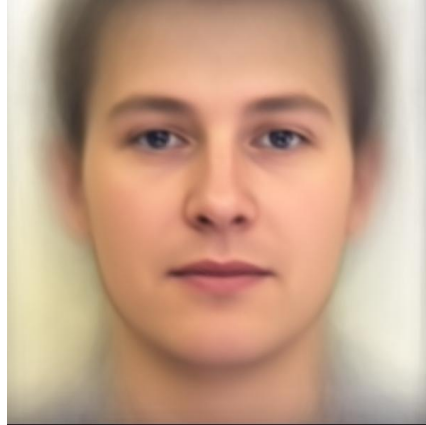


# Machine Learning HW7 Report





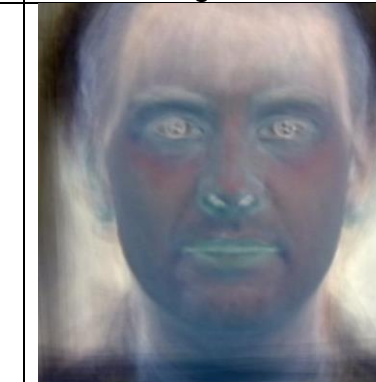
學號：R07942091 系級：電信碩一 姓名：許博閔

## 1. PCA of color faces:

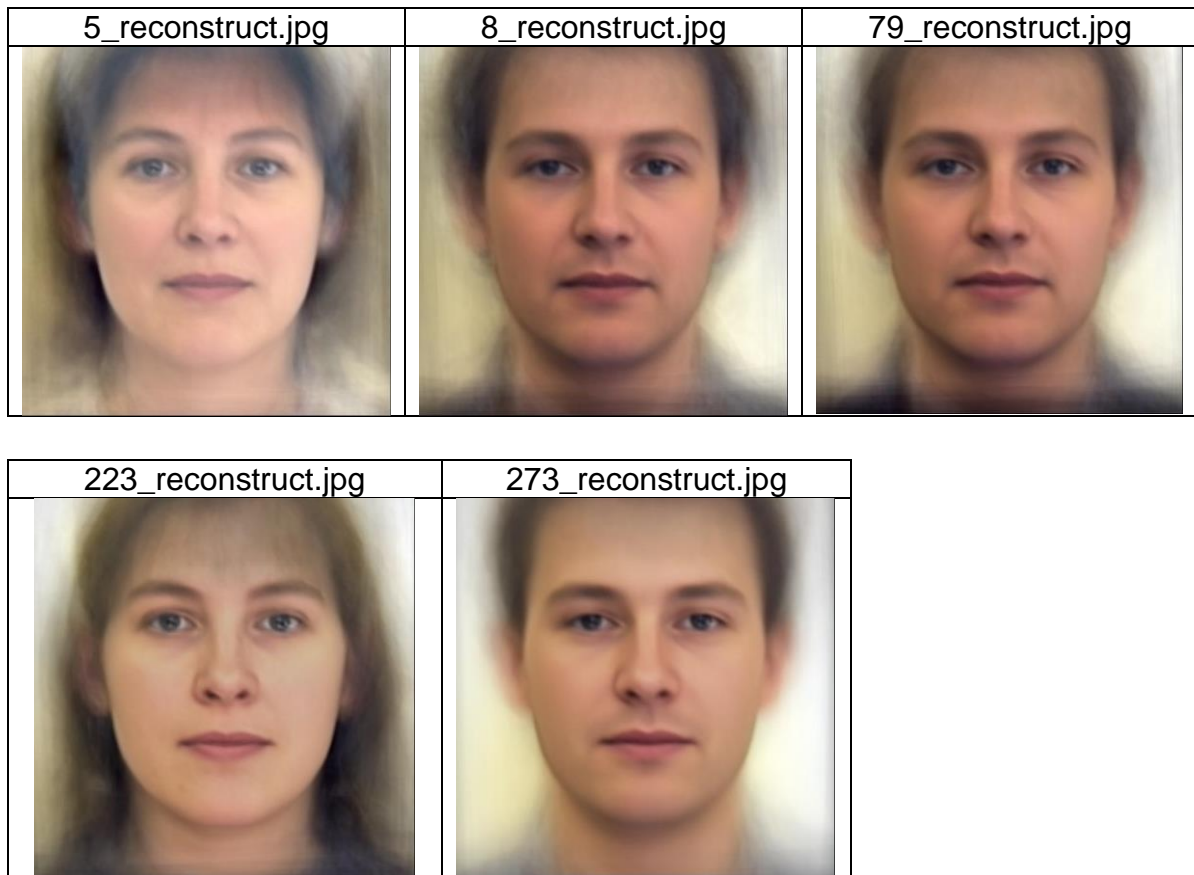
a. 請畫出所有臉的平均。



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

1 Eigenfaces		2 Eigenfaces		3 Eigenfaces	
					
4 Eigenfaces		5 Eigenfaces			
					

- 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。



- 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。

1 Eigenfaces : 21.5%  
 2 Eigenfaces : 2.7%  
 3 Eigenfaces : 2.4%  
 4 Eigenfaces : 1.8%  
 5 Eigenfaces : 1.8%

## 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)  
 我的兩種方法差在 cluster 的方法，第一中用 kmean，第二種用 Agglomerative clustering，兩者皆用 sklearn 來達成。

Reconstruction loss: 兩者相同，我用 `nn.L1Loss` 來計算，平均每張圖的 loss 為 0.03565

Accuracy :

Kmean : kaggle public : 0.97060 , kaggle private : 0.97037

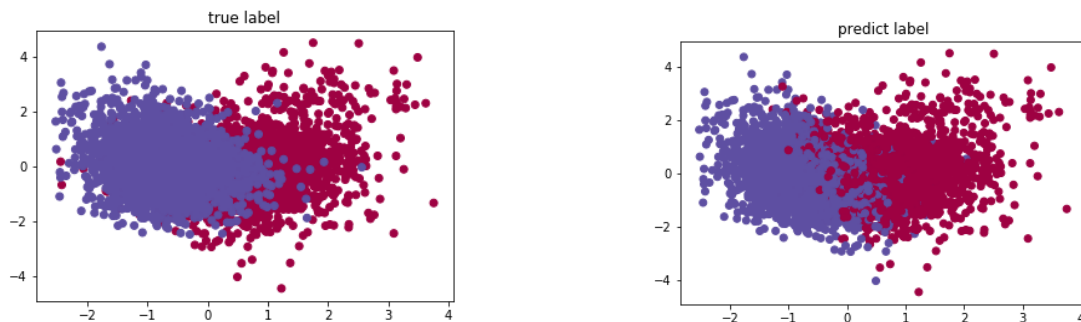
Agglomerative clustering:kaggle public : 0.88816 , kaggle private :0.88669

- 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。

我的 model 預測出來的結果正確率為 79.66%

下面是用 PCA 將抽出來的 feature 投影到二維和 label 作圖的結果



左圖為 feature 和 true label 作圖的結果，右圖為 feature 和 model predict label 作圖的結果。

比較後可看出判斷錯誤的點大多集中在中間的部分，以及一些零星散落的點，錯誤主要集中在中間應該是合理的現象，因為用 kmean 分群的方法下，這些點可能因為一點微小的差距就被分類到錯誤的 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。

我的 model 用 VAE 來實作，encoder 用 convolution，將 convolution 後的結果通過一層 FC 後得到維度為 100 的 latent code，decoder 則是 Transposed convolution，用來重建圖片。

Loss function : 我用 L1 loss 來計算圖片 reconstruct loss，並計算 latent code 和 gaussian distribution 的 KL divergence loss，將兩者相加就是 model 最終的 loss。

以下是 model 的完整架構

```

VAE(
  (conv_stage): Sequential(
    (0): Conv2d(3, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01, inplace)
    (3): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): LeakyReLU(negative_slope=0.01, inplace)
    (6): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): LeakyReLU(negative_slope=0.01, inplace)
    (9): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (10): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): LeakyReLU(negative_slope=0.01, inplace)
  )
  (fcMean): Linear(in_features=1024, out_features=100, bias=True)
  (fcStd): Linear(in_features=1024, out_features=100, bias=True)
  (fcDecode): Linear(in_features=100, out_features=1024, bias=True)
  (trans_conv_stage): Sequential(
    (0): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01, inplace)
    (3): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): LeakyReLU(negative_slope=0.01, inplace)
    (6): ConvTranspose2d(64, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): LeakyReLU(negative_slope=0.01, inplace)
    (9): ConvTranspose2d(32, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
  )
)

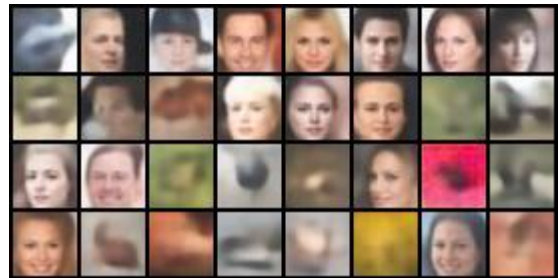
```

因為 VAE 的關係，**reconstruct** 的圖片都會變模糊，但通常人臉重建的圖片比較成功，另一個 **dataset** 重建的圖片有時會看不出原本是什麼東西。

原始圖片



重建圖片



P.S. 因為 PCA 是用 sklearn 做的，萬一 **data shuffle** 的結果使 **reproduce** 相差超過 0.2%，大概會是這個原因。(即使 **data** 沒有 **shuffle**，我在兩台電腦跑出的結果差了約 0.03%)