Assignment 3 Report

Amirhossein Sohrabbeig - STD: 1744420

2022-03-24

1 Problem 1

1.1 Is the VAE model has convolutional layers? Please sketch the VAE model.

No. Our VAE model does not have any convolutional layer. You can find the sketch of encoder and decoder below:

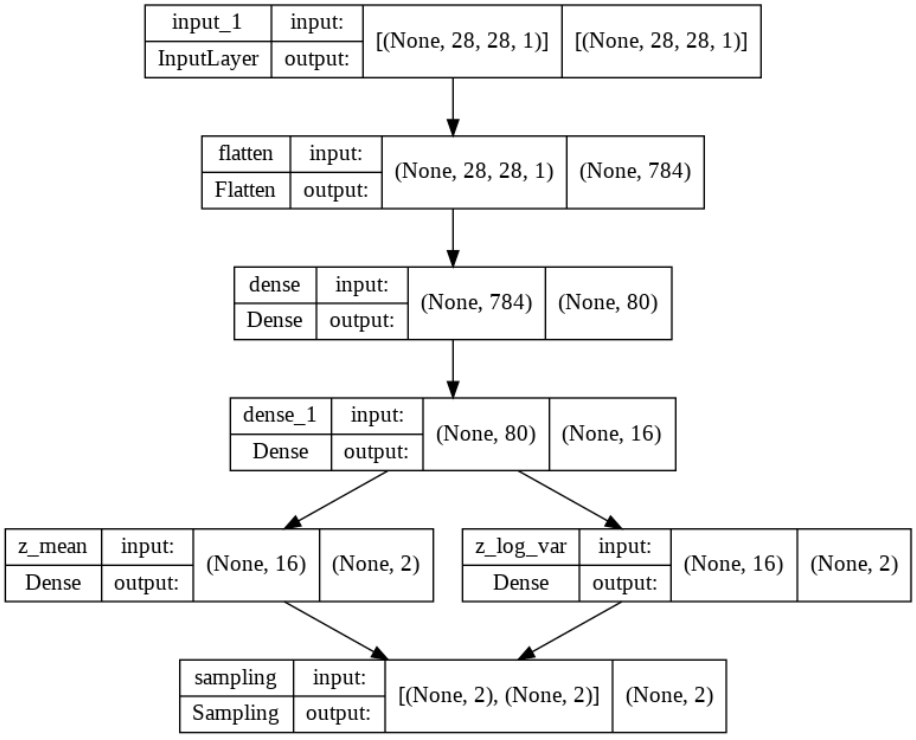

Figure 1: Model plot for encoder

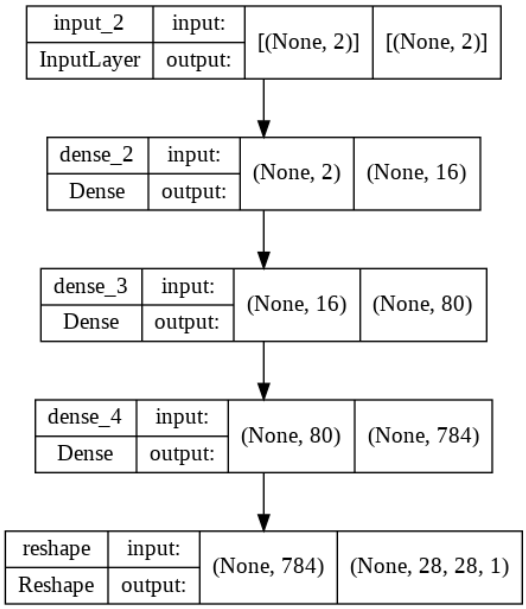

Figure 2: Model plot for decoder

1.2 2D plot of digits

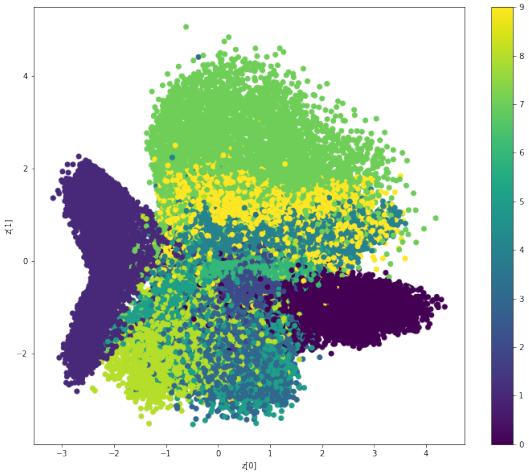

Figure 3: 2-D plot of the distribution of digit classes in the latent space

As you can see, digit classes are centered around (0,0). For training Variational Auto-Encoder(VAE), we add a regularization term to the loss function that

forces class distributions to be as close as possible to the standard normal distribution, which has a mean of zero. That is why our distributions are centered around (0,0).

1.3 Grid of generated sampled digit images

To find active regions for latent variables, we can simply call min(), and max() functions from the NumPy library on the latent values computed in the last step.

```
z_mean, _, _ = encoder.predict(x_train)
x_lower = z_mean[:, 0].min()
x_upper = z_mean[:, 0].max()
y_lower = z_mean[:, 1].min()
y_upper = z_mean[:, 1].max()
```

Listing 1: Finding the active regions for latent variables

The result of generating images with different latent values is as follows:

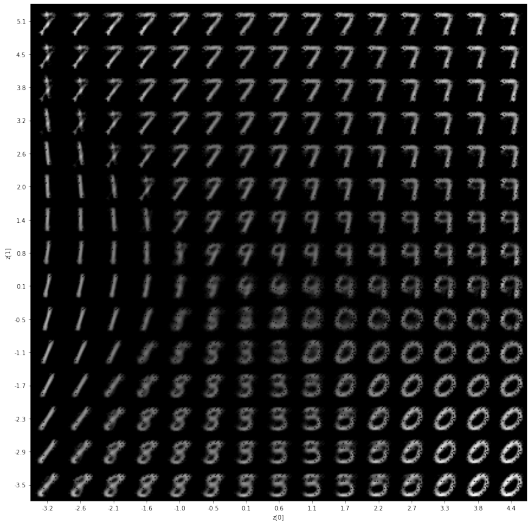

Figure 4: Grid of sampled digit images generated by different latent variables

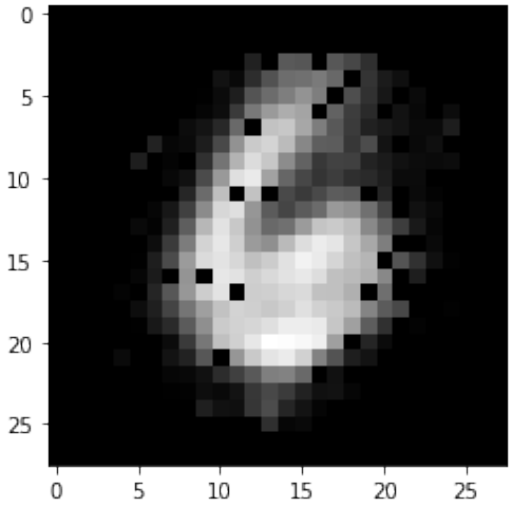

Figure 5: Output of (0,0) with simple VAE model

1.4 Implement the Convolutional VAE

To implement Convolutional VAE, we have to replace some of the Dense layers with Conv2D layers from Keras. The codes for Encoder and Decoder are as follows:

```
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(14, 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=
    "cencoder")
encoder.summary()
```

Listing 2: Encoder for Convolutional VAE

```
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,
    padding="same")(x)
x = layers.Conv2DTranspose(32, 3, activation="relu", strides=2,
    padding="same")(x)
decoder_outputs = layers.Conv2DTranspose(1, 3, activation="sigmoid"
    , padding="same")(x)
decoder = keras.Model(latent_inputs, decoder_outputs, name="
    cdecoder")
```

```
8 decoder.summary()
```

Listing 3: Decoder for Convolutional VAE

1.5 Repeat b) and c) for your Convolutional VAE(CVAE)

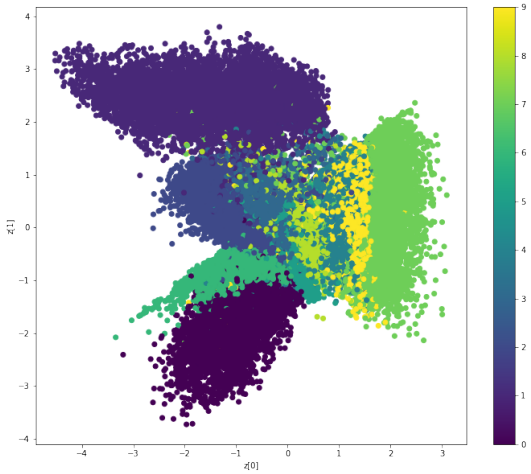

Figure 6: 2-D plot of the distribution of digit classes in the latent space(CVAE)

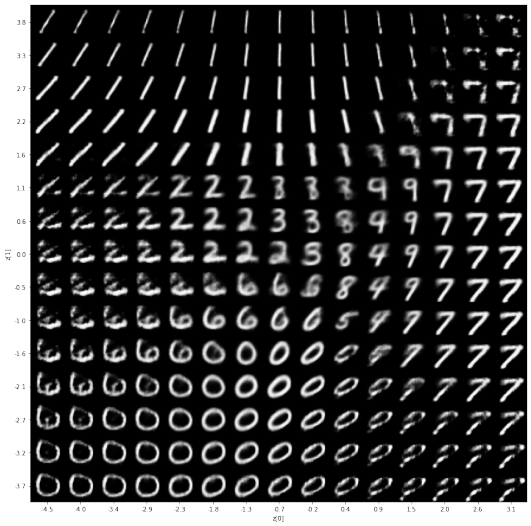

Figure 7: Grid of sampled digit images generated by different latent variables (CVAE)

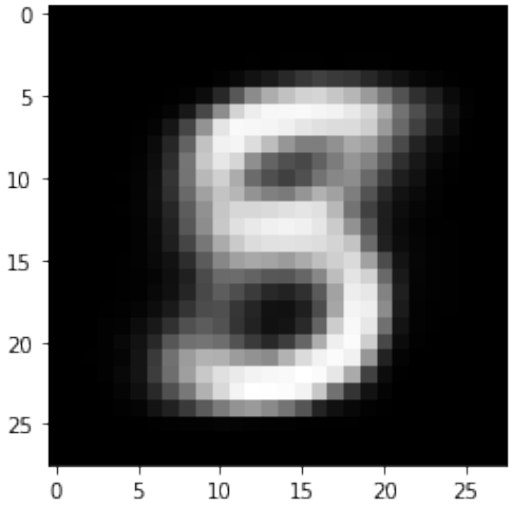

Figure 8: Output of (0,0) with Convolutional VAE(CVAE) model

By carefully looking at digits generated by the Convolutional VAE(CVAE) model, we can easily find out that the newly generated digits have better quality compared to the results of simple VAE.

1.6 Which model has a better performance?

We can compare the performance of these two models(CVAE and VAE) from different aspects. By looking at the generated digits, we can easily determine that CVAE generates digits of higher quality. In terms of training loss, CVAE achieves loss values of **141.69**, **3.70**, and **145.39** for reconstruction loss, KL loss, and total, respectively, which is almost half of the loss values for simple VAE(**282.07**, **3**, **285.07** for reconstruction loss, KL loss, and total, respectively). From the training time point of view, simple VAE is better since each epoch takes about **3** seconds, while this value for CVAE is about **10** seconds(almost three times slower).

1.7 Conditional Convolutional VAE(CCVAE)

To implement Conditional Convolutional VAE(CCVAE), we have to concatenate labels of the images to the input for both the encoder and the decoder. Doing this enables us to have better control over the generating digits by explicitly specifying which digit we expect the model to generate.

To achieve this goal, first, the labels should be transformed into one-hot vector representations and then concatenated with the output of the first convolutional layers in the encoder:

```
encoder_inputs = keras.Input(shape=(28, 28, 1))
```

```
c = keras.Input(shape=(mnist_labels.shape[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 = concat([x, c])
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, c], [z_mean, z_log_var, z],
    name="ccencoder")
encoder.summary()
```

Listing 4: Encoder of CCVAE

And in the Decoder, we have to concatenate the one-hot representation of our desired digit with the sampled latent variables and then generate the digit out of it.

```
latent_inputs = keras.Input(shape=(latent_dim,))
x = concat([latent_inputs, c])
x = layers.Dense(7 * 7 * 64, activation="relu")(x)
x = layers.Reshape((7, 7, 64))(x)
x = layers.Conv2DTranspose(64, 3, activation="relu", strides=2,
    padding="same")(x)
x = layers.Conv2DTranspose(32, 3, activation="relu", strides=2,
    padding="same")(x)
decoder_outputs = layers.Conv2DTranspose(1, 3, activation="sigmoid"
    , padding="same")(x)
decoder = keras.Model([latent_inputs, c], decoder_outputs, name="
    ccdecoder")
decoder.summary()
```

Listing 5: Decoder of CCVAE

1.8 Repeat b) and c) for your Conditional Convolutional VAE(CCVAE)

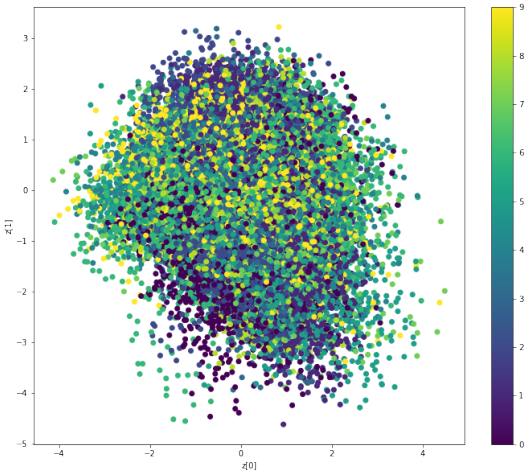

Figure 9: 2-D plot of the distribution of digit classes in the latent space(CCVAE)

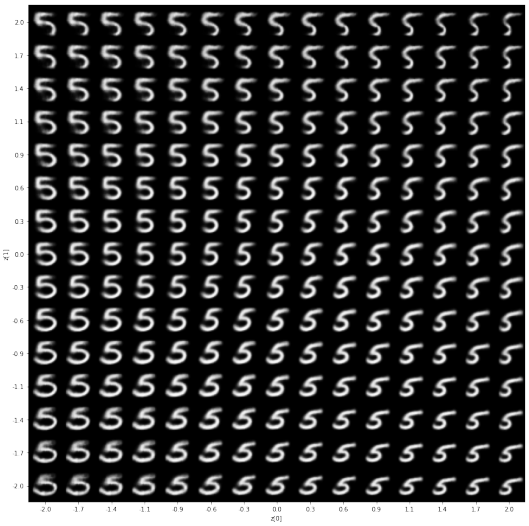

Figure 10: Grid of sampled digit images generated by different latent variables for number 5(CCVAE)

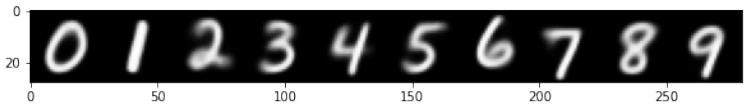

Figure 11: Output of (0,0) with Convolutional VAE(CCVAE) model for different digits

Conditioning our model on the image label is somehow like learning different tasks individually. This way, the distribution of the latent values of the digits can overlap, and one can not find a decision boundary between them easily. You can see this in Fig. 9. Training times for CVAE and CCVAE are similar; however, CCVAE obtain lower loss values of **124.63**, **2.79**, and **127.41** for reconstruction loss, KL loss, and total, respectively. These results show that CCVAE is performing better than CVAE.