Report

R10945001
陳欣頤

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

   VAE 在編碼過程增加了一些噪音，透過 mean 和 standard deviation 進行參數化，使生成的向量遵從高斯分佈，因此 VAE 可以控制想要生成的圖片:
   輸出 mean、standard deviation 與 normal distribution 產生的三個向量
   ⇨ exp(standard deviation) x normal distribution + mean

   advantage: VAE 透過高斯分佈對目前的編碼方式進行改進，成為連續的編碼，因此在高斯分佈所示範圍內的圖片都有可能被採樣到，decoder 也能在訓練時把這些影像都盡可能還原成和原圖相似的圖片。

   problem: VAE 較適用於符合標準常態分佈的隱變量，且由於它沒有對抗網路的抗衡，因此可能產生較為模糊的圖像。

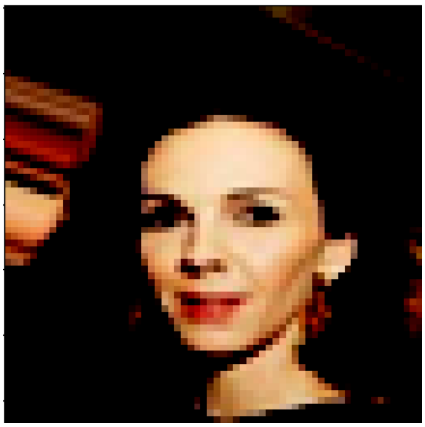
2. Train a fully connected autoencoder and adjust at least two different element of the latent representation. Show your model architecture, plot out the original image, the reconstructed images for each adjustment and describe the differences.

model architecture :

```
class multi_fcn_autoencoder(nn.Module):
    def __init__(self):
        super(multi_fcn_autoencoder, self).__init__()
        self.encoder1 = nn.Sequential(
            nn.Linear(64 * 64 * 3, 1280),
            nn.ReLU(),
            nn.Linear(1280, 640),
            nn.ReLU(),
            nn.Linear(640, 360),
            nn.ReLU(),
            nn.Linear(360, 198)
        )

        self.encoder2 = nn.Sequential(
            nn.Linear(64 * 64 * 3, 1920),
            nn.ReLU(),
            nn.Linear(1920, 512),
            nn.ReLU(),
            nn.Linear(512, 300),
            nn.ReLU(),
            nn.Linear(300, 198)
        )

        self.encoder3 = nn.Sequential(
            nn.Linear(64 * 64 * 3, 640),
            nn.ReLU(),
            nn.Linear(640, 320),
            nn.ReLU(),
            nn.Linear(320, 198)
        )

        self.decoder = nn.Sequential(
```

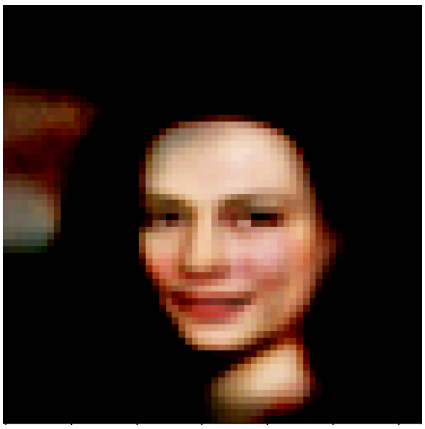
```
            nn.Linear(3*66, 360),
            nn.ReLU(),
            nn.Linear(360, 640),
            nn.ReLU(),
            nn.Linear(640, 1280),
            nn.ReLU(),
            nn.Linear(1280, 64 * 64 * 3),
            nn.Tanh()
        )

    def forward(self, x):
        x1 = self.encoder1(x)
        x2 = self.encoder2(x)
        x3 = self.encoder3(x)
        x = torch.cat((x1, x2, x3), 0)
        x = self.decoder(x)
        a, b, c = x.chunk(3, 0)

        return (a + b + c)/3
```

original image:

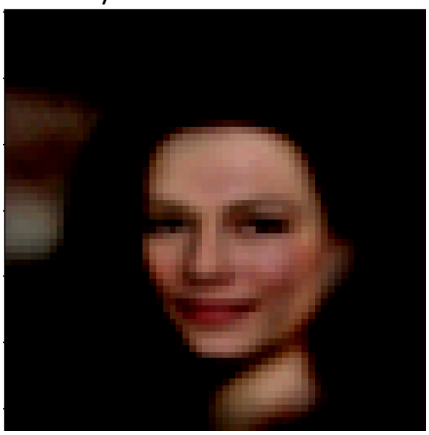


reconstructed image:

1. 將 encoder 的結果直接丟入 decoder

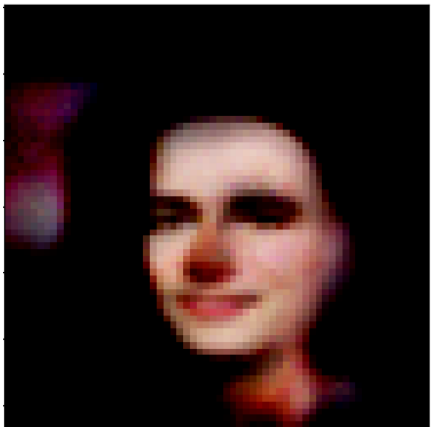


2. 將 encoder 的結果除以 2 再丟入 decoder
   x= x/2



與上圖 1 相比，影像的亮度變暗

3. 將 encoder 結果的最後一維放大 20 倍再丟入 decoder
   x[:,197] = x[:,197]*20



與上圖 1 相比，眼睛輪廓的地方變深了(不同的 latent representation 對應不同特徵的改變)