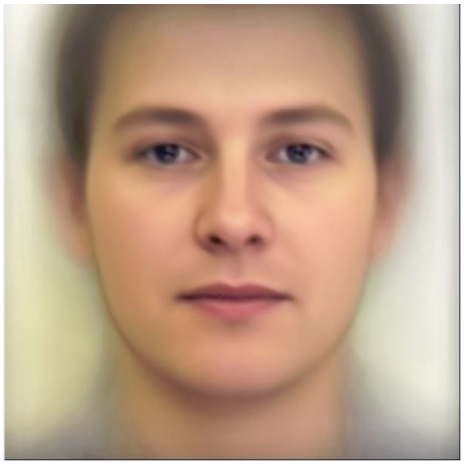
Machine Learning HW7 Report

學號：R07943150　　系級：電子碩一　　姓名：吳辰鋐

1. PCA of color faces:

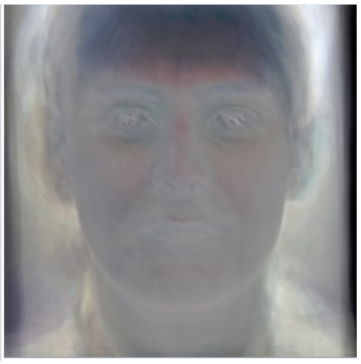
a. 請畫出所有臉的平均。



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

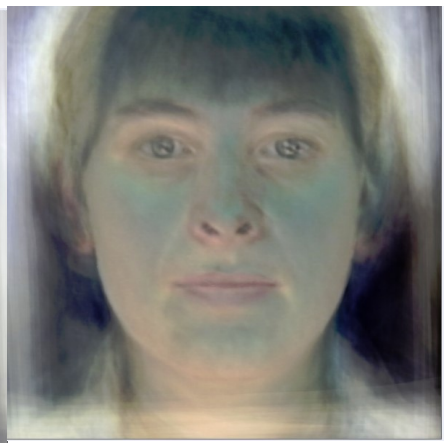
1

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


1


10


22


37


72

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。

前五大 Eigenfaces 比重依序為: 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)

Model 1 號如下

```
input_img = Input(shape=(32, 32, 3))
#Encoder:
conv_1 = Conv2D(64, (3,3), strides=(1,1))(input_img)
act_1 = Activation('relu')(conv_1)
maxpool_1 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(act_1)
conv_2 = Conv2D(64, (3,3), strides=(1,1), padding='same')(maxpool_1)
act_2 = Activation('relu')(conv_2)
maxpool_2 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(act_2)
flat_1 = Flatten()(maxpool_2)
fc_1 = Dense(512)(flat_1)
act_3 = Activation('relu')(fc_1)
fc_2 = Dense(256)(act_3)
act_4 = Activation('relu')(fc_2)
#Decoder:
fc_4 = Dense(512)(act_4)
act_5 = Activation('relu')(fc_4)
fc_5 = Dense(3136)(act_5)
act_6 = Activation('relu')(fc_5)
reshape_1 = Reshape((7,7,64))(act_6)
upsample_1 = UpSampling2D((2, 2))(reshape_1)
deconv_1 = Conv2DTranspose(64, (3, 3), strides=(1, 1))(upsample_1)
act_7 = Activation('relu')(deconv_1)
upsample_2 = UpSampling2D((2, 2))(act_7)
deconv_2 = Conv2DTranspose(64, (3, 3), strides=(1, 1))(upsample_2)
act_8 = Activation('relu')(deconv_2)
conv_3 = Conv2D(3, (3, 3), strides=(1, 1))(act_8)
act_9 = Activation('sigmoid')(conv_3)
autoencoder = Model(input_img, act_9)
autoencoder.compile(optimizer='rmsprop', loss='mae')
```

Encoder 出來 256 維資料用 PCA 降成 128 維再用 Kmeans 分成 2 群

Public score:96.734%　　Private score:96.718%

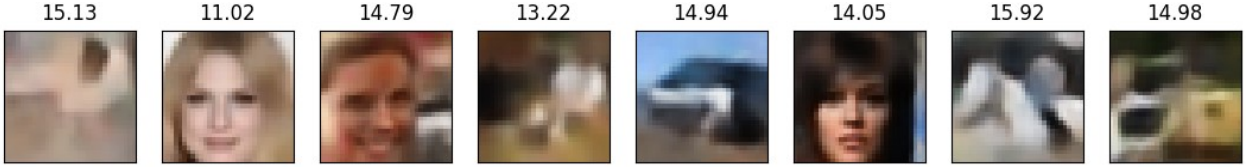
然而一樣的 data 用 hierarchical clustering 分成 2 群

Public score: 85.907%　　Private score: 86.029%

分數明顯降低許多

Original:

Decoder reconstruct:

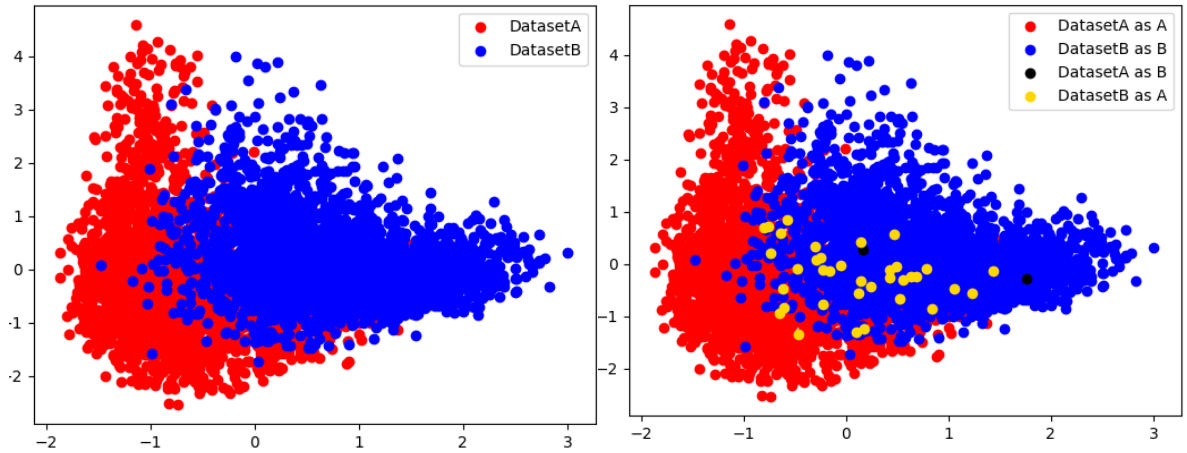


Original:

Decoder reconstruct:

Decoder reconstruct 圖片上面小數代表平均和原圖相差多少色值(reconstruction loss)
Model2 號模型架構、分群方式、準確率、reconstruction loss 都在題 C
Model1 號相較之下小很多，少了很多參數，也沒有使用 Conv2DTranspose 和經過 BatchNormalization 層，模型訓練完後 MAE 為 0.0501，是 Model2 的兩倍，因此重建結果差異較大。

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。



左圖是原始 data 透過 encoder 變成 4096 維再用 PCA 降成 2 維的分布圖
右圖是將 data 透過 encoder 變成 4096 維再用 PCA 降成 300 維後利用 Kmeans 分群的結果畫到左圖上，其中黃點和黑點為誤判的資料，準確率達到 99.24%。

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 從 input_img 到 act_3，總共兩層(Conv2D+ BatchNormalization +Maxpooling) 最後攤平接上 Dense
Decoder 從 fc_5 到 act_9，由 Dense reshape 為 7*7*128 再接回兩層 (Upsampling+Conv2DTranspose+BatchNormalization)
Encoder 及 Decoder 的 Activate function 皆為 ReLu，Optimizer 使用 rmsprop，Loss function 為 Mean Absolute Error，訓練結束時模型 MAE= 0.0255，等同是每張圖平均相差 0.0255/(1/255) = 6.5025 色值

```
K_result = KMeans(n_clusters=2, max_iter=500, n_init=50, verbose=0, n_jobs=-1, random_state=seed).fit(PCA_data)
```

Encode 完的 data 有 4096 維，再透過 PCA 降成 800、1100、1400、1700、2000 後用 Kmeans 分成兩群去 ensemble，最後結果
Public score: 99.314%　　Private score: 99.328%

```
#Encoder:
input_img = Input(shape=(32, 32, 3))
conv_1 = Conv2D(128, (3,3), strides=(1,1))(input_img)
bn_1 = BatchNormalization()(conv_1)
act_1 = Activation('relu')(bn_1)
maxpool_1 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(act_1)

conv_2 = Conv2D(128, (3,3), strides=(1,1), padding='same')(maxpool_1)
bn_2 = BatchNormalization()(conv_2)
act_2 = Activation('relu')(bn_2)
maxpool_2 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(act_2)

flat_1 = Flatten()(maxpool_2)

fc_1 = Dense(4096)(flat_1)
bn_3 = BatchNormalization()(fc_1)
act_3 = Activation('relu')(bn_3)
encoder = Model(input_img, act_3)
```

```
#Decoder:
fc_5 = Dense(3136*2)(act_3)
bn_4 = BatchNormalization()(fc_5)
act_6 = Activation('relu')(bn_4)

reshape_1 = Reshape((7,7,128))(act_6)

upsample_1 = UpSampling2D((2, 2))(reshape_1)
deconv_1 = Conv2DTranspose(128, (3, 3), strides=(1, 1))(upsample_1)
bn_5 = BatchNormalization()(deconv_1)
act_7 = Activation('relu')(bn_5)

upsample_2 = UpSampling2D((2, 2))(act_7)
deconv_2 = Conv2DTranspose(128, (3, 3), strides=(1, 1))(upsample_2)
bn_6 = BatchNormalization()(deconv_2)
act_8 = Activation('relu')(bn_6)

conv_3 = Conv2D(3, (3, 3), strides=(1, 1))(act_8)
bn_7 = BatchNormalization()(conv_3)
act_9 = Activation('sigmoid')(bn_7)

autoencoder = Model(input_img, act_9)
autoencoder.compile(optimizer='rmsprop', loss='mae')
```

Original:

Decoder reconstruct:

Original:

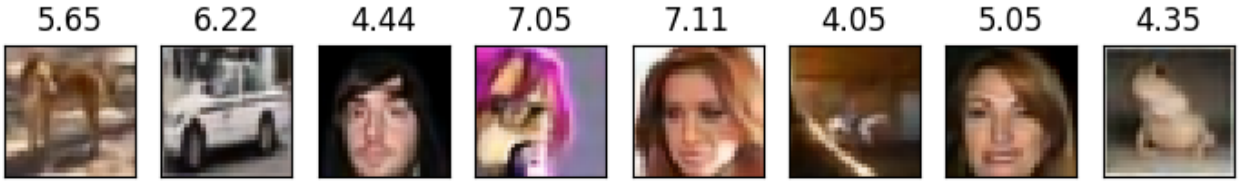
Decoder reconstruct:

Original:

Decoder reconstruct:

Original:

Decoder reconstruct:



Decoder reconstruct 圖片上面小數代表平均和原圖相差多少色值，相似度極高可以推測是因為中間拉出來的 Dense 維度很高，4096 已經比原本 32*32*3 更高，因此還原能力很強。