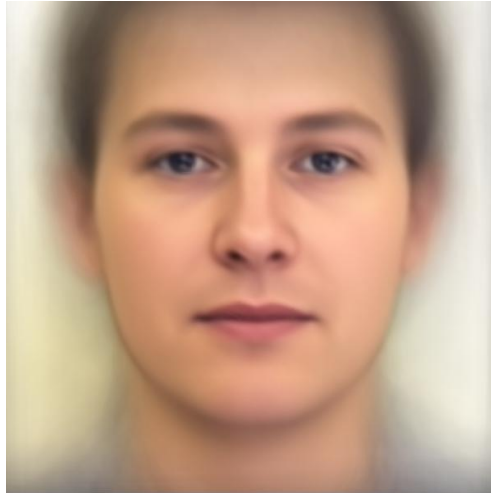


Machine Learning HW7 Report

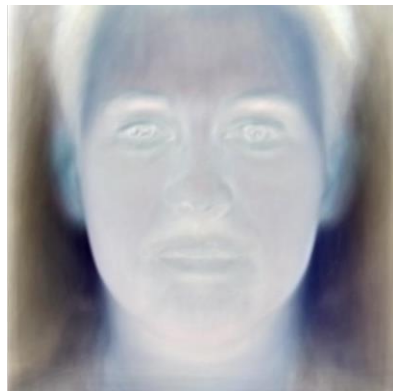
學號：B04104040 系級：工海三 姓名：解正安

1. PCA of color faces:

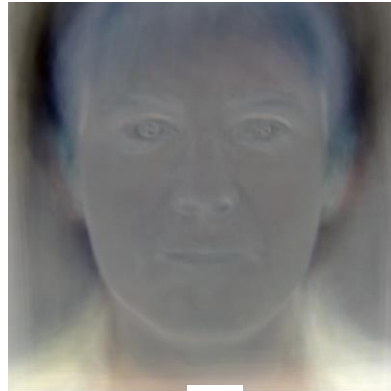
a. 請畫出所有臉的平均。



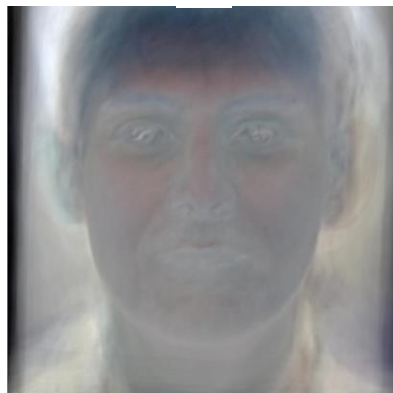
b. 請畫出前五個 Eigenfaces，也就是對應到前五大 Eigenvalues 的 Eigenvectors。



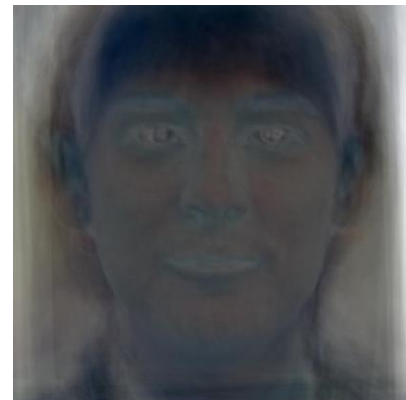
1



2



3









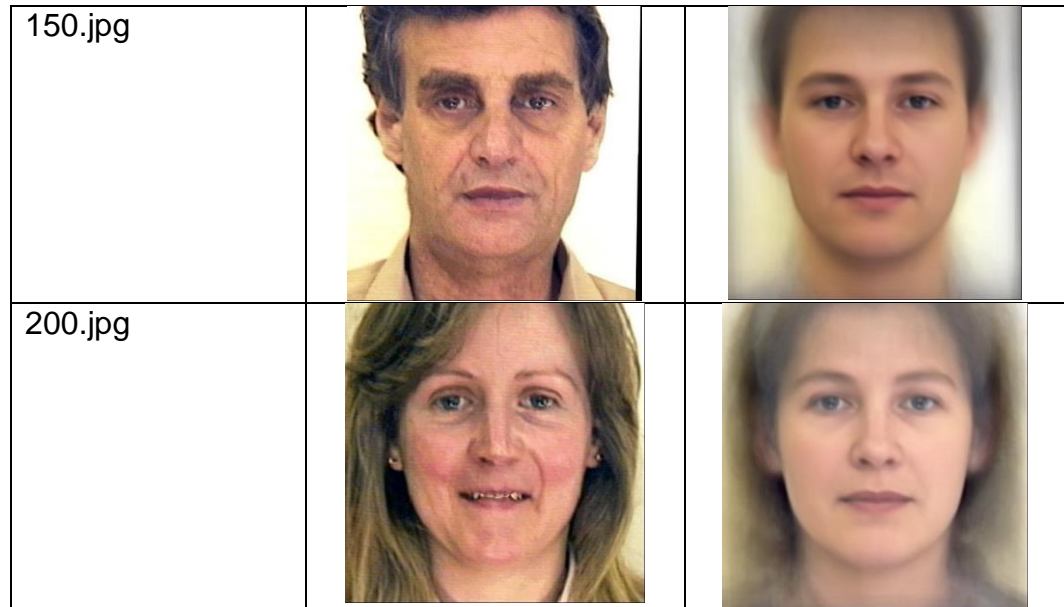
4



5

- 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。

	Original		Construction	
100.jpg				
101.jpg				
130.jpg				



- 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。

1st	2nd	3rd	4th	5th
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)

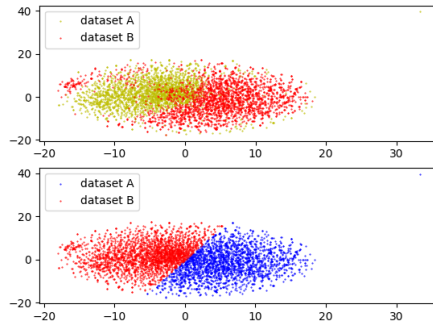
	Reconstruction loss	Reconstruction Similarity	Accuracy (Public)
AutoEncoder+PCA+Kmeans	5968.7817	0.96	0.95865
AutoEncoder+TSNE+Kmeans	5968.7817	0.96	0.86564

這裡採用 **autoencoder** 的 **reconstruction loss** 和 **similarity**。**AutoEncoder** 的架構兩者都相同，細部內容同 **c.**小題。

第一部分利用 **PCA** 降維至 128，有 **whiten**。而分群採用 **k means** 的方式，**n_clusters=2,max_iter=3000**，這裡發現 **iteration** 如果照預設效果差蠻多的，必須將 **iteration** 調大效果才夠好。這部分結果較佳，利用距離直接分群可以免除人為找 **threshold** 的困難。

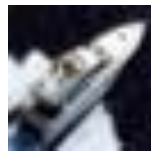
第二部分採用 **TSNE**，希望利用非線性的方式去降維，但結果並沒有更好，可能需要在調整參數讓 **autoencoder** 維度降更少，找到更合適的維度。也有可能在本次作業不適合非線性的降維方法。

- 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。



上方圖為 pca 降維成 128 後取前兩維度做 x-y 圖。從圖中可以發現兩個 dataset 在某些圖片的 feature 非常的像，可能會讓 Model 無法區分。而在利用 tsne 和 k-means 區分後，仍有很少數的點會跑到對方的區域，但其實 model 並沒有受到太大的影響。

下圖分別為 index=0 跟 index=4999 的圖片



index=0

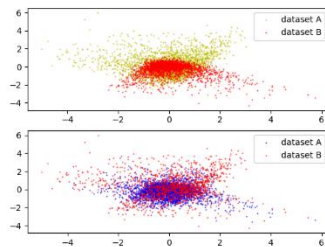


index=4999

經過重建後，發現圖片本身似乎比原先 data 更為模糊，不確定助教是否有做過處理，讓圖片 feature 弱化，使的部分 feature 會很類似。

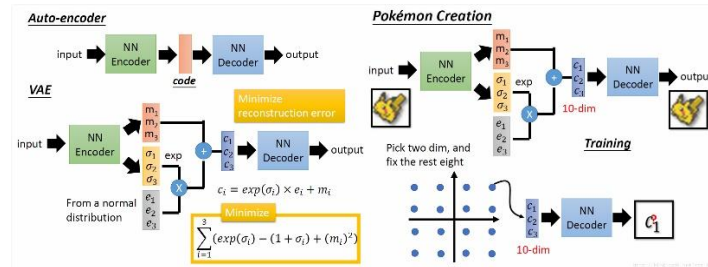
補充:若沒有用 tsne，效果會非常的差:

因為 pca 是線性，如果特徵之間是非線性，很有可能會有 underfitting 的發生。



- 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。

Encoder and Decoder: 採用 Variational autoencoder，如教授上課所提到的方式，



Learning rate = 0.001, BATCH_SIZE = 32, weight_decay=1e-5

```
# Encoder
self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1, bias=False)
self.bn1 = nn.BatchNorm2d(16)
self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=2, padding=1, bias=False)
self.bn2 = nn.BatchNorm2d(32)
self.conv3 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1, bias=False)
self.bn3 = nn.BatchNorm2d(64)
self.conv4 = nn.Conv2d(64, 16, kernel_size=3, stride=2, padding=1, bias=False)
self.bn4 = nn.BatchNorm2d(16)

self.fc1 = nn.Linear(8*8*16, 512)
self.fc_bn1 = nn.BatchNorm1d(512)
self.fc21 = nn.Linear(512, 512)
self.fc22 = nn.Linear(512, 512)
```

Encoder:總共 4 層，利用 batchNorm2d 去 normalize，未加上 dropout。

Fully Connected Layer:共 2 層，直接降維至 512

VAE 內部部分，求出平均和標準差後，再利用 normal distribution 求出一個向量，乘上標準差的 exp 再加上平均，由此去得中間降維後 vector



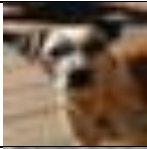


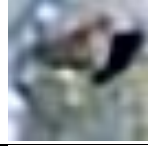


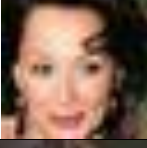
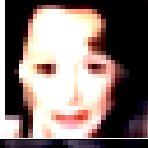







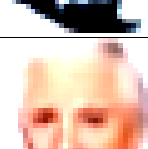

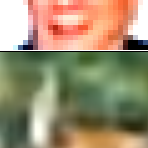


```
# Decoder
self.conv5 = nn.ConvTranspose2d(16, 64, kernel_size=3, stride=2, padding=1, output_padding=1, bias=False)
self.bn5 = nn.BatchNorm2d(64)
self.conv6 = nn.ConvTranspose2d(64, 32, kernel_size=3, stride=1, padding=1, bias=False)
self.bn6 = nn.BatchNorm2d(32)
self.conv7 = nn.ConvTranspose2d(32, 16, kernel_size=3, stride=2, padding=1, output_padding=1, bias=False)
self.bn7 = nn.BatchNorm2d(16)
self.conv8 = nn.ConvTranspose2d(16, 3, kernel_size=3, stride=1, padding=1, bias=False)
self.relu = nn.ReLU()
```




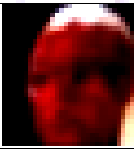
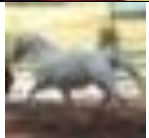



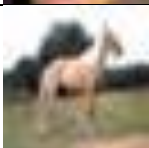

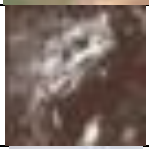
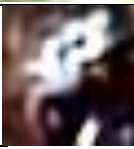


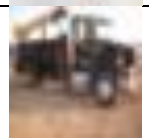
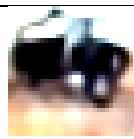
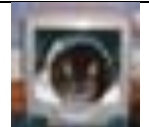

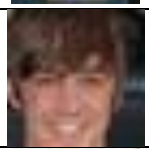
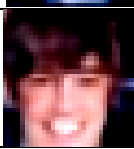




Decoder:同樣 4 層，利用 batchNorm2d 去 normalize，未加上 dropout，最後加上 relu 去確保輸出值界在 0~1 之間。





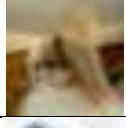
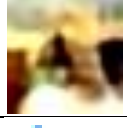






Loss function 的部分，採用 MSELoss。最後的 decoder 輸出和原圖片的 cosine similarity 達到平均 96%的相似度。

在 VAE 我降維至 512，再用 sklearn PCA 降維至 128，whiten=True, svd_solver='full'。最後利用 k-mean 分群，iteration=3000。最後準確率達到 0.95865。下圖為 32 張圖片：

	Original Photo		Reconstruct Photo	
9.jpg				
18.jpg				
27.jpg				

36.jpg					
45.jpg					
54.jpg					
63.jpg					
72.jpg					
81.jpg					
90.jpg					
99.jpg					
108.jpg					
117.jpg					
126.jpg					

135.jpg		
144.jpg		
153.jpg		
180.jpg		
225.jpg		
234.jpg		
243.jpg		
252.jpg		
261.jpg		
270.jpg		
279.jpg		
288.jpg		

297.jpg		
306.jpg		
315.jpg		
324.jpg		
333.jpg		
342.jpg		

基本上在肉眼上，兩者圖片在物體輪廓並沒有變化太多，人還是可以分辨是人，非人物體也能分辨。只是 **reconstruct** 在影像上略為模糊，可能是去掉一些不重要的雜點或是顏色，著重在一些重要的物體上。而在算 **cosine similarity** 的部分，也確實達到 **96%** 相似度，也就是說降維取到的 **feature** 應該算是不錯的。