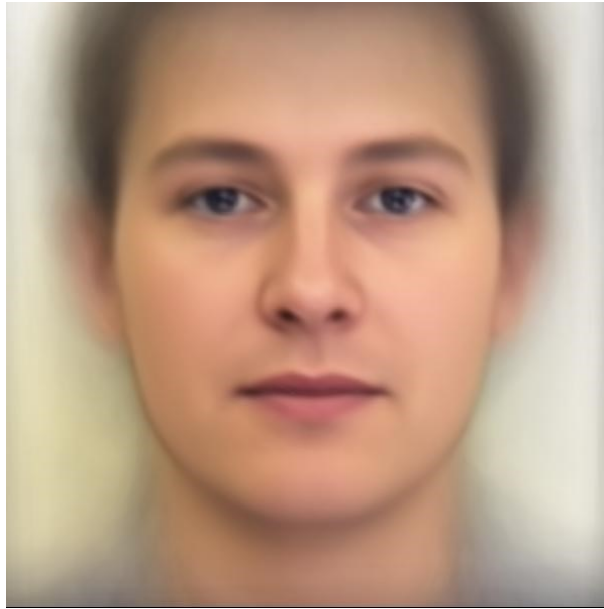


A. PCA of colored faces

A.1. (.5%) 請畫出所有臉的平均。

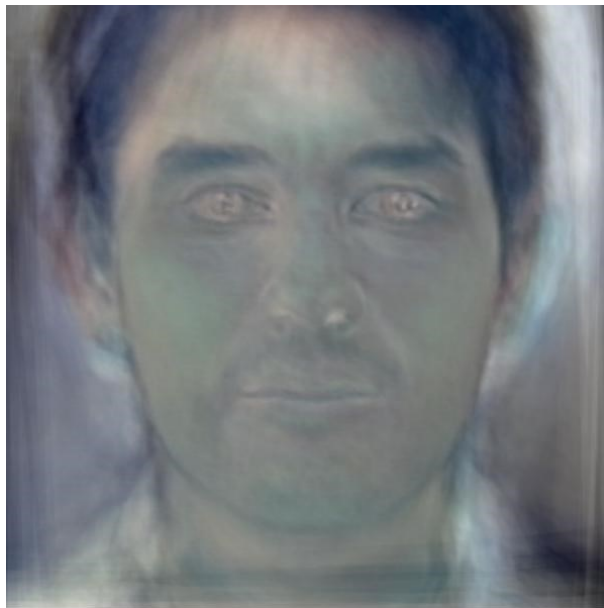


圖、共 415 張圖的平均臉

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

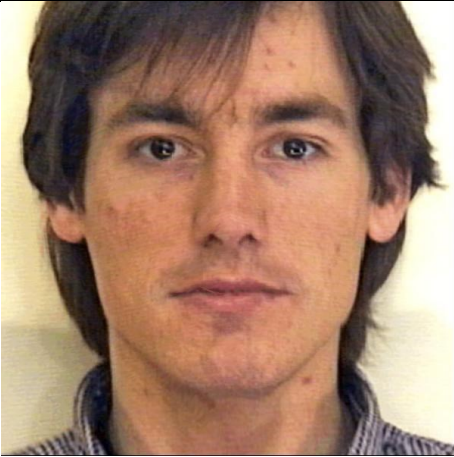
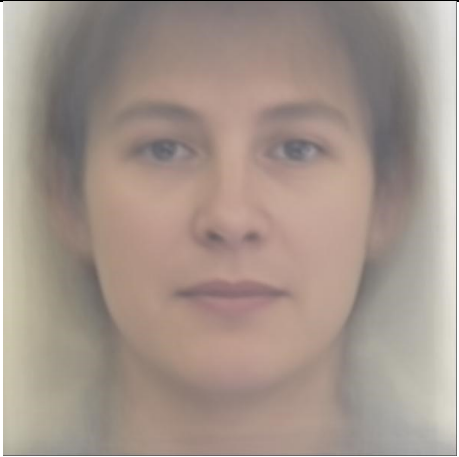

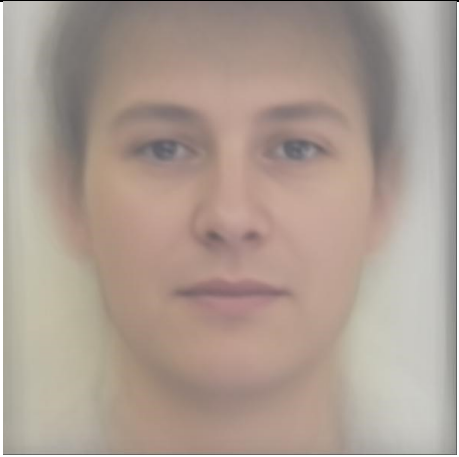






圖、由左上至右下分別為編號第 0, 1, 2, 3 個(前四大)Eigenfaces



圖、第十大(編號第 9 個)Eigenfaces，與助教的圖幾乎相同

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。

Image name	Original image		Reconstruction image	
0.jpg				
100.jpg				
200.jpg				
300.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。

表、前四大 Eigenfaces 與所有 Eigenfaces 總和

s[0]	540369.686584
s[1]	384451.083146
s[2]	311306.053868
s[3]	287854.919730
sum(s)	13037843.174500

表、前四大 Eigenfaces 各自所佔的比重

s[0]	4.144625 %
s[1]	2.948732 %
s[2]	2.387711 %
s[3]	2.207842 %

表、前四大 Eigenfaces 各自所佔的比重(四捨五入到小數點後一位)

s[0]	4.1 %
s[1]	2.9 %
s[2]	2.4 %
s[3]	2.2 %

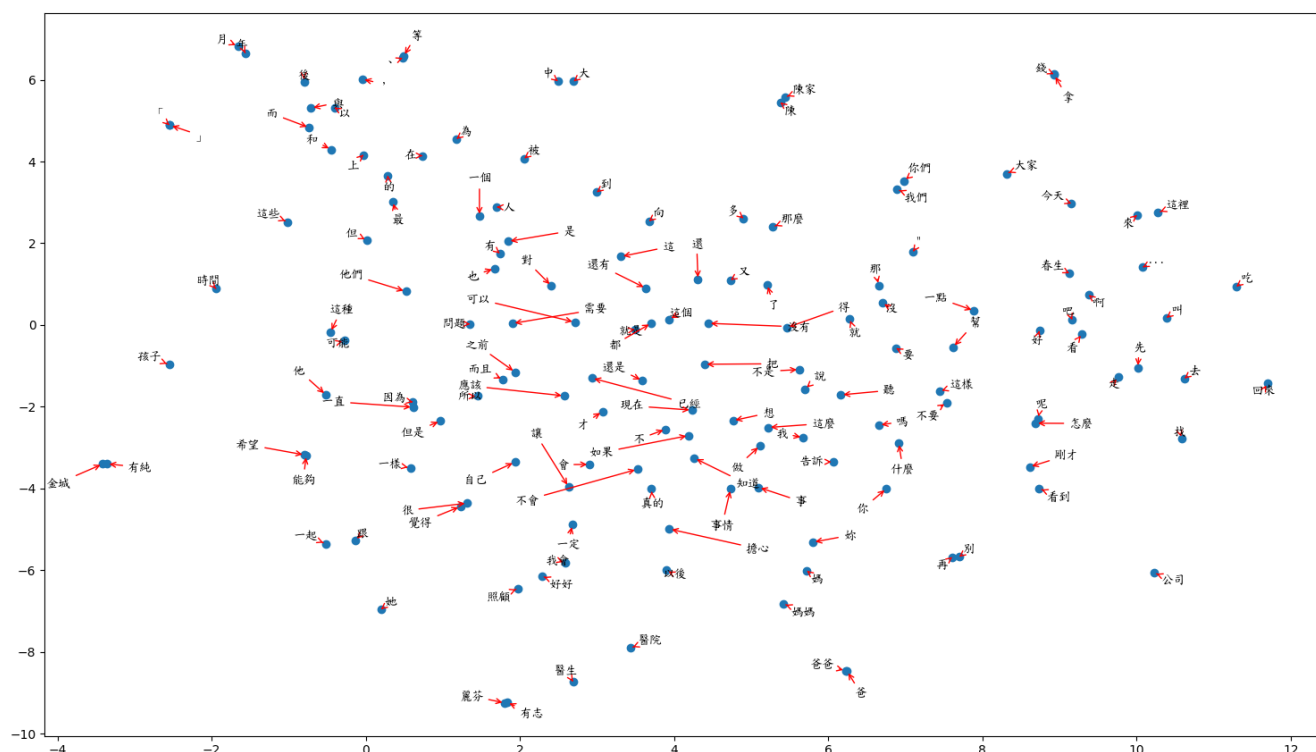
B. Visualization of Chinese word embedding

B.1. (.5%) 請說明你用哪一個 word2vec 套件，並針對你有調整的參數說明那個參數的意義。

```
12 sentences = word2vec.Text8Corpus(TRAIN_SEG_FILE_PATH)
13 print ('Training model...')
14 model = word2vec.Word2Vec(sentences,
15                             size=128, window=8, sample=1e-4,
16                             negative=10, hs=0, sg=0, iter=15,
17                             workers=8, min_count=5)
18 print ('Saving model...')
19 model.save(W2V_MODEL_FILE_PATH)
```

我使用 jieba 先將中文句子依 word 切開(TRAIN_SEG_FILE_PATH)，再使用 gensim 的 word2vec 來 train，最後將 model 存起來(W2V_MODEL_FILE_PATH)。其中 size=128 就是每一個 word 對應到的 vector 有多少 dimension、window=8 是一個句子中考慮的最長距離(左右最多看多少字)、sample=1e-4 是高頻率單字被隨機 down sampled 的 threshold、negative=10 是多少 noise words 會被 negative sampled、hs=0 會啟用 negative sampling、sg=0 會使用 CBOW training algorithm、iter=15 代表會 train 15 個 epochs、workers=8 代表我使用 8 個 threads 一起 train、min_count=5 代表出現頻率低於 5 的 word 會被忽略。而最後 visualization 時只針對出現 3000 次以上的字做視覺化。

B.2. (.5%) 請在 Report 上放上你 visualization 的結果。



B.3. (.5%) 請討論你從 visualization 的結果觀察到什麼。

圖有點大張，助教可能需要放大來看，圖中可以發現類似意思的字有被畫到相近的位置上，像是「你們、我們」、「很、覺得」、「而、和、與、以、等、逗號、頓號」、「上引號、下引號」、「年、月」、「爸爸、爸」、「媽媽、媽」等，都有被分在一塊。其中人名都有明顯被獨立分開自己一群，連接詞與標點符號也有被分開自己一群，沒有混在字詞之間。然後最右邊的區塊幾乎都是動詞居多、左下則是名詞居多、左上則是連接詞居多，中間的區塊則有點分不開，且箭頭交錯使得畫面有些混亂了。

C. Image clustering

C.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)

我原本是使用 **Convolutional autoencoder** 將 28x28x1 的每一張 image 降至 14x14x4、再降至 7x7x2、再降至 4x4x2，也就是 **32 維**，最後用 Kmeans 分成兩群，但是這樣的 F1 score 在 private/public 上只有 **0.54726/0.54697** 而已。架構圖如圖 C1.1.所示。

後來助教公告 sample code 之後，我也跟著改用 **Deep autoencoder** 將 784 維的每一張 image 降至 128、再降至 64、再降至 **32 維**，用 Kmeans 分成兩群，結果這樣的 F1 score 在 private/public 上竟然達到 **1.00000/1.00000**。但是我沒有用助教的 learning rate(adam)，且跑了 3072 個 epochs。架構圖如圖 C1.2.所示。

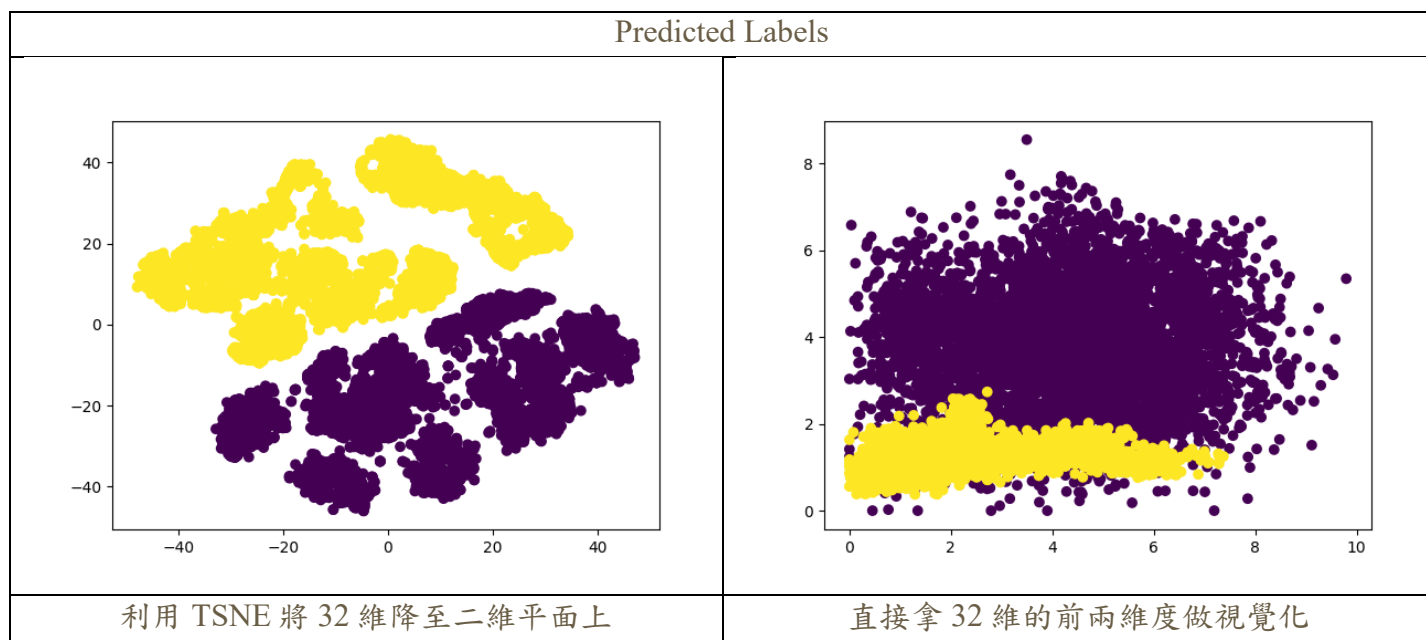
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 28, 28, 1)	0
conv2d_1 (Conv2D)	(None, 28, 28, 4)	40
leaky_re_lu_1 (LeakyReLU)	(None, 28, 28, 4)	0
max_pooling2d_1 (MaxPooling2)	(None, 14, 14, 4)	0
conv2d_2 (Conv2D)	(None, 14, 14, 2)	74
leaky_re_lu_2 (LeakyReLU)	(None, 14, 14, 2)	0
max_pooling2d_2 (MaxPooling2)	(None, 7, 7, 2)	0
conv2d_3 (Conv2D)	(None, 7, 7, 2)	38
leaky_re_lu_3 (LeakyReLU)	(None, 7, 7, 2)	0
max_pooling2d_3 (MaxPooling2)	(None, 4, 4, 2)	0
conv2d_4 (Conv2D)	(None, 4, 4, 2)	38
leaky_re_lu_4 (LeakyReLU)	(None, 4, 4, 2)	0
up_sampling2d_1 (UpSampling2)	(None, 8, 8, 2)	0
conv2d_5 (Conv2D)	(None, 8, 8, 2)	38
leaky_re_lu_5 (LeakyReLU)	(None, 8, 8, 2)	0
up_sampling2d_2 (UpSampling2)	(None, 16, 16, 2)	0
conv2d_6 (Conv2D)	(None, 14, 14, 4)	76
leaky_re_lu_6 (LeakyReLU)	(None, 14, 14, 4)	0
up_sampling2d_3 (UpSampling2)	(None, 28, 28, 4)	0
conv2d_7 (Conv2D)	(None, 28, 28, 1)	37

圖 C.1.1. Convolutional autoencoder 架構圖

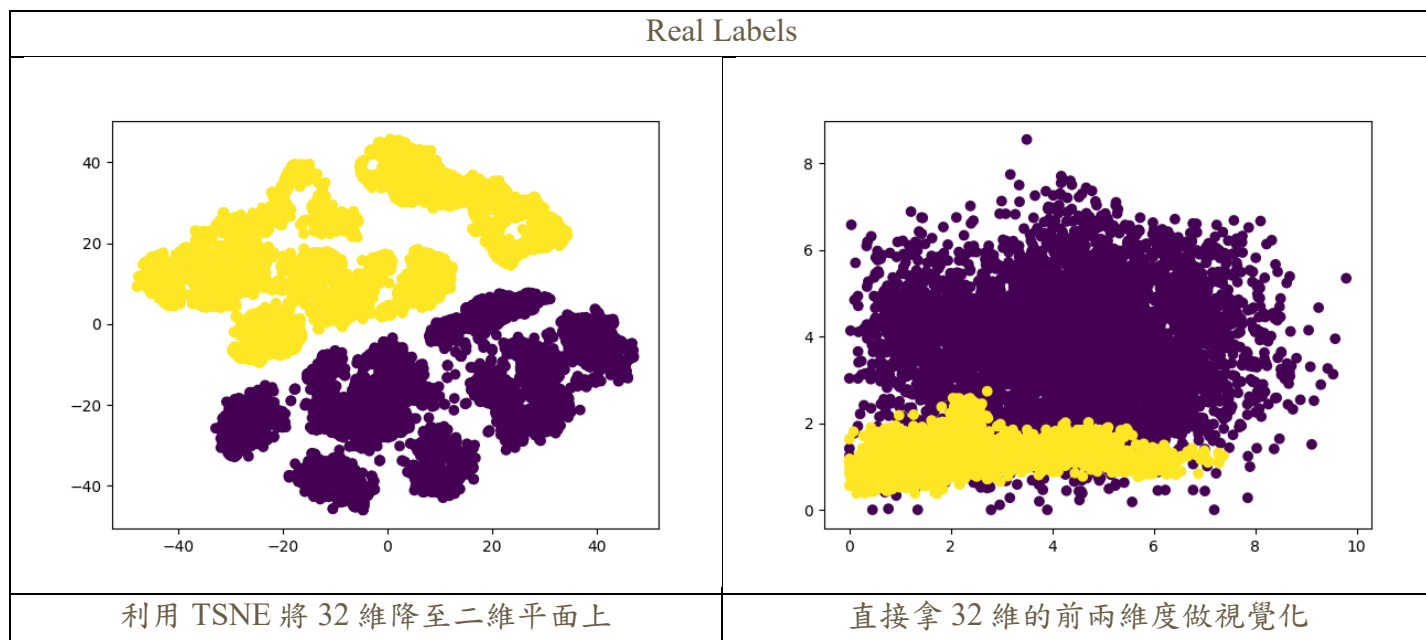
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 784)	0
dense_1 (Dense)	(None, 128)	100480
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 32)	2080
dense_4 (Dense)	(None, 64)	2112
dense_5 (Dense)	(None, 128)	8320
dense_6 (Dense)	(None, 784)	101136

圖 C1.2. Deep autoencoder 架構圖

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



C.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。



因為我的 DNN Auto Encoder 降至 32 維再配上 Kmeans 的分成 2 群的方法在 Kaggle 的 public 與 private 上的 F1 Score 都是 1.00000，所以 C.3. 的圖片會跟 C.2. 的圖片一模一樣（但我真的有分別去跑兩種 labels），所以我在這邊討論一下左右兩張圖有何不同。左邊的圖是用 TSNE 將 32 維降至二維平面上，點與點在高維度空間上的關係在平面上還有被保留下來，所以原本應該被分開的兩群，被降至兩維後還是能分成兩群。而右邊的圖直接拿 32 維中的其中兩維來作圖，就會直接捨去剩下 30 維的資訊，所以才會導致在左下角的部分兩群資料都混在一起了。