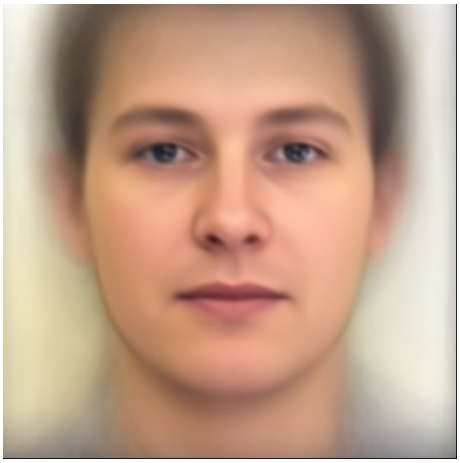
HW4

學號：R06921038 系級：電機所碩一 姓名：謝宗宏

A. PCA of colored faces

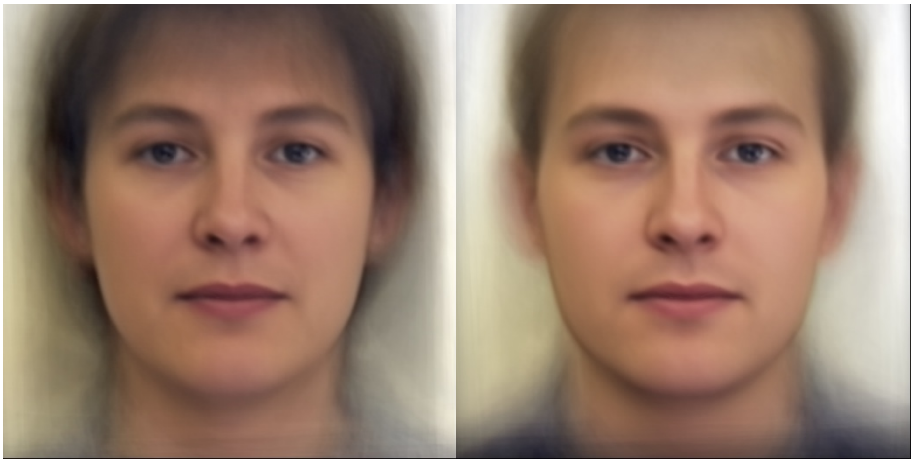
A.1. (.5%) 請畫出所有臉的平均。

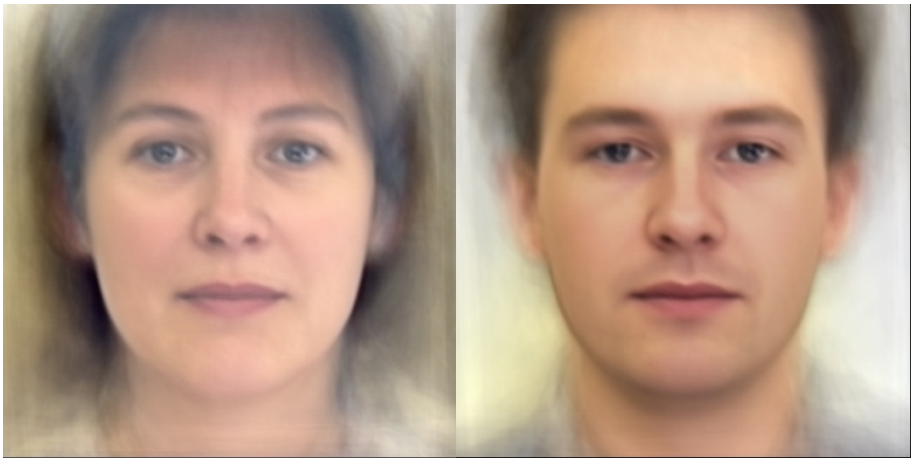


A.2. (.5%) 請畫出前四個 Eigenfaces，也就是對應到前四大 Eigenvalues 的 Eigenvectors。

左排由上至下為前四個中的最大、次大、次小、最小。右排為左排向量取負號之結果。

A.3. (.5%) 請從數據集中挑出任意四個圖片，並用前四大 Eigenfaces 進行 reconstruction，並畫出結果。





左上：0.jpg, 右上：3.jpg, 左下：5.jpg, 右下：9.jpg。

A.4. (.5%) 請寫出前四大 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。

前四大 eigenfaces 對應的 singular values 所佔的比重分別為: 0.04147478, 0.02950849, 0.0238936 , 0.02209329

以百分比表示四捨五入到小數點後一位：4.1%, 3.0%, 2.4%, 2.2%

前四大 eigenfaces 對應的 eigenvalues 所佔的比重分別為: 0.2159447, 0.10931213, 0.07167001, 0.06127666

以百分比表示四捨五入到小數點後一位：21.6%, 10.9%, 7.2%, 6.1%

B. Image clustering

B.1. (.5%) 請比較至少兩種不同的 feature extraction 及其結果。(不同的降維方法或不同的 cluster 方法都可以算是不同的方法)

方法一：使用 pca 降到 50 維再使用 tsne 降到 2 維，最後用 kmeans 分成兩群。此方法(public, private) = (0.92359, 0.92358)

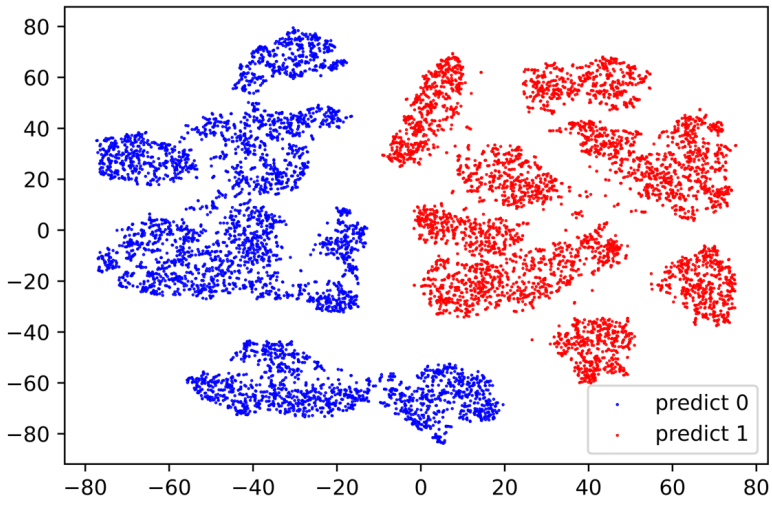
基本上都是使用 sklearn default 的參數，只有 pca 中下的 ‘algorithm’參數為 ‘full’ 而非 ’auto’。

其實這個結果也已經相當不錯，都能超過 strong baseline。儘管第一步很暴力地用 pca 降到 50 維，但是似乎沒有損失太多的資訊 (大部分重要的資訊可以用前 50 維重建) ，第二步再使用 tsne 是考慮到要將 data 分得稍微開一點，避免下一步 kmeans 的結果不穩定，並且因為 tsne 非常耗時，若是直接從原本圖片的維度降下來，其運算時間無法忍受。

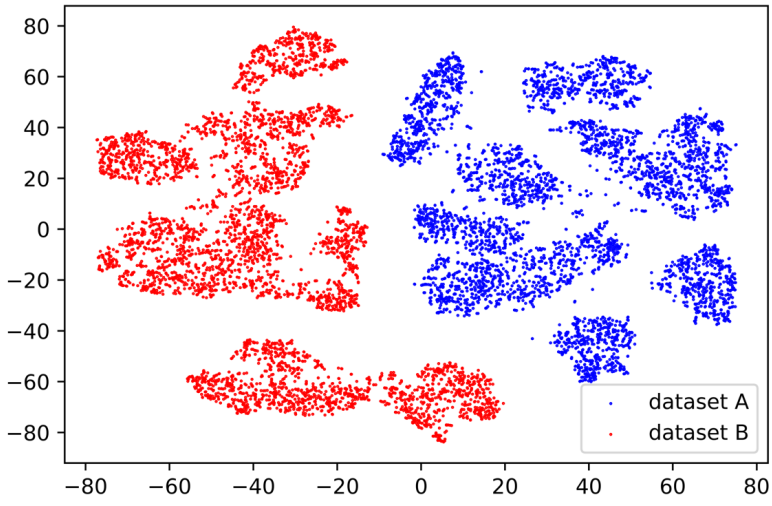
方法二 (本次最好的方法)：使用 autoencoder 降到 32 維再用 kmeans 分成兩群。此方法(public, private) = (1.0, 1.0)

所使用的 autoencoder 參考投影片中的架構，input 與 output 都是 784 維，中間層的 unit 數為 128, 64, 32, 64, 128，以 32 維那層作為 feature space。使用 nadam 而非 adam optimizer， loss 使用 mse。在 private 與 public 上都達到 1.0 的分數，表示此方法能取出有意義的 features。

B.2. (.5%) 預測 visualization.npy 中的 label，在二維平面上視覺化 label 的分佈。
下圖為用 autoencoder 降到 32 維後用 kmeans 分成兩群，再用 tsne 投影到二維，紅色與藍色代表 kmeans 預測成不同的 label。



B.3. (.5%) visualization.npy 中前 5000 個 images 跟後 5000 個 images 來自不同 dataset。請根據這個資訊，在二維平面上視覺化 label 的分佈，接著比較和自己預測的 label 之間有何不同。
下圖為使用 autoencoder 降到 32 維後再用 tsne 投影到 2 維的結果，並根據真實的 label 上色。與前一題比較，可發現結果是一致的 (除了顏色上不同) 。表示 autoencoder 取的 32 維 feature 確實有將重要的資訊找出來。



C. Ensemble learning

C.1. (1.5%) 請在 hw1/hw2/hw3 的 task 上擇一實作 ensemble learning，請比較其與未使用 ensemble method 的模型在 public/private score 的表現並詳細說明你實作的方法。（所有跟 ensemble learning 有關的方法都可以，不需要像 hw3 的要求硬塞到同一個 model 中）

以 hw2 的 task 實作。從原有的 N 筆訓練資料中隨機取樣 N 筆資料出來 (取樣後放回)作為一個新的訓練集，以此方式總共產生 11 個新的訓練集。使用作業二時的 sklearn logistic regression 對 11 個訓練集進行訓練，最後以 11 個 model 的 voting 作為預測的結果。
訓練時的設置為：epoch=200, lambda=1/C=32, stochastic average gradient descent, sklearn default step size。
未使用 bagging 時的 (public, private) = (0.85761, 0.84842)
使用 bagging 的 (public, private) = (0.85687, 0.84903)
可以發現兩者差距其實不大，而多次測試下來，bagging 在 public 的表現大多比較差，而在 private 上較好。而 0.84903 的 private score 則是比我試過的其他所有方法都要高。