03_variational_autoencoder

September 29, 2021

1 Variational Autoencoder on Fashion MNIST data using Feed-forward NN

Adapted from Building Autoencoders in Keras by Francois Chollet who created Keras.

1.1 Imports & Settings

[1]:
```
%matplotlib inline

from pathlib import Path
import os
import numpy as np

import tensorflow as tf
from tensorflow.keras.layers import Lambda, Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.losses import mse
from tensorflow.keras.utils import plot_model
from tensorflow.keras import backend as K

import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
gpu_devices = tf.config.experimental.list_physical_devices('GPU')
if gpu_devices:
    print('Using GPU')
    tf.config.experimental.set_memory_growth(gpu_devices[0], True)
else:
    print('Using CPU')
```

```
Using CPU
```

[3]:
```
sns.set_style('white')
```

[4]:
```
results_path = Path('results', 'fashion_mnist')
if not results_path.exists():
```

```
    results_path.mkdir(parents=True)
```

1.2 Sampling

[5]:
```
# instead of sampling from Q(z|X), sample eps = N(0,I)
# z = z_mean + sqrt(var)*eps
def sampling(args):
    """Reparameterization trick by sampling fr an isotropic unit Gaussian.

    # Arguments
        args (tensor): mean and log of variance of Q(z|X)

    # Returns
        z (tensor): sampled latent vector
    """

    z_mean, z_log_var = args
    batch = K.shape(z_mean)[0]
    dim = K.int_shape(z_mean)[1]
    # by default, random_normal has mean=0 and std=1.0
    epsilon = K.random_normal(shape=(batch, dim))
    return z_mean + K.exp(0.5 * z_log_var) * epsilon
```

1.3 Load Fashion MNIST Data

[6]:
```
# MNIST dataset
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

image_size = x_train.shape[1]
original_dim = image_size * image_size
x_train = np.reshape(x_train, [-1, original_dim])
x_test = np.reshape(x_test, [-1, original_dim])
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
```

1.4 Define Variational Autoencoder Architecture

1.4.1 Network Parameters

[7]:
```
input_shape = (original_dim,)
intermediate_dim = 512
batch_size = 128
latent_dim = 2
epochs = 50
```

1.4.2 Encoder model

Define Layers

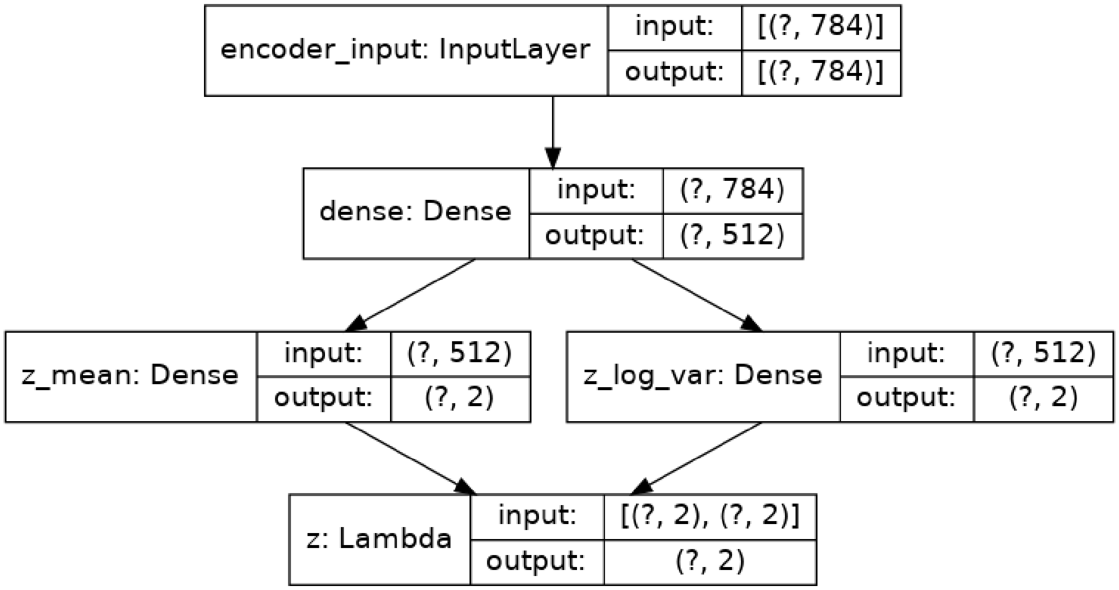
[8]:
```
inputs = Input(shape=input_shape, name='encoder_input')
x = Dense(intermediate_dim, activation='relu')(inputs)
z_mean = Dense(latent_dim, name='z_mean')(x)
z_log_var = Dense(latent_dim, name='z_log_var')(x)

# use reparameterization trick to push the sampling out as input
# note that "output_shape" isn't necessary with the TensorFlow backend
z = Lambda(sampling, output_shape=(latent_dim,), name='z')([z_mean, z_log_var])
```

Instantiate Model

[9]:
```
encoder = Model(inputs, [z_mean, z_log_var, z], name='encoder')
encoder.summary()
plot_model(encoder, to_file=(results_path / 'vae_mlp_encoder.png').as_posix(),␣
 ↪show_shapes=True)
```

```
Model: "encoder"

--------------------------------------------------------------------------------
------------------
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
encoder_input (InputLayer)      [(None, 784)]        0

--------------------------------------------------------------------------------
------------------
dense (Dense)                   (None, 512)          401920
encoder_input[0][0]

--------------------------------------------------------------------------------
------------------
z_mean (Dense)                  (None, 2)            1026        dense[0][0]

--------------------------------------------------------------------------------
------------------
z_log_var (Dense)               (None, 2)            1026        dense[0][0]

--------------------------------------------------------------------------------
------------------
z (Lambda)                      (None, 2)            0           z_mean[0][0]
                                                                 z_log_var[0][0]
================================================================================
==================
Total params: 403,972
Trainable params: 403,972
Non-trainable params: 0

--------------------------------------------------------------------------------
------------------
```

[9]:



1.4.3 Decoder Model

Define Layers

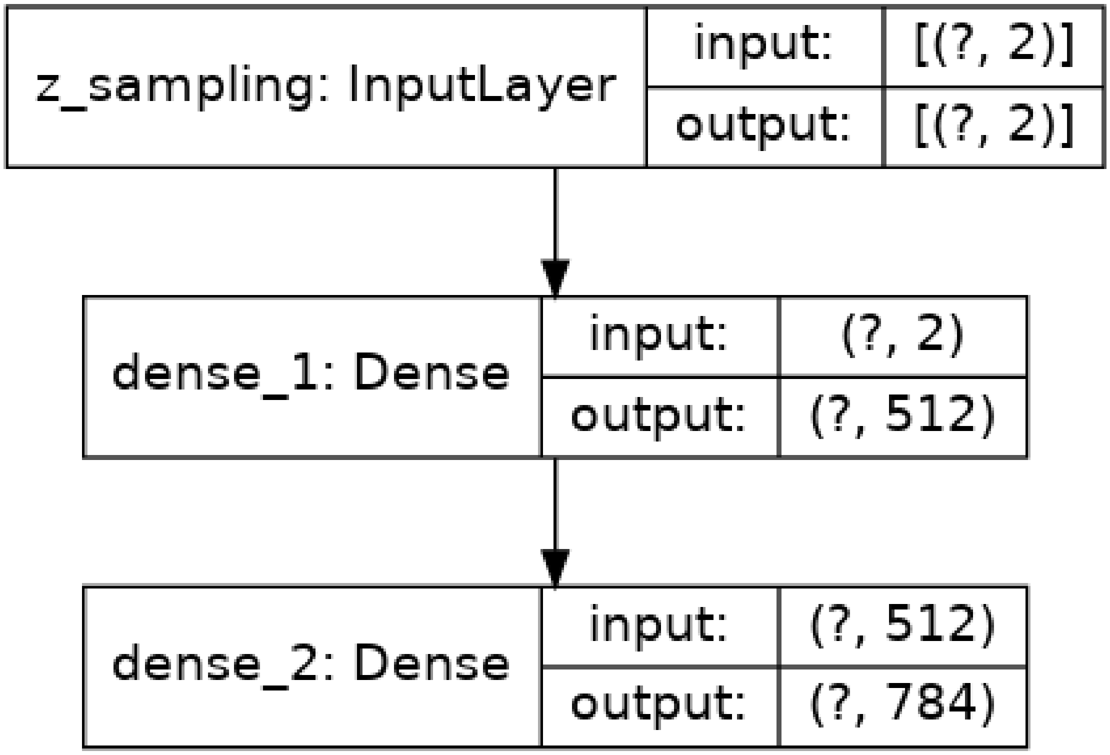
[10]:
```
latent_inputs = Input(shape=(latent_dim,), name='z_sampling')
x = Dense(intermediate_dim, activation='relu')(latent_inputs)
outputs = Dense(original_dim, activation='sigmoid')(x)
```

Instantiate model

[11]:
```
decoder = Model(latent_inputs, outputs, name='decoder')
decoder.summary()
plot_model(decoder, to_file=(results_path / 'vae_mlp_decoder.png').as_posix(),␣
 ↪show_shapes=True)
```

```
Model: "decoder"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
z_sampling (InputLayer)      [(None, 2)]               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1536
_________________________________________________________________
dense_2 (Dense)              (None, 784)               402192
=================================================================
Total params: 403,728
Trainable params: 403,728
Non-trainable params: 0
_________________________________________________________________
```

[11]:



1.4.4 Combine Encoder and Decoder to VAE model

[12]:
```
outputs = decoder(encoder(inputs)[2])
vae = Model(inputs, outputs, name='vae_mlp')
```

[13]:
```
models = (encoder, decoder)
```

1.5 Train Model

[14]:
```
data = (x_test, y_test)

reconstruction_loss = mse(inputs, outputs)
reconstruction_loss *= original_dim

kl_loss = 1 + z_log_var - K.square(z_mean) - K.exp(z_log_var)
kl_loss = K.sum(kl_loss, axis=-1)
kl_loss *= -0.5
vae_loss = K.mean(reconstruction_loss + kl_loss)
vae.add_loss(vae_loss)
vae.compile(optimizer='adam')
vae.summary()
```

```
Model: "vae_mlp"
__________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
==================
encoder_input (InputLayer)      [(None, 784)]        0
__________________________________________________________________________________
__________________
encoder (Model)                 [(None, 2), (None, 2 403972
encoder_input[0][0]
__________________________________________________________________________________
__________________
decoder (Model)                 (None, 784)          403728      encoder[1][2]
__________________________________________________________________________________
__________________
dense (Dense)                   (None, 512)          401920
encoder_input[0][0]
__________________________________________________________________________________
__________________
z_log_var (Dense)               (None, 2)            1026        dense[0][0]
__________________________________________________________________________________
__________________
z_mean (Dense)                  (None, 2)            1026        dense[0][0]
__________________________________________________________________________________
__________________
tf_op_layer_AddV2 (TensorFlowOp [(None, 2)]          0           z_log_var[0][0]
__________________________________________________________________________________
__________________
tf_op_layer_Square (TensorFlowO [(None, 2)]          0           z_mean[0][0]
__________________________________________________________________________________
__________________
tf_op_layer_Sub (TensorFlowOpLa [(None, 2)]          0
tf_op_layer_AddV2[0][0]
tf_op_layer_Square[0][0]
__________________________________________________________________________________
__________________
tf_op_layer_Exp (TensorFlowOpLa [(None, 2)]          0           z_log_var[0][0]
__________________________________________________________________________________
__________________
tf_op_layer_SquaredDifference ( [(None, 784)]        0           decoder[1][0]
encoder_input[0][0]
__________________________________________________________________________________
__________________
tf_op_layer_Sub_1 (TensorFlowOp [(None, 2)]          0
tf_op_layer_Sub[0][0]
tf_op_layer_Exp[0][0]
__________________________________________________________________________________
```

```
------------------
tf_op_layer_Mean (TensorFlowOpL [(None,)]            0
tf_op_layer_SquaredDifference[0][
________________________________________________________________________________
------------------
tf_op_layer_Sum (TensorFlowOpLa [(None,)]            0
tf_op_layer_Sub_1[0][0]
________________________________________________________________________________
------------------
tf_op_layer_Mul (TensorFlowOpLa [(None,)]            0
tf_op_layer_Mean[0][0]
________________________________________________________________________________
------------------
tf_op_layer_Mul_1 (TensorFlowOp [(None,)]            0
tf_op_layer_Sum[0][0]
________________________________________________________________________________
------------------
tf_op_layer_AddV2_1 (TensorFlow [(None,)]            0
tf_op_layer_Mul[0][0]
tf_op_layer_Mul_1[0][0]
________________________________________________________________________________
------------------
tf_op_layer_Mean_1 (TensorFlowO [()]                 0
tf_op_layer_AddV2_1[0][0]
________________________________________________________________________________
------------------
add_loss (AddLoss)              ()                   0
tf_op_layer_Mean_1[0][0]
================================================================================
==================
Total params: 807,700
Trainable params: 807,700
Non-trainable params: 0
________________________________________________________________________________
------------------
```

[15]:
```python
vae.fit(x_train,
        epochs=epochs,
        batch_size=batch_size,
        validation_data=(x_test, None))
vae.save_weights((results_path / 'vae_mlp_mnist.h5').as_posix())
```

```
Epoch 1/50
469/469 [==============================] - 2s 4ms/step - loss: 42.6044 -
val_loss: 34.2133
Epoch 2/50
469/469 [==============================] - 1s 2ms/step - loss: 33.4674 -
val_loss: 32.6424
```

```
Epoch 3/50
469/469 [==============================] - 1s 2ms/step - loss: 32.3152 -
val_loss: 32.0217
Epoch 4/50
469/469 [==============================] - 1s 2ms/step - loss: 31.6140 -
val_loss: 31.4146
Epoch 5/50
469/469 [==============================] - 1s 2ms/step - loss: 31.1505 -
val_loss: 31.0679
Epoch 6/50
469/469 [==============================] - 1s 2ms/step - loss: 30.7917 -
val_loss: 30.6559
Epoch 7/50
469/469 [==============================] - 1s 2ms/step - loss: 30.5014 -
val_loss: 30.2441
Epoch 8/50
469/469 [==============================] - 1s 2ms/step - loss: 30.2827 -
val_loss: 30.3005
Epoch 9/50
469/469 [==============================] - 1s 2ms/step - loss: 30.0447 -
val_loss: 29.8976
Epoch 10/50
469/469 [==============================] - 1s 2ms/step - loss: 29.9301 -
val_loss: 29.9924
Epoch 11/50
469/469 [==============================] - 1s 2ms/step - loss: 29.7390 -
val_loss: 29.6914
Epoch 12/50
469/469 [==============================] - 1s 2ms/step - loss: 29.5829 -
val_loss: 29.6263
Epoch 13/50
469/469 [==============================] - 1s 2ms/step - loss: 29.4653 -
val_loss: 29.8084
Epoch 14/50
469/469 [==============================] - 1s 2ms/step - loss: 29.3385 -
val_loss: 29.3024
Epoch 15/50
469/469 [==============================] - 1s 1ms/step - loss: 29.2424 -
val_loss: 29.2218
Epoch 16/50
469/469 [==============================] - 1s 1ms/step - loss: 29.1593 -
val_loss: 29.4312
Epoch 17/50
469/469 [==============================] - 1s 1ms/step - loss: 29.0132 -
val_loss: 29.1509
Epoch 18/50
469/469 [==============================] - 1s 2ms/step - loss: 28.9567 -
val_loss: 28.9628
```

```
Epoch 19/50
469/469 [==============================] - 1s 1ms/step - loss: 28.9033 -
val_loss: 29.0355
Epoch 20/50
469/469 [==============================] - 1s 1ms/step - loss: 28.8013 -
val_loss: 28.9237
Epoch 21/50
469/469 [==============================] - 1s 1ms/step - loss: 28.7235 -
val_loss: 28.7236
Epoch 22/50
469/469 [==============================] - 1s 2ms/step - loss: 28.6351 -
val_loss: 28.5885
Epoch 23/50
469/469 [==============================] - 1s 2ms/step - loss: 28.5886 -
val_loss: 28.7825
Epoch 24/50
469/469 [==============================] - 1s 1ms/step - loss: 28.5513 -
val_loss: 28.6018
Epoch 25/50
469/469 [==============================] - 1s 1ms/step - loss: 28.4255 -
val_loss: 28.5597
Epoch 26/50
469/469 [==============================] - 1s 1ms/step - loss: 28.3960 -
val_loss: 28.4661
Epoch 27/50
469/469 [==============================] - 1s 2ms/step - loss: 28.3348 -
val_loss: 28.5090
Epoch 28/50
469/469 [==============================] - 1s 2ms/step - loss: 28.2976 -
val_loss: 28.4132
Epoch 29/50
469/469 [==============================] - 1s 2ms/step - loss: 28.2094 -
val_loss: 28.4153
Epoch 30/50
469/469 [==============================] - 1s 2ms/step - loss: 28.2082 -
val_loss: 28.5260
Epoch 31/50
469/469 [==============================] - 1s 2ms/step - loss: 28.1639 -
val_loss: 28.4743
Epoch 32/50
469/469 [==============================] - 1s 2ms/step - loss: 28.0970 -
val_loss: 28.2689
Epoch 33/50
469/469 [==============================] - 1s 2ms/step - loss: 28.0877 -
val_loss: 28.2841
Epoch 34/50
469/469 [==============================] - 1s 1ms/step - loss: 28.0367 -
val_loss: 28.4918
```

```
Epoch 35/50
469/469 [==============================] - 1s 1ms/step - loss: 28.0261 -
val_loss: 28.2841
Epoch 36/50
469/469 [==============================] - 1s 2ms/step - loss: 27.9823 -
val_loss: 28.1451
Epoch 37/50
469/469 [==============================] - 1s 1ms/step - loss: 27.9752 -
val_loss: 28.3467
Epoch 38/50
469/469 [==============================] - 1s 2ms/step - loss: 27.9074 -
val_loss: 28.2239
Epoch 39/50
469/469 [==============================] - 1s 2ms/step - loss: 27.8626 -
val_loss: 28.1703
Epoch 40/50
469/469 [==============================] - 1s 2ms/step - loss: 27.8487 -
val_loss: 28.0867
Epoch 41/50
469/469 [==============================] - 1s 1ms/step - loss: 27.8264 -
val_loss: 28.0996
Epoch 42/50
469/469 [==============================] - 1s 1ms/step - loss: 27.7949 -
val_loss: 28.0658
Epoch 43/50
469/469 [==============================] - 1s 2ms/step - loss: 27.7961 -
val_loss: 27.9862
Epoch 44/50
469/469 [==============================] - 1s 1ms/step - loss: 27.7430 -
val_loss: 28.1035
Epoch 45/50
469/469 [==============================] - 1s 1ms/step - loss: 27.7110 -
val_loss: 28.0929
Epoch 46/50
469/469 [==============================] - 1s 1ms/step - loss: 27.7035 -
val_loss: 27.9722
Epoch 47/50
469/469 [==============================] - 1s 2ms/step - loss: 27.6859 -
val_loss: 27.9249
Epoch 48/50
469/469 [==============================] - 1s 1ms/step - loss: 27.6500 -
val_loss: 28.0056
Epoch 49/50
469/469 [==============================] - 1s 1ms/step - loss: 27.6530 -
val_loss: 27.9606
Epoch 50/50
469/469 [==============================] - 1s 1ms/step - loss: 27.6200 -
val_loss: 27.8897
```

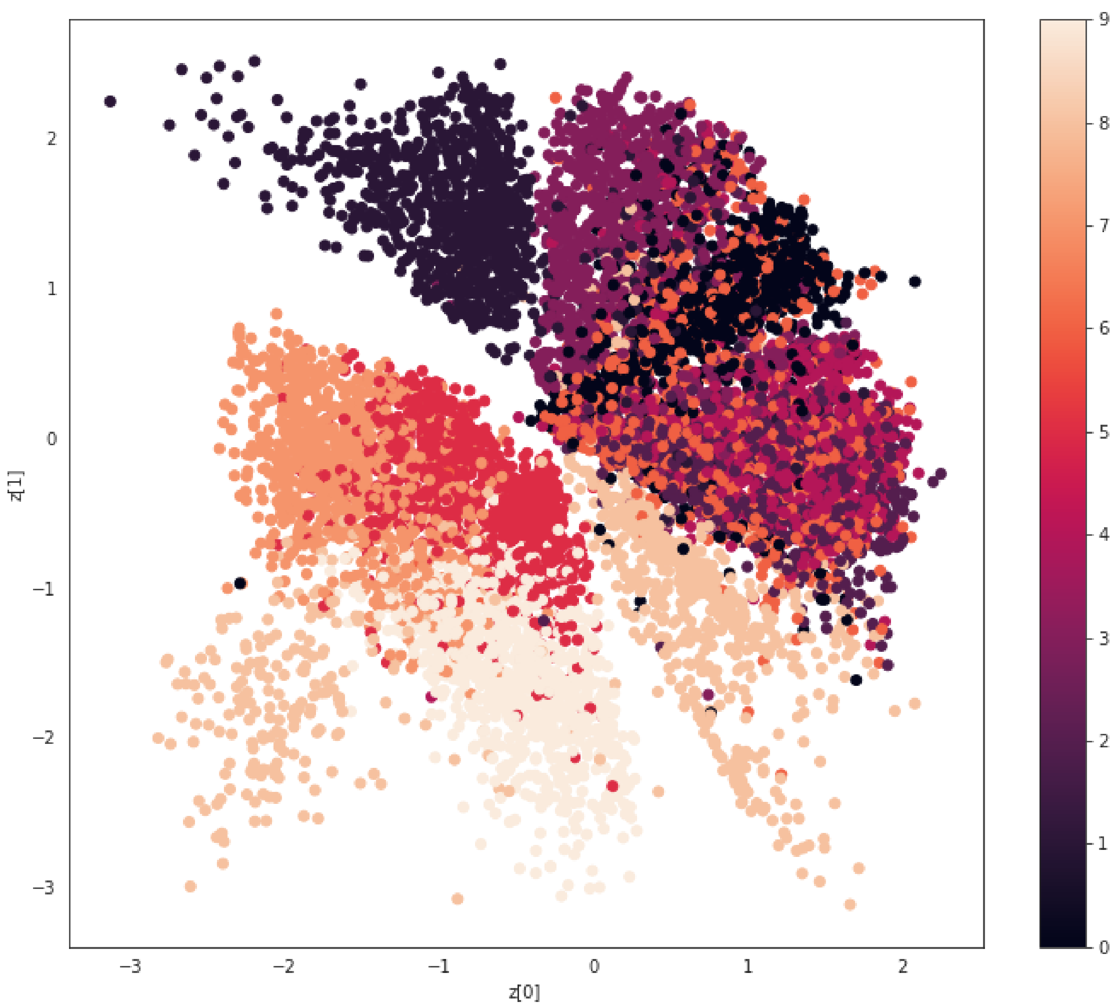
1.6 Plot Results

[16]:
```
def plot_results(models,
                 data,
                 batch_size=128,
                 model_name="vae_mnist"):
    """Plots labels and MNIST digits as function of 2-dim latent vector

    # Arguments
        models (tuple): encoder and decoder models
        data (tuple): test data and label
        batch_size (int): prediction batch size
        model_name (string): which model is using this function
    """

    encoder, decoder = models
    x_test, y_test = data
    os.makedirs(model_name, exist_ok=True)

    filename = results_path / 'vae_mean'
    # display a 2D plot of the digit classes in the latent space
    z_mean, _, _ = encoder.predict(x_test,
                                   batch_size=batch_size)
    plt.figure(figsize=(12, 10))
    plt.scatter(z_mean[:, 0], z_mean[:, 1], c=y_test)
    plt.colorbar()
    plt.xlabel("z[0]")
    plt.ylabel("z[1]")
    plt.savefig(filename, dpi=300)
    plt.show()

    filename = results_path / 'digits_over_latent'
    # display a 30x30 2D manifold of digits
    n = 30
    digit_size = 28
    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(-4, 4, n)
    grid_y = np.linspace(-4, 4, 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=(10, 10))
    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(filename, dpi=300)
    plt.show()
```

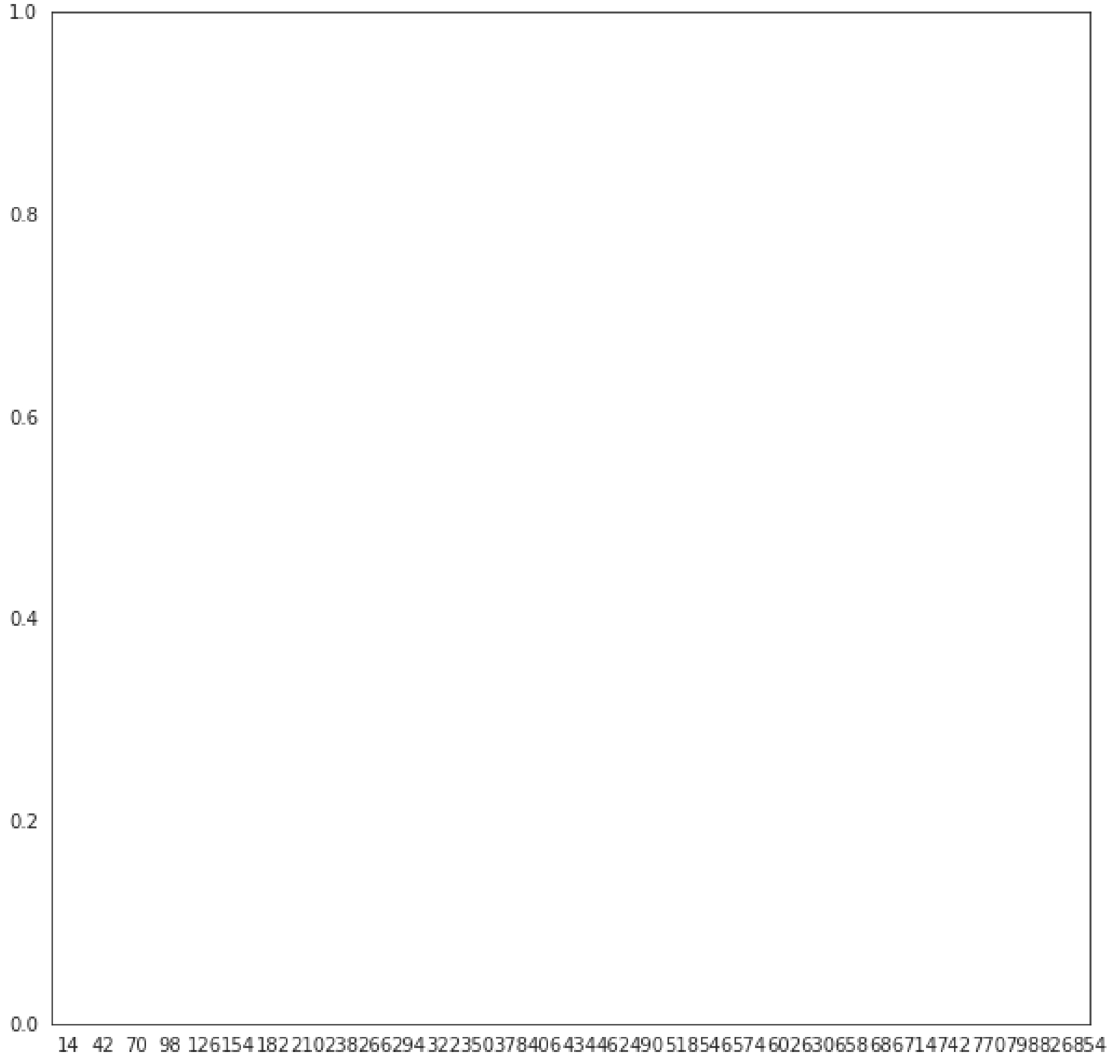
[17]:

```
plot_results(models,
             data,
             batch_size=batch_size,
             model_name="vae_mlp")
```




```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-17-8b58c392dfb7> in <module>
----> 1 plot_results(models,

      2              data,
      3              batch_size=batch_size,
      4              model_name="vae_mlp")

<ipython-input-16-57e41dd7bc68> in plot_results(models, data, batch_size,␣
 ↪model_name)
     52     sample_range_x = np.round(grid_x, 1)
     53     sample_range_y = np.round(grid_y, 1)
---> 54     plt.xticks(pixel_range, sample_range_x)
     55     plt.yticks(pixel_range, sample_range_y)
     56     plt.xlabel("z[0]")
```

```
~/.pyenv/versions/miniconda3-latest/envs/ml4t-dl/lib/python3.8/site-packages/
↪matplotlib/pyplot.py in xticks(ticks, labels, **kwargs)
   1657         labels = ax.get_xticklabels()
   1658     else:
-> 1659         labels = ax.set_xticklabels(labels, **kwargs)
   1660     for l in labels:
   1661         l.update(kwargs)

~/.pyenv/versions/miniconda3-latest/envs/ml4t-dl/lib/python3.8/site-packages/
↪matplotlib/axes/_base.py in wrapper(self, *args, **kwargs)
     61
     62         def wrapper(self, *args, **kwargs):
---> 63             return get_method(self)(*args, **kwargs)
     64
     65         wrapper.__module__ = owner.__module__

~/.pyenv/versions/miniconda3-latest/envs/ml4t-dl/lib/python3.8/site-packages/
↪matplotlib/cbook/deprecation.py in wrapper(*args, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

~/.pyenv/versions/miniconda3-latest/envs/ml4t-dl/lib/python3.8/site-packages/
↪matplotlib/axis.py in _set_ticklabels(self, labels, fontdict, minor, **kwargs
   1794         if fontdict is not None:
   1795             kwargs.update(fontdict)
-> 1796         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1797
   1798     @cbook._make_keyword_only("3.2", "minor")

~/.pyenv/versions/miniconda3-latest/envs/ml4t-dl/lib/python3.8/site-packages/
↪matplotlib/axis.py in set_ticklabels(self, ticklabels, minor, **kwargs)
   1715             # remove all tick labels, so only error for > 0 ticklabels
   1716             if len(locator.locs) != len(ticklabels) and len(ticklabels) !
↪= 0:
-> 1717                 raise ValueError(
   1718                     "The number of FixedLocator locations"
   1719                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (31), usually from a call to␣
↪set_ticks, does not match the number of ticklabels (30).
```


1.0
0.8
0.6
0.4
0.2
0.0
14 42 70 98 126154182210238266294322350378406434462490518546574602630658686714742770798826854