1. Make a brief introduction about variational autoencoder (VAE). List one advantage comparing with vanilla autoencoder and one problem of VAE.

   VAE learns probability distribution of the data whereas autoencoders learns a function to map each input to a number and decoder learns the reverse mapping. VAE encodes the data to be Gaussian distribution instead of a single point so that it can express the latent regularization.

   1) Compared to vanilla autoencoder which the latent space is not continuous, VAE can avoid overfitting because it encodes data as a distribution over the latent space, this property also ensures the latent space to be able to enable generative process.
   2) However, VAE reconstructs data by sampling from the latent distribution, it normally generates a much more blurred images than those images from GAN. In addition, The KL-divergence component of the VAE loss function may cause an under-exploitation of the network capacity since it typically induces a parsimonious use of latent variables which may be altogether neglected by the decoder.

2. Train a fully connected autoencoder and adjust at least two different element of the latent representation

Model Architecture

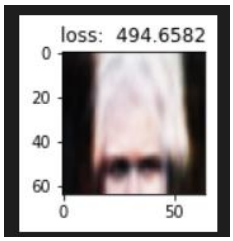
```
class fcn_autoencoder(nn.Module):
    def __init__(self):
        super(fcn_autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(64 * 64 * 3, 1024),
            nn.ReLU(),
            # nn.Linear(12, 24),
            # nn.ReLU(),
        )

        self.decoder = nn.Sequential(
            # nn.Linear(24, 12),
            # nn.ReLU(),
            nn.Linear(1024, 64 * 64 * 3),
            nn.Tanh()
        )

    def forward(self, x):
        # print(x.shape)
        # raise
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

```
tf = transforms.Compose([
            transforms.Lambda(lambda x: x.to(torch.float32)),
            transforms.Lambda(lambda x: crop(x,10,0,28,64)),
            transforms.Resize((64,64)),
            transforms.Lambda(lambda x: 2. * x/255. - 1.),
           ])
```

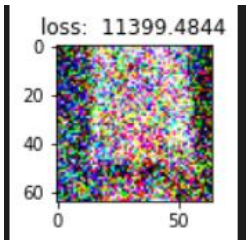
Original: Original reconstructed:



Latent shape: torch.Size([1, 1024])

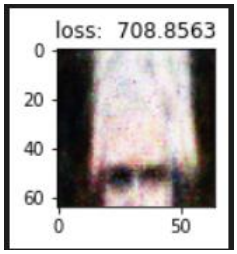
1. Latent + 0.5: A lot of RGB Points emerge after the adjustment.

```
latent = model.encoder(data)
latent = latent + 0.5
out_array = model.decoder(latent)
```



2. latent[:,3], latent[:,1000] = 2, 8: Some noise points emerge after the adjustment

```
latent = model.encoder(data)
latent[:,3], latent[:,1000] = 2, 8
out_array = model.decoder(latent)
```



Reference:

https://link.springer.com/article/10.1007/s42979-021-00702-9#Sec4

https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

https://www.quora.com/Whats-the-difference-between-a-Variational-Autoencoder-VAE-and-an-Autoencoder