Assignment12_VermaGourav

March 6, 2021

Gourav Verma DSC 650 Assignment 12 : variational autoencoder

[2]:

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import keras
from keras import layers
from keras import backend as K
from keras.models import Model
import numpy as np
from pathlib import Path
import time
start_time = time.time()
# Needed the following as caused CUDA DNN errors
#physical_devices = tf.config.list_physical_devices('GPU')
#tf.config.experimental.set_memory_growth(physical_devices[0], True)
```

[3]:

```
results_dir = Path('C:/Users/goura/Desktop/GARV ML/DSC 650/
 ↪Assignment12_VermaGourav/').joinpath('results').joinpath('vae')
results_dir.mkdir(parents=True, exist_ok=True)
```

[4]:

```
img_shape = (28,28, 1)
batch_size = 16
latent_dim = 2

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)

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)
```

[5]:

```
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])

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)
```

[6]:

```
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
```

[7]:

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

[8]:

```
from keras.datasets import mnist
vae = Model(input_img, y)
vae.compile(optimizer='rmsprop', loss=None)
vae.summary()

(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))
```

```
WARNING:tensorflow:Output custom_variational_layer 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.
Model: "model_1"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            [(None, 28, 28, 1)]  0
________________________________________________________________________________
__________________
conv2d (Conv2D)                 (None, 28, 28, 32)   320         input_1[0][0]
________________________________________________________________________________
__________________
conv2d_1 (Conv2D)               (None, 14, 14, 64)   18496       conv2d[0][0]
________________________________________________________________________________
__________________
conv2d_2 (Conv2D)               (None, 14, 14, 64)   36928       conv2d_1[0][0]
________________________________________________________________________________
__________________
conv2d_3 (Conv2D)               (None, 14, 14, 64)   36928       conv2d_2[0][0]
________________________________________________________________________________
__________________
flatten (Flatten)               (None, 12544)        0           conv2d_3[0][0]
________________________________________________________________________________
__________________
dense (Dense)                   (None, 32)           401440      flatten[0][0]
________________________________________________________________________________
__________________
dense_1 (Dense)                 (None, 2)            66          dense[0][0]
________________________________________________________________________________
__________________
dense_2 (Dense)                 (None, 2)            66          dense[0][0]
________________________________________________________________________________
__________________
lambda (Lambda)                 (None, 2)            0           dense_1[0][0]
                                                                 dense_2[0][0]
________________________________________________________________________________
__________________
model (Functional)              (None, 28, 28, 1)    56385       lambda[0][0]
```

```
--------------------------------------------------------------------------------
------------------
custom_variational_layer (Custo (None, 28, 28, 1)    0           input_1[0][0]
                                                                 model[0][0]
================================================================================
==================
Total params: 550,629
Trainable params: 550,629
Non-trainable params: 0
--------------------------------------------------------------------------------
------------------
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - ETA: 0s - loss: 2791277435.5020
```

```
C:\Users\goura\AppData\Roaming\Python\Python37\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. '
```

```
60000/60000 [==============================] - 321s 5ms/sample - loss:
2791277435.5020 - val_loss: 0.1996
Epoch 2/10
60000/60000 [==============================] - 320s 5ms/sample - loss: 0.1947 -
val_loss: 0.1918
Epoch 3/10
  176/60000 […] - ETA: 5:28 - loss: 0.1850
```

```
       ␣
 ↪---------------------------------------------------------------------------

        KeyboardInterrupt                         Traceback (most recent call␣
 ↪last)

        <ipython-input-8-23b7311d3749> in <module>
         11 x_test = x_test.reshape(x_test.shape + (1,))
         12
    ---> 13 vae.fit(x=x_train, y=None, shuffle=True, epochs=10,␣
 ↪batch_size=batch_size, validation_data=(x_test, None))


       ␣
 ↪~\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\training_v1.
 ↪py in fit(self, x, y, batch_size, epochs, verbose, callbacks,␣
 ↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
 ↪initial_epoch, steps_per_epoch, validation_steps, validation_freq,␣
 ↪max_queue_size, workers, use_multiprocessing, **kwargs)
```

```
    806         max_queue_size=max_queue_size,
    807         workers=workers,
--> 808         use_multiprocessing=use_multiprocessing)
    809
    810   def evaluate(self,


      ␣
↪~\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\training_arra
↪py in fit(self, model, x, y, batch_size, epochs, verbose, callbacks,␣
↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
↪initial_epoch, steps_per_epoch, validation_steps, validation_freq, **kwargs)
    662         validation_steps=validation_steps,
    663         validation_freq=validation_freq,
--> 664         steps_name='steps_per_epoch')
    665
    666   def evaluate(self,


      ␣
↪~\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\engine\training_arra
↪py in model_iteration(model, inputs, targets, sample_weights, batch_size,␣
↪epochs, verbose, callbacks, val_inputs, val_targets, val_sample_weights,␣
↪shuffle, initial_epoch, steps_per_epoch, validation_steps, validation_freq,␣
↪mode, validation_in_fit, prepared_feed_values_from_dataset, steps_name,␣
↪**kwargs)
    382
    383         # Get outputs.
--> 384         batch_outs = f(ins_batch)
    385         if not isinstance(batch_outs, list):
    386           batch_outs = [batch_outs]


      ␣
↪~\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\keras\backend.
↪py in __call__(self, inputs)
   3955
   3956     fetched = self._callable_fn(*array_vals,
-> 3957                                 run_metadata=self.run_metadata)
   3958     self._call_fetch_callbacks(fetched[-len(self._fetches):])
   3959     output_structure = nest.pack_sequence_as(


      ␣
↪~\AppData\Roaming\Python\Python37\site-packages\tensorflow\python\client\session.
↪py in __call__(self, *args, **kwargs)
```

```
      1480         ret = tf_session.TF_SessionRunCallable(self._session.
↪_session,
      1481                                                self._handle, args,
   -> 1482                                                run_metadata_ptr)
      1483         if run_metadata:
      1484           proto_data = tf_session.TF_GetBuffer(run_metadata_ptr)


       KeyboardInterrupt:
```

[9]:

```
import matplotlib.pyplot as plt
from scipy.stats import norm
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))
print("grid_x")
print(grid_x)
grid_y = norm.ppf(np.linspace(0.05, 0.95, n))
print("grid_y")
print(grid_y)

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')
img_file = results_dir.joinpath('Assignment_12_15x15_Grid.png')
plt.savefig(img_file)
plt.show()
```

```
grid_x
[-1.64485363e+00 -1.20404696e+00 -9.20822976e-01 -6.97141435e-01
 -5.03965367e-01 -3.28072108e-01 -1.61844167e-01 -1.39145821e-16
  1.61844167e-01  3.28072108e-01  5.03965367e-01  6.97141435e-01
  9.20822976e-01  1.20404696e+00  1.64485363e+00]
grid_y
[-1.64485363e+00 -1.20404696e+00 -9.20822976e-01 -6.97141435e-01
 -5.03965367e-01 -3.28072108e-01 -1.61844167e-01 -1.39145821e-16
  1.61844167e-01  3.28072108e-01  5.03965367e-01  6.97141435e-01
  9.20822976e-01  1.20404696e+00  1.64485363e+00]
```

```
C:\Users\goura\AppData\Roaming\Python\Python37\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. '
```

```
<Figure size 1000x1000 with 1 Axes>
```

[ ]: