Dimension Reduction

- **PCA, ICA and LLE comparison**
- **Eigenface Algorithm**

Usage

- **dm_reduction.py**

```
$ python3 dm_reduction.py [-h]
```

optional Options	Description
-h, --help	show this help message and exit
-s, SCALING	svm kernel,default=rbf
-nc, N_COMPONENTS	PCA,ICA,LLE n_components, default = 25
-n, NUMBER	The number you want to plot in Scatter plot, 0…9 or all , default = 2
-gs, GRID_SIZE	The grid size of scatter plot for number images, default = 10
-dr, DR_TECHNIQUES	The techniques of dimension reduction, default = pca, pca=(Principal Component Analysis) , ica =(Independent Component Analysis), lle =(Local Linear Embedding), all(pca, ica, lle)
-ims, IMG_SHOW	Show image, default = false

python3 dm_reduction.py -n all -dr all
會執行全部方法的全部數字，結果圖片存在 result_a 資料夾內

- **eigenface_algo.py**

```
$ python3 eigenface_algo.py [-h]
```

optional Options	Description
-h, --help	show this help message and exit
-nc, N_COMPONENTS	PCA,ICA,LLE n_components, default = 25

用-nc 來指定降維的數量。
結果會在 result_b 資料夾內。

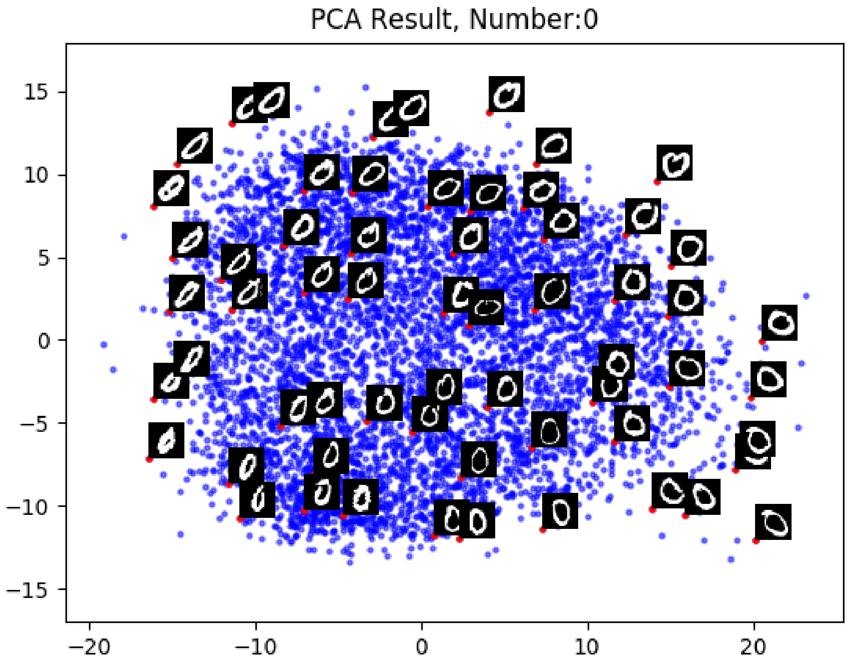
Reoprt

- **Apply PCA (Principal Component Analysis) to the MNIST database of hand written digits and write a report to analyze the characteristics of each main axis in the reduced 2D-space.**

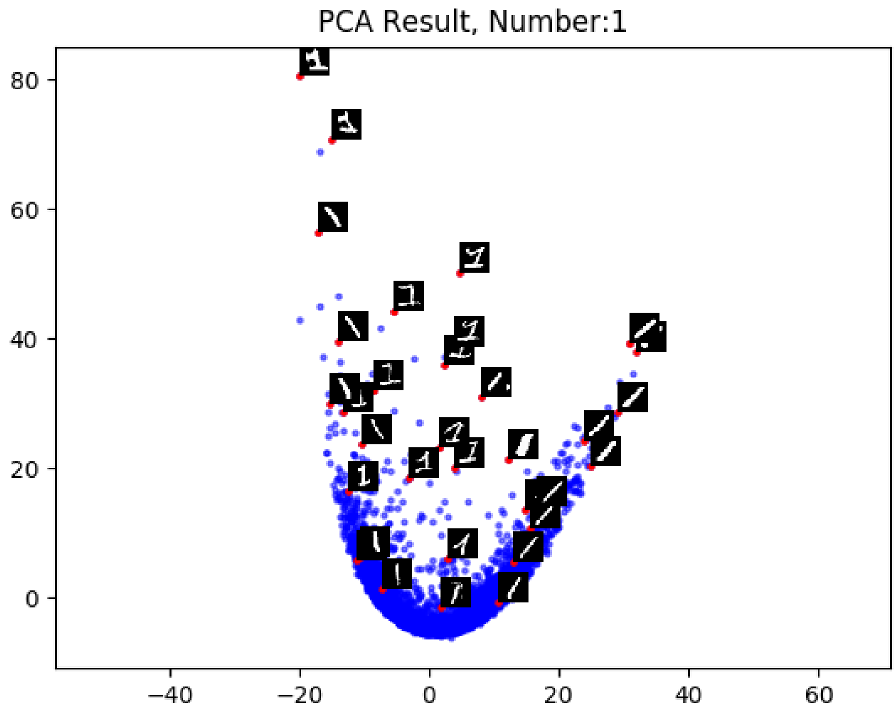
- **MNIST data set**
  train-images.idx3-ubyte

- **Brief description of development environment**
  DISTRIB_ID=Ubuntu
  DISTRIB_RELEASE=18.04
  DISTRIB_CODENAME=bionic
  DISTRIB_DESCRIPTION="Ubuntu 18.04.1 LTS"
  Architecture: x86_64
  CPU op-mode(s): 32-bit, 64-bit
  CPU(s): 12
  Model name: Intel® Core™ i7-8700 CPU @ 3.20GHz
  L1d cache: 32K
  L1i cache: 32K
  L2 cache: 256K
  L3 cache: 12288K

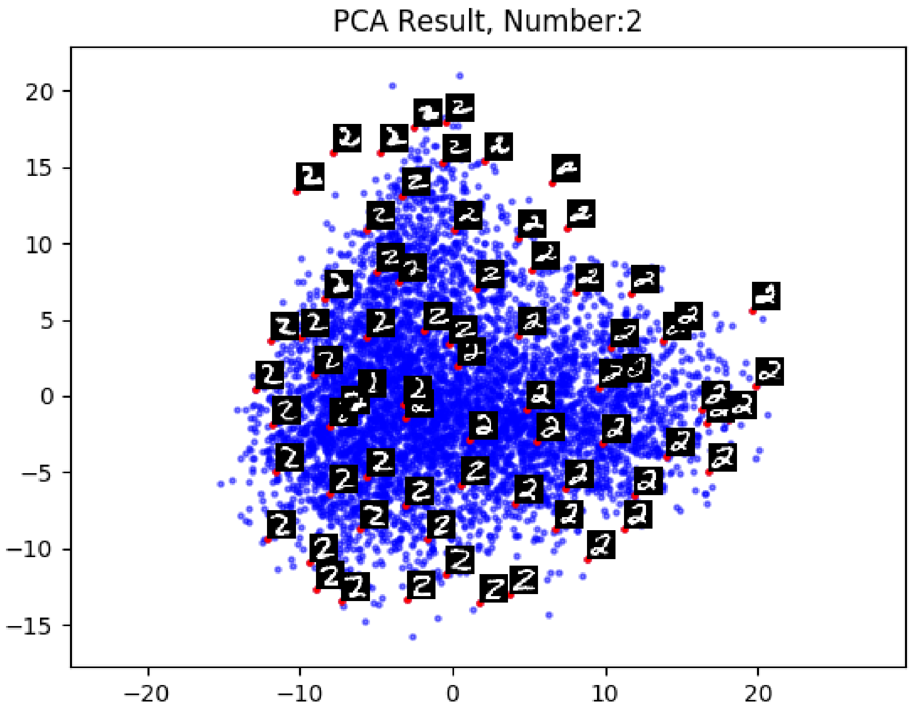
- **The result figure as the example in page 2**
- **Description of your observations**



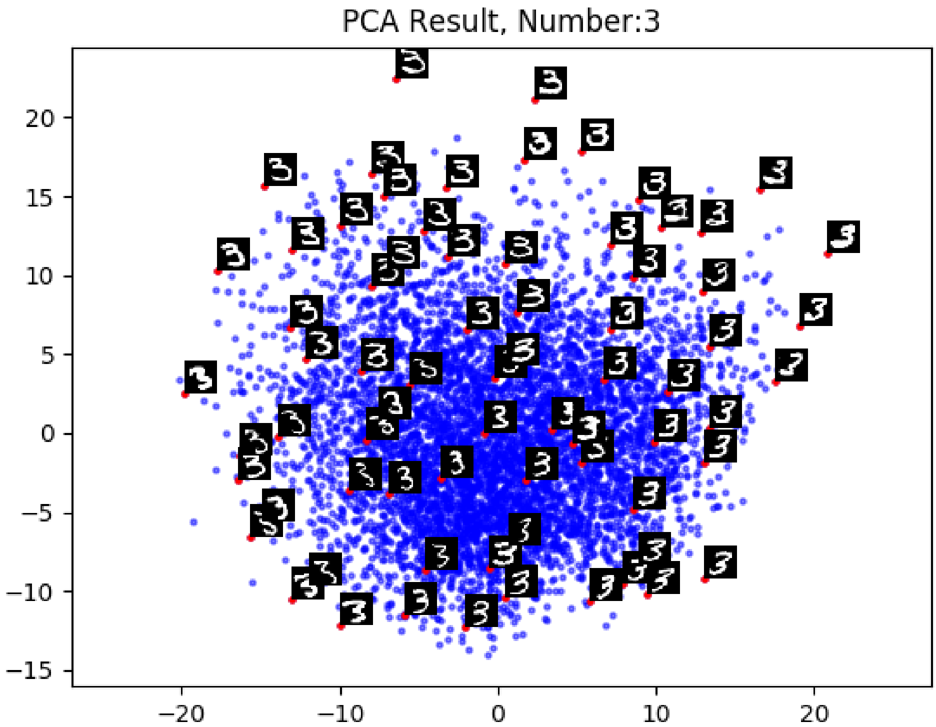
此張圖可以觀察到，在結果的左邊部分的 0 都往右邊傾斜且較瘦長，靠中間的 0 較為直立且圓，而右邊的 0 往左邊傾斜。



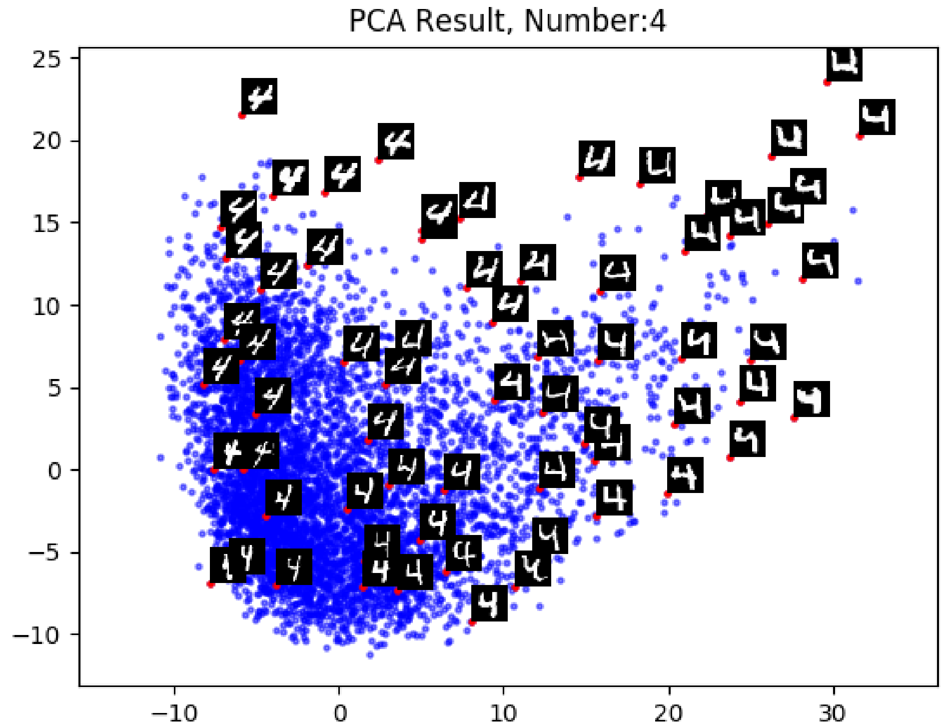
此張圖可以觀察到，在結果的左邊部分的 1 都往左邊傾斜，中間的 1 直立，且頭部會突出，而右邊 1 的往右邊傾斜。



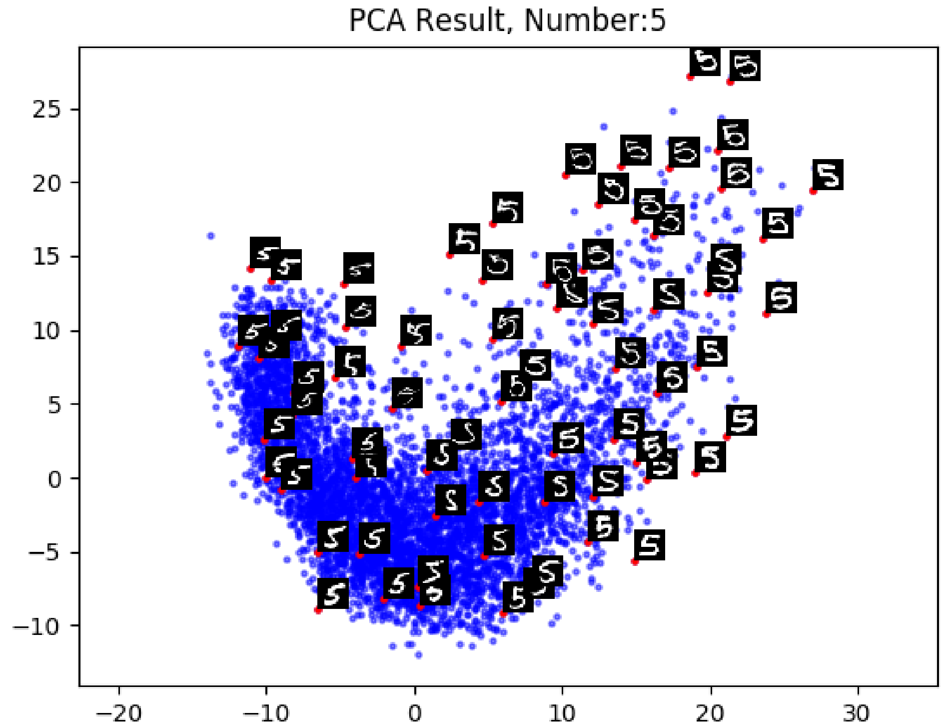
此張圖可以觀察到，在右邊的 2 底部會形成一個小圈圈，頭也比較彎，而左邊的 2 頭和底部都比較直，上面的 2 頭部較靠前，而下面的 2 頭部較靠後，也頭比較寬。



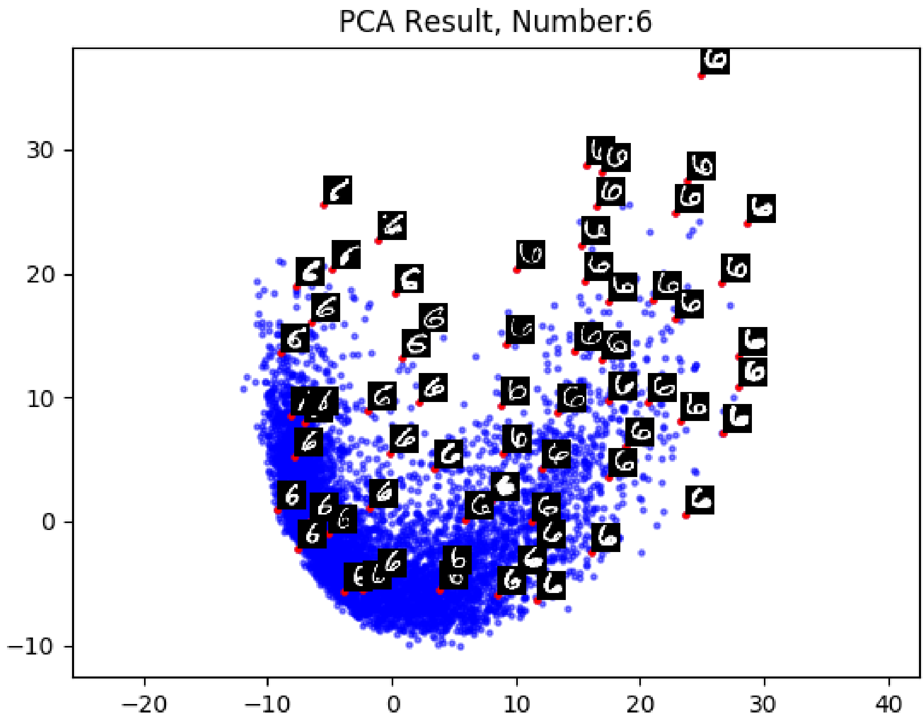
圖片可以觀察到，右邊的 3 較為向右傾斜，左邊的 3 較為向左傾斜，上面的 3 底較寬，下面的 3 頭部較寬。



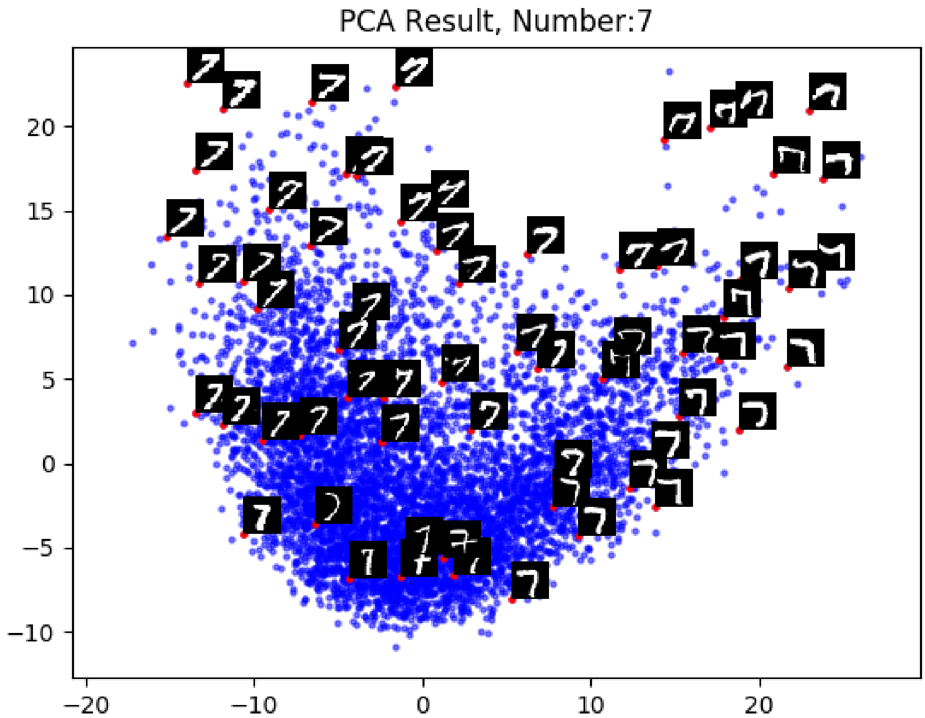
此張圖可以觀察到，在右邊的 4 線的厚度較厚並向左傾斜，左邊 4 的線則比較薄且向右傾斜，靠下的 4 較為瘦高，靠上的 4 較為寬胖。



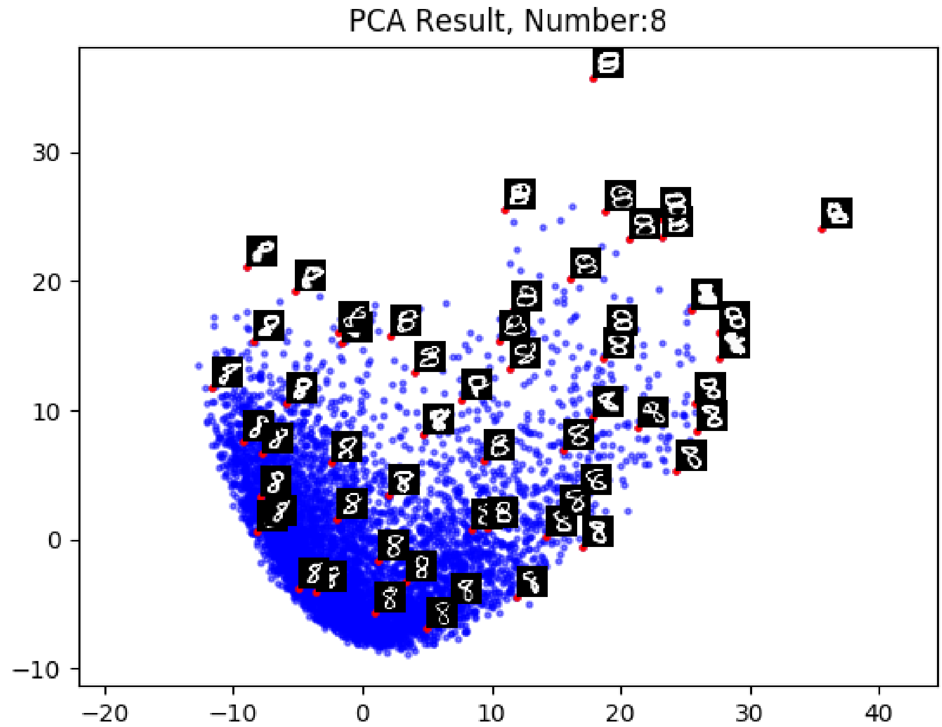
此張圖可以觀察到，在右邊的 5 較靠左傾斜厚度較厚，且底部的圓較圓滿，而左邊的 5 則靠右傾斜厚度較薄， 且底部的圓則比較扁。



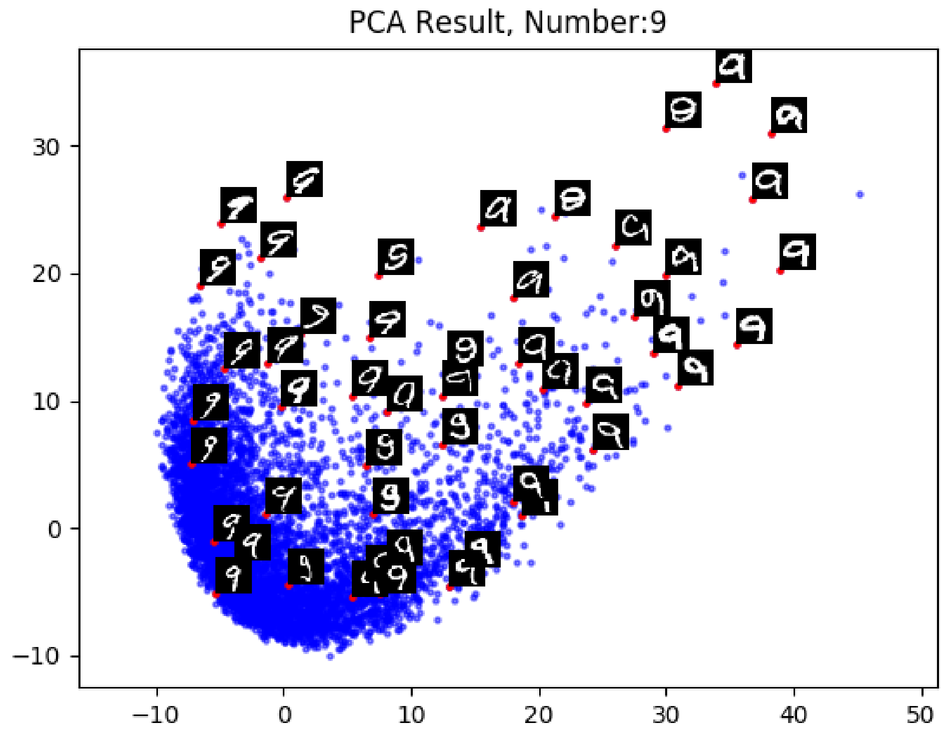
此張圖可以觀察到，在左邊的 6 向右傾斜，右邊的 6 向左傾斜，越靠下面的 6 則越正越標準。



此張圖可以觀察到，在左邊的 7 較為向右傾斜，右邊的 7 則比較正，而靠上面的 7 頭部會有突出，而下面的 7 頭部較平。

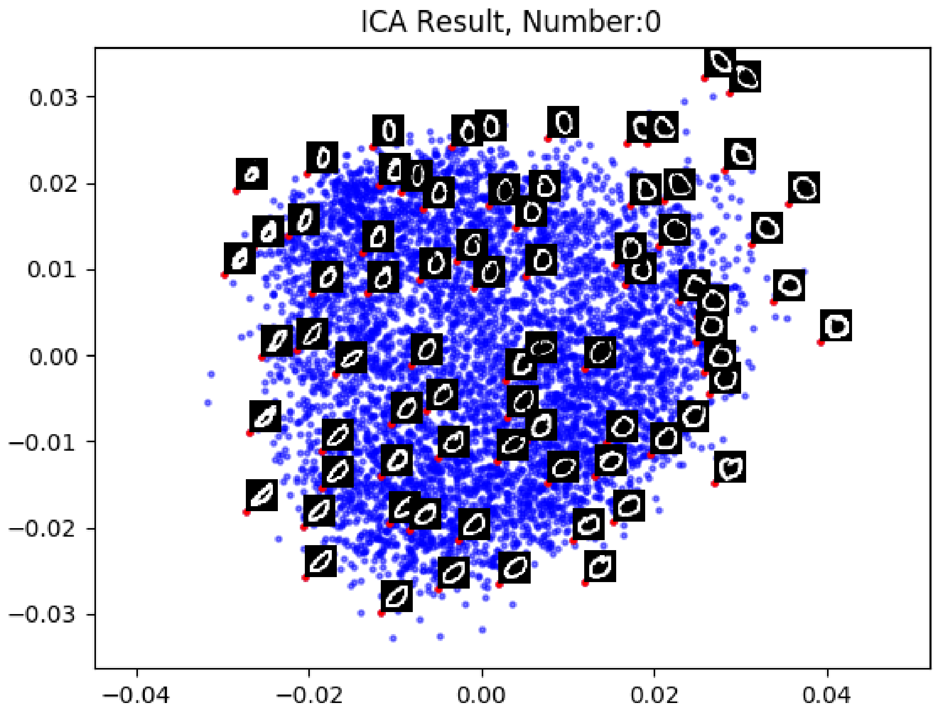


此張圖可以觀察到，在左邊的 8 較為向右傾斜，右邊的 8 則向左傾斜，而靠上面的 8 較寬，靠下面的 8 較瘦較直。

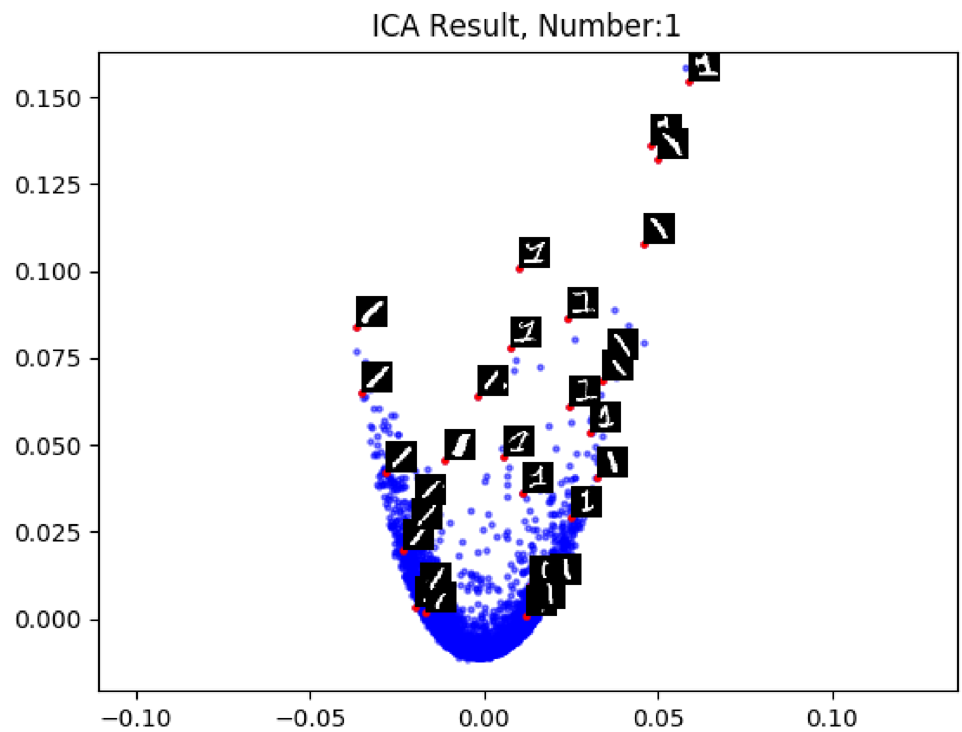


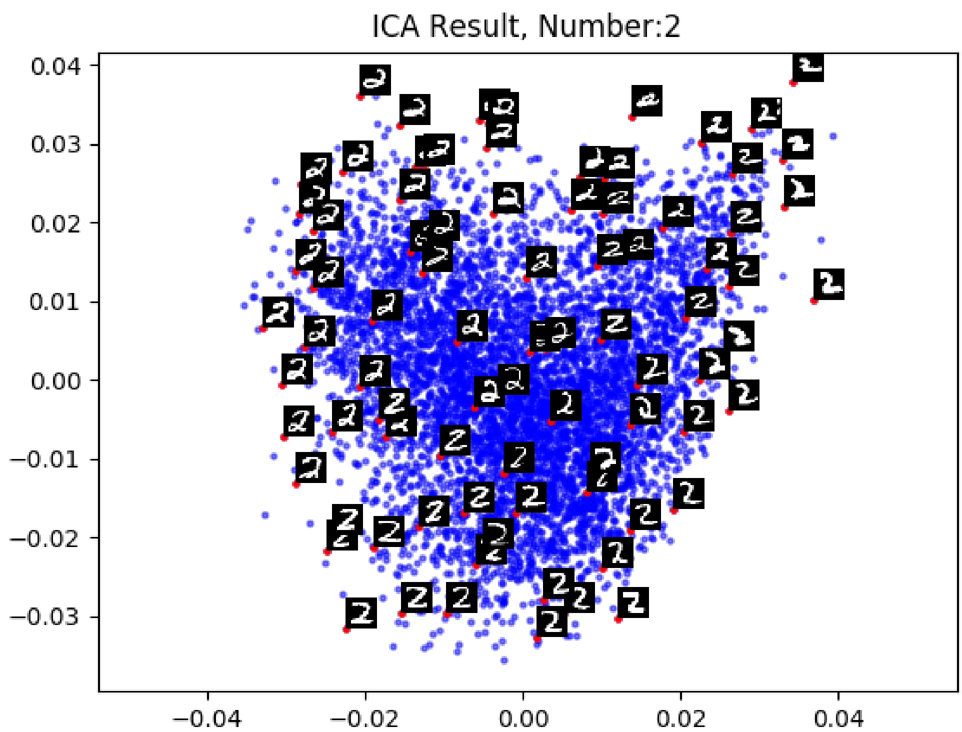
此張圖可以觀察到，在左邊的 9 較為向右傾斜，右邊的 9 則比較正，而靠上面的 9 比較扁頭比較大，而下面的 9 較為瘦高。

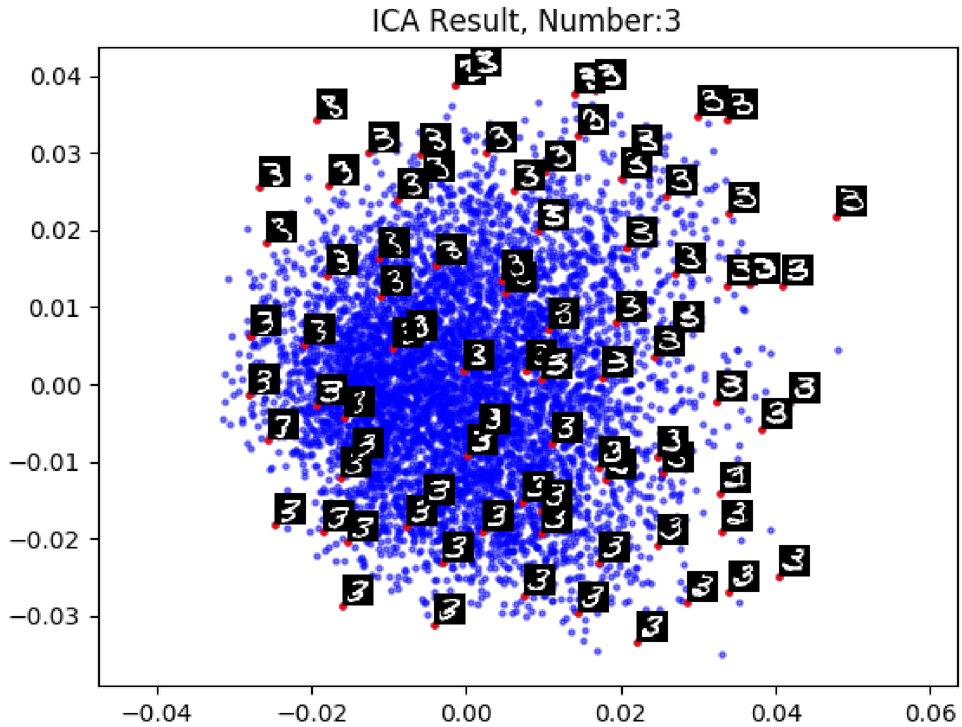
- **Bonus : Apply ICA and LLE to the same dataset and compare the results with PCA**
- **ICA Result:**

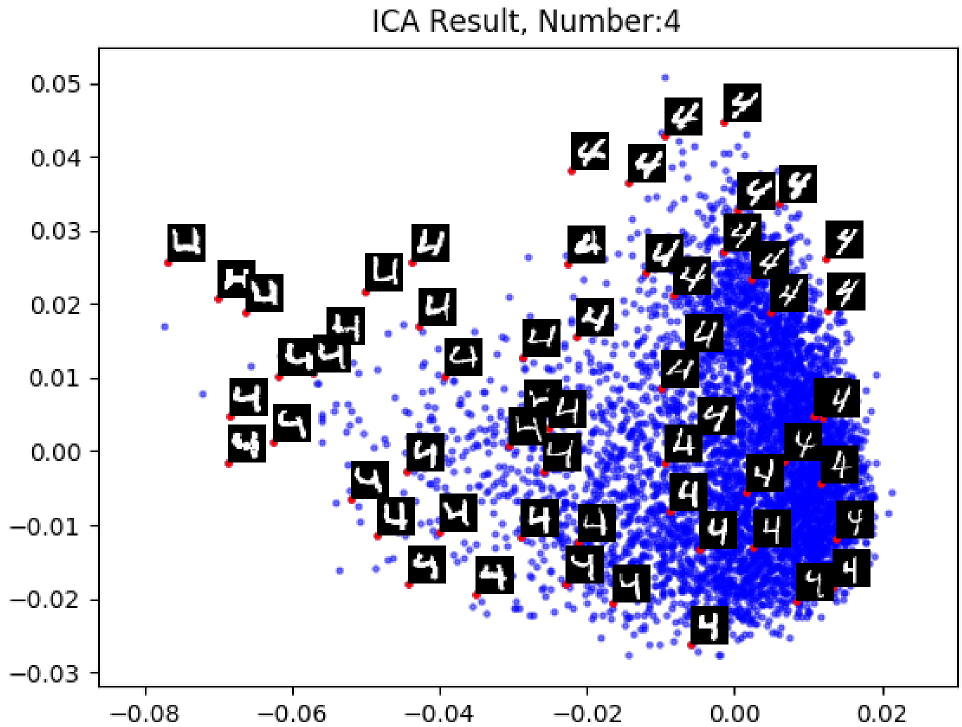


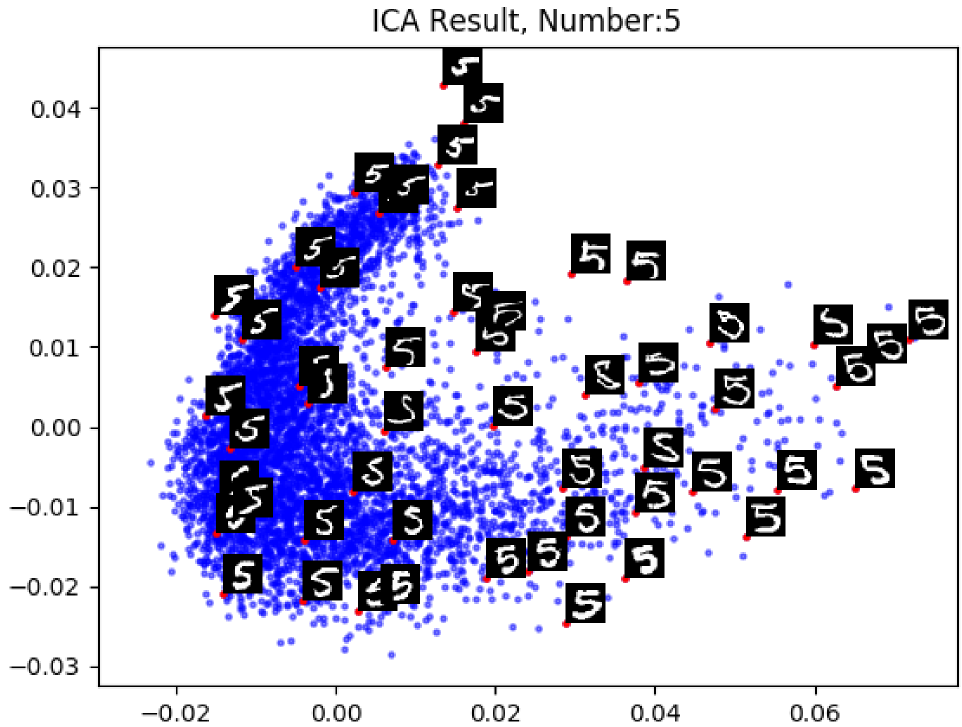
使用此圖來代表性概述 ICA 的結果圖，可以看到靠左的 0 較為向右傾斜，靠右的 0 較為向左傾斜，而中間區域的 0 有比較圓的趨勢。

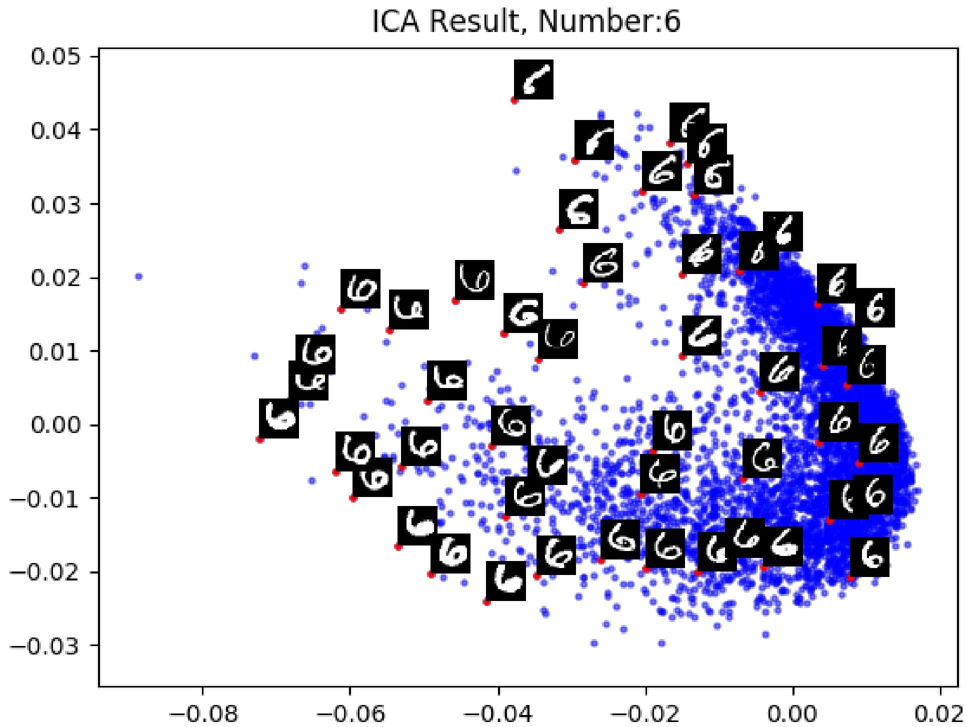

ICA Result, Number:1
0.150
0.125
0.100
0.075
0.050
0.025
0.000
−0.10
−0.05
0.00
0.05
0.10

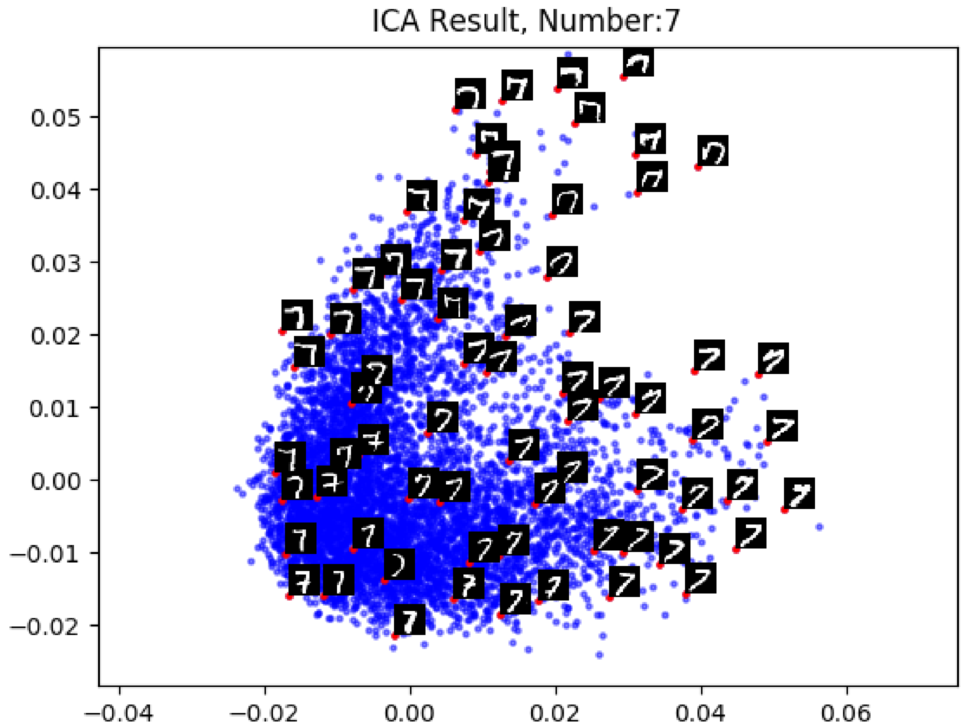

ICA Result, Number:2
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.02
0.00
0.02
0.04

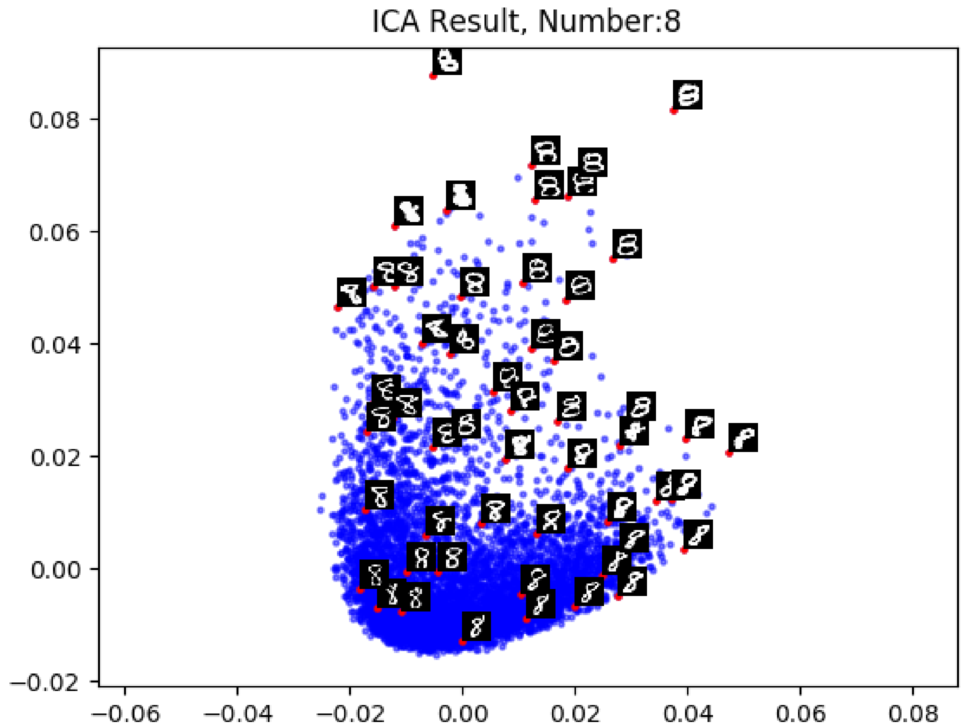

ICA Result, Number:3
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
0.00
0.02
0.04
0.06

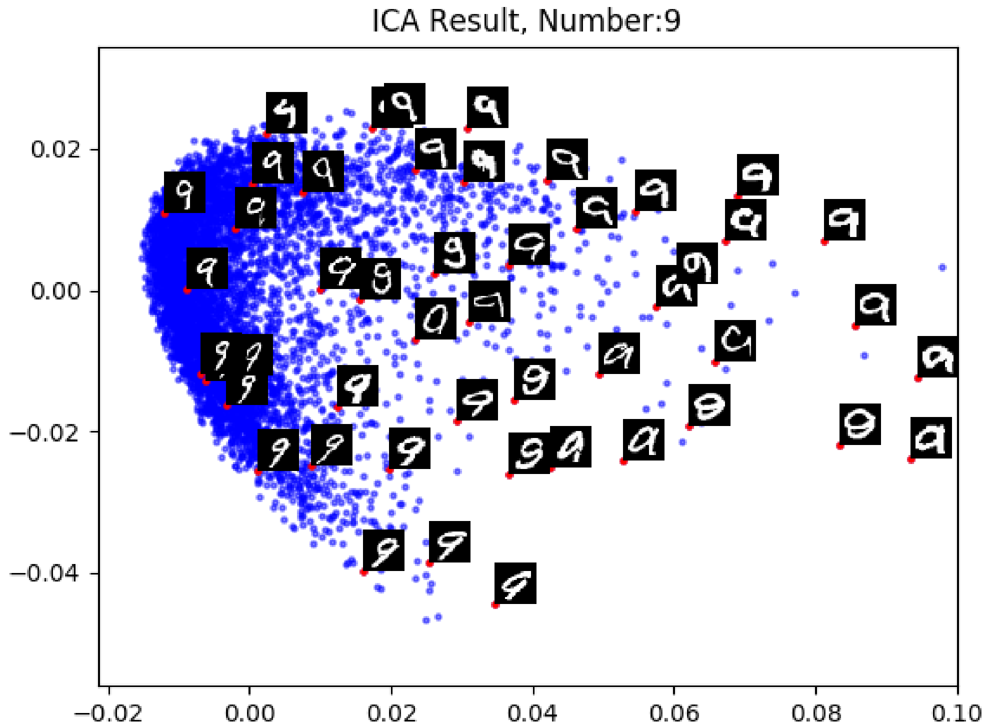

ICA Result, Number:4
0.05
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.08
−0.06
−0.04
−0.02
0.00
0.02


ICA Result, Number:5
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.02
0.00
0.02
0.04
0.06


ICA Result, Number:6
0.05
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.08
−0.06
−0.04
−0.02
0.00
0.02

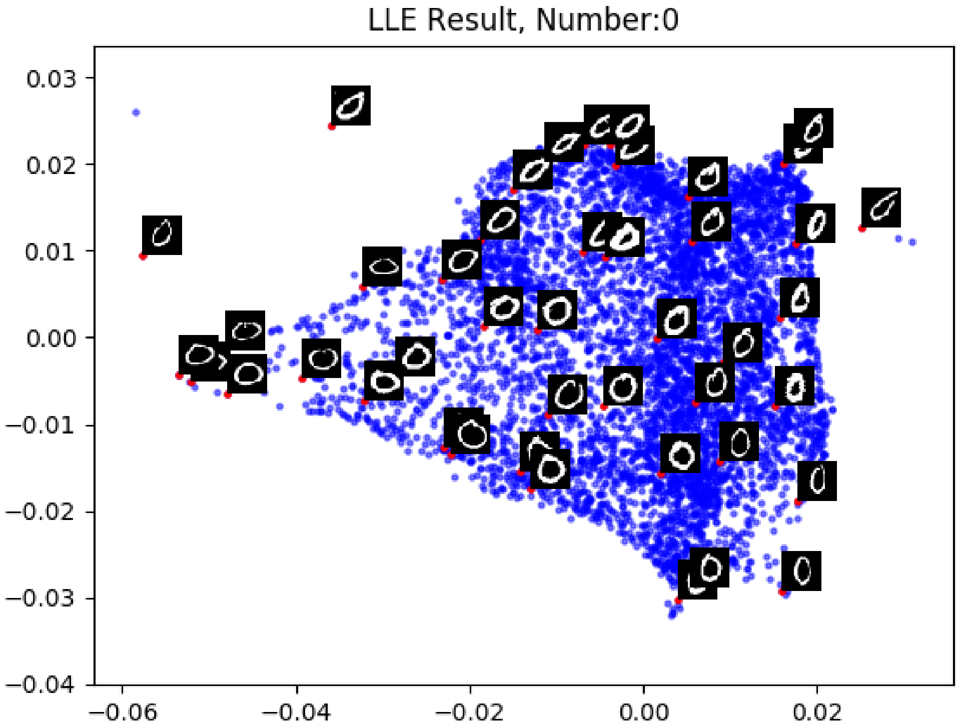

ICA Result, Number:7
0.05
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.04
−0.02
0.00
0.02
0.04
0.06


ICA Result, Number:8
0.08
0.06
0.04
0.02
0.00
−0.02
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08

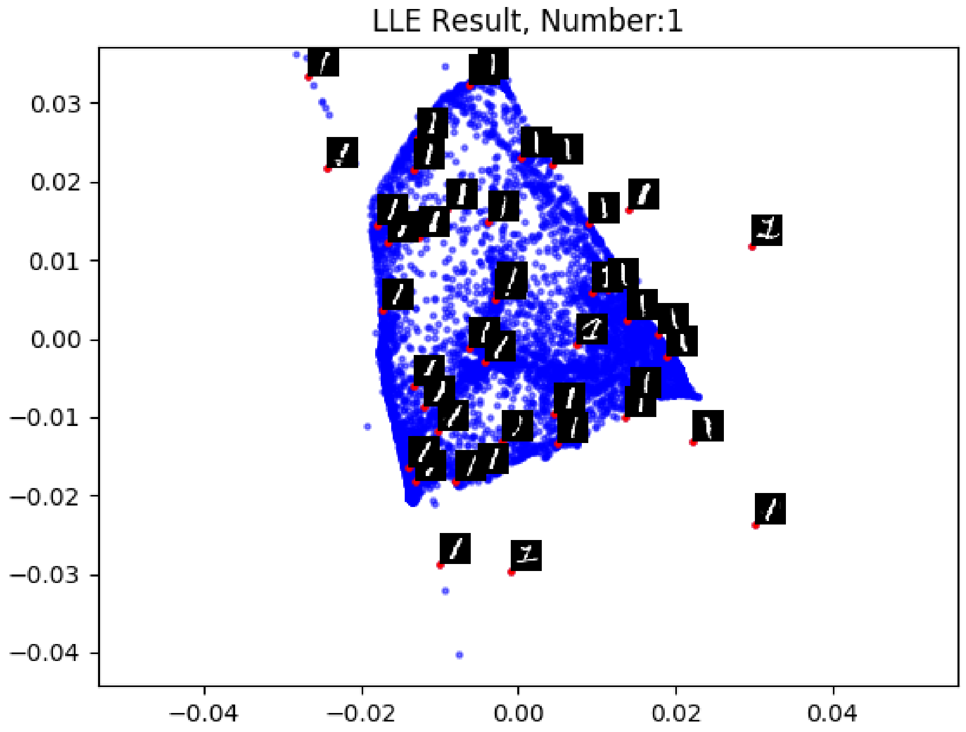


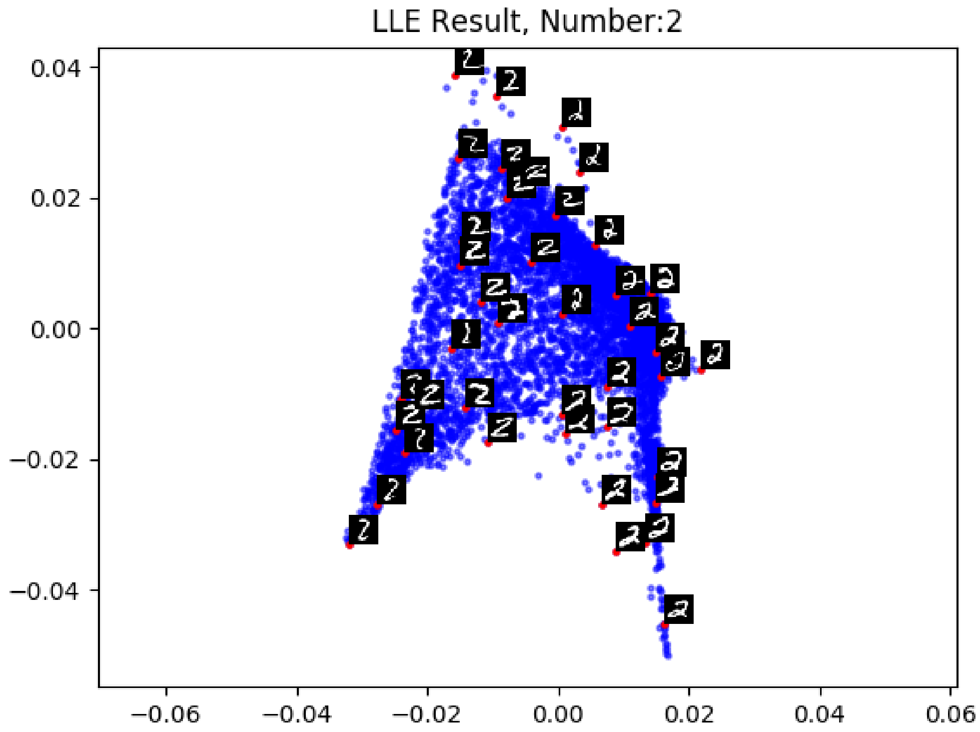
ICA Result, Number:9
0.02
0.00
−0.02
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
0.10

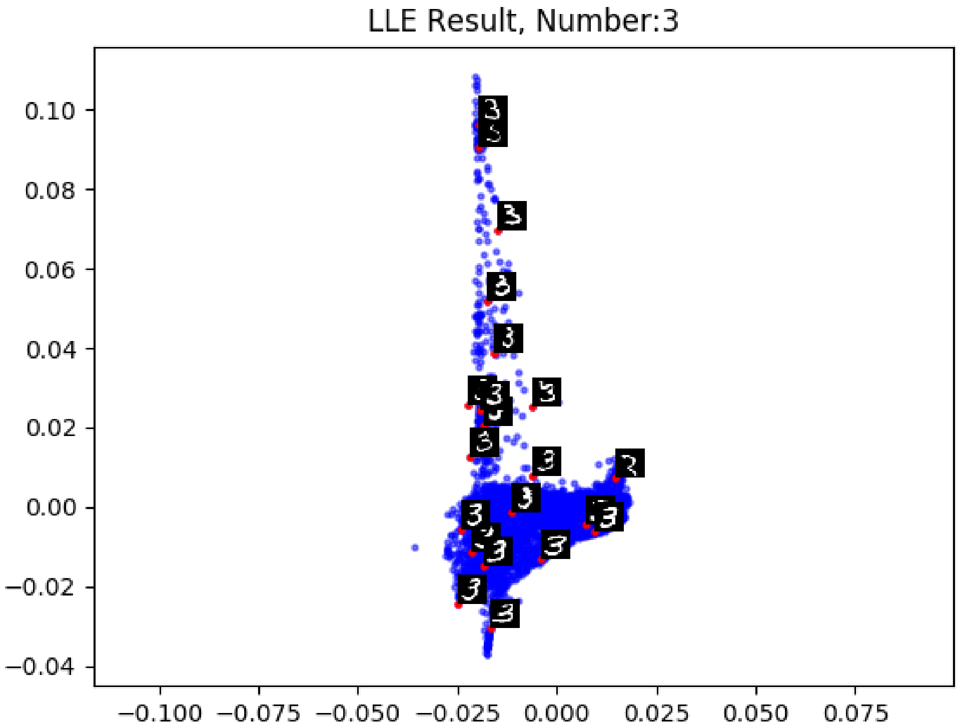
- **LLE Result:**

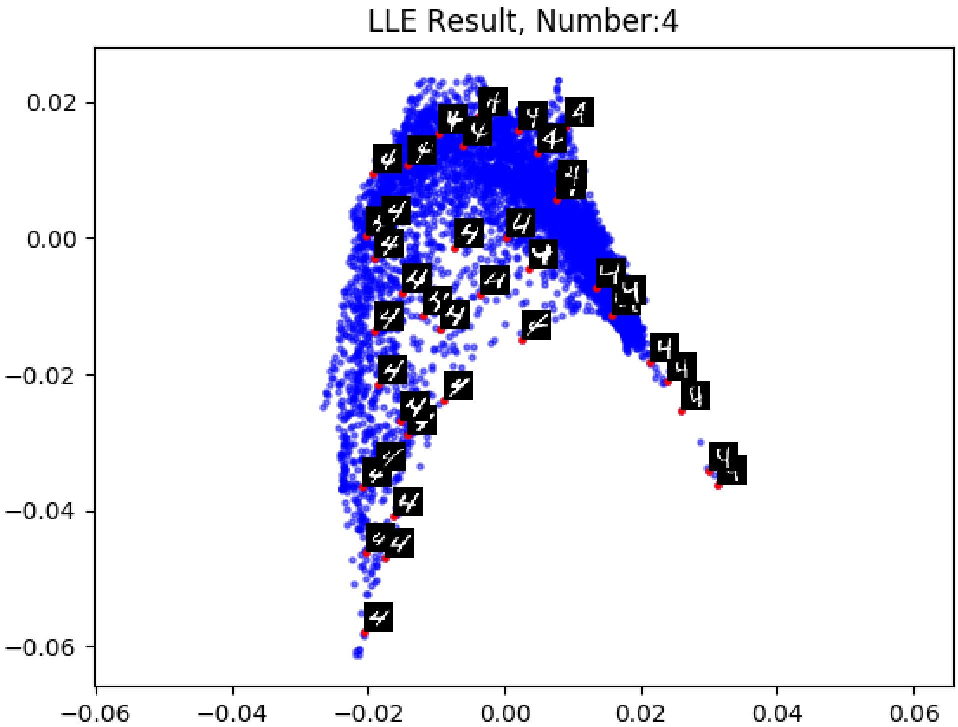


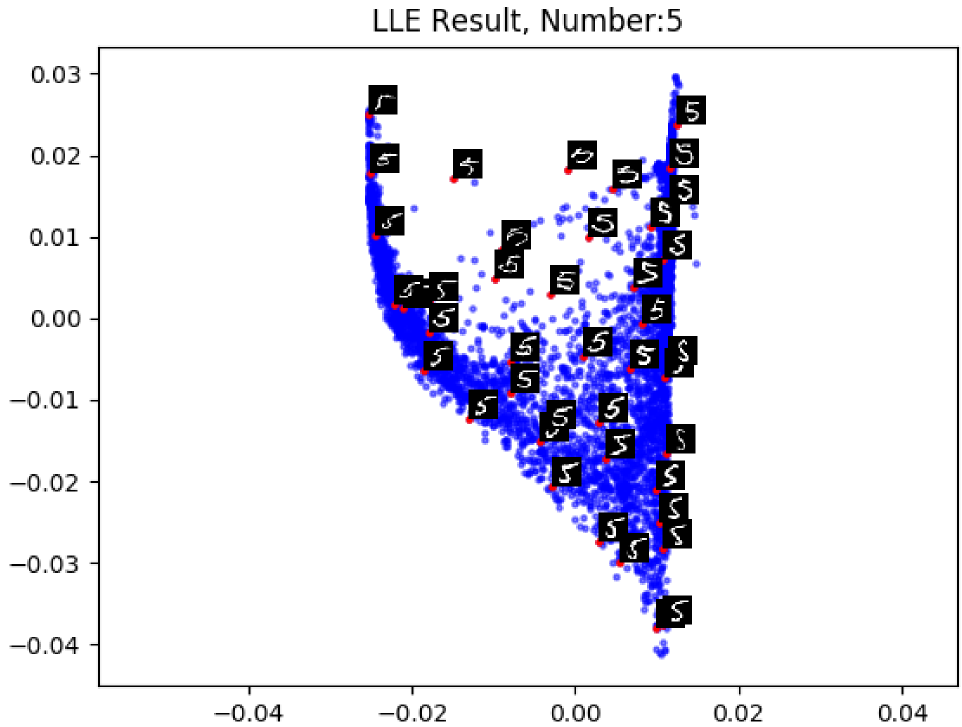
使用此圖來代表性概述 LLE 的結果圖，可以看到靠左的 0 比較圓，而靠右的 0 比較瘦長，靠上的 0 往右傾斜。

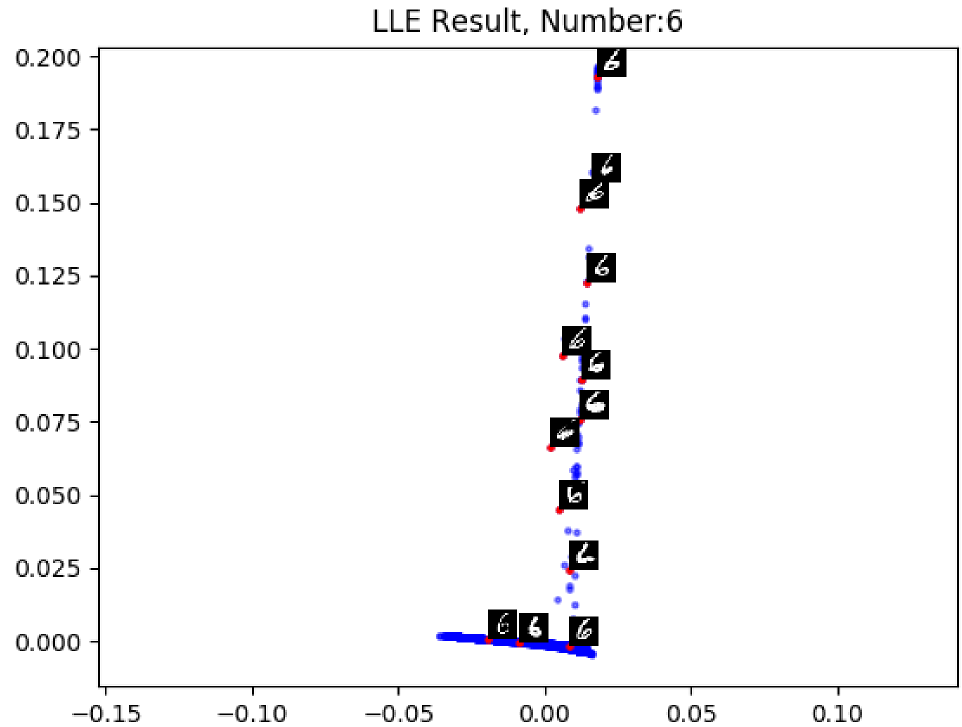

LLE Result, Number:1
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.04
−0.02
0.00
0.02
0.04

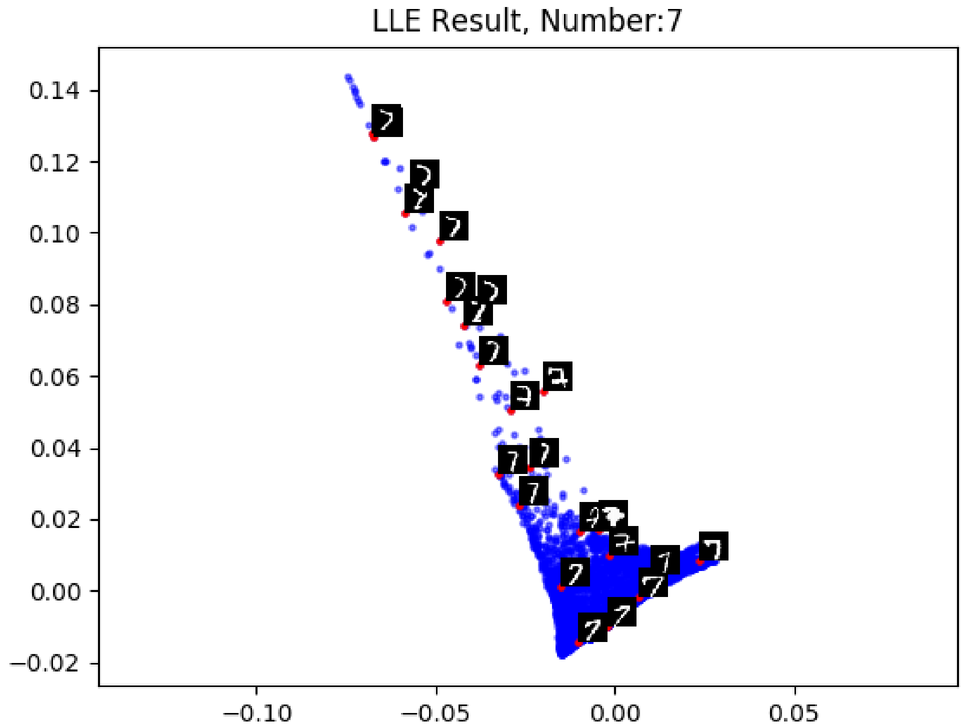

LLE Result, Number:2
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06

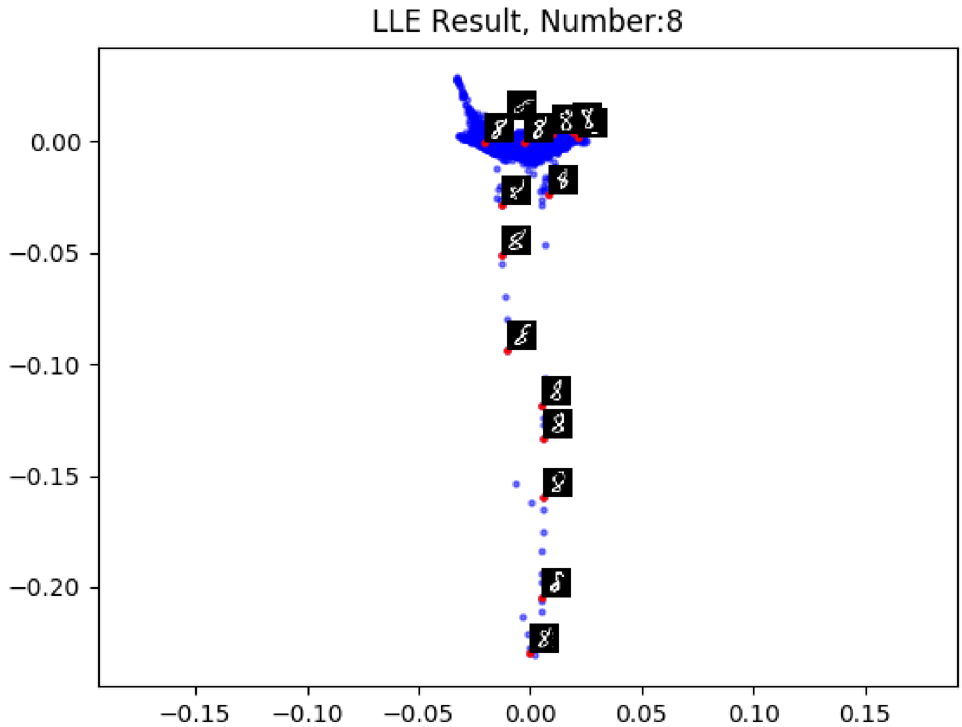

LLE Result, Number:3
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075

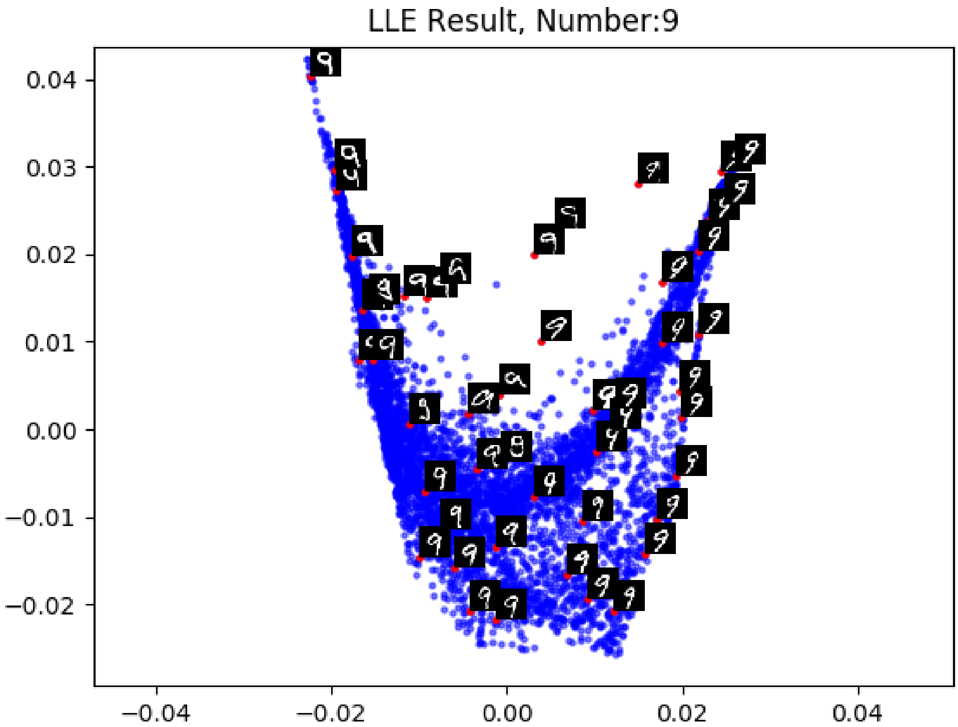

LLE Result, Number:4
0.02
0.00
−0.02
−0.04
−0.06
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06


LLE Result, Number:5
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04
−0.04
−0.02
0.00
0.02
0.04

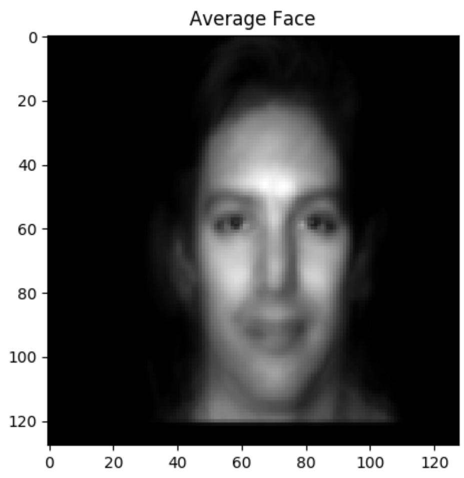

LLE Result, Number:6
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
−0.15
−0.10
−0.05
0.00
0.05
0.10


LLE Result, Number:7
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.10
−0.05
0.00
0.05

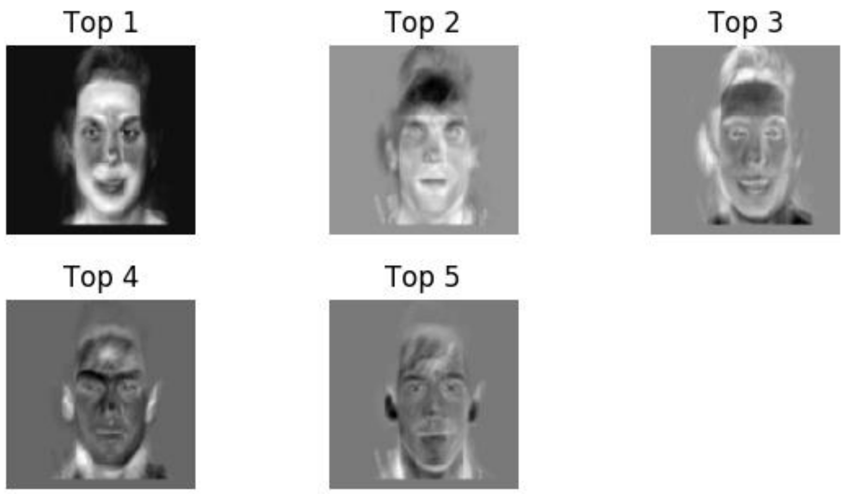

LLE Result, Number:8
0.00
−0.05
−0.10
−0.15
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15


LLE Result, Number:9
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.04
−0.02
0.00
0.02
0.04

- **Implement the eigenface algorithm (cf. page5) using the train.db database and do the following tasks.**
  - **Show the mean (average) face, top 5 eigenfaces and their corresponding eigenvalues in a descending order.**
    - **Average Face**



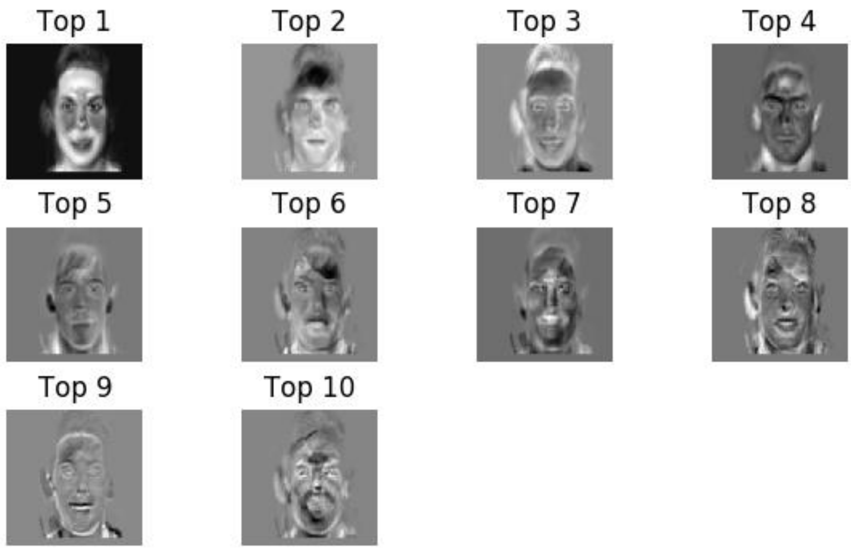
    - **Components = 5**
      - **Top 5 eigenfaces**



      - **corresponding eigenvalues in a descending order.**

```
Eigenvalues: [6146308.48282181 1662399.91841882  982888.71998104  499407.24694748
  437356.31540362]
```

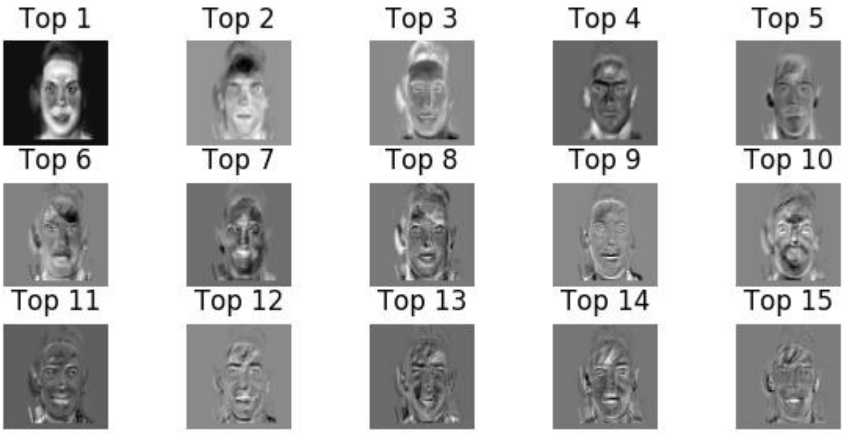
- **Components = 10**
  - **Top 10 eigenfaces**



  - **corresponding eigenvalues in a descending order.**

```
Eigenvalues: [6146308.48282181 1662399.9184196   982888.71999253  499407.25026548
  437356.3255072   267322.76777969  231461.60798345  178086.26986992
  136514.79366593  123411.39099485]
```

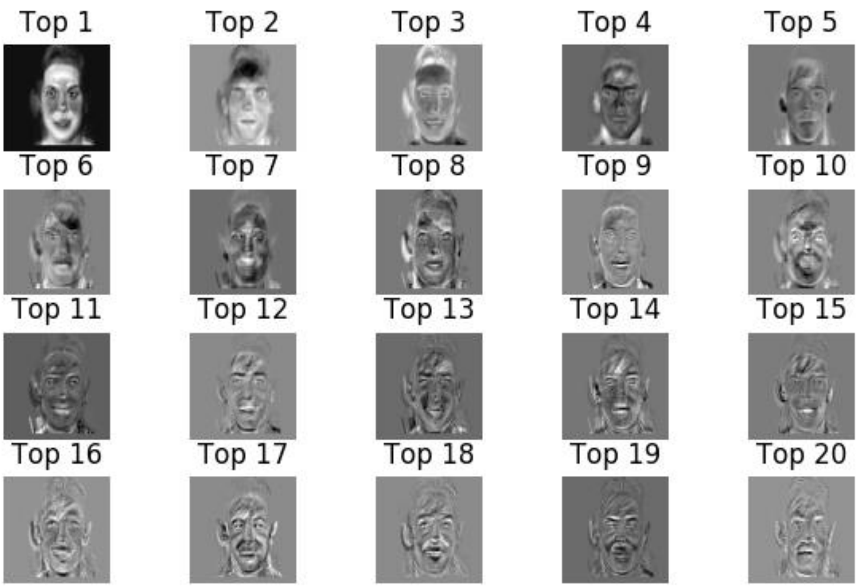
- **Components = 15**
  - **Top 15 eigenfaces**



- **corresponding eigenvalues in a descending order.**

```
Eigenvalues: [6146308.48282181 1662399.9184196   982888.71999289  499407.25026777
  437356.32555151  267322.77279614  231461.62439122  178086.35688938
  136516.19660503  123411.98359418  107905.0267098    98003.04146673
   82892.15208294   72838.33458125   68042.28019411]
```

- **Components = 20**
  - **Top 20 eigenfaces**

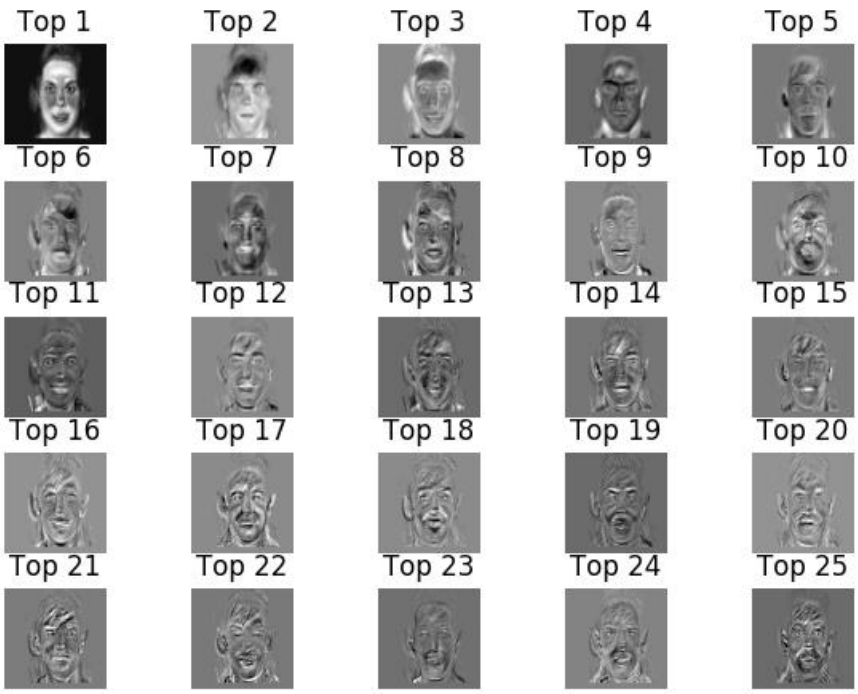


- **corresponding eigenvalues in a descending order.**

```
Eigenvalues: [6146308.48282181 1662399.9184196   982888.71999289  499407.25026785
  437356.32555527  267322.77282029  231461.62453521  178086.36332672
  136516.20211673  123411.99201207  107905.03238102   98003.2264027
   82893.25874754   72838.36765446   68042.32657949   60643.62646818
   53385.47980797   48093.50734452   42816.99549413   38805.33377982]
```

- **Components = 25**

    - **Top 25 eigenfaces**



- **corresponding eigenvalues in a descending order.**

```
Eigenvalues: [6146308.48282181 1662399.9184196   982888.71999289  499407.25026785
  437356.32555528  267322.77282045  231461.62453532  178086.36335403
  136516.20215778  123411.9920277   107905.0323984    98003.22696342
   82893.2600994    72838.37090522   68042.3280686    60643.82167666
   53385.56139995   48093.57485004   42817.71992532   38805.99571147
   37686.3654112    31873.45142717   28418.26943068   25093.06167213
   21345.82509245]
```

- **Given a test image (hw03-test.tif, or you can use your own image), compute the top 10 eigenface coefficients**
    - **Components = 5**

```
Top 5 eigenface coefficients:
[ 4519.32336656   166.30725411 -1335.98912539  -795.64870175
  -675.57532704]
```

    - **Components = 10**

```
Top 10 eigenface coefficients:
[ 4519.323366     166.30766891 -1335.99112193  -795.68639245
  -675.6749674   -105.792348     -94.17889777    64.79650699
   450.1779545   -382.21944595]
```

    - **Components = 15**

```
Top 15 eigenface coefficients:
[ 4519.32336602   166.30766237 -1335.99110128  -795.68775387
  -675.67674752  -105.79397822   -94.21944979    64.77347526
   450.35120641  -382.72108392  -421.74957408   428.69331941
  -510.76178963   497.65839966   371.98555368]
```

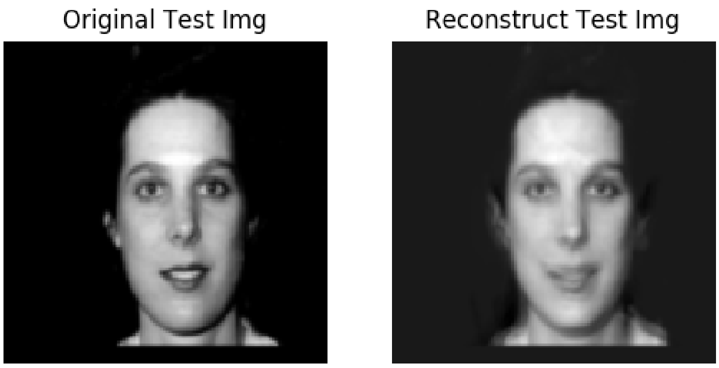
    - **Components = 20**

```
Top 20 eigenface coefficients:
[ 4.51932337e+03  1.66307663e+02 -1.33599110e+03 -7.95687818e+02
 -6.75676217e+02 -1.05794126e+02 -9.42147362e+01  6.47429506e+01
  4.50311032e+02 -3.82684791e+02 -4.21743552e+02  4.28921124e+02
 -5.11385089e+02  4.97599362e+02  3.71859030e+02 -1.42956045e+02
 -2.68806463e+02  1.81795711e+01  6.71796462e+01 -4.39064220e+00]
```

    - **Components = 25**

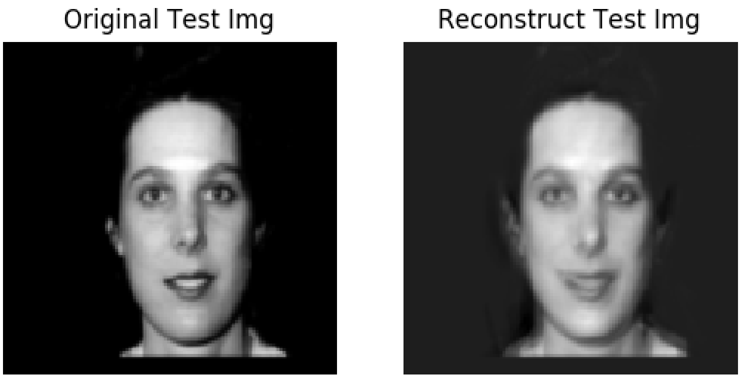
```
Top 25 eigenface coefficients:
[ 4.51932337e+03  1.66307663e+02 -1.33599110e+03 -7.95687819e+02
 -6.75676217e+02 -1.05794131e+02 -9.42147112e+01  6.47432756e+01
  4.50311043e+02 -3.82684222e+02 -4.21744074e+02  4.28917284e+02
 -5.11390234e+02  4.97606907e+02  3.71857702e+02 -1.43029813e+02
 -2.68875570e+02  1.81519729e+01  6.69645443e+01 -4.25197641e+00
 -3.49873485e+01 -7.70082215e+01 -3.86976511e+02  2.73020172e+02
 -5.12018102e+01]
```

- **Keep only first K (K=5,10,15,20, and 25) coefficients and use them to reconstruct the image in the pixel domain. Compare the reconstructed image with the original image by PSNR (Peak Signal to Noise Ration) value.**
    - **Components = 5**



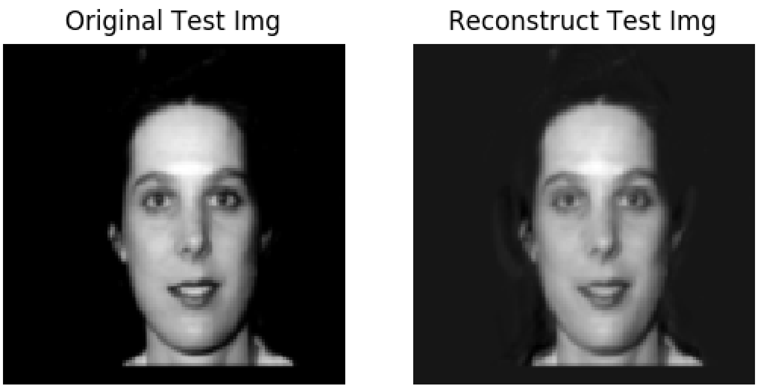
- **PSNR= 27.876904467956688**

- **Components = 10**



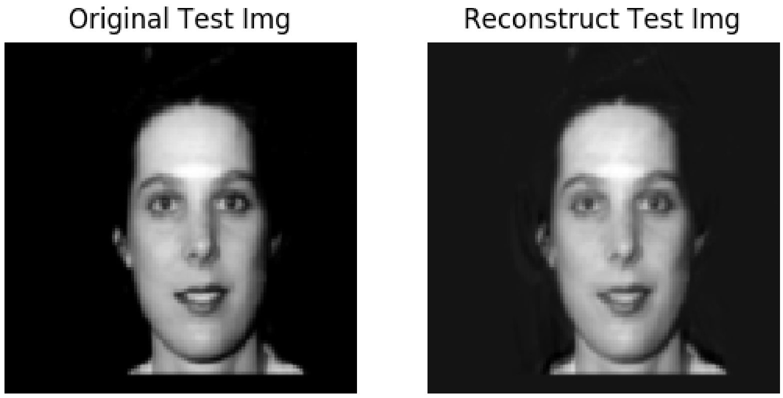
  - **PSNR= 28.924126598844122**

- **Components = 15**



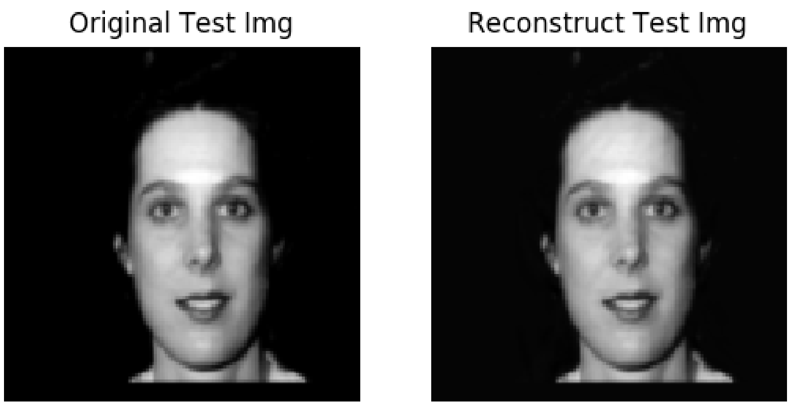
  - **PSNR= 34.77344107274608**

- **Components = 20**



  - **PSNR= 36.17912261729825**

- **Components = 25**



  - **PSNR= 46.70577829969278**

從上述的的結果中可以觀察到，Components 用越大，降維後能保留住越多人臉圖片原有的特徵，在使用 PCA 來進行 reconstruct 時，就越能夠還原出與原來 test data 人臉更為相近的臉，理所當然 PSNR 的值也會越大。