assignment12

August 11, 2021

[11]:
```
import keras
from keras import layers
import numpy as np
import tensorflow.compat.v1.keras.backend as K
import tensorflow as tf
tf.compat.v1.disable_eager_execution()
from keras.datasets import mnist
from tensorflow.keras.models import Model
```

[12]:
```
img_shape = (28, 28, 1)
batch_size = 16
latent_dim = 2
```

[13]:
```
input_img = keras.Input(shape=img_shape)

x = layers.Conv2D(32, 3, padding='same', activation='relu')(input_img)
x = layers.Conv2D(64, 3, padding='same', activation='relu', strides=(2, 2))(x)
x = layers.Conv2D(64, 3, padding='same', activation='relu')(x)
x = layers.Conv2D(64, 3, padding='same', activation='relu')(x)
shape_before_flattening = K.int_shape(x)
```

[14]:
```
x = layers.Flatten()(x)
x = layers.Dense(32, activation='relu')(x)

z_mean = layers.Dense(latent_dim)(x)
z_log_var = layers.Dense(latent_dim)(x)
```

[15]:
```
def sampling(args):
    z_mean, z_log_var = args
    epsilon = K.random_normal(shape=(K.shape(z_mean)[0], latent_dim),
                              mean=0., stddev=1.)
    return z_mean + K.exp(z_log_var) * epsilon

z = layers.Lambda(sampling)([z_mean, z_log_var])
```

[16]:
```
decoder_input = layers.Input(K.int_shape(z)[1:])
x = layers.Dense(np.prod(shape_before_flattening[1:]),
```

```
                activation='relu')(decoder_input)
x = layers.Reshape(shape_before_flattening[1:])(x)
x = layers.Conv2DTranspose(32, 3, padding='same', activation='relu',␣
 ↪strides=(2,2))(x)
x = layers.Conv2D(1, 3, padding='same', activation='sigmoid')(x)
decoder = Model(decoder_input, x)
z_decoded = decoder(z)
```

[17]:
```
class CustomVariationalLayer(keras.layers.Layer):

    def vae_loss(self, x, z_decoded):
        x = K.flatten(x)
        z_decoded = K.flatten(z_decoded)
        xent_loss = keras.metrics.binary_crossentropy(x, z_decoded)
        kl_loss = -5e-4 * K.mean(
            1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
        return K.mean(xent_loss + kl_loss)

    def call(self, inputs):
        x = inputs[0]
        z_decoded = inputs[1]
        loss = self.vae_loss(x, z_decoded)
        self.add_loss(loss, inputs=inputs)
        return x

y = CustomVariationalLayer() ([input_img, z_decoded])
```

[18]:
```
vae = Model(input_img, y)
vae.compile(optimizer='rmsprop', loss=None)
vae.summary()
```

```
WARNING:tensorflow:Output custom_variational_layer_1 missing from loss
dictionary. We assume this was done on purpose. The fit and evaluate APIs will
not be expecting any data to be passed to custom_variational_layer_1.
Model: "model_3"
_________________________________________________________________________________
__________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================
==================
input_3 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________________________
__________________
conv2d_5 (Conv2D)            (None, 28, 28, 32)        320         input_3[0][0]
_________________________________________________________________________________
__________________
conv2d_6 (Conv2D)            (None, 14, 14, 64)        18496       conv2d_5[0][0]
```

```
__________________________________________________________________________________
__________________
conv2d_7 (Conv2D)               (None, 14, 14, 64)   36928       conv2d_6[0][0]
__________________________________________________________________________________
__________________
conv2d_8 (Conv2D)               (None, 14, 14, 64)   36928       conv2d_7[0][0]
__________________________________________________________________________________
__________________
flatten_1 (Flatten)             (None, 12544)        0           conv2d_8[0][0]
__________________________________________________________________________________
__________________
dense_4 (Dense)                 (None, 32)           401440      flatten_1[0][0]
__________________________________________________________________________________
__________________
dense_5 (Dense)                 (None, 2)            66          dense_4[0][0]
__________________________________________________________________________________
__________________
dense_6 (Dense)                 (None, 2)            66          dense_4[0][0]
__________________________________________________________________________________
__________________
lambda_1 (Lambda)               (None, 2)            0           dense_5[0][0]
                                                                 dense_6[0][0]
__________________________________________________________________________________
__________________
model_2 (Functional)            (None, 28, 28, 1)    56385       lambda_1[0][0]
__________________________________________________________________________________
__________________
custom_variational_layer_1 (Cus (None, 28, 28, 1)    0           input_3[0][0]
                                                                 model_2[0][0]
==================================================================================
==================
Total params: 550,629
Trainable params: 550,629
Non-trainable params: 0
__________________________________________________________________________________
__________________
```

[19]:
```
(x_train, _), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255.
x_train = x_train.reshape(x_train.shape + (1,))
x_test = x_test.astype('float32') / 255.
x_test = x_test.reshape(x_test.shape + (1,))

vae.fit(x=x_train, y=None,
        shuffle=True,
        epochs=10,
```

```
        batch_size=batch_size,
        validation_data=(x_test, None))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 85s 1ms/sample - loss:
14400405.7198 - val_loss: 0.2003
Epoch 2/10
60000/60000 [==============================] - 83s 1ms/sample - loss: 0.1975 -
val_loss: 0.1955
Epoch 3/10
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.1929 -
val_loss: 0.1909
Epoch 4/10
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.1903 -
val_loss: 0.1888
Epoch 5/10
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.1886 -
val_loss: 0.1876
Epoch 6/10
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.1875 -
val_loss: 0.1884
Epoch 7/10
60000/60000 [==============================] - 83s 1ms/sample - loss: 0.1865 -
val_loss: 0.1875
Epoch 8/10
60000/60000 [==============================] - 83s 1ms/sample - loss: 0.1858 -
val_loss: 0.1867
Epoch 9/10
60000/60000 [==============================] - 83s 1ms/sample - loss: 0.1851 -
val_loss: 0.1854
Epoch 10/10
60000/60000 [==============================] - 84s 1ms/sample - loss: 0.1846 -
val_loss: 0.1844
```

[19]:
```
<tensorflow.python.keras.callbacks.History at 0x7f3cab8c5370>
```

[20]:
```
import matplotlib.pyplot as plt
from scipy.stats import norm
```

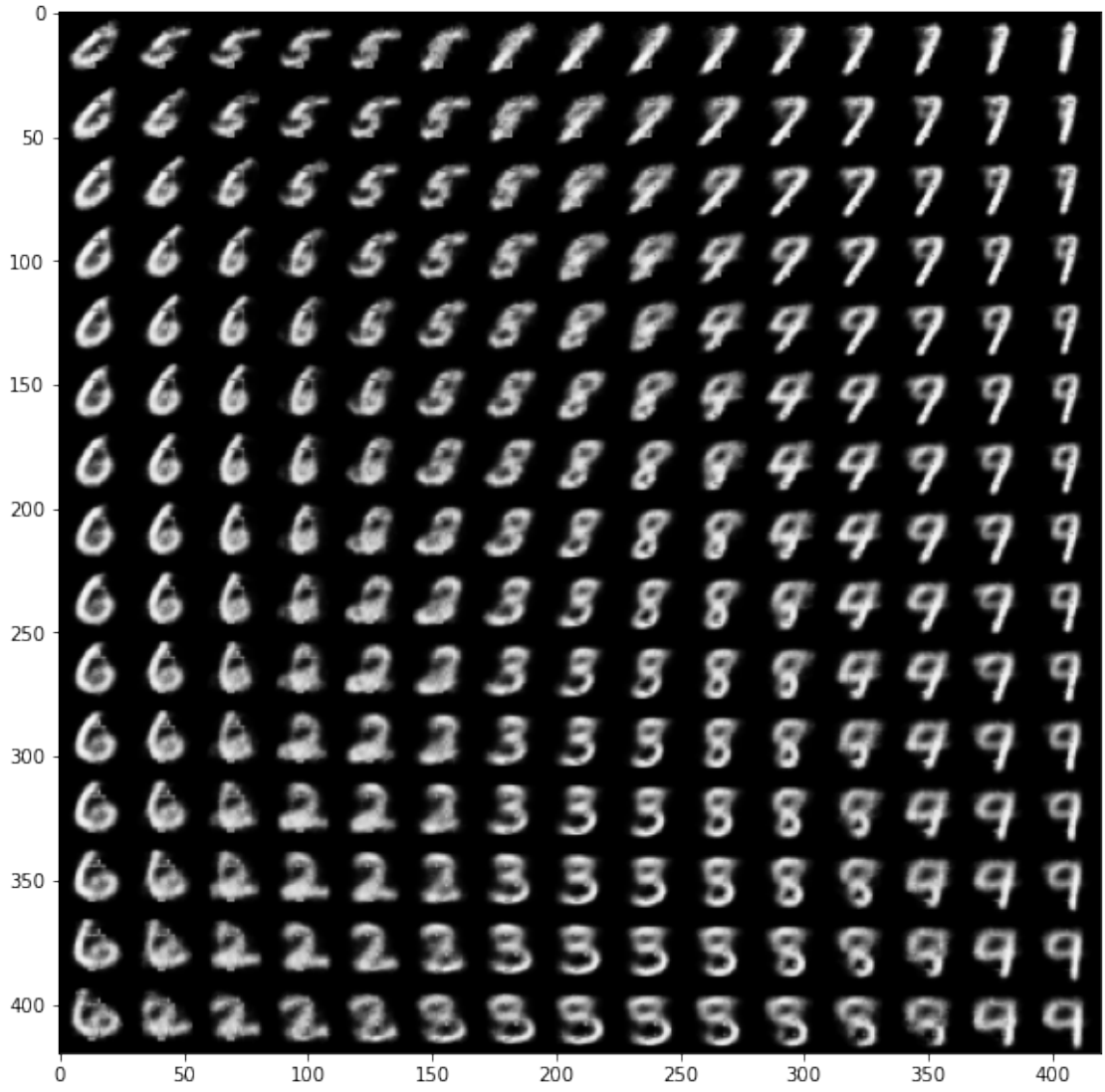
[21]:
```
n = 15
digit_size = 28
figure = np.zeros((digit_size * n, digit_size * n))
grid_x = norm.ppf(np.linspace(0.05, 0.95, n))
grid_y = norm.ppf(np.linspace(0.05, 0.95, n))

for i, yi in enumerate(grid_x):
```

```
    for j, xi in enumerate(grid_y):
        z_sample = np.array([[xi, yi]])
        z_sample = np.tile(z_sample, batch_size).reshape(batch_size, 2)
        x_decoded = decoder.predict(z_sample, batch_size=batch_size)
        digit = x_decoded[0].reshape(digit_size, digit_size)
        figure[i * digit_size: (i + 1) * digit_size,
               j * digit_size: (j + 1) * digit_size] = digit

plt.figure(figsize=(10,10))
plt.imshow(figure, cmap='Greys_r')
plt.show()
```

```
/opt/conda/lib/python3.8/site-
packages/tensorflow/python/keras/engine/training.py:2325: UserWarning:
`Model.state_updates` will be removed in a future version. This property should
not be used in TensorFlow 2.0, as `updates` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
```


0
50
100
150
200
250
300
350
400
0
50
100
150
200
250
300
350
400

[ ]: