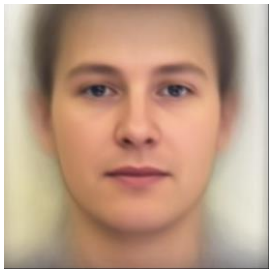
HW4

學號：r06522828　系級： 機械碩一 姓名：王榆昇

Collaborator: 王宇昕、徐瑋廷

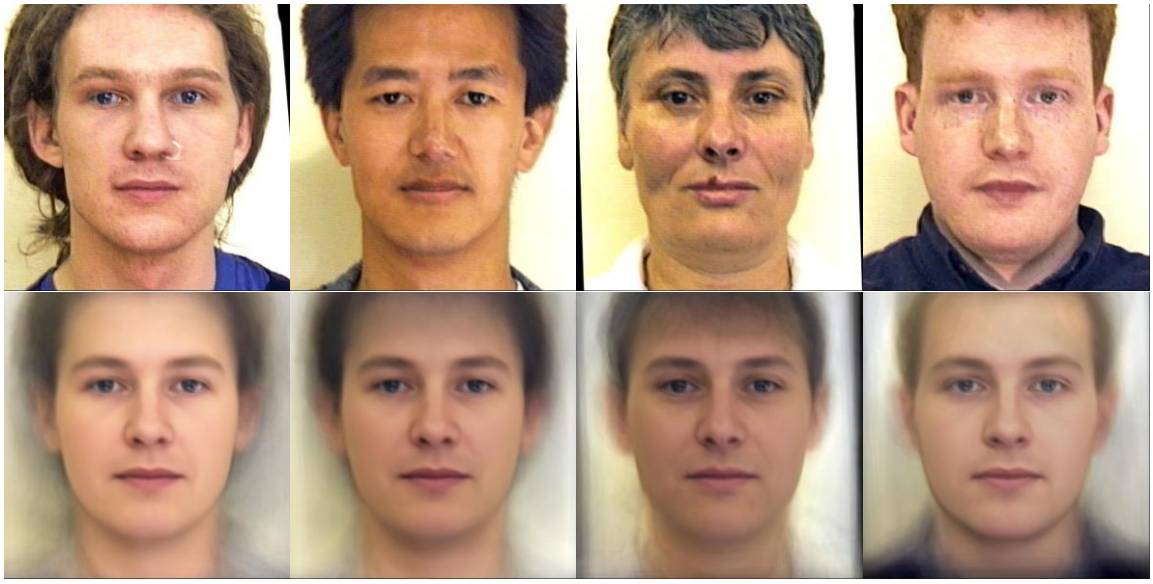
A. PCA of colored faces

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。

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。



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。

Eigenface	1	2	3	4
比重	4.1%	2.9%	2.4%	2.2%

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)

<法一>

使用 PCA 降至 400 維，並做 whiten 處理後，用 kmeans 方法分成兩類，過程都是使用 sklearn 套件:

```
pca = PCA(n_components=pca_component, copy=True, whiten=True);
z = pca.fit_transform(x);
```

```
kmeans = KMeans(n_clusters=2,random_state=0).fit(z);
```

<法二>

先訓練 3 層 encoder、3 層 decoder 的 autoencoder，採用 adam 演算法，設定 batch_size=256、epochs=50。

```
encoded = Dense(256, activation='relu')(input_img);
encoded = Dense(128, activation='relu')(encoded);
encoded = Dense(64, activation='relu')(encoded);
decoded = Dense(128, activation='relu')(encoded);
decoded = Dense(256, activation='relu')(decoded);
decoded = Dense(784, activation='sigmoid')(decoded);
```

用訓練好的 3 層 encoder 將資料降至 64 維， 再用 kmaens 將降維後的結果分成 2 類。

```
encoded_imgs = encoder.predict(x);
kmeans = KMeans(n_clusters=2, random_state=0).fit(encoded_imgs);
```

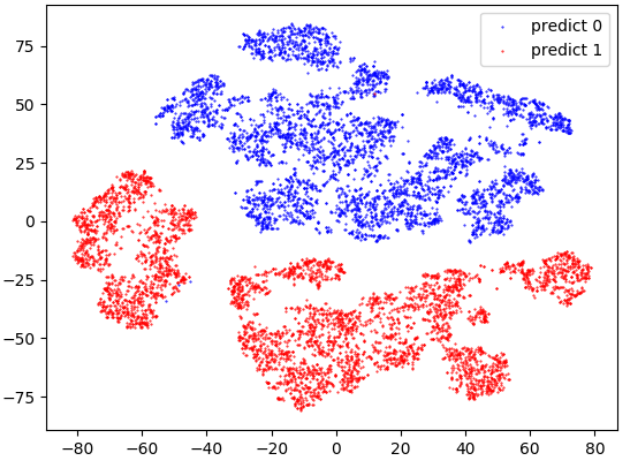
比較:

	Public	Private
PCA	0.99998	0.99998
Autoencoder	0.99972	0.99971

由上表可見，兩個演算法在這個資料的分類上都非常成功，關鍵在於 PCA 需要選到足夠具代表性的 eigenvectors，而 autoencoder 則須設計適當且穩定的架構。

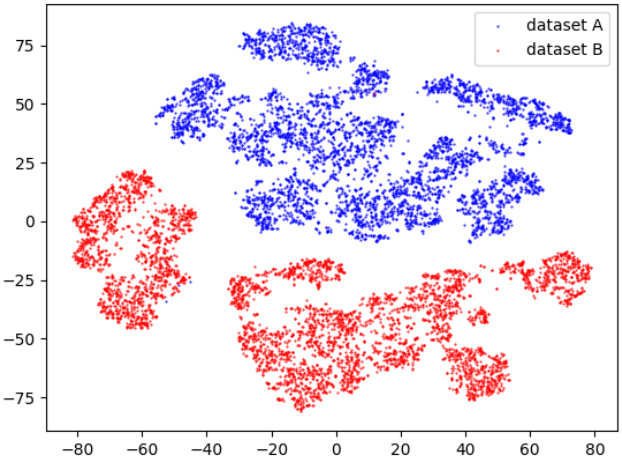
B.2 (.5%) 預測 visualization.npy 中的 label，在二維平面上視覺化 label 的分佈。

預測 ： 使用上一題得到的轉換矩陣 W(前 400 個 eigenvectors)對 visualization.npy 中的圖片做降維(降成 400 維)，再使用 kmeans 分成兩類預測。
作圖 ： 先由 PCA 降至 20 維後，再做 tsne 降成 2 維，帶入上面預測結果用不同顏色標記作圖。



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。

作圖方法同上題，直接將前、後 5000 筆資料標成不同顏色作圖。



比較 B.2 的圖，可以發現使用 PCA 做預測的結果幾乎與 B.3 標上 label 的圖相同，可以證明 PCA 在這個資料的分類效果也非常好。

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)

首先，我分別用 3 種不同的 CNN 架構，設定 batch_size=256, epochs=30、steps_per_epoch=3* len(x_train) // bacthsize，並使用 keras 內建的 ImageDataGenerator 處理 training data，訓練出 3 個不同的模型。

利用這 3 個模型，我可以得到 3 個預測結果(predict1、predict2、predict3)，針對每個 testing data 我選用多數決的方法，由這 3 個 predicted result 投票，並選最高票的那個 label 作為輸出結果，若遇到相同票數情況則選擇第一個輸出，如下:

```
227  for i in range(len(y_predict1)):
228      vote = [0,0,0,0,0,0,0];
229      vote = np.array(vote);
230      vote[y_predict1] += 1;
231      vote[y_predict2] += 1;
232      vote[y_predict3] += 1;
233      index = np.where(vote == np.max(vote));
234      result.writerow([(i), index[0][0]] );
```

結果比較 :

	Model1	Model2	Model3	Ensemble Model
Public	0.65589	0.65171	0.64781	0.66202
Private	0.66258	0.66397	0.65505	0.66703

可以看到，ensemble 過後的 model 與原本的 model 做比較下，在 kaggle 的表現上不論是 public、private 都獲得 0.01~0.005 的增長，說明了即使是多數決這樣一個簡單的 ensemble model，也能適當的修補單一 model 犯錯的地方，讓整合的模型更準確且更穩定。