Assignment 12

Brandon Sams

21Nov2020

Using section 8.4 in Deep Learning with Python as a guide, implement a variational autoencoder using the MNIST data set and save a grid of 15 x 15 digits to the results/vae directory. If you would rather work on a more interesting dataset, you can use the CelebFaces Attributes Dataset instead.

In [8]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

In [2]:
```
class Sampling(layers.Layer):
    """Uses (z_mean, z_log_var) to sample z, the vector encoding a dig
it."""

    def call(self, inputs):
        z_mean, z_log_var = inputs
        batch = tf.shape(z_mean)[0]
        dim = tf.shape(z_mean)[1]
        epsilon = tf.keras.backend.random_normal(shape=(batch, dim))
        return z_mean + tf.exp(0.5 * z_log_var) * epsilon
```

In [3]:
```
latent_dim = 2

encoder_inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(32, 3, activation="relu", strides=2, padding="same")
(encoder_inputs)
x = layers.Conv2D(64, 3, activation="relu", strides=2, padding="same")
(x)
x = layers.Flatten()(x)
x = layers.Dense(16, activation="relu")(x)
z_mean = layers.Dense(latent_dim, name="z_mean")(x)
z_log_var = layers.Dense(latent_dim, name="z_log_var")(x)
z = Sampling()([z_mean, z_log_var])
encoder = keras.Model(encoder_inputs, [z_mean, z_log_var, z], name="en
coder")
encoder.summary()
```

```
Model: "encoder"
____________________________________________________________________
______________________________
Layer (type)                    Output Shape         Param #     Con
nected to
====================================================================
==============================
input_1 (InputLayer)            [(None, 28, 28, 1)]  0

____________________________________________________________________
______________________________
conv2d (Conv2D)                 (None, 14, 14, 32)   320         inp
ut_1[0][0]
____________________________________________________________________
______________________________
conv2d_1 (Conv2D)               (None, 7, 7, 64)     18496       con
v2d[0][0]
____________________________________________________________________
______________________________
flatten (Flatten)               (None, 3136)         0           con
v2d_1[0][0]
____________________________________________________________________
______________________________
dense (Dense)                   (None, 16)           50192       fla
tten[0][0]
____________________________________________________________________
______________________________
z_mean (Dense)                  (None, 2)            34          den
se[0][0]
____________________________________________________________________
______________________________
z_log_var (Dense)               (None, 2)            34          den
se[0][0]
____________________________________________________________________
______________________________
sampling (Sampling)             (None, 2)            0           z_m
ean[0][0]
                                                                 z_l
og_var[0][0]
====================================================================
==============================
Total params: 69,076
Trainable params: 69,076
Non-trainable params: 0
____________________________________________________________________
______________________________
```

In [4]:
```
latent_inputs = keras.Input(shape=(latent_dim,))
x = layers.Dense(7 * 7 * 64, activation="relu")(latent_inputs)
x = layers.Reshape((7, 7, 64))(x)
x = layers.Conv2DTranspose(64, 3, activation="relu", strides=2, paddin
g="same")(x)
x = layers.Conv2DTranspose(32, 3, activation="relu", strides=2, paddin
g="same")(x)
decoder_outputs = layers.Conv2DTranspose(1, 3, activation="sigmoid", p
adding="same")(x)
decoder = keras.Model(latent_inputs, decoder_outputs, name="decoder")
decoder.summary()
```

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense_1 (Dense)              (None, 3136)              9408
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 14, 14, 64)        36928
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 28, 28, 32)        18464
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 1)         289
=================================================================
Total params: 65,089
Trainable params: 65,089
Non-trainable params: 0
_________________________________________________________________
```

In [5]:
```
class VAE(keras.Model):
    def __init__(self, encoder, decoder, **kwargs):
        super(VAE, self).__init__(**kwargs)
        self.encoder = encoder
        self.decoder = decoder

    def train_step(self, data):
        if isinstance(data, tuple):
            data = data[0]
        with tf.GradientTape() as tape:
            z_mean, z_log_var, z = encoder(data)
            reconstruction = decoder(z)
            reconstruction_loss = tf.reduce_mean(
                keras.losses.binary_crossentropy(data, reconstruction)
            )
            reconstruction_loss *= 28 * 28
            kl_loss = 1 + z_log_var - tf.square(z_mean) - tf.exp(z_log
_var)
            kl_loss = tf.reduce_mean(kl_loss)
            kl_loss *= -0.5
            total_loss = reconstruction_loss + kl_loss
        grads = tape.gradient(total_loss, self.trainable_weights)
        self.optimizer.apply_gradients(zip(grads, self.trainable_weigh
ts))
        return {
            "loss": total_loss,
            "reconstruction_loss": reconstruction_loss,
            "kl_loss": kl_loss,
        }
```

In [6]:
```
(x_train, _), (x_test, _) = keras.datasets.mnist.load_data()
mnist_digits = np.concatenate([x_train, x_test], axis=0)
mnist_digits = np.expand_dims(mnist_digits, -1).astype("float32") / 25
5

vae = VAE(encoder, decoder)
vae.compile(optimizer=keras.optimizers.Adam())
vae.fit(mnist_digits, epochs=30, batch_size=128)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-k
eras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
Epoch 1/30
547/547 [==============================] - 4s 6ms/step - loss: 202.3
682 - reconstruction_loss: 199.4540 - kl_loss: 2.9142
Epoch 2/30
547/547 [==============================] - 3s 6ms/step - loss: 162.5
128 - reconstruction_loss: 159.4889 - kl_loss: 3.0239
Epoch 3/30
```

```
547/547 [==============================] - 3s 6ms/step - loss: 157.5
287 - reconstruction_loss: 154.3713 - kl_loss: 3.1575
Epoch 4/30
547/547 [==============================] - 3s 6ms/step - loss: 155.2
329 - reconstruction_loss: 151.9954 - kl_loss: 3.2376
Epoch 5/30
547/547 [==============================] - 3s 6ms/step - loss: 153.7
180 - reconstruction_loss: 150.4177 - kl_loss: 3.3003
Epoch 6/30
547/547 [==============================] - 3s 6ms/step - loss: 152.6
100 - reconstruction_loss: 149.2735 - kl_loss: 3.3365
Epoch 7/30
547/547 [==============================] - 3s 6ms/step - loss: 151.6
836 - reconstruction_loss: 148.3357 - kl_loss: 3.3480
Epoch 8/30
547/547 [==============================] - 3s 6ms/step - loss: 151.0
421 - reconstruction_loss: 147.6427 - kl_loss: 3.3994
Epoch 9/30
547/547 [==============================] - 3s 6ms/step - loss: 150.3
833 - reconstruction_loss: 146.9797 - kl_loss: 3.4037
Epoch 10/30
547/547 [==============================] - 3s 6ms/step - loss: 149.8
730 - reconstruction_loss: 146.4460 - kl_loss: 3.4270
Epoch 11/30
547/547 [==============================] - 3s 6ms/step - loss: 149.4
690 - reconstruction_loss: 146.0347 - kl_loss: 3.4342
Epoch 12/30
547/547 [==============================] - 3s 6ms/step - loss: 148.9
961 - reconstruction_loss: 145.5439 - kl_loss: 3.4522
Epoch 13/30
547/547 [==============================] - 3s 6ms/step - loss: 148.6
019 - reconstruction_loss: 145.1272 - kl_loss: 3.4746
Epoch 14/30
547/547 [==============================] - 3s 6ms/step - loss: 148.3
547 - reconstruction_loss: 144.8601 - kl_loss: 3.4946
Epoch 15/30
547/547 [==============================] - 3s 6ms/step - loss: 148.1
077 - reconstruction_loss: 144.6002 - kl_loss: 3.5075
Epoch 16/30
547/547 [==============================] - 3s 6ms/step - loss: 147.7
308 - reconstruction_loss: 144.2117 - kl_loss: 3.5191
Epoch 17/30
547/547 [==============================] - 3s 6ms/step - loss: 147.5
377 - reconstruction_loss: 144.0077 - kl_loss: 3.5300
Epoch 18/30
547/547 [==============================] - 3s 6ms/step - loss: 147.3
641 - reconstruction_loss: 143.8094 - kl_loss: 3.5546
Epoch 19/30
547/547 [==============================] - 3s 6ms/step - loss: 147.0
738 - reconstruction_loss: 143.5065 - kl_loss: 3.5673
```

```
Epoch 20/30
547/547 [==============================] - 3s 6ms/step - loss: 146.8
453 - reconstruction_loss: 143.2773 - kl_loss: 3.5680
Epoch 21/30
547/547 [==============================] - 3s 6ms/step - loss: 146.7
664 - reconstruction_loss: 143.2013 - kl_loss: 3.5651
Epoch 22/30
547/547 [==============================] - 3s 6ms/step - loss: 146.5
439 - reconstruction_loss: 142.9481 - kl_loss: 3.5958
Epoch 23/30
547/547 [==============================] - 4s 6ms/step - loss: 146.3
700 - reconstruction_loss: 142.7789 - kl_loss: 3.5911
Epoch 24/30
547/547 [==============================] - 4s 7ms/step - loss: 146.2
900 - reconstruction_loss: 142.6831 - kl_loss: 3.6070
Epoch 25/30
547/547 [==============================] - 4s 7ms/step - loss: 145.9
991 - reconstruction_loss: 142.3880 - kl_loss: 3.6111
Epoch 26/30
547/547 [==============================] - 4s 6ms/step - loss: 145.9
081 - reconstruction_loss: 142.2842 - kl_loss: 3.6240
Epoch 27/30
547/547 [==============================] - 3s 6ms/step - loss: 145.8
139 - reconstruction_loss: 142.1850 - kl_loss: 3.6290
Epoch 28/30
547/547 [==============================] - 3s 6ms/step - loss: 145.6
528 - reconstruction_loss: 142.0203 - kl_loss: 3.6326
Epoch 29/30
547/547 [==============================] - 3s 6ms/step - loss: 145.5
576 - reconstruction_loss: 141.9096 - kl_loss: 3.6480
Epoch 30/30
547/547 [==============================] - 3s 6ms/step - loss: 145.4
119 - reconstruction_loss: 141.7675 - kl_loss: 3.6444

Out[6]: <tensorflow.python.keras.callbacks.History at 0x7f945e2b3780>
```

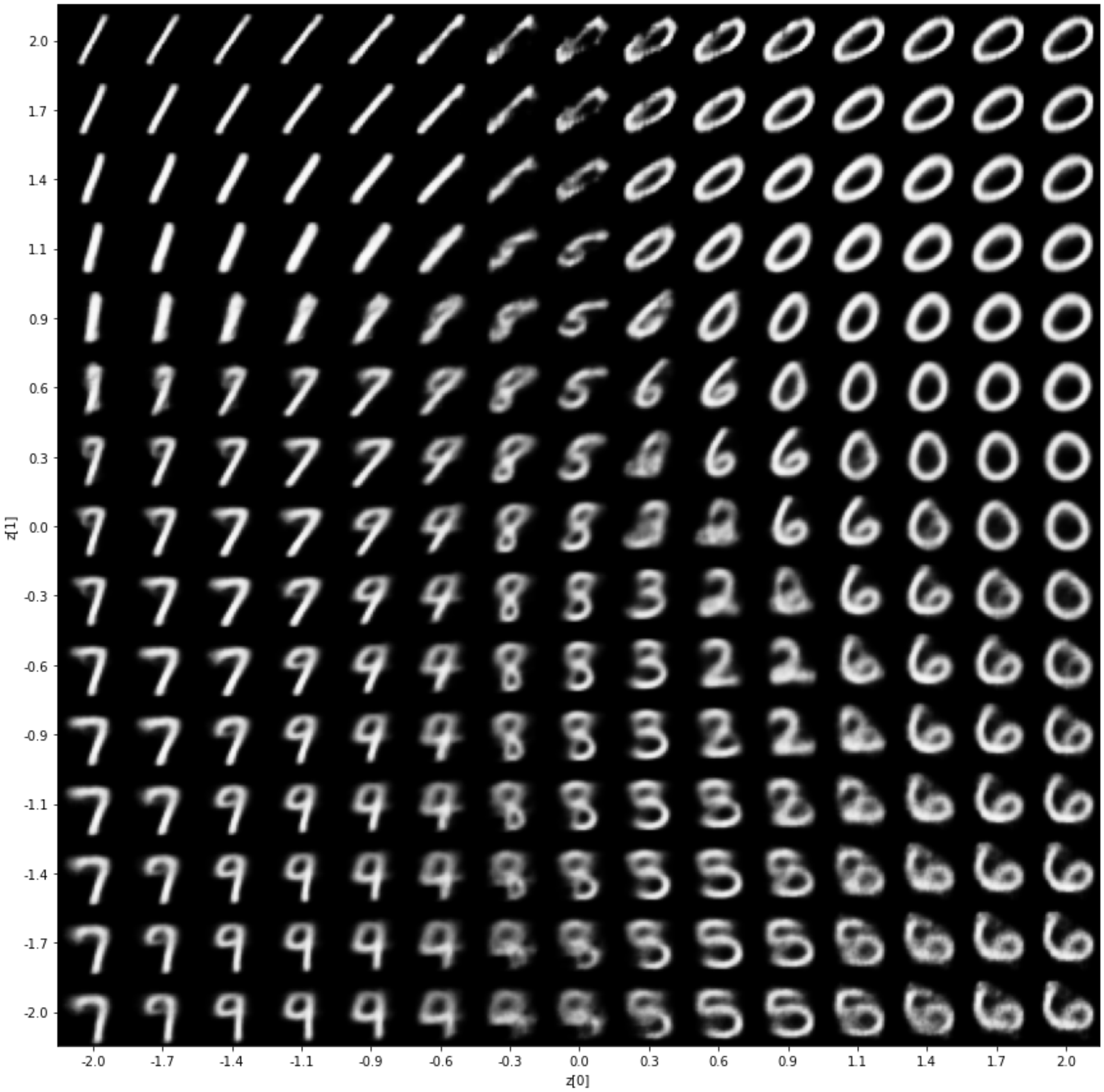
In [7]:
```python
import matplotlib.pyplot as plt


def plot_latent(encoder, decoder):
    # display a n*n 2D manifold of digits
    n = 15
    digit_size = 28
    scale = 2.0
    figsize = 15
    figure = np.zeros((digit_size * n, digit_size * n))
    # linearly spaced coordinates corresponding to the 2D plot
    # of digit classes in the latent space
    grid_x = np.linspace(-scale, scale, n)
    grid_y = np.linspace(-scale, scale, n)[::-1]

    for i, yi in enumerate(grid_y):
        for j, xi in enumerate(grid_x):
            z_sample = np.array([[xi, yi]])
            x_decoded = decoder.predict(z_sample)
            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=(figsize, figsize))
    start_range = digit_size // 2
    end_range = n * digit_size + start_range + 1
    pixel_range = np.arange(start_range, end_range, digit_size)
    sample_range_x = np.round(grid_x, 1)
    sample_range_y = np.round(grid_y, 1)
    plt.xticks(pixel_range, sample_range_x)
    plt.yticks(pixel_range, sample_range_y)
    plt.xlabel("z[0]")
    plt.ylabel("z[1]")
    plt.imshow(figure, cmap="Greys_r")
    plt.savefig("mnist_vae.png")
    plt.show()


plot_latent(encoder, decoder)
```


2.0
1.7
1.4
1.1
0.9
0.6
0.3
0.0
-0.3
-0.6
-0.9
-1.1
-1.4
-1.7
-2.0
z[1]
-2.0 -1.7 -1.4 -1.1 -0.9 -0.6 -0.3 0.0 0.3 0.6 0.9 1.1 1.4 1.7 2.0
z[0]

In [7]: