Python file "Assignment_5_student_id" information.
python functions:

Input_read. Read data and preprocess data.
Noise_addition. Adding noise to the data
Plot1. Plots the learning curves
Plot2. Plots the original image, noisy image and reconstructed image
Plot 3. Reconstructs images by varying the value of the latent variable.
Architecture_1_Variation_AE. first VAE model discussed in question 2-3
Architecture_2_Variation_AE. Second VAE model discussed in question 2-3
Architecture_2_Variation_AE. third VAE model discussed in question 2-3
Architecture_1_VAE. Its an autoencoder model, coded just for cross checking the models.

Q1. Preprocessing dataset.

As mentioned in the question we have to use balance dataset of emnist. So, I used that data in this whole report. The following step by step process is carried out in "input_data" function of "data_prepro.py" file.

- First to download the dataset, the emnist library was installed using "pip install emnist" in command window.
- I imported the emnist "balanced" training and test samples.
- The training samples are 112800, while the test samples are 18800. All t he images have shape (28, 28)
- After analyzing the dataset, I found that that numerical data is labelled from 0-9. The capital alphabets are labelled from 10 to 35 and onward from 35, the labels are given to small alphabets and special characters.
- As we have to use capital alphabets, so I extracted capital alphabets are those data sets.
- The training samples of alphabets only balance dataset is now 62400 while that of testing set are 10400.
- After obtaining our data, then I analyzed the image pixels. I found that pixel value ranges from 0-255.
- So, the final thing which I did was to normalizing each data by dividing it by 255, and changed the datatype to float values.

The data_prepro.py also contain the following two function.
noise_addition: It generate noisy copies of our dataset. In that function we only add gaussian noise to image and the clip it between 0 and 1.

Note: Question 2 and question 3 are answered simultaneously here.
Q2. and Q3. The VAE Model and its learning curve plots.

I trained different VAE architectures. Three of them are below.

- Architecture 1:

  The architecture is shown in fig.1. This encoder has two convolutional and two maxpooling layers in the encoder. The latent dimension were 2 and the batch_size was 100. The decoder has only dense layers.

  The Train loss, validation loss is shown in figure 2. While train accuracy and validation accuracy are shown in figure 3.

Encoder

Layer (type)	Output Shape	Param #	Connected to
input_19 (InputLayer)	(None, 28, 28, 1)	0	
conv2d_34 (Conv2D)	(None, 28, 28, 16)	160	input_19[0][0]
max_pooling2d_33 (MaxPooling2D)	(None, 14, 14, 16)	0	conv2d_34[0][0]
conv2d_35 (Conv2D)	(None, 14, 14, 8)	1160	max_pooling2d_33[0][0]
max_pooling2d_34 (MaxPooling2D)	(None, 7, 7, 8)	0	conv2d_35[0][0]
flatten (Flatten)	(None, 392)	0	max_pooling2d_34[0][0]
mean (Dense)	(None, 2)	786	flatten[0][0]
var (Dense)	(None, 2)	786	flatten[0][0]
sampling (Lambda)	(None, 2)	0	mean[0][0] var[0][0]

Total params: 2,892
Trainable params: 2,892
Non-trainable params: 0

Decoder

Layer (type)	Output Shape	Param #
sampling (InputLayer)	(None, 2)	0
decoding (Dense)	(None, 512)	1536
decoding1 (Dense)	(None, 256)	131328
flat_decoded (Dense)	(None, 784)	201488
out_decoded (Reshape)	(None, 28, 28, 1)	0

Total params: 334,352
Trainable params: 334,352
Non-trainable params: 0

VAE

Layer (type)	Output Shape	Param #
input_19 (InputLayer)	(None, 28, 28, 1)	0
encoder (Model)	[(None, 2), (None, 2)]	2892
decoder (Model)	(None, 28, 28, 1)	334352

Total params: 337,244
Trainable params: 337,244
Non-trainable params: 0

Figure 1. VAE Architecture 1

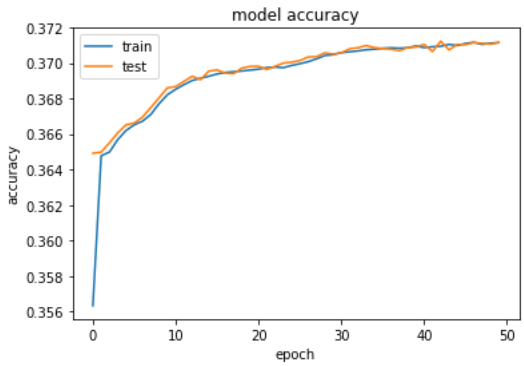


Figure 2. Architecture 1 accuracy.

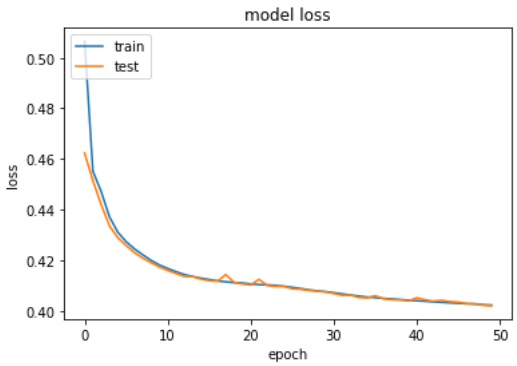


Figure 3. Architecture 1 loss

- Architecture 2:

The architecture is shown in fig.4. This encoder has only one dense layer. The decoder also has a single dense layer. The latent dimension were 2 and the batch_size was 100.

The Train loss, validation loss is shown in figure 5. While train accuracy and validation accuracy are shown in figure 6. The loss in model is decreasing but accuracy is highly fluctuating.

```
Encoder
_________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================
input_encoder (InputLayer)   (None, 28, 28, 1)         0
_________________________________________________________________
flat_incoder (Flatten)       (None, 784)               0         input_encoder[0][0]
_________________________________________________________________
encoding_dense (Dense)       (None, 512)               401920    flat_incoder[0][0]
_________________________________________________________________
mean (Dense)                 (None, 2)                 1026      encoding_dense[0][0]
_________________________________________________________________
dense_var (Dense)            (None, 2)                 1026      encoding_dense[0][0]
_________________________________________________________________
sampling (Lambda)            (None, 2)                 0         mean[0][0]
                                                                 dense_var[0][0]
=================================================================
Total params: 403,972
Trainable params: 403,972
Non-trainable params: 0
_________________________________________________________________

Decoder
=================================================================
Decoder
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_decoder (InputLayer)   (None, 2)                 0
_________________________________________________________________
decoding_Dense (Dense)       (None, 512)               1536
_________________________________________________________________
flat_decoder (Dense)         (None, 784)               402192
_________________________________________________________________
output_decoder (Reshape)     (None, 28, 28, 1)         0
=================================================================
Total params: 403,728
Trainable params: 403,728
Non-trainable params: 0
_________________________________________________________________

VAE
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_encoder (InputLayer)   (None, 28, 28, 1)         0
_________________________________________________________________
encoder (Model)              [(None, 2), (None, 2)]    403972
_________________________________________________________________
decoder (Model)              (None, 28, 28, 1)         403728
=================================================================
Total params: 807,700
Trainable params: 807,700
Non-trainable params: 0
```

Figure 4. Architecture 2

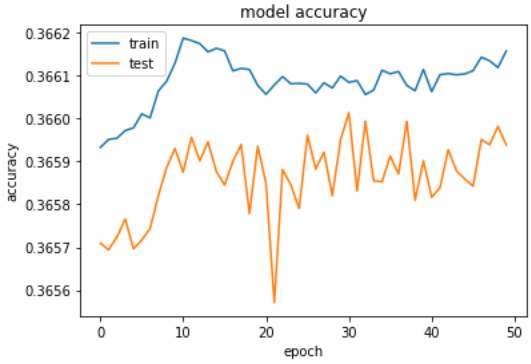


Figure 5. Architecture 2 accuracy

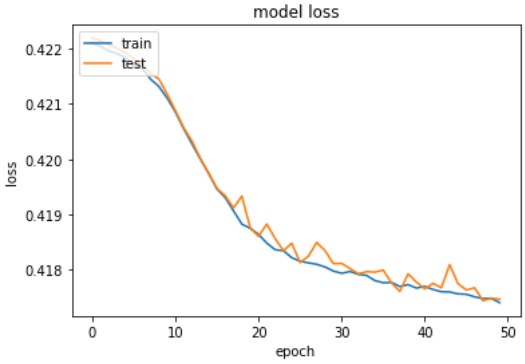


Figure 6. Architecture 2 loss

As I didn't get the required result so I increased the depth of the model and included some convolutional layers in both encoder and decoder. The architecture 3 shows that model. This model gave us some better results.

- Architecture 3:

  The architecture is shown in fig.7. This encoder has three convolutional and three maxpooling layers in the encoder. The latent dimension were 2 and the batch_size was 100. The decoder also has three convolutional layers and also upsampling has been done.

  - The Train loss, validation loss is shown in figure 8. While train accuracy and validation accuracy are shown in figure 9.

Encoder

Layer (type)	Output Shape	Param #	Connected to
input_encoder (InputLayer)	(None, 28, 28, 1)	0	
encoder_conv1 (Conv2D)	(None, 28, 28, 16)	160	input_encoder[0][0]
encoder_maxpool1 (MaxPooling2D)	(None, 14, 14, 16)	0	encoder_conv1[0][0]
encoder_conv2 (Conv2D)	(None, 14, 14, 8)	1160	encoder_maxpool1[0][0]
encoder_maxpool2 (MaxPooling2D)	(None, 7, 7, 8)	0	encoder_conv2[0][0]
encoder_conv3 (Conv2D)	(None, 7, 7, 8)	584	encoder_maxpool2[0][0]
encoder_maxpool3 (MaxPooling2D)	(None, 4, 4, 8)	0	encoder_conv3[0][0]
flatten_5 (Flatten)	(None, 128)	0	encoder_maxpool3[0][0]
mean (Dense)	(None, 2)	258	flatten_5[0][0]
sampling (Dense)	(None, 2)	258	flatten_5[0][0]
z (Lambda)	(None, 2)	0	mean[0][0] sampling[0][0]

Total params: 2,420
Trainable params: 2,420
Non-trainable params: 0

Decoder

Layer (type)	Output Shape	Param #
decoding_input (InputLayer)	(None, 2)	0
decoder_dense1 (Dense)	(None, 128)	384
decoder_reshape (Reshape)	(None, 4, 4, 8)	0
decoder_conv1 (Conv2D)	(None, 4, 4, 8)	584
decoder_upsammpling1 (UpSamp	(None, 8, 8, 8)	0
decoder_conv2 (Conv2D)	(None, 8, 8, 8)	584
decoder_upsammpling2 (UpSamp	(None, 16, 16, 8)	0
decoder_conv3 (Conv2D)	(None, 14, 14, 16)	1168
decoder_upsammpling3 (UpSamp	(None, 28, 28, 16)	0
decoder_conv4 (Conv2D)	(None, 28, 28, 1)	145

Total params: 2,865
Trainable params: 2,865
Non-trainable params: 0

VAE

Layer (type)	Output Shape	Param #
input_encoder (InputLayer)	(None, 28, 28, 1)	0
encoder (Model)	[(None, 2), (None, 2), (N	2420
decoder (Model)	(None, 28, 28, 1)	2865

Total params: 5,285
Trainable params: 5,285
Non-trainable params: 0

Figure 7. Architecture 3

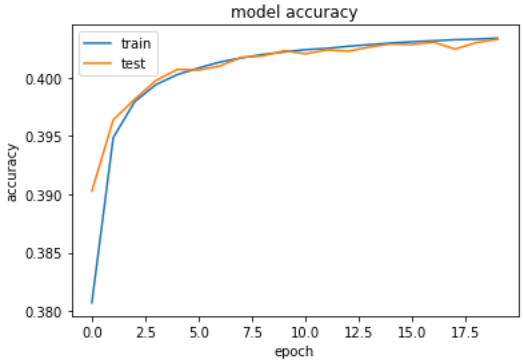


Figure 2.Architecture 3 accuracy

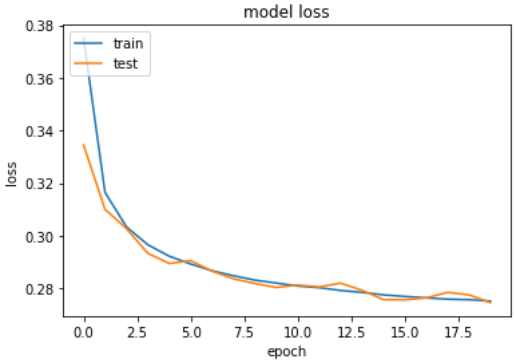
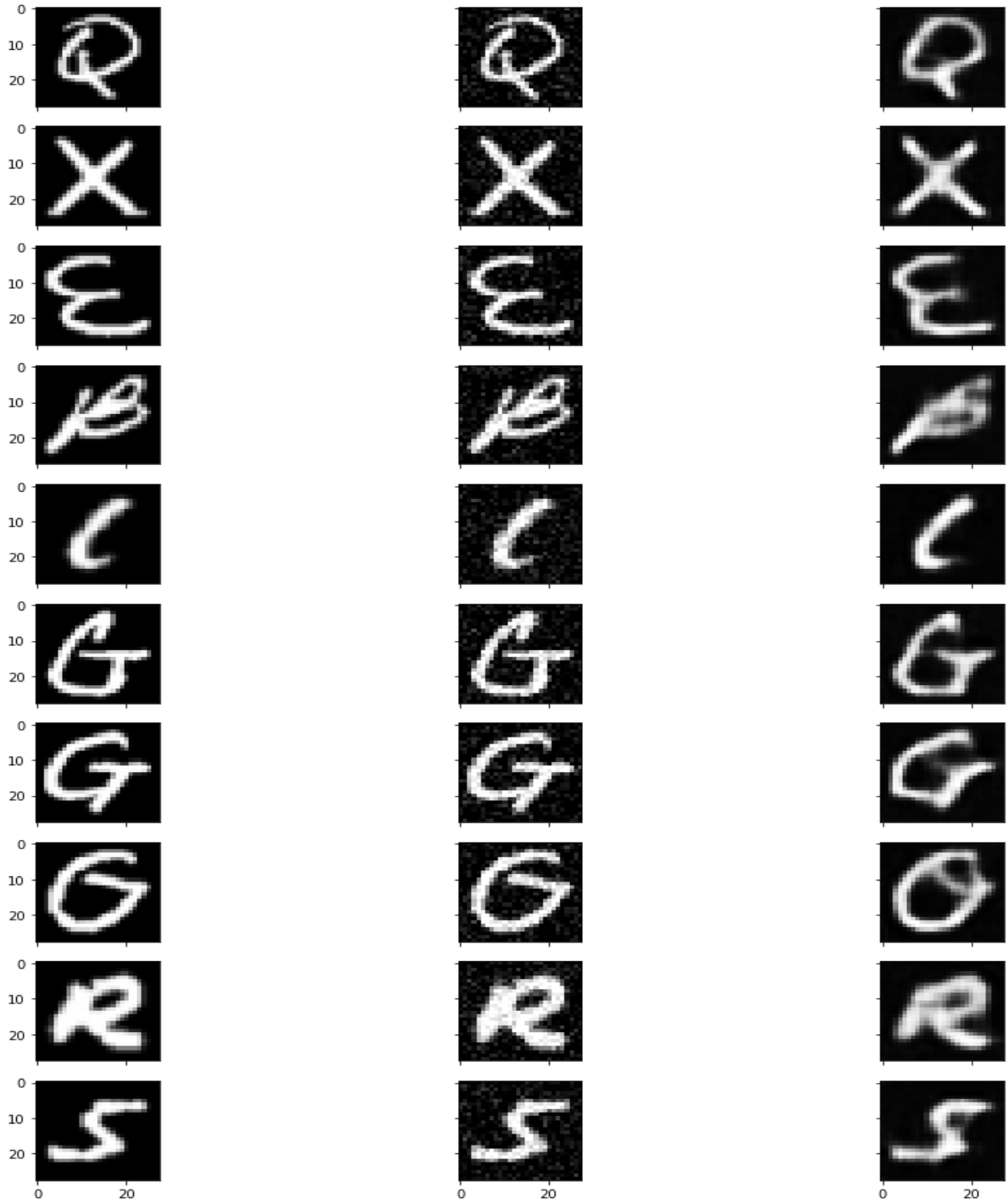


Figure 3.Architecture 3 loss

Q4. The column1 shows original images, column 2 shows noised images, column 3 shows regenerated images



Question 5. Figure 1 and figure 2 both are from different architectures, The figure 1 is so far best result with us

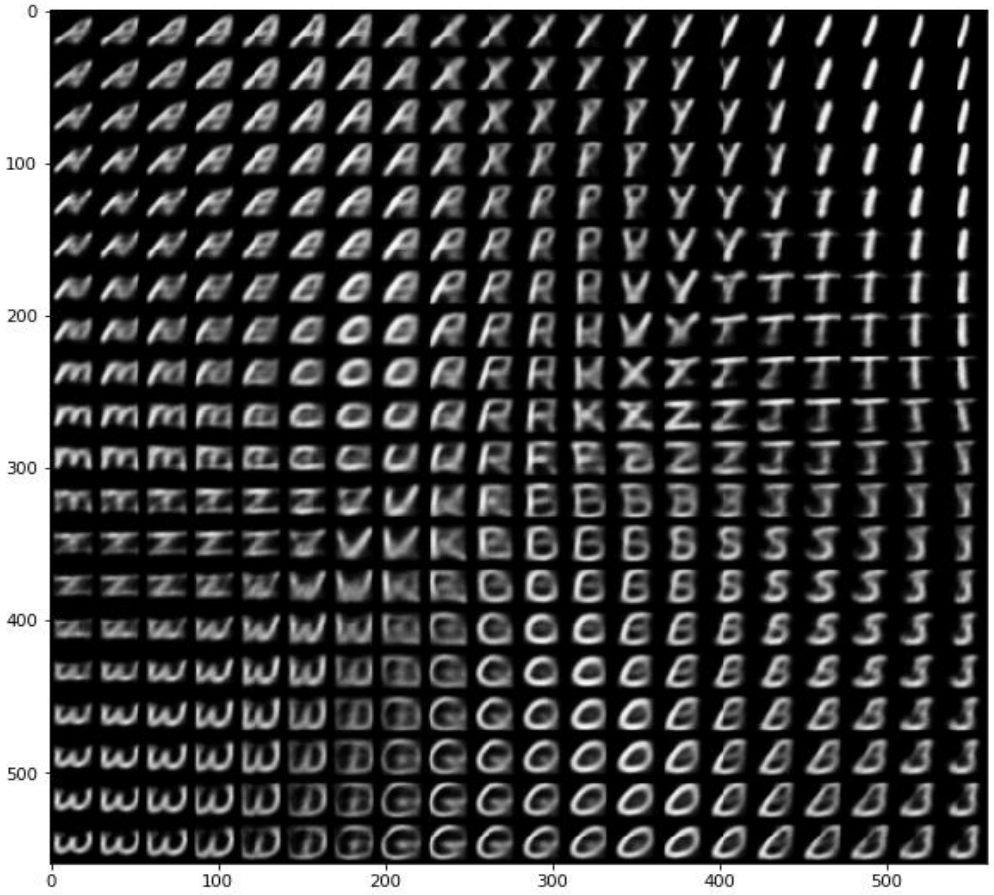


Figure 1

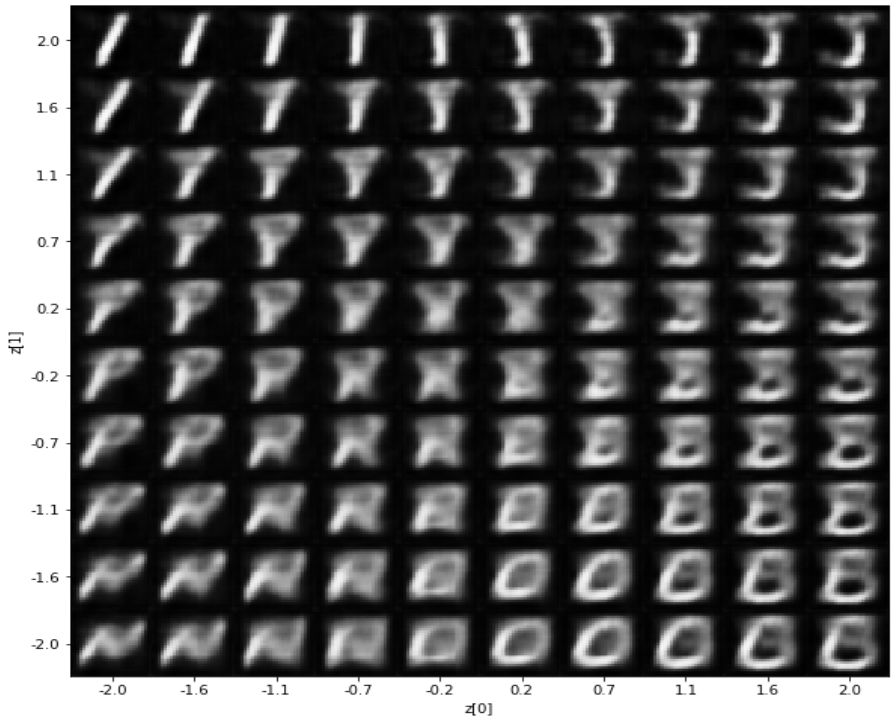


Figure 2

Question 6.