BDAA-ICP8-wiki report : Yousef Almutairi
Date: 10/25/2021

1. <u>What you learned in the ICP:</u>

   I learned to use the Autoencoder algorithm and built the 3 components such as encoder, code, and c decoder. The Autoencoder is used to compress the input into a lower-dimensional code and then reconstruct the output from this representation.

2. <u>Screen shots that shows the successful execution of each required step of your code:</u>

   Since the source code has been given by the instructor and asked us to add only 2 encoder and decoder layers I did not take a screenshot for the part that I did not change.

a. Encoder:

In this part, I have added Conv2D with 32 filters and 1 strides in order to move the filters to 1 pixel at a time. The reason that I increased the filter size is to capture more combinations. Also, I have added BatchNormalization in order to maintain the mean output close to 0 and the output standard deviation close to 1 and enhance the neural network performance.

```
# Encoder Definition
i       = Input(shape=input_shape, name='encoder_input')
cx      = Conv2D(filters=8, kernel_size=3, strides=2, padding='same', activation='relu')(i)
cx      = BatchNormalization()(cx)
cx      = Conv2D(filters=16, kernel_size=3, strides=2, padding='same', activation='relu')(cx)
cx      = BatchNormalization()(cx)
          # New 2 layer: I added Conv2D with 32 filters and BatchNormalization to help coordinate the update of multiple layers in the model.
cx      = Conv2D(filters=32, kernel_size=3, strides=1, padding='same', activation='relu')(cx)
cx      = BatchNormalization()(cx)
x       = Flatten()(cx)
x       = Dense(20, activation='relu')(x)
x       = BatchNormalization()(x)
mu      = Dense(latent_dim, name='latent_mu')(x)
sigma   = Dense(latent_dim, name='latent_sigma')(x)
```

+ Code   + Text

Encoder summary:

```
[ ] # Instantiate encoder
    encoder = Model(i, [mu, sigma, z], name='encoder')
    encoder.summary()
```

```
Model: "encoder"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
encoder_input (InputLayer)      [(None, 28, 28, 1)]  0
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 14, 14, 8)    80          encoder_input[0][0]
__________________________________________________________________________________________________
batch_normalization_16 (BatchNo (None, 14, 14, 8)    32          conv2d_6[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 7, 7, 16)     1168        batch_normalization_16[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, 7, 7, 16)     64          conv2d_7[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 7, 7, 32)     4640        batch_normalization_17[0][0]
__________________________________________________________________________________________________
batch_normalization_18 (BatchNo (None, 7, 7, 32)     128         conv2d_8[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 1568)         0           batch_normalization_18[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 20)           31380       flatten_2[0][0]
__________________________________________________________________________________________________
batch_normalization_19 (BatchNo (None, 20)           80          dense_4[0][0]
__________________________________________________________________________________________________
latent_mu (Dense)               (None, 2)            42          batch_normalization_19[0][0]
__________________________________________________________________________________________________
latent_sigma (Dense)            (None, 2)            42          batch_normalization_19[0][0]
__________________________________________________________________________________________________
z (Lambda)                      (None, 2)            0           latent_mu[0][0]
                                                                 latent_sigma[0][0]
==================================================================================================
Total params: 37,656
Trainable params: 37,504
Non-trainable params: 152
__________________________________________________________________________________________________
```

b. Decoder:

In this part, I have the same layers that I added to the encoder. After the model has trained I used the Conv2DTranspose in the decoder to upsample its input and to arise from the desire to use a transformation going in the opposite direction of a normal convolution.

```
# Decoder Definition
d_i   = Input(shape=(latent_dim, ), name='decoder_input')
x     = Dense(conv_shape[1] * conv_shape[2] * conv_shape[3], activation='relu')(d_i)
x     = BatchNormalization()(x)
x     = Reshape((conv_shape[1], conv_shape[2], conv_shape[3]))(x)
      # I start the decoder with the one that I have added in the encoder.
cx    = Conv2DTranspose(filters=32, kernel_size=3, strides=1, padding='same', activation='relu')(x)
cx    = BatchNormalization()(cx)
cx    = Conv2DTranspose(filters=16, kernel_size=3, strides=2, padding='same', activation='relu')(cx)
cx    = BatchNormalization()(cx)
cx    = Conv2DTranspose(filters=8, kernel_size=3, strides=2, padding='same',  activation='relu')(cx)
cx    = BatchNormalization()(cx)
o     = Conv2DTranspose(filters=num_channels, kernel_size=3, activation='sigmoid', padding='same', name='decoder_output')(cx)
```

BDAA-ICP8-wiki report : Yousef Almutairi
Date: 10/25/2021

Decoder summary:

```
[ ] # Instantiate decoder
    decoder = Model(d_i, o, name='decoder')
    decoder.summary()
```

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
decoder_input (InputLayer)   [(None, 2)]               0
_________________________________________________________________
dense_5 (Dense)              (None, 1568)              4704
_________________________________________________________________
batch_normalization_20 (Batc (None, 1568)              6272
_________________________________________________________________
reshape_2 (Reshape)          (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_transpose_6 (Conv2DTr (None, 7, 7, 32)          9248
_________________________________________________________________
batch_normalization_21 (Batc (None, 7, 7, 32)          128
_________________________________________________________________
conv2d_transpose_7 (Conv2DTr (None, 14, 14, 16)        4624
_________________________________________________________________
batch_normalization_22 (Batc (None, 14, 14, 16)        64
_________________________________________________________________
conv2d_transpose_8 (Conv2DTr (None, 28, 28, 8)         1160
_________________________________________________________________
batch_normalization_23 (Batc (None, 28, 28, 8)         32
_________________________________________________________________
decoder_output (Conv2DTransp (None, 28, 28, 1)         73
=================================================================
Total params: 26,305
Trainable params: 23,057
Non-trainable params: 3,248
_________________________________________________________________
```

BDAA-ICP8-wiki report : Yousef Almutairi
Date: 10/25/2021

In this screenshot (Instantiate VAE) we can see the shape of the input and output is the same.

```
# Instantiate VAE
vae_outputs = decoder(encoder(i)[2])
vae         = Model(i, vae_outputs, name='vae')
vae.summary()
```

```
Model: "vae"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   [(None, 28, 28, 1)]       0
_________________________________________________________________
encoder (Functional)         [(None, 2), (None, 2), (N 37656
_________________________________________________________________
decoder (Functional)         (None, 28, 28, 1)         26305
=================================================================
Total params: 63,961
Trainable params: 60,561
Non-trainable params: 3,400
_________________________________________________________________
```

Then I trained the model using fit function and I got (loss: 0.1874 - val_loss: 0.1929) which means the model's performance is good.

```
tf.config.run_functions_eagerly(True)
# Compile VAE
vae.compile(optimizer='adam', loss='binary_crossentropy')

# Train autoencoder
vae.fit(input_train, input_train, epochs = no_epochs, batch_size = batch_size, validation_split = validation_split)
```

```
40/40 [==============================] - 4s 100ms/step - loss: 0.1995 - val_loss: 0.2027
Epoch 23/50
40/40 [==============================] - 4s 89ms/step - loss: 0.1980 - val_loss: 0.2036
Epoch 24/50
40/40 [==============================] - 3s 88ms/step - loss: 0.1980 - val_loss: 0.2012
Epoch 25/50
40/40 [==============================] - 4s 99ms/step - loss: 0.1975 - val_loss: 0.2045
Epoch 26/50
40/40 [==============================] - 4s 100ms/step - loss: 0.2009 - val_loss: 0.2176
Epoch 27/50
40/40 [==============================] - 4s 91ms/step - loss: 0.1972 - val_loss: 0.2049
Epoch 28/50
40/40 [==============================] - 3s 87ms/step - loss: 0.1978 - val_loss: 0.2015
Epoch 29/50
40/40 [==============================] - 4s 98ms/step - loss: 0.1951 - val_loss: 0.1994
Epoch 30/50
40/40 [==============================] - 4s 100ms/step - loss: 0.1947 - val_loss: 0.1974
Epoch 31/50
40/40 [==============================] - 4s 99ms/step - loss: 0.1933 - val_loss: 0.1983
Epoch 32/50
40/40 [==============================] - 4s 99ms/step - loss: 0.1944 - val_loss: 0.1970
Epoch 33/50
40/40 [==============================] - 4s 90ms/step - loss: 0.1930 - val_loss: 0.1980
Epoch 34/50
40/40 [==============================] - 4s 99ms/step - loss: 0.1947 - val_loss: 0.1981
Epoch 35/50
40/40 [==============================] - 4s 98ms/step - loss: 0.1931 - val_loss: 0.1967
Epoch 36/50
40/40 [==============================] - 4s 99ms/step - loss: 0.1919 - val_loss: 0.1966
Epoch 37/50
40/40 [==============================] - 3s 88ms/step - loss: 0.1916 - val_loss: 0.1961
Epoch 38/50
40/40 [==============================] - 4s 88ms/step - loss: 0.1915 - val_loss: 0.1968
Epoch 39/50
40/40 [==============================] - 4s 98ms/step - loss: 0.1896 - val_loss: 0.1939
Epoch 40/50
40/40 [==============================] - 4s 96ms/step - loss: 0.1902 - val_loss: 0.1956
```

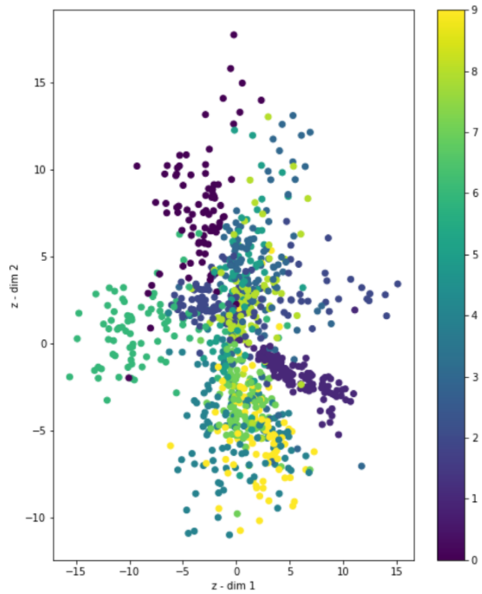
BDAA-ICP8-wiki report : Yousef Almutairi
Date: 10/25/2021

From the screenshot below I can notice the model is doing great job because 2 things:

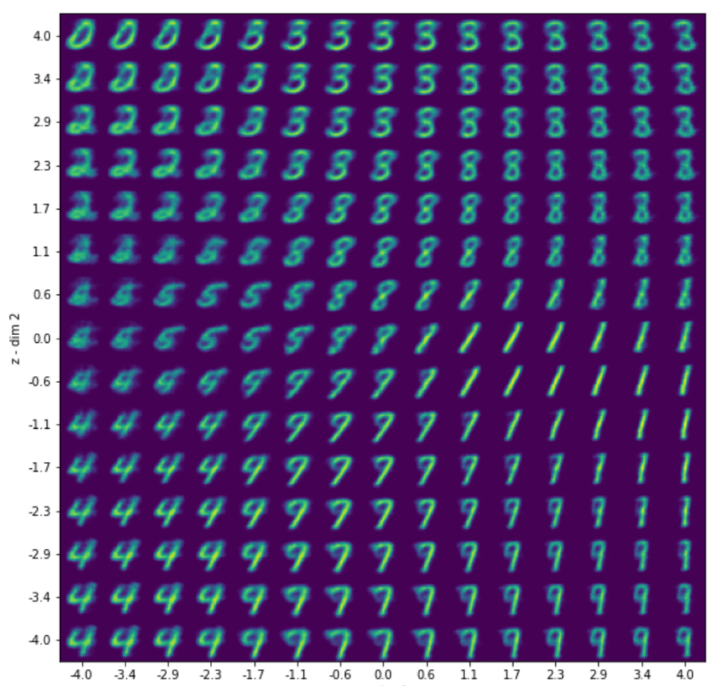
1. The cluster is close to each other and that means the generated value is good.

```
# Plot results
data = (input_test, target_test)
viz_latent_space(encoder, data)
viz_decoded(encoder, decoder, data)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/data/ops
  "Even though the `tf.config.experimental_run_functions_eagerly`
```



2. The sampled that is displayed is show how the model is performing. Although there is some noising, but I can see all the numbers.



3. Conclusion:

   I have Successfully executed the code and made autoencoder and built the convolutional and denoising autoencoders with the MNIST dataset in Keras. In this ICP I learned more how downsamples the input using the Encoder part, and upsamples its input in Decoder part.

4. Video link:

https://drive.google.com/file/d/1VLytTwouTeNcisrG1tClKWPcCaPOuJy0/view?usp=sharing