A Variational Autoencoder model is developed on CIFAR10 dataset for this problem.
For building an auto encoder there are 3 essential parts including an encoder in which maps the input (x) into a lower dimensional feature vector(z) which is latent space, and a decoder in which reconstructs output (x') from z. Then the model gets train based on comparing x and x' and optimize parameters so that increase the similarities. We do not use batch normalization in auto encoder due to the correlation it may cause for encoding and decoding, we want the encoding of each image to be independent of other images.

a)

The latent dimensionality is considered 2 for this problem which means the encoder will encode the data into 2 dimensions for latent space and must be able to decode from 2d to upper space to reconstruct the output image.

The encoder in this example consists of 7 convolutional layer to map the data to lower feature vector space z, last layer of decoder must be Flattened.

Build sampling layer input mean vector and std vector, mean and std vector will be computed from the encoder. In VAE the encoder doesn't directly map to latent space instead, it first uses the input data to output two values mu and sigma. This will prevent the model to memorize everything about the input and generalize model.

Then to build the decoder the input dimensionality must match the output of encoder, hence at the start of decoder process the latent variable is reshaped. Encoder acts like mirror of decoder so we apply transpose of convolutional layer and do up sampling for decoder unlike encoder which convolutional layer and MaxPooling was applied. In the following, summary of encoder and decoder are displayed. The encoder maps data into latent space and decoder maps from latent space into original image

[127] Model: "encoder"

Layer (type)	Output Shape	Param #	Connected to
input_20 (InputLayer)	[(None, 32, 32, 3)]	0	[]
conv2d_46 (Conv2D)	(None, 32, 32, 512)	14336	['input_20[0][0]']
max_pooling2d_38 (MaxPooling2D)	(None, 16, 16, 512)	0	['conv2d_46[0][0]']
conv2d_47 (Conv2D)	(None, 16, 16, 256)	1179904	['max_pooling2d_38[0][0]']
max_pooling2d_39 (MaxPooling2D)	(None, 8, 8, 256)	0	['conv2d_47[0][0]']
conv2d_48 (Conv2D)	(None, 8, 8, 128)	295040	['max_pooling2d_39[0][0]']
max_pooling2d_40 (MaxPooling2D)	(None, 4, 4, 128)	0	['conv2d_48[0][0]']
conv2d_49 (Conv2D)	(None, 4, 4, 64)	73792	['max_pooling2d_40[0][0]']
max_pooling2d_41 (MaxPooling2D)	(None, 2, 2, 64)	0	['conv2d_49[0][0]']
conv2d_50 (Conv2D)	(None, 2, 2, 32)	18464	['max_pooling2d_41[0][0]']
max_pooling2d_42 (MaxPooling2D)	(None, 1, 1, 32)	0	['conv2d_50[0][0]']
conv2d_51 (Conv2D)	(None, 1, 1, 16)	4624	['max_pooling2d_42[0][0]']
conv2d_52 (Conv2D)	(None, 1, 1, 8)	1160	['conv2d_51[0][0]']
flatten_5 (Flatten)	(None, 8)	0	['conv2d_52[0][0]']
z_mean (Dense)	(None, 2)	18	['flatten_5[0][0]']
z_log_var (Dense)	(None, 2)	18	['flatten_5[0][0]']
sampling_5 (Sampling)	(None, 2)	0	['z_mean[0][0]', 'z_log_var[0][0]']

Total params: 1,587,356
Trainable params: 1,587,356
Non-trainable params: 0

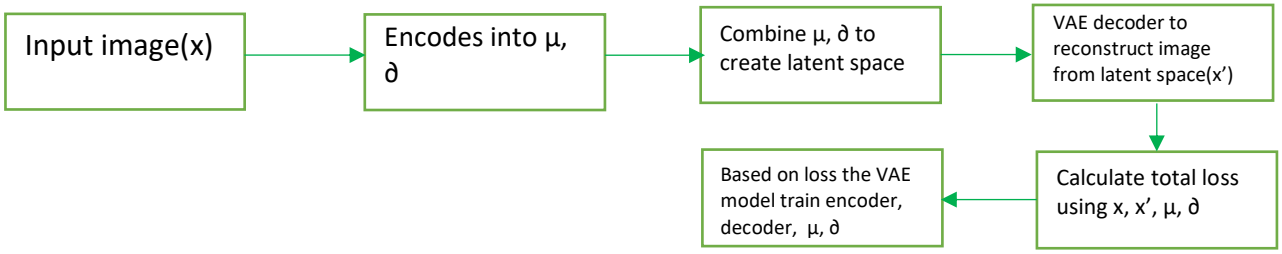
```
Model: "decoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_21 (InputLayer)       [(None, 2)]               0

 dense_23 (Dense)            (None, 8)                 24

 reshape_13 (Reshape)        (None, 1, 1, 8)           0

 conv2d_transpose_84 (Conv2D  (None, 1, 1, 8)          584
 Transpose)

 up_sampling2d_64 (UpSamplin  (None, 2, 2, 8)          0
 g2D)

 conv2d_transpose_85 (Conv2D  (None, 2, 2, 16)         1168
 Transpose)

 up_sampling2d_65 (UpSamplin  (None, 4, 4, 16)         0
 g2D)

 conv2d_transpose_86 (Conv2D  (None, 4, 4, 32)         4640
 Transpose)

 up_sampling2d_66 (UpSamplin  (None, 8, 8, 32)         0
 g2D)

 conv2d_transpose_87 (Conv2D  (None, 8, 8, 64)         18496
 Transpose)

 up_sampling2d_67 (UpSamplin  (None, 16, 16, 64)       0
 g2D)

 conv2d_transpose_88 (Conv2D  (None, 16, 16, 128)      73856
 Transpose)

 up_sampling2d_68 (UpSamplin  (None, 32, 32, 128)      0
 g2D)

 conv2d_transpose_89 (Conv2D  (None, 32, 32, 3)        3459
 Transpose)

=================================================================
Total params: 102,227
Trainable params: 102,227
Non-trainable params: 0
_________________________________________________________________
```

b)

Then we build VEA model with decoder and encoder and calculating reconstruction loss, **Kullback-Leibler Divergence** (KL) loss and the total loss which is sum of two previous mentioned loss values and train the weights. Reconstruction loss is the difference between original image(x) and reconstructed image(x') so that higher values for this loss depicts the lower quality reconstructed image so the goal is to minimize the reconstruction loss.
KL loss is to build generalized latent space which came from learning mu and sigma.
Then we compile the model using 'adam' as optimizer running 100 epochs with batch size as 128. Below is the first few epoch values.

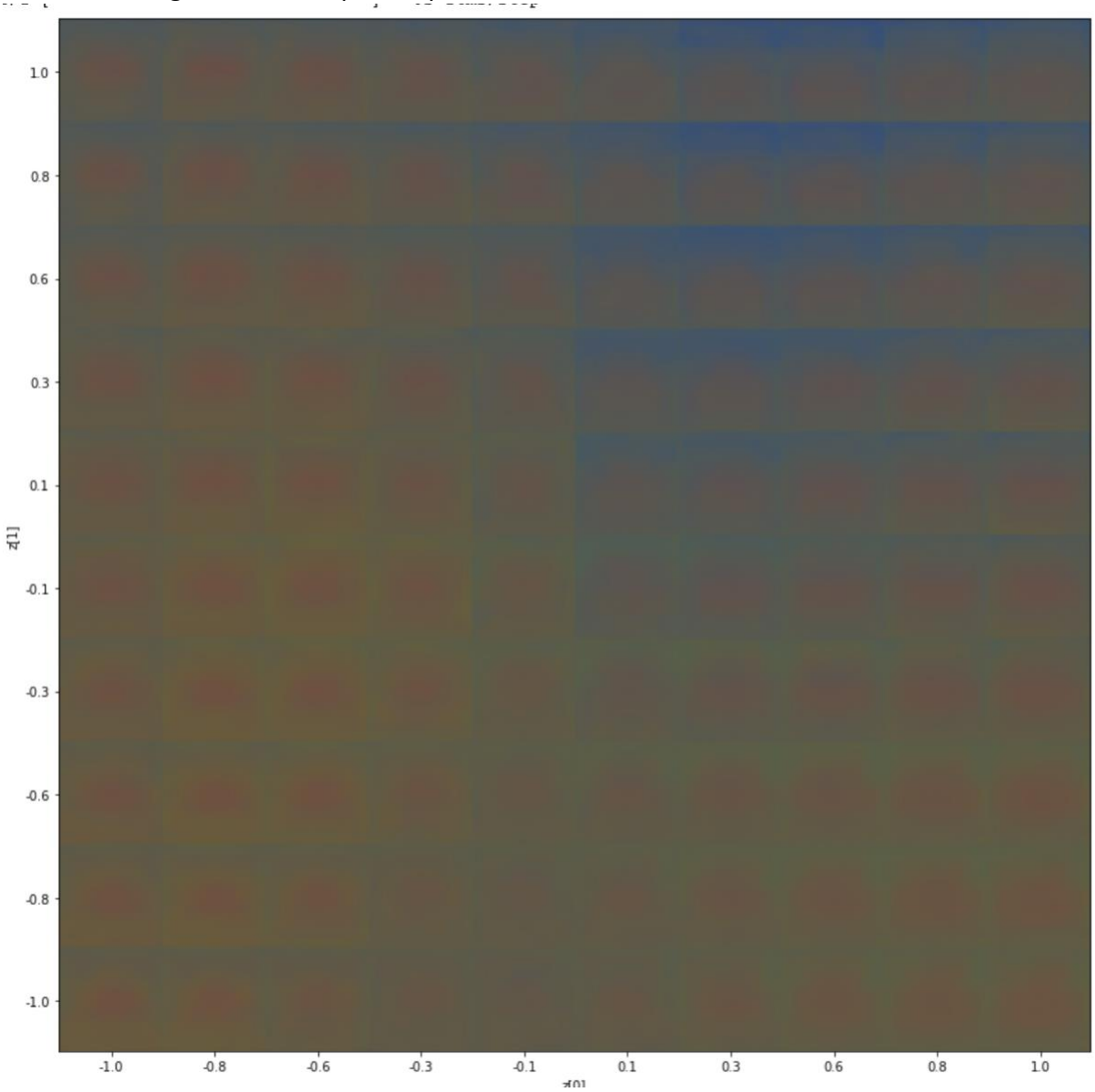
```
(50000, 32, 32, 3)
Epoch 1/100
469/469 [==============================] - 29s 58ms/step - loss: 684.8542 - reconstruction_loss: 665.1818 - kl_loss: 3.5802
Epoch 2/100
469/469 [==============================] - 27s 57ms/step - loss: 655.6526 - reconstruction_loss: 651.0784 - kl_loss: 4.1630
Epoch 3/100
469/469 [==============================] - 28s 60ms/step - loss: 654.7843 - reconstruction_loss: 650.5775 - kl_loss: 4.2807
Epoch 4/100
469/469 [==============================] - 27s 57ms/step - loss: 655.0215 - reconstruction_loss: 650.2675 - kl_loss: 4.4618
Epoch 5/100
469/469 [==============================] - 27s 57ms/step - loss: 654.7099 - reconstruction_loss: 650.2840 - kl_loss: 4.5700
Epoch 6/100
469/469 [==============================] - 27s 57ms/step - loss: 655.0526 - reconstruction_loss: 650.3416 - kl_loss: 4.7346
Epoch 7/100
469/469 [==============================] - 27s 57ms/step - loss: 655.8075 - reconstruction_loss: 651.5242 - kl_loss: 4.8163
Epoch 8/100
469/469 [==============================] - 26s 56ms/step - loss: 655.9978 - reconstruction_loss: 652.5031 - kl_loss: 4.7539
Epoch 9/100
469/469 [==============================] - 26s 56ms/step - loss: 658.3849 - reconstruction_loss: 652.7234 - kl_loss: 4.6027
Epoch 10/100
469/469 [==============================] - 26s 56ms/step - loss: 655.5945 - reconstruction_loss: 650.5509 - kl_loss: 4.6947
Epoch 11/100
469/469 [==============================] - 26s 56ms/step - loss: 655.0189 - reconstruction_loss: 649.9813 - kl_loss: 4.7990
Epoch 12/100
469/469 [==============================] - 26s 56ms/step - loss: 655.7752 - reconstruction_loss: 650.8139 - kl_loss: 4.9702
Epoch 13/100
469/469 [==============================] - 26s 56ms/step - loss: 657.6267 - reconstruction_loss: 651.8823 - kl_loss: 4.6652
Epoch 14/100
469/469 [==============================] - 26s 56ms/step - loss: 655.5377 - reconstruction_loss: 650.5037 - kl_loss: 4.3411
Epoch 15/100
469/469 [==============================] - 26s 56ms/step - loss: 655.0706 - reconstruction_loss: 650.6757 - kl_loss: 4.4806
Epoch 16/100
469/469 [==============================] - 26s 56ms/step - loss: 657.5882 - reconstruction_loss: 653.2723 - kl_loss: 4.9680
Epoch 17/100
469/469 [==============================] - 26s 56ms/step - loss: 657.2949 - reconstruction_loss: 651.4780 - kl_loss: 4.7242
Epoch 18/100
469/469 [==============================] - 26s 56ms/step - loss: 654.5868 - reconstruction_loss: 649.9915 - kl_loss: 4.3531
```

The process is like following diagram:



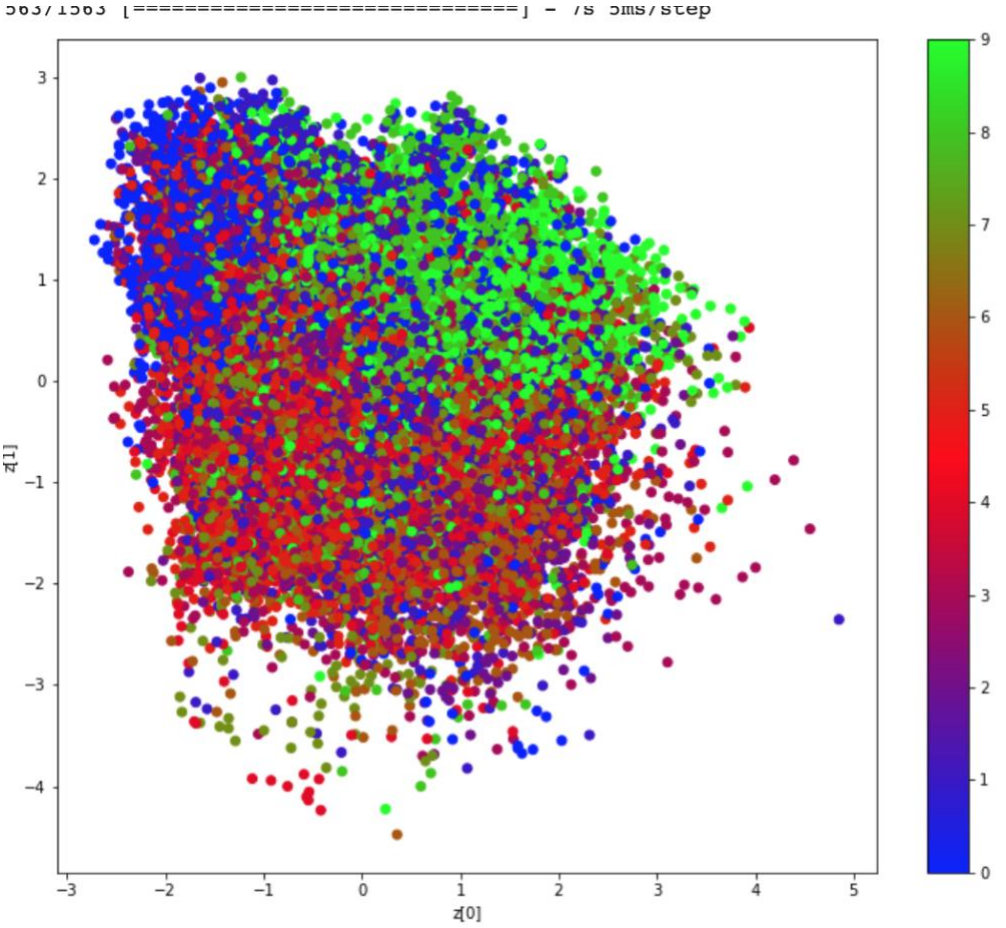
This process runs multiple times so that VAE model optimize itself in each run by encode and decode images.

Then a 10X10 grid of latent space is plotted as below:



Also different classes as clusters in latent space is plotted.
Color bar coded as below:

```
y_labels = {0: 'airplane',1: 'automobile', 2: 'bird', 3: 'cat', 4: 'deer',
5: 'dog', 6: 'frog', 7: 'horse', 8: 'ship', 9: 'truck'}
```



c)

VAE can generate new images besides encoding and decoding and that is resulted by picking random values for latent space as input to VAE decoder. The generated images are constructed from random latent space so they might not be understandable.

Below is 10 new fake images:

Sources:

1. https://keras.io/examples/generative/vae/
2. SJSU Data 255 - Deep learning - Dr. Mohammad Masum 'AE.ipynb'
3. https://becominghuman.ai/variational-autoencoders-simply-explained-46e6f97947ed