

Machine Learning HW7 Report

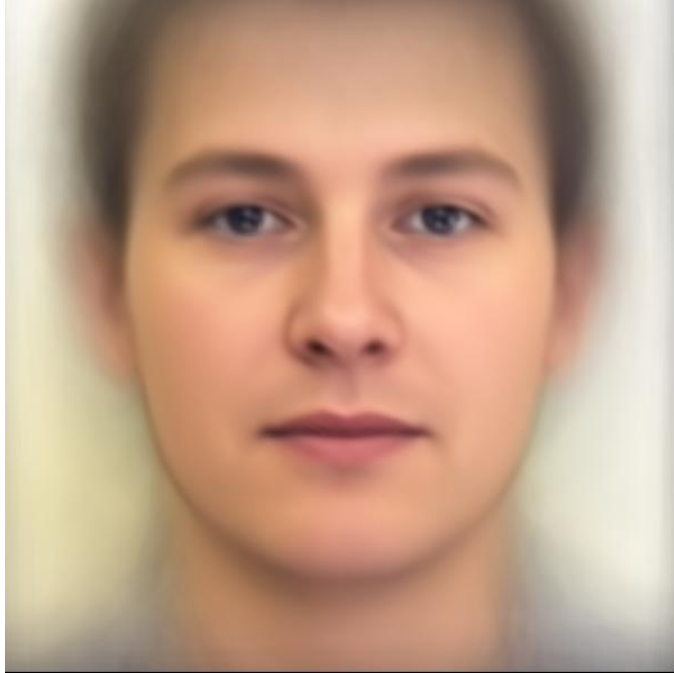
學號：B05901111

系級：電機三

姓名：陳建成

1. PCA of color faces:

- a. 請畫出所有臉的平均。



- b. 請畫出前五個 Eigenfaces，也就是對應到前五大 Eigenvalues 的 Eigenvectors。



- c. 請從數據集中挑出任意五張圖片，並用前五大 Eigenfaces 進行 reconstruction，並畫出結果。

12, 18, 48, 57, 73



- d. 請寫出前五大 Eigenfaces 各自所佔的比重，請用百分比表示並四捨五入到小數點後一位。

4.1%, 2.9%, 2.4%, 2.2%, 2.1%

2. Image clustering:

- a. 請實作兩種不同的方法，並比較其結果(reconstruction loss, accuracy)。
(不同的降維方法或不同的 cluster 方法都可以算是不同的方法)

因為感覺用 VAE 去 reconstruct 比較困難，而 cluster 或是 PCA 似乎比較準確，因此我是都用上傳 Kaggle 的 VAE 去做 encoding，然後 latent 的 96 維向量中，使用不同的方法做降維 / cluster。

I. 96 維 —[PCA]→ 48 維 —————[K-Means]————→ 2 個 cluster 的 label

II. 96 維 —[PCA]→ 2 維 —[比較第一維與第二維的值]→ 2 個 cluster 的 label

reconstruction loss: 578.9316

其算法是以 VAE 的 loss，除了本身每個 pixel 的 BCE loss 之外還加上 KL divergence 的 loss。因為兩個方法是用同樣的 AE 因此 reconstruction loss 相同。

```
xent_loss = 32 * 32 * metrics.binary_crossentropy(
    K.flatten(input_img), K.flatten(output_img))
kl_loss = - 0.5 * K.sum(1 + z_log_var -
    K.square(z_mean) - K.exp(z_log_var), axis=-1)
vae_loss = K.mean(xent_loss + kl_loss)
```

accuracy (private, public):

I. 0.95375, 0.95377 (Kaggle 上面)

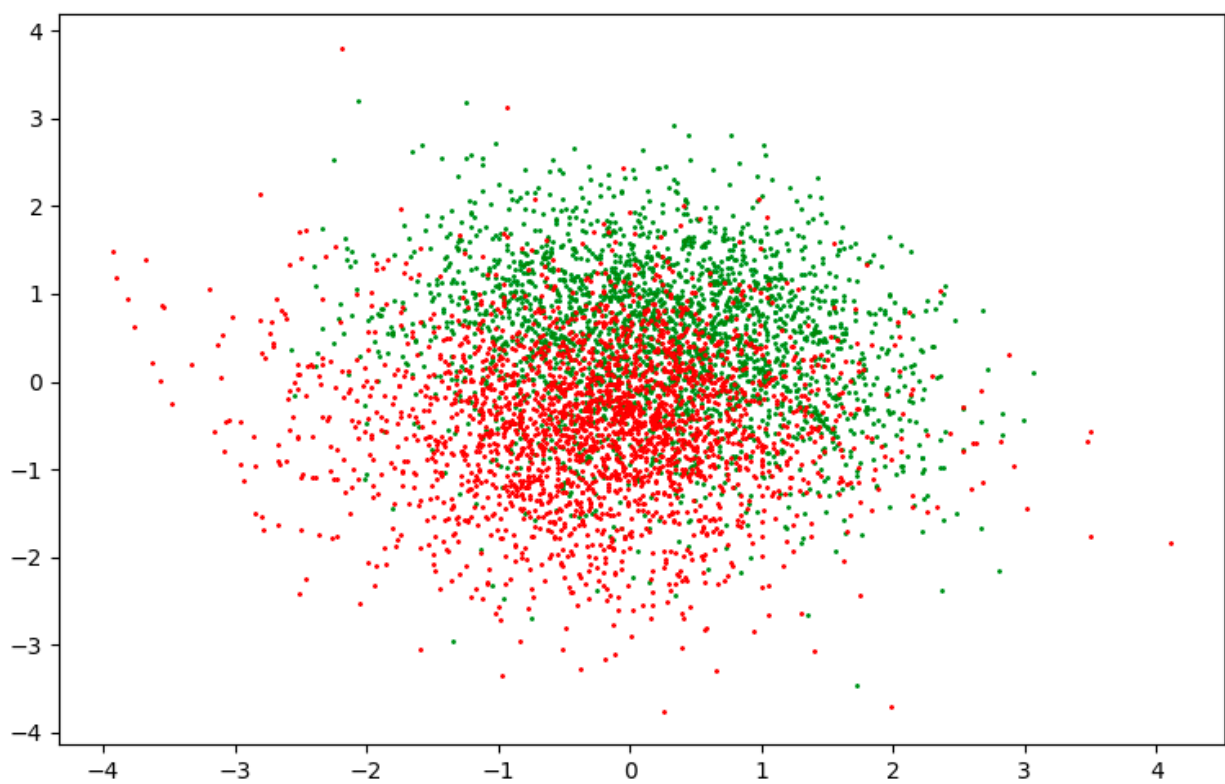
II. 0.50087, 0.49944

- b. 預測 visualization.npy 中的 label，在二維平面上視覺化 label 的分佈。
(用 PCA, t-SNE 等工具把你抽出來的 feature 投影到二維，或簡單的取前兩維 2 的 feature)
其中 visualization.npy 中前 2500 個 images 來自 dataset A，後 2500 個 images 來自 dataset B，比較和自己預測的 label 之間有何不同。

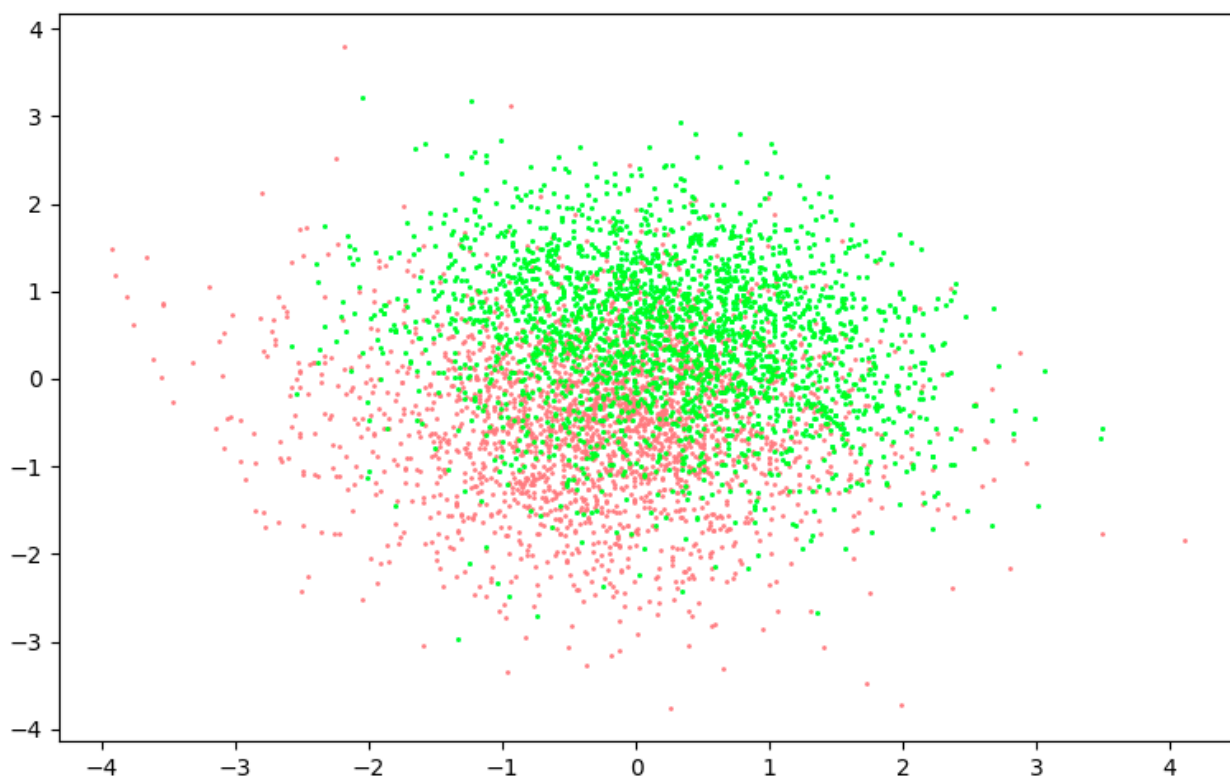
把 visualization 的結果放入 encoder 進行 label prediction 之後，可以得到一個預測的結果，跟 ground truth 比較後發現其正確率為 97.22%。另外，dataset A 被預測有 2539 張而 dataset B 有 2461 張，看來整套 cluster 方法比較偏好預測為 dataset A。

下面兩張視覺化分別是根據 ground truth 和 predicted label，直接經過 PCA 去降維的結果。紅色/皮膚色的是 dataset A，綠色/亮綠色的是 dataset B。

Ground truth label:



Predicted label:



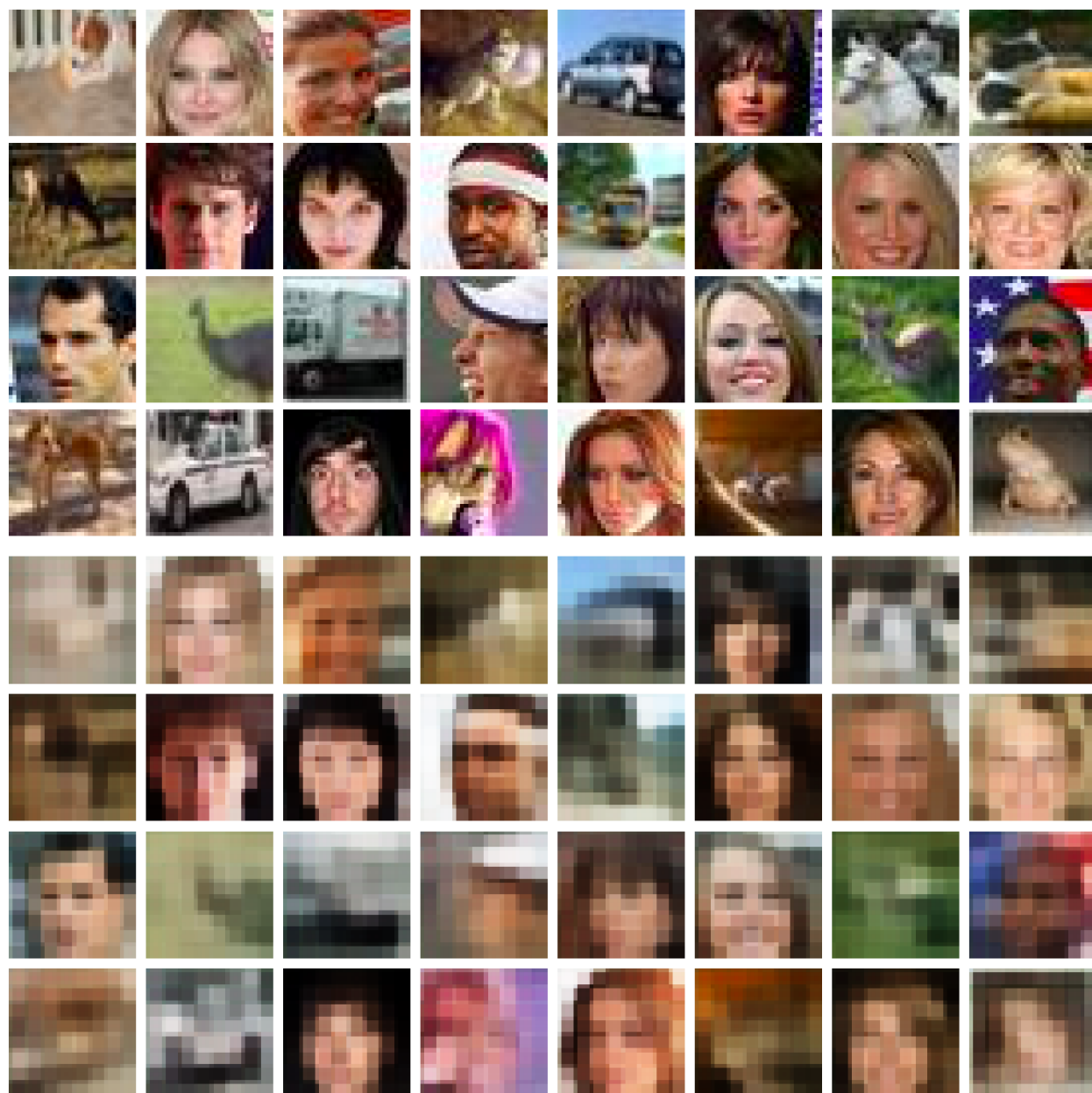
- c. 請介紹你的 model 架構(encoder, decoder, loss function...), 並選出任意 32 張圖片, 比較原圖片以及用 decoder reconstruct 的結果。

以下是 VAE model 架構, 其中到 lambda 層是 encoder, 後面是 decoder, 整個 model 是 jointly trained。Loss function 採用 VAE 的式子實作 (code 前面有附在 a. 小題, 變數與 GitHub 裡的 code 稍有不同)。

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 32, 32, 3)	0	
conv2d (Conv2D)	(None, 32, 32, 3)	39	input_1[0][0]
conv2d_1 (Conv2D)	(None, 16, 16, 64)	832	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 8, 8, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 8, 8, 48)	27696	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 8, 8, 16)	6928	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 16)	0	conv2d_3[0][0]
flatten (Flatten)	(None, 256)	0	max_pooling2d_1[0][0]
dense (Dense)	(None, 300)	77100	flatten[0][0]
dense_1 (Dense)	(None, 125)	37625	dense[0][0]
dense_2 (Dense)	(None, 96)	12096	dense_1[0][0]
dense_3 (Dense)	(None, 96)	12096	dense_1[0][0]
lambda (Lambda)	(None, 96)	0	dense_2[0][0] dense_3[0][0]
dense_4 (Dense)	(None, 125)	12125	lambda[0][0]
dense_5 (Dense)	(None, 300)	37800	dense_4[0][0]
dense_6 (Dense)	(None, 256)	77056	dense_5[0][0]
reshape (Reshape)	(None, 4, 4, 16)	0	dense_6[0][0]
up_sampling2d (UpSampling2D)	(None, 8, 8, 16)	0	reshape[0][0]
conv2d_transpose (Conv2DTranspo	(None, 8, 8, 48)	6960	up_sampling2d[0][0]
conv2d_transpose_1 (Conv2DTrans	(None, 8, 8, 64)	27712	conv2d_transpose[0][0]
up_sampling2d_1 (UpSampling2D)	(None, 16, 16, 64)	0	conv2d_transpose_1[0][0]
conv2d_transpose_2 (Conv2DTrans	(None, 32, 32, 3)	771	up_sampling2d_1[0][0]
conv2d_transpose_3 (Conv2DTrans	(None, 32, 32, 3)	39	conv2d_transpose_2[0][0]
Total params: 336,875			
Trainable params: 336,875			
Non-trainable params: 0			

總之就是怎麼把維度縮下去的就怎麼長回來，幾乎都是對稱的，然後 padding 都用'same'處理。Lambda 層則是分成 mean 跟 variance 處理。

再來是原圖和 reconstructed 的圖片比較，因為隨機選取容易只選到 Celeb A 的圖，因此直接以前 32 張進行比較。



感覺是人臉的感覺比較有 reconstruct 回來，但東西的部分反而幾乎都明顯模糊掉了。不過感覺大概還足以看得出 dataset 的差異。