:4, Image: 7
Digit1:7, Digit2:4, Image: 8

Code for interpolation and plotting:

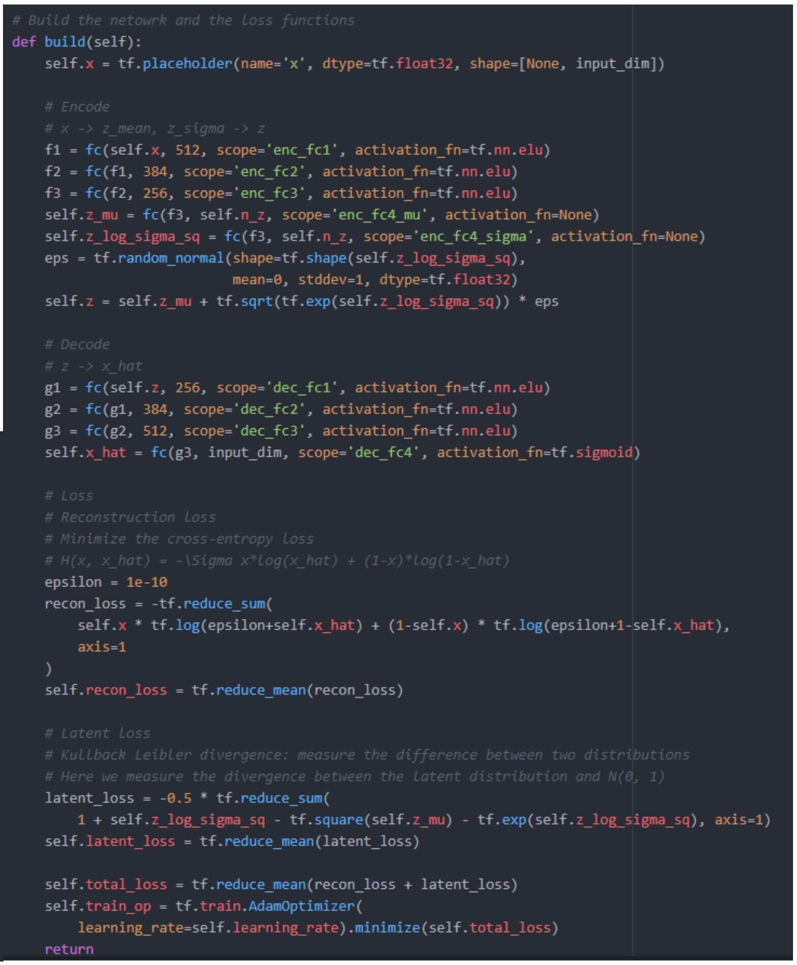


```python
# pairs of different digit
f, b = plt.subplots(10,9)
labels = np.argmax(mnist.test.labels, axis=1)
for i in range(10):
    indice = np.random.choice(labels.shape[0],2,replace = False)
    z = model.transformer(mnist.test.images[indice])
    a_z = np.zeros((9,z.shape[1]))
    diff = (z[1,:] - z[0,:])/8
    a_z[0,:] = z[0]
    a_z[8,:] = z[1]
    for j in range(1,8):
        a_z[j,:] = a_z[j-1,:] + diff
    g = model.generator(a_z)
    for j in range(9):
        b[i,j].imshow(g[j].reshape(28,28))
        b[i,j].set_title("Digit1:" + str(labels[indice[0]]) + ", " + "Digit2:" + str(labels[indice[1]]) + ", " + "Image: "+ str(j))
f.subplots_adjust(hspace = 1, wspace = 1)
f.set_figheight(50)
f.set_figwidth(40)
plt.savefig("p2.png")
```

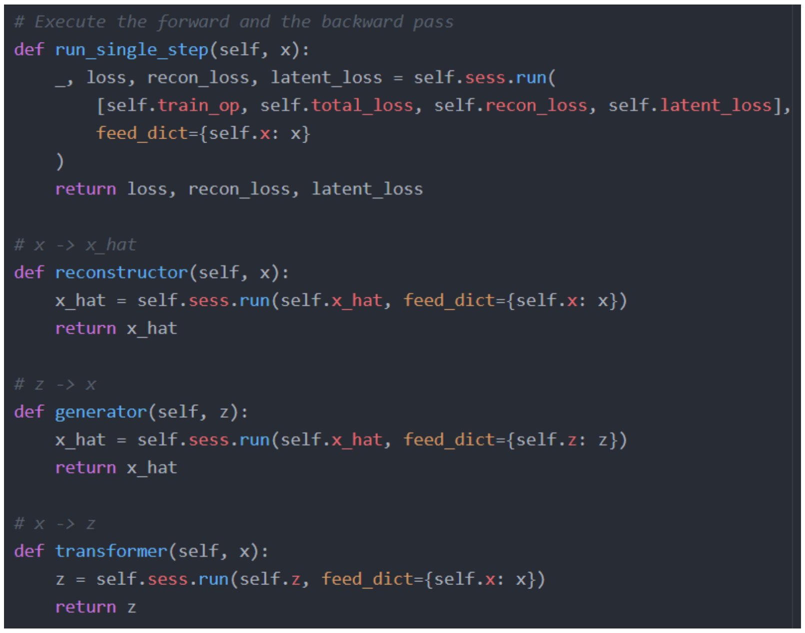
Entire code:



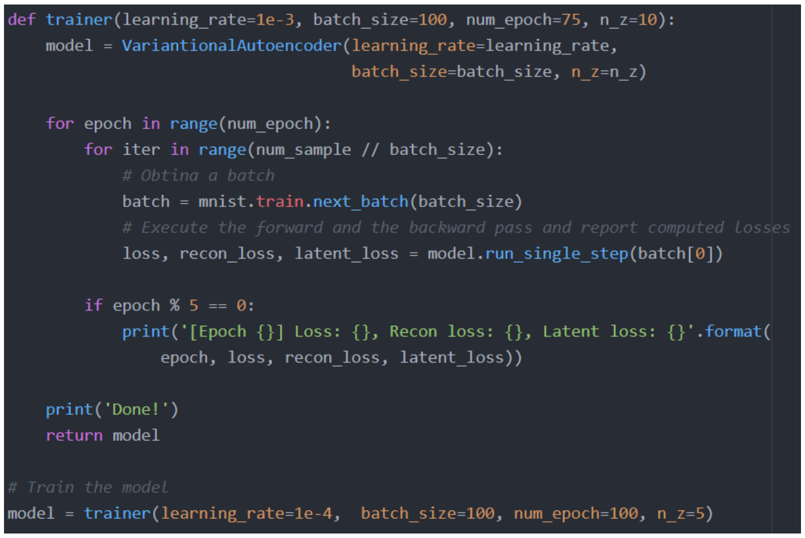
```python
import numpy as np
import tensorflow as tf
from tensorflow.contrib.slim import fully_connected as fc
import matplotlib.pyplot as plt
get_ipython().magic('matplotlib inline')

mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
num_sample = mnist.train.num_examples
input_dim = mnist.train.images[0].shape[0]
w = h = 28

class VariantionalAutoencoder(object):

    def __init__(self, learning_rate=1e-3, batch_size=100, n_z=10):
        self.learning_rate = learning_rate
        self.batch_size = batch_size
        self.n_z = n_z

        self.build()

        self.sess = tf.InteractiveSession()
        self.sess.run(tf.global_variables_initializer())
```



```python
# Build the netowrk and the loss functions
def build(self):
    self.x = tf.placeholder(name='x', dtype=tf.float32, shape=[None, input_dim])

    # Encode
    # x -> z_mean, z_sigma -> z
    f1 = fc(self.x, 512, scope='enc_fc1', activation_fn=tf.nn.elu)
    f2 = fc(f1, 384, scope='enc_fc2', activation_fn=tf.nn.elu)
    f3 = fc(f2, 256, scope='enc_fc3', activation_fn=tf.nn.elu)
    self.z_mu = fc(f3, self.n_z, scope='enc_fc4_mu', activation_fn=None)
    self.z_log_sigma_sq = fc(f3, self.n_z, scope='enc_fc4_sigma', activation_fn=None)
    eps = tf.random_normal(shape=tf.shape(self.z_log_sigma_sq),
                           mean=0, stddev=1, dtype=tf.float32)
    self.z = self.z_mu + tf.sqrt(tf.exp(self.z_log_sigma_sq)) * eps

    # Decode
    # z -> x_hat
    g1 = fc(self.z, 256, scope='dec_fc1', activation_fn=tf.nn.elu)
    g2 = fc(g1, 384, scope='dec_fc2', activation_fn=tf.nn.elu)
    g3 = fc(g2, 512, scope='dec_fc3', activation_fn=tf.nn.elu)
    self.x_hat = fc(g3, input_dim, scope='dec_fc4', activation_fn=tf.sigmoid)

    # Loss
    # Reconstruction loss
    # Minimize the cross-entropy loss
    # H(x, x_hat) = -\Sigma x*log(x_hat) + (1-x)*log(1-x_hat)
    epsilon = 1e-10
    recon_loss = -tf.reduce_sum(
        self.x * tf.log(epsilon+self.x_hat) + (1-self.x) * tf.log(epsilon+1-self.x_hat),
        axis=1
    )
    self.recon_loss = tf.reduce_mean(recon_loss)

    # Latent loss
    # Kullback Leibler divergence: measure the difference between two distributions
    # Here we measure the divergence between the latent distribution and N(0, 1)
    latent_loss = -0.5 * tf.reduce_sum(
        1 + self.z_log_sigma_sq - tf.square(self.z_mu) - tf.exp(self.z_log_sigma_sq), axis=1)
    self.latent_loss = tf.reduce_mean(latent_loss)

    self.total_loss = tf.reduce_mean(recon_loss + latent_loss)
    self.train_op = tf.train.AdamOptimizer(
        learning_rate=self.learning_rate).minimize(self.total_loss)
    return
```



```python
# Execute the forward and the backward pass
def run_single_step(self, x):
    _, loss, recon_loss, latent_loss = self.sess.run(
        [self.train_op, self.total_loss, self.recon_loss, self.latent_loss],
        feed_dict={self.x: x}
    )
    return loss, recon_loss, latent_loss

# x -> x_hat
def reconstructor(self, x):
    x_hat = self.sess.run(self.x_hat, feed_dict={self.x: x})
    return x_hat

# z -> x
def generator(self, z):
    x_hat = self.sess.run(self.x_hat, feed_dict={self.z: z})
    return x_hat

# x -> z
def transformer(self, x):
    z = self.sess.run(self.z, feed_dict={self.x: x})
    return z
```



```python
def trainer(learning_rate=1e-3, batch_size=100, num_epoch=75, n_z=10):
    model = VariantionalAutoencoder(learning_rate=learning_rate,
                                    batch_size=batch_size, n_z=n_z)

    for epoch in range(num_epoch):
        for iter in range(num_sample // batch_size):
            # Obtina a batch
            batch = mnist.train.next_batch(batch_size)
            # Execute the forward and the backward pass and report computed losses
            loss, recon_loss, latent_loss = model.run_single_step(batch[0])

        if epoch % 5 == 0:
            print('[Epoch {}] Loss: {}, Recon loss: {}, Latent loss: {}'.format(
                epoch, loss, recon_loss, latent_loss))

    print('Done!')
    return model

# Train the model
model = trainer(learning_rate=1e-4,  batch_size=100, num_epoch=100, n_z=5)
```