

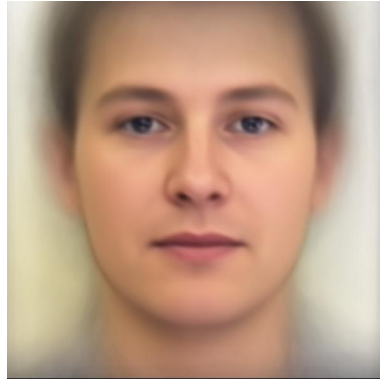
HW4

學號：b04902063 系級：資工三 姓名：陳昱儒

(collaborator：資工三 b04902055 陳巧蓁)

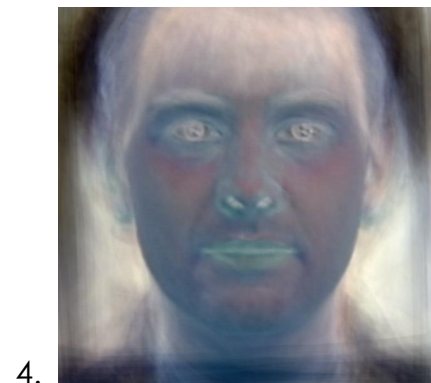
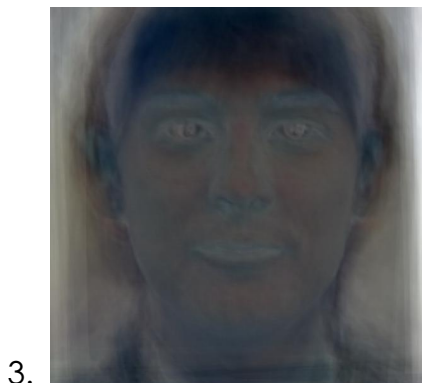
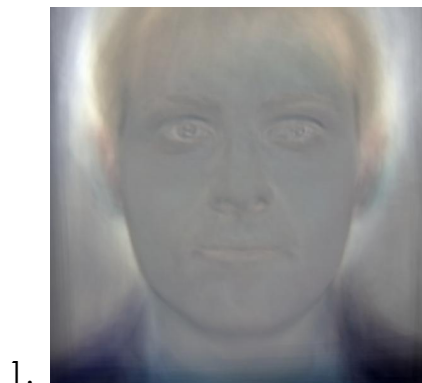
A. PCA of colored faces

1. (.5%) 請畫出所有臉的平均。




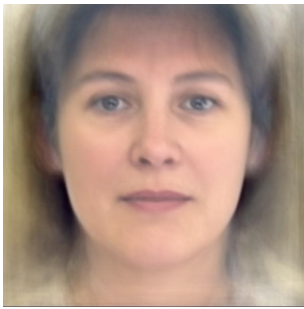
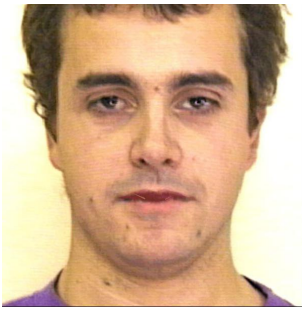
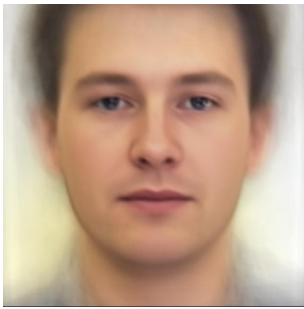
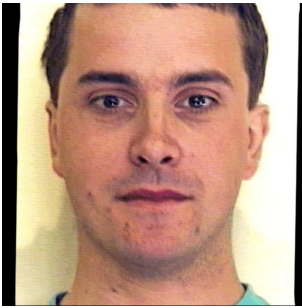
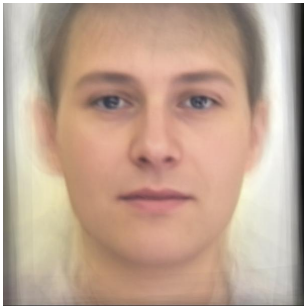

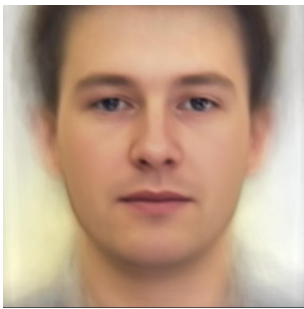
上圖為將 415 個圖片做平均的結果

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



以上 1-4 圖分別代表第一大至第四大 Eigenvalues 對應到的 Eigenface

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。

原圖	reconstruction 結果
	
	
	
	

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。

第一大	第二大	第三大	第四大
4.1%	2.9%	2.4%	2.2%

B. Image clustering

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)
在降維方法上，除了原本使用的 PCA，我試了另外一個 autoencoder，implement 的方式主要是 follow 助教在 sample code 當中的做法，在 encode 和 decode 的部分各建了三層 layers，而 clustering 的演算法依然使用 kmeans 將其分為兩類，以下為兩個不同的做法在 public data set 上和 private data set 上的表現：

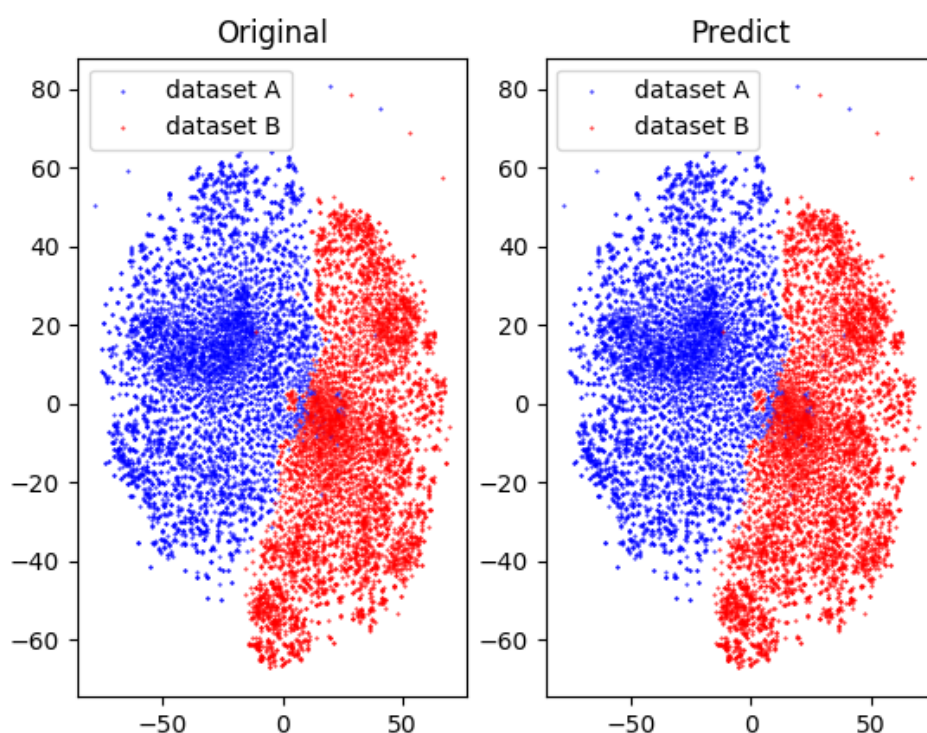
	private score	public score
PCA	1.00000	1.00000
autoencoder	0.98966	0.98971

可以發現 autoencoder 的做法相較於 PCA 的 performance 稍微低一些，我自己猜想可能是因為 autoencoder 的 model 較複雜，壓縮了較多的資訊所以在降維上漏掉了比較多資訊，也有可能是因為我的 PCA 實作上有使用 PCA-whitening 這個預處理的動作，先除去了 feature 間的相關性，使得在降維時不會造成這麼多偏誤。

2. (.5%) 預測 visualization.npy 中的 label，在二維平面上視覺化 label 的分佈。

(以下兩圖均以助教提供的 sample code 做更改)

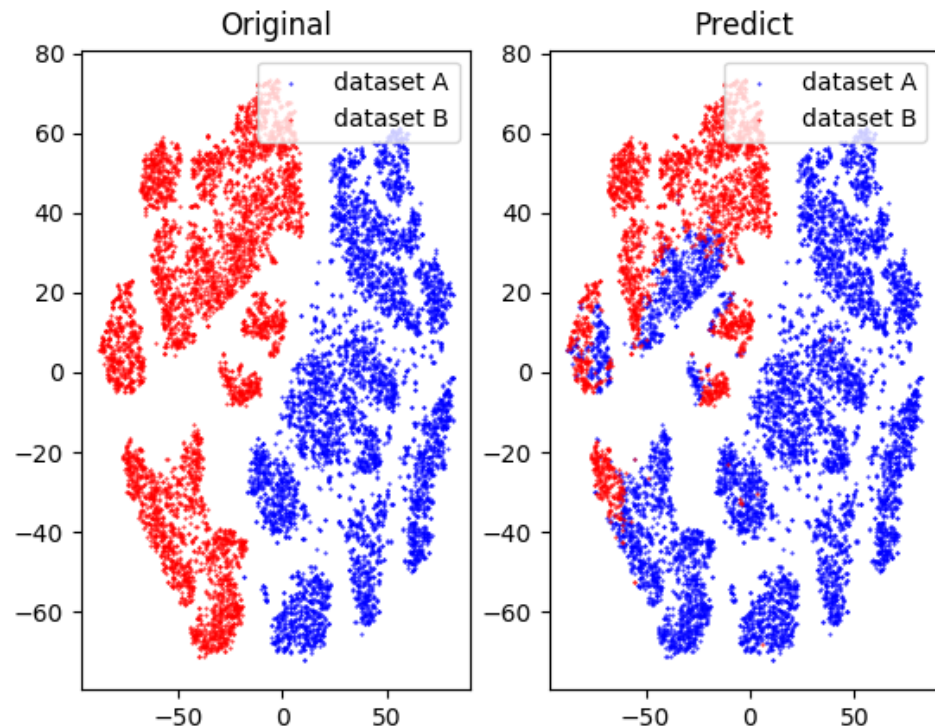
如以下附圖<Predict>的部分，先使用 sklearn 內建的 PCA 建立可降維至 400 維的 model，並利用 image.npy 中的圖片資料訓練，再來使用訓練完後的 model 將 visualization.npy 中的資料降維，並使用了 tSNE 將降維後的資料投影至二維做圖，最後使用 Kmeans 將原本被降至 400 維的資料分為兩類，然後將 Kmeans 預測出來的 label 標上，data set A 為 0，data set B 為 1。



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。

如以上附圖<Original>的部分，使用了和上面一樣由 image.npy train 出來的 PCA 降維 model 對 visualization.npy 中的資料做降維，再使用 tSNE 投影到二維上，前 5000 標示為 data set A，後 5000 標示為 data set B。（使用助教提供的 sample code，另外因 visualization.npy 中只知道前 5000 筆資料和後 5000 筆資料來自不同 data set，但沒有像 kmeans 的結果會將其標號，所以也會有兩組顏色剛剛好顛倒的狀況）。我發現我的作法剛好在 visualization.npy 這個 data set 上能做到 100% 的正確率，所以其實上面的兩個分佈圖出來的結果是一樣的，我有特別再判斷過自己預測出來的 label，剛好就是以前 5000 筆資料和後 5000 筆資料為分界點。

另外我又實做了一個沒有做 PCA-whitening 的投影圖，維度依然是降至 400 維，也使用 Kmeans 做分類，就可以顯著的發現本來<Original>的 data set 分群十分明顯，而我 predict 出來的投影圖只有大致上以紅色在左上角，藍色在右下角分佈，且不少本來應該要被判為 data set B(紅)的都被判為藍色了。



C. Ensemble learning

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）

我選擇實作 ensemble learning 的作業為 hw2，使用的方法為 bagging 算法，對於原始三萬多筆 training data 隨機採樣，取後放回，製作出四組與原本訓練資料數相同的 training data，但因為是隨機取樣且取後放回，可能取到重複資料以及部分資料不取，平均來說其包含的資料約是原始數據集的 63%，對其分別進行 logistic regression 的訓練，製作出 4 個 model，再使用這四個 model 對 testing data 進行投票，若遇到同分的狀況，則直接用四個 model 算出來的平均判斷，以下是有做 ensemble learning 以及沒有做

ensemble learning 的 model 在 public/private data set 上的表現：

	public score	private score
實作 ensemble	0.85749	0.84731
無 ensemble	0.85737	0.84866

結果發現，model 的分數與原本未實作 ensemble learning 的 model 分數不相上下，我後來又試了製作 10 個 model 以及改隨機取七成的 training data 也是一樣的結果，我想可能是因為 model 本身的 variance 就不高，對於 bagging 算法期望達到投過平均降低 variance 的效果不大，所以才會出現沒有什麼 improve 的狀況。

作業四參考 Reference:

<https://blog.csdn.net/u012162613/article/details/42192293>

<https://blog.csdn.net/walilk/article/details/69660129>

<https://ithelp.ithome.com.tw/articles/10187314>