Lab 7 : Practic Autoencoder and Variational Autoencoders

Function API

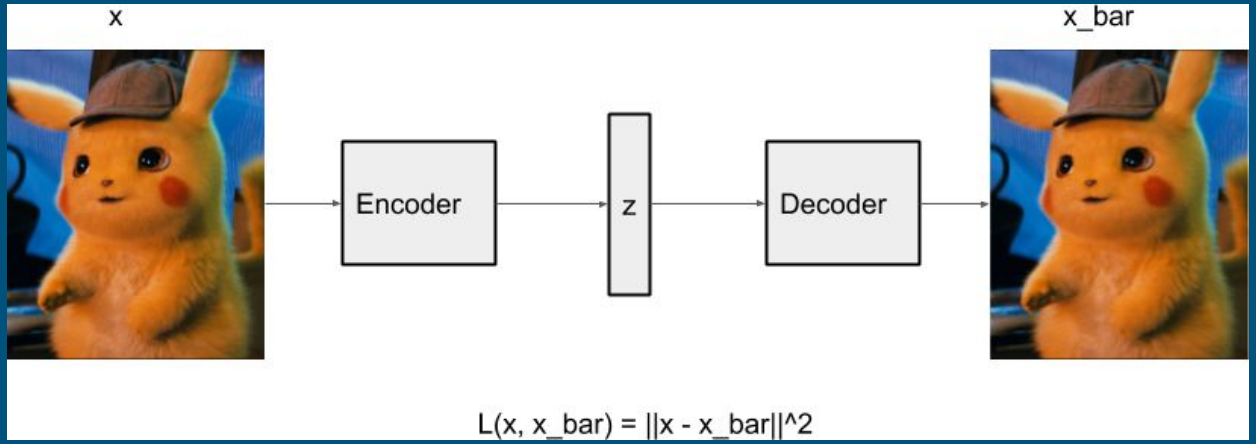
```
input_img = layers.Input(shape=input_shape)

x = layers.Conv2D(32,3,padding='same',activation='relu')(input_img)
x = layers.Conv2D(32,3,padding='same',strides=(2,2),activation='relu')(x)
x = layers.Conv2D(64,3,padding='same',activation='relu')(x)
x = layers.Conv2D(64,3,padding='same',activation='relu')(x)
```

No Function API

```
cnn.add(layers.Conv2D(32,kernel_size=(3, 3), padding="same",input_shape=(32, 32, 3), activation = 'relu'))
cnn.add(layers.Conv2D(32, kernel_size=(3, 3), padding="same",activation='relu'))
cnn.add(layers.Conv2D(64, kernel_size=(3, 3), padding="same",activation='relu'))
cnn.add(layers.Conv2D(64, kernel_size=(3, 3), padding="same",activation='relu'))
```

Autoencoder



https://medium.com/analytics-vidhya/building-a-convolutional-autoencoder-using-keras-using-conv2dtranspose-ca403c8d144e

Encoder

```
inputs = Input(shape=(28, 28,1))
x = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs) #28 x 28 x 32
x = MaxPooling2D((2, 2))(x) #14 x 14 x 32
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x) #14 x 14 x 32
encoded = MaxPooling2D((2, 2))(x) #7 x 7 x 32
# At this point the representation is (7, 7, 32)
```

Decoder

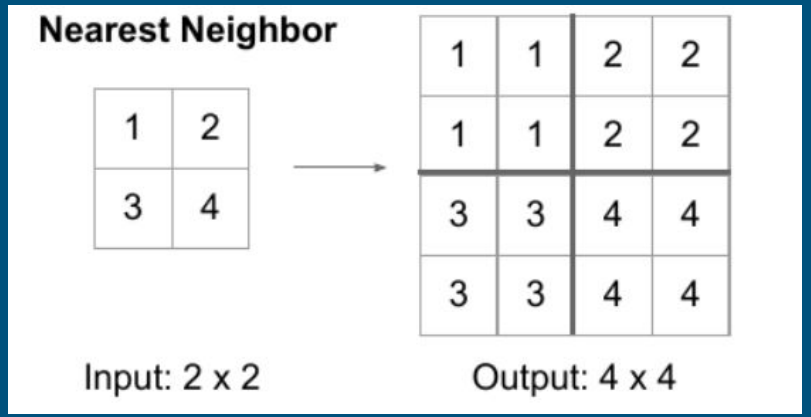
```
x = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded) #7 x 7 x 32
x = UpSampling2D((2, 2))(x)  #14 x 14 x 32
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x) # 14 x 14 x 32
x = UpSampling2D((2, 2))(x) #28 x 28 x 32
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x) #28 x 28 x 1
```

Upsampling

`keras.layers.UpSampling2D(size)`

Size = The upsampling factors for rows and columns.

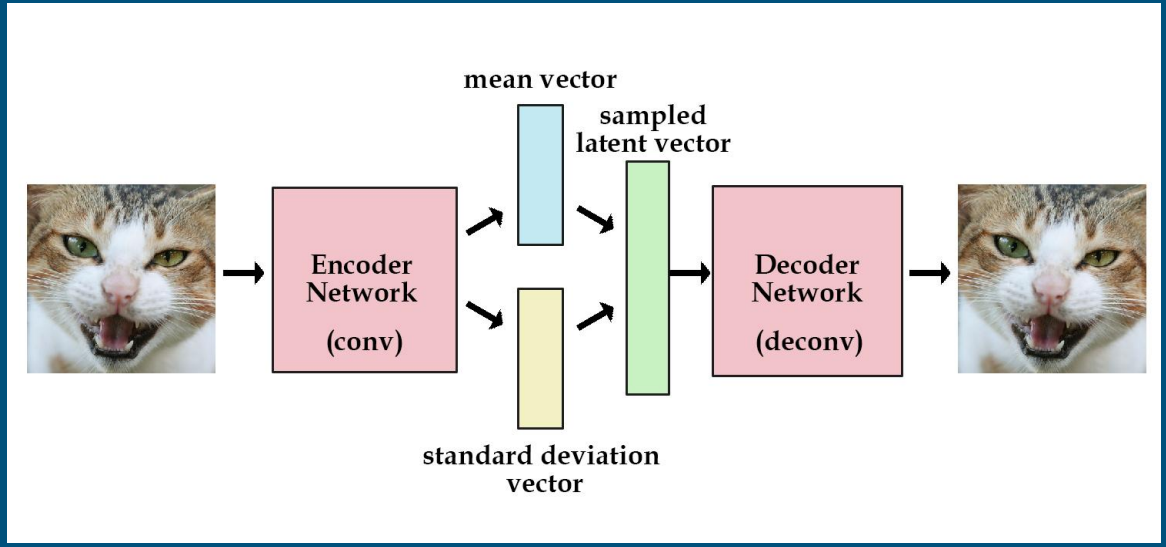
```
x = UpSampling2D((2, 2))(x)  #14 x 14 x 32
```



Autoencoder

```
autoencoder = Model(inputs , decoded)
```

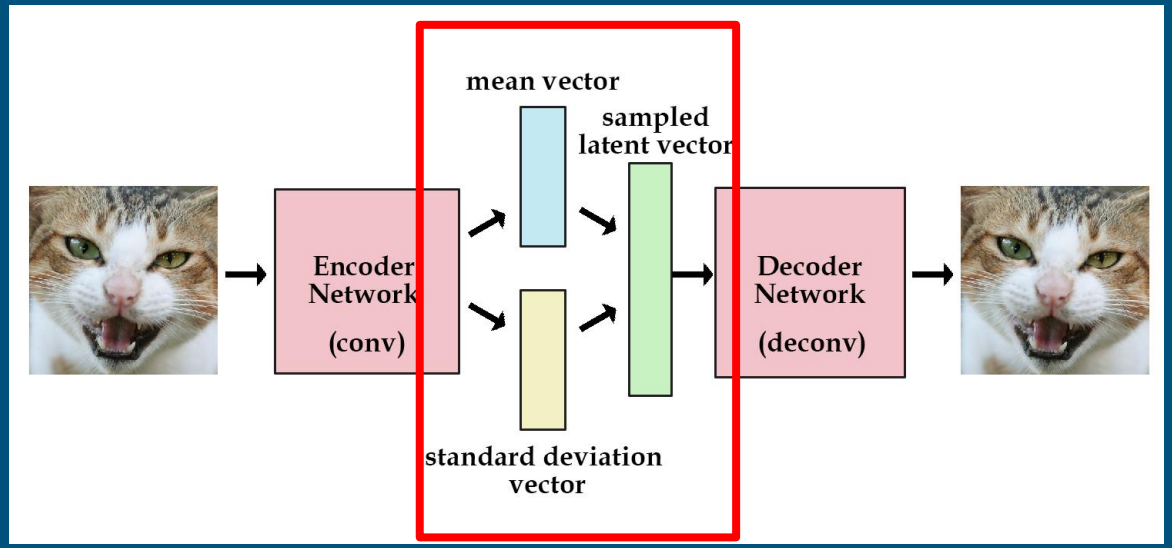
VAE



http://kvfrans.com/variational-autoencoders-explained/

latent space

```
def sampling(arg):
    z_mean, z_log_var = arg
    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])
```

Tensorboard

```
history = vae.fit(x_train,x_train,shuffle=True,
                  epochs=10,
                 validation_data=(x_val,x_val),
                  verbose=False,
                 callbacks=[tensorboard])
```

Tensorboard